# A/UX® System Administration

## Participant's Guide

## Developed for A/UX Release 2.0

Printed in U.S.A. July, 1990

## Copyright Notice

## Limited Warranty

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

APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT OR THE SOFTWARE IT DESCRIBES, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT DESCRIBES SOFTWARE THAT IS SOLD "AS IS," AND THE PURCHASER ASSUMES THE ENTIRE RISK AS TO ITS QUALITY AND PERFORMANCE.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT IN THIS DOCUMENT OR THE SOFTWARE IT DESCRIBES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN PARTICULAR, APPLE SHALL HAVE NO LIABILITY FOR ANY PROGRAMS OR DATA STORED OR USED WITH APPLE PRODUCTS, INCLUDING THE COSTS OF RECOVERING SUCH PROGRAMS OR DATA.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. NO APPLE DEALER, AGENT, OR EMPLOYEE, IS AUTHORIZED TO MAKE ANY MODIFICATION, EXTENSION, OR ADDITION TO THIS WARRANTY.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES OR LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO THE OWNER OF THIS DOCUMENT OR THE SOFTWARE IT DESCRIBES. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

# Course Table of Contents

## Module 0—Preface

## Module 0A—Preface

## Module 1—Introduction to System Administration

## Module 2—Installating and Configuring A/UX

## Module 3—User Administration

## Module 4—Disk Drive Administration

## Module 5—Backup and Restore

## Module 6—Printer Administration

## Module 7—Terminal Administration

## Module 8—Autorecovery

## Module 9—Security

## Module 10—The Kernel

## Module 11—Process Management

## Module 12—Files

## Quick Reference

## Glossary

## Additional Resources

## Answers to Exercises

# Module 0
# Preface

# Table of Contents

# Welcome to A/UX System Administration

A/UX® 2.0

## Notes

# Preface

**Objectives**

By the time you complete this module, you will be able to:

❑ describe the goals of the course

❑ describe the notational conventions used in this course

**Activities**

Lecture/Discussion

**References**

*A/UX Essentials*

# Preface

A/UX® 2.0

## Notes

# Goals

By the time you have finished this course you will have learned and practiced the following skills:

❑ Install A/UX
❑ Expand the system
❑ Use the Startup utility
❑ Add users to the system
❑ Add an external disk drive
❑ Backup files using several methods
❑ Add printers to the system
❑ Configure a Macintosh to act as a terminal
❑ Use the autorecovery features of A/UX
❑ Explain kernel tuning

By the end of the course you'll have practical, hands-on experience in administering the A/UX operating system.

# Goals

- ❑ Install A/UX
- ❑ Expand the system
- ❑ Use the Startup utility
- ❑ Add users to the system
- ❑ Add an extemal disk drive
- ❑ Back up files using several methods
- ❑ Add printers to the system
- ❑ Configure a Macintosh to act as a terminal
- ❑ Use the autorecovery features of A/UX
- ❑ Explain kemel tuning

3

## Notes

# Course Format

The course is taught in a series of modules, each covering a general subject. It is expected we'll spend half of the class time in hands-on exercises, and the other half in lecture. The material is presented in a logical order, with later modules building on the information presented in prior modules. Thus, if you have a question, bring it up right away so your understanding will not be hampered as we go on. As you can already see, the course materials are provided in binders.

We'll begin each module with a brief lecture. At the end of each module are checkpoints, presented as a list of questions. These cover the objectives of each module, so we'll go over them to make sure you know the answers. (Answers are provided at the end of the binder.) Once we've finished this review, we'll dive into the hands-on exercises.

At the end of the course we'll conduct a test covering the materials in this course.

# Course Format

□ 50% Lecture

□ 50% Hands-on

- Checkpoints
- Exercises

□ Post-test

## Notes

# Agenda

The course begins with a brief introduction to A/UX system administration, the whys and wherefores. We then cover configuring and starting A/UX, including the setting the software to automatically boot A/UX. After this, we discuss user administration, the adding and removing of users.

A practical module on adding disk drives is next. A critical topic, file backups, are then covered and four methods for doing this are practiced.

We then cover connectivity with other devices, printers and terminals, and how to configure the system for their use. The subject of A/UX autorecovery is discussed, followed by a module on security, covering such topics as passwords, superuser privileges and file permissions.

We then review the concepts of processes (introduced in Shell Programming) and we wrap up the course with a look at tunable kernel parameters.

**Review**

Once you have completed this course, you will have a practical understanding of administering the A/UX operating system, which will enable you to better perform your job.

By participating in the exercises and lectures, and reviewing your own progress with each module's checkpoints, you will not only be able to pass the course post-test, but will see many ways that this new knowledge is directly applicable to your profession.

The intent of the course is to help you succeed, because after all, Apple's success is based on <u>your</u> success.

# Agenda

Day 1, Morning:
- Preface
- Introduction
- Installing and Configuring A/UX
- User Administration

Day 1, Afternoon:
- Disk Drive Administration
- Backups

Day 2, Morning:
- Printer Administration
- Terminal Administration
- Autorecovery

Day 2, Afternoon:
- Security
- The Kernel
- Process Management
- Post-test and Evaluation

## Notes

# Notational Conventions

Several kinds of notational conventions are used in this course. Among them are:

## Font Usage

Words that you would see on a display screen appear in `Courier font`. Additionally, in examples, the data typed in by the user is shown in **`Courier boldface`**. Words that you would replace with a value appropriate to a particular set of circumstances appear in *`Courier italics`*.

New terminology, whose definition appear in the Glossary, is shown in **Times boldface** the first time it is used in the text. References to publications are given in *Times Italic.*

## Key Presses

Certain keys are identified with names on the keyboard. These modifier and character keys perform functions, and are often used in combination with other keys. The names of these keys appear in SMALL CAPS foımat.

If a key combination is to be formed, that is, if a key is to be used in combination with another key, the two keycap specifications will be separated by a hyphen. An example is CONTROL-C. To correctly form such a combination, press and hold down the leftmost key of the combination, then press and release the rightmost key of the combination.

# Notational Conventions

| | |
|---|---|
| Courier | data displayed on screen |
| **Courier Bold** | user-entered data |
| *Courier Italic* | text to be replaced |
| **Times Bold** | glossary item |
| *Times Italic* | publication reference |
| SMALL CAPS | keyboard keys |
| KEY-COMBINATION | multiple keyboard keys |

## Notes

**Terminology**

In A/UX Manuals, a certain term can represent a specific set of actions. For example, the word Enter indicates that you type in an entry and press the RETURN key. Here is a list of common terms and their corresponding actions:

**Type** Type in the letter or letters *without* pressing the RETURN key.

**Enter** Type the entry and press the RETURN key.

**Press** Press a *single* letter or key *without* pressing the RETURN key.

**Click** Press and then immediately release the mouse button.

**Select** Position the pointer on an icon, and click the mouse button.

**Drag** Position the pointer on an icon, and then press and hold down the mouse button while moving the mouse. Release the mouse button when you reach the desired position.

**Choose** Activate a command title in the menu bar. While holding down the mouse button, drag the pointer to a command name in the indicated menu and then release the mouse button.

# Terminology

Type

Enter

Press

Click

Select

Drag

Choose

7

## Notes

**Syntax Notation**

A/UX commands follow a specific order of entry. A typical command has this form:

**Syntax**

```
command {req1 | req2} [option...] [argument ...]
```

**`command`** is the command name.

**{ }** (braces) - items that are enclosed in braces (curly brackets) are required items. If this is given with a command name, one of the items so enclosed must be provided. Mutually exclusive options are separated by a vertical bar ( | ).

***`option`*** is one or more arguments that modify the command. Most options have the form [ -opt] where opt is a letter representing an option. Commands can take zero or more options.

***`argument`*** is a modification or a specification of the command; usually a filename or symbols representing one or more filenames.

**[ ]** (brackets) - items that are enclosed in square brackets are optional items. If this is given with a command name, items so enclosed may be provided but are not necessary to the functioning of the command.

**. . .** (ellipses) - follow an argument that may be repeated.

**Command Reference**

Reference manual pages are available in hardcopy as the *A/UX Command Reference*. In this format, the commands are grouped into sections, listed alphabetically within each section. Throughout the course, you will find references to *cmd(n)*. This refers to the command named `cmd` in section n of the manual. Thus *date(1)* refers to the `date` command in manual section 1. Just because you do not find a particular command name alphabetically ordered does not necessarily mean that the command is not documented. It is possible that it is in another section.

Because of the way the *Command Reference* manual is maintained, page numbering is not used. Each command starts on a separate page 1.

# Syntax and References

```
command {req1 | req2...} [ option...] [argument ...]
```

*reference(1)*

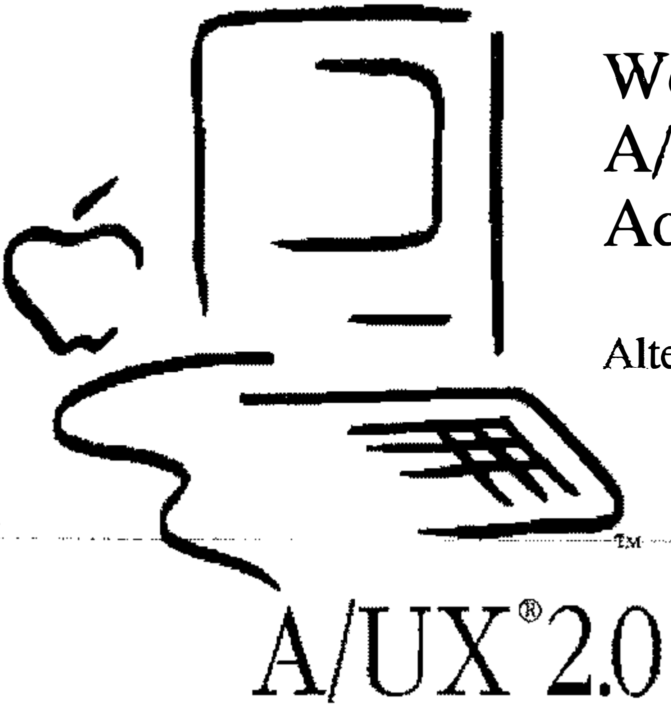
8

## Notes

# Module 0A
# Preface

## Alternate Learning Path

# Table of Contents



# Welcome to A/UX System Administration

Alternate Learning Path

## Notes

# Preface

**Objectives**

By the time you complete this module, you will be able to:

❑ describe the goals of the course

❑ describe the notational conventions used in this course

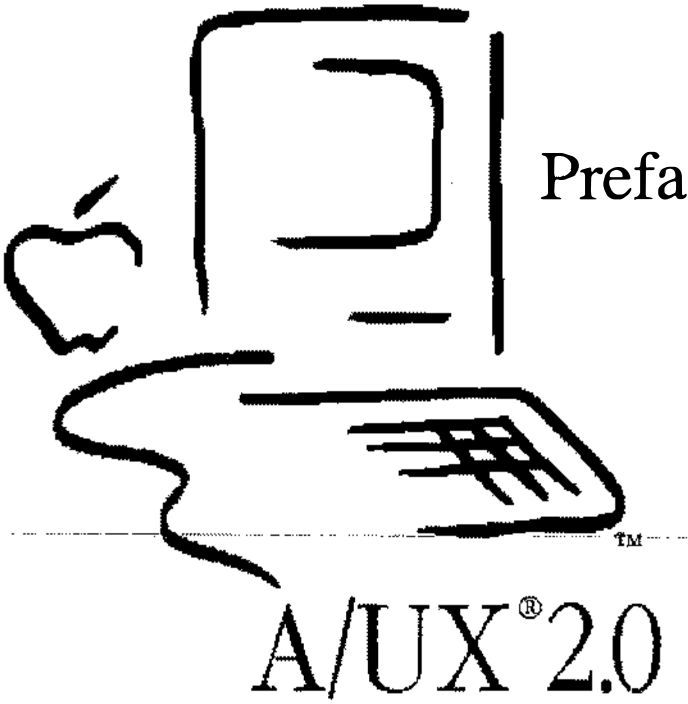
**Activities**

Lecture/Discussion

**References**

*A/UX Essentials*

# Preface



## Notes

# Goals

By the time you have finished this course you will have learned and practiced the following skills:

❑ Install A/UX
❑ Use the Startup utility
❑ Add an external disk drive
❑ Add printers to the system
❑ Use the autorecovery features of A/UX
❑ Discuss A/UX security issues

By the end of the course you'll have practical, hands-on experience in administering the A/UX operating system.

# Goals

- ❑ Install A/UX
- ❑ Use the Startup utility
- ❑ Add an external disk drive
- ❑ Add printers to the system
- ❑ Use the autorecovery features of A/UX
- ❑ Discuss A/UX security issues
- ❑ Explain kernel tuning

3

## Notes

# Course Format

The course is taught in a series of modules, each covering a general subject. It is expected we'll spend half of the class time in hands-on exercises, and the other half in lecture. The material is presented in a logical order, with later modules building on the information presented in prior modules. Thus, if you have a question, bring it up right away so your understanding will not be hampered as we go on. As you can already see, the course materials are provided in binders.

We'll begin each module with a brief lecture. At the end of each module are checkpoints, presented as a list of questions. These cover the objectives of each module, so we'll go over them to make sure you know the answers. (Answers are provided at the end of the binder.) Once we've finished this review, we'll dive into the hands-on exercises.

At the end of the course we'll conduct a test covering the materials in this course.

# Course Format

- ❑ 50% Lecture
- ❑ 50% Hands-on
  - Checkpoints
  - Exercises
- ❑ Post-test

A/UX

4

## Notes

# Agenda

The course begins with configuring and starting A/UX, including the setting the software to automatically boot A/UX.

A module on adding disk drives is next, including some information on the dp utility, which can be used on third-party drives.

We then cover connectivity with printers, and how to configure the system for their use. After this, the subject of A/UX autorecovery is discussed.

A module on security is next, covering the special problems of security on an A/UX system, as well as passwords, superuser privileges and file permissions. The course ends with a discussion of how kernels are created and adjusted.

## Review

Once you have completed this course, you will have a practical understanding of administering the A/UX operating system, which will enable you to better perform your job.

By participating in the exercises and lectures, and reviewing your own progress with each module's checkpoints, you will not only be able to pass the course post-test, but will see many ways that this new knowledge is directly applicable to your profession.

The intent of the course is to help you succeed, because after all, Apple's success is based on <u>your</u> success.

# Agenda

Morning:
Preface
Installing and Configuring A/UX
Disk Drive Administration
Printer Administration

Afternoon:
Printer Administration
Autorecovery
Security
The Kernel
Post-test and Evaluation

5

## Notes

# Notational Conventions

Several kinds of notational conventions are used in this course. Among them are:

## Font Usage

Words that you would see on a display screen appear in `Courier font`. Additionally, in examples, the data typed in by the user is shown in **`Courier boldface`**. Words that you would replace with a value appropriate to a particular set of circumstances appear in *`Courier italics`*.

New terminology, whose definition appear in the Glossary, is shown in **Times boldface** the first time it is used in the text. References to publications are given in *Times Italic.*

## Key Presses

Certain keys are identified with names on the keyboard. These modifier and character keys perform functions, and are often used in combination with other keys. The names of these keys appear in SMALL CAPS format.

If a key combination is to be formed, that is, if a key is to be used in combination with another key, the two keycap specifications will be separated by a hyphen. An example is CONTROL-C. To correctly form such a combination, press and hold down the leftmost key of the combination, then press and release the rightmost key of the combination.

# Notational Conventions

| | |
|---|---|
| `Courier` | data displayed on screen |
| **`Courier Bold`** | user-entered data |
| *`Courier Italic`* | text to be replaced |
| **Times Bold** | glossary item |
| *Times Italic* | publication reference |
| SMALL CAPS | keyboard keys |
| KEY–COMBINATION | multiple keyboard keys |

6

## Notes

**Terminology**

In A/UX Manuals, a certain term can represent a specific set of actions. For example, the word Enter indicates that you type in an entry and press the RETURN key. Here is a list of common terms and their corresponding actions:

**Type** Type in the letter or letters *without* pressing the RETURN key.

**Enter** Type the entry and press the RETURN key.

**Press** Press a *single* letter or key *without* pressing the RETURN key.

**Click** Press and then immediately release the mouse button.

**Select** Position the pointer on an icon, and click the mouse button.

**Drag** Position the pointer on an icon, and then press and hold down the mouse button while moving the mouse. Release the mouse button when you reach the desired position.

**Choose** Activate a command title in the menu bar. While holding down the mouse button, drag the pointer to a command name in the indicated menu and then release the mouse button.

# Terminology

Type

Enter

Press

Click

Select

Drag

Choose

7

## Notes

**Syntax Notation**

A/UX commands follow a specific order of entry. A typical command has this form:

**Syntax**

```
command {req1 | req2} [option...] [argument ...]
```

`command` is the command name.

**{ }** (braces) - items that are enclosed in braces (curly brackets) are required items. If this is given with a command name, one of the items so enclosed must be provided. Mutually exclusive options are separated by a vertical bar ( | ).

***`option`*** is one or more arguments that modify the command. Most options have the form [ -opt] where opt is a letter representing an option. Commands can take zero or more options.

***`argument`*** is a modification or a specification of the command; usually a filename or symbols representing one or more filenames.

**[ ]** (brackets) - items that are enclosed in square brackets are optional items. If this is given with a command name, items so enclosed may be provided but are not necessary to the functioning of the command.

**. . .** (ellipses) - follow an argument that may be repeated.

**Command Reference**

Reference manual pages are available in hardcopy as the *A/UX Command Reference*. In this format, the commands are grouped into sections, listed alphabetically within each section. Throughout the course, you will find references to *cmd(n)*. This refers to the command named `cmd` in section n of the manual. Thus *date(1)* refers to the `date` command in manual section 1. Just because you do not find a particular command name alphabetically ordered does not necessarily mean that the command is not documented. It is possible that it is in another section.

Because of the way the *Command Reference* manual is maintained, page numbering is not used. Each command starts on a separate page 1.

# Syntax and References

```
command {req1 | req2...} [option...] [argument ...]
```

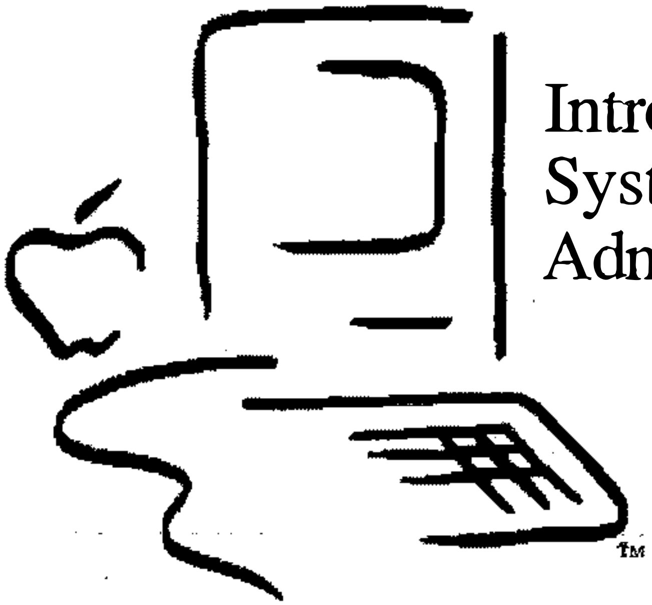
*reference(1)*

A/UX

## Notes

# Module 1
# Introduction to System Administration

# Table of Contents



# Introduction to System Administration

A/UX® 2.0

## Notes

# Introduction to System Administration

**Goal**

The goal of Module 1 is to introduce the concepts of system administration for A/UX systems.

**Objectives**

By the end of this module, participants will be able to:

- list the duties of an A/UX local system administrator
- describe the differences between root and user privileges
- identify the manuals containing administration information

**Activities**

Lecture/Discussion/Checkpoints

# Objectives

- □ List duties of an A/UX local system administrator
- □ Describe differences between root and user privileges
- □ Identify the manuals containing administration information

2

## Notes

# Administering A/UX

The health and productivity of a UNIX® computer system depends heavily on its system administrator. The system administrator, whose administrative login is root, has ultimate power throughout the system.

System administration in A/UX is much the same as any UNIX system, although many procedures are streamlined in A/UX. Below are some of the A/UX features that make the system administrator’s job easier:

- A/UX Startup utility
- autorecovery
- simplified procedures for adding users
- simplified procedures for adding devices
- autoconfiguration

In order to efficiently administer a UNIX system, one user is given access permission to all files on the system. This user is called the superuser and normally has the login name “root”. When you are logged in as root you can remove any file (or group of files) on the entire system. This means you have great power, and must exercise great caution in all of your activities as a system administrator.

To avoid accidental catastrophe, you should log in as root only when necessary. Difficulties can arise from a naive system administrator’s activities.

**References** *A/UX Local System Administration*

# Administering A/UX

- □ Ultimate power on A/UX system
- □ Similar to system administration on other UNIX systems
- □ Administration simplified with A/UX added value features
- □ Requires care and caution in everything you do

3

## Notes

# Responsibilities of the System Administrator

The system administrator is occupied from the time of booting up the system for its first time, through shut down, software updates, the addition of new users and new peripherals, etc.

These are some of the commands commonly used by System Administrators to complete various tasks. Those unique to A/UX are shown in boldface.

- System startup and shut down
  **`Startup`**, **`launch`**, **`restart`**, `shutdown`

- Add, move, and remove users;
  **`adduser`**, `vipw`, `cpio`, `tar`, `pax`

- Device management
  **`newconfig`**, **`setport`**, **`fsentry`**, `newfs`, `mkfs`, `mount`, `mknod`

- Maintain data and file system integrity
  **`esch`**, **`escher`**, **`eu`**, **`eupdate`**, `fsck`

- Backup and restore
  `tar`, `cpio`, `pax`

- Monitor system usage
  `uptime`, `ruptime`, `rwho`, `w`, `ps`, `whodo`, `pstat`

- Install new software and updates
  `tar`, `cpio`, `pax`

**References**

*A/UX Local System Administration*
*A/UX Network System Administration*

# Administrator Responsibilities

- ❑ System installation
- ❑ User accounts and environments
- ❑ Peripheral device management
- ❑ File system integrity
- ❑ Backup and restore
- ❑ Software/update maintenance
- ❑ System activity and accounting

4

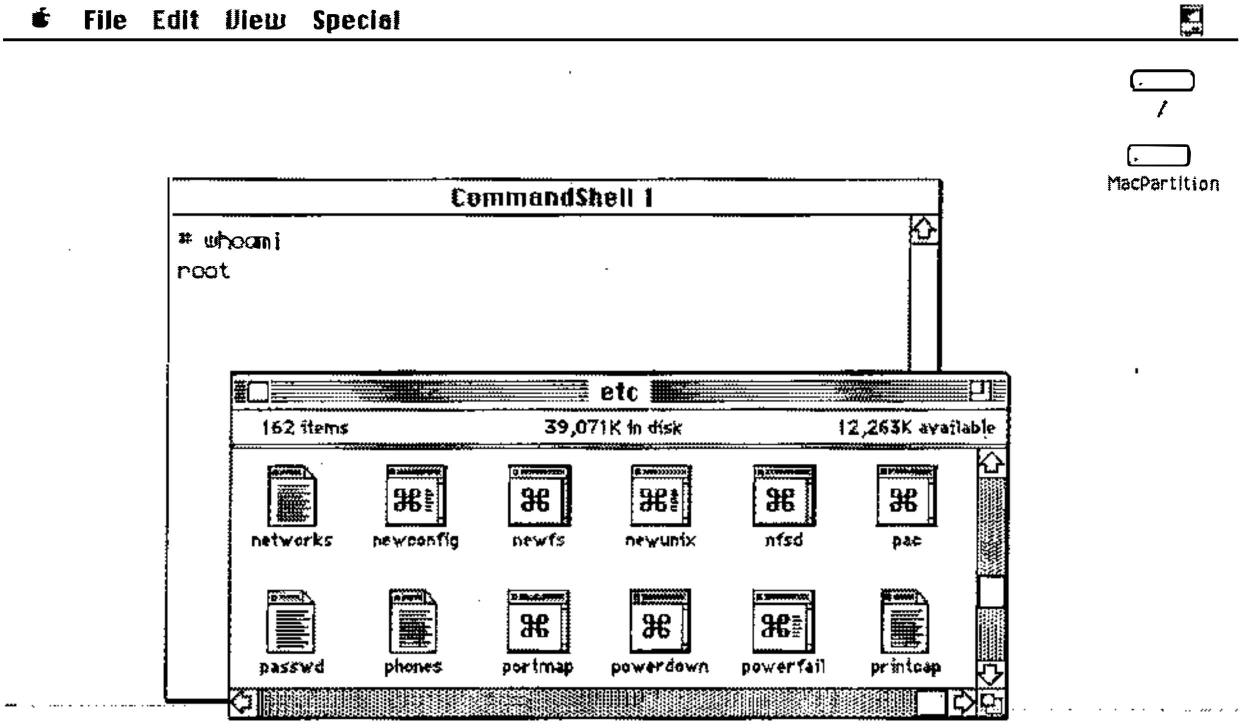
## Notes

# Administrator Privileges

In the course of their normal duties, system administrators need to have access to all of the files on a system. It is one of the strengths of A/UX that if you don't have read permission on a file, you can't make a copy of it. While this is an advantage in a normal working environment, it can get in the way of making backups of entire disks, which system administrators must periodically do. Therefore, the system administration login root has read, write, and execute permission on all files.

This implies that the system administrator can run any command on the system. This is not something that we want regular users to be able to do. A/UX has several commands that can destroy data on disks and, by the nature of UNIX, do so silently. This means that the potential for accidents is high unless access to these powerful commands is restricted.

**References** *Local System Administration*



```
# whoami
root
```

## Superuser Privileges

# Notes

# Standard User Privileges

Again, due to the nature of UNIX, users must sometimes be permitted to access the powerful features of the operating system. Rather than bothering the system administrator for a temporary permission change, A/UX allows certain programs to appear as though the superuser were running the program. When the program exits, the user's permissions have not changed. The two types of programs that do this are called setuid (set user id) and setgid (set group id). (We will discuss group and user ids later in this course.)

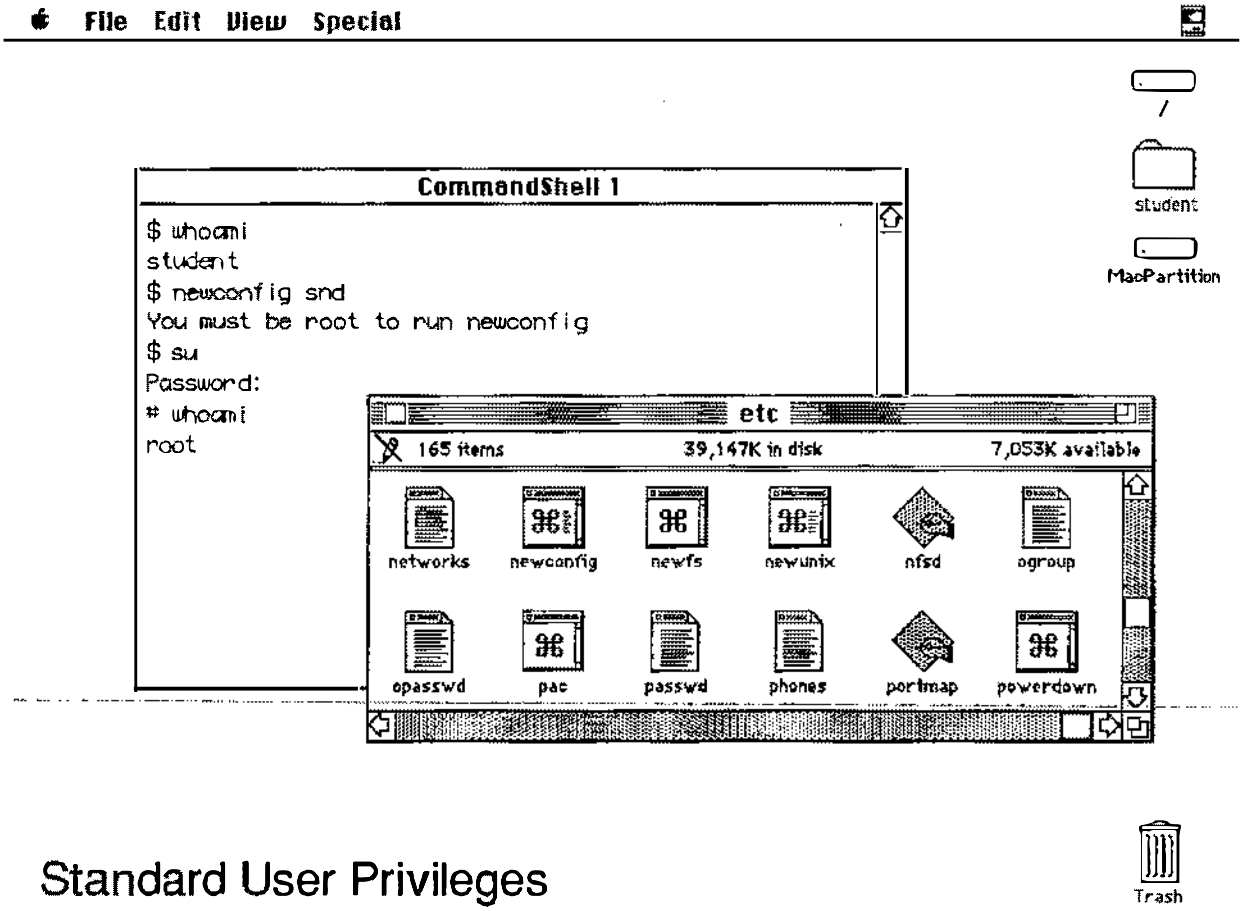
For example, the password file, containing the lists of users and their encrypted passwords, is normally readable by everyone. This is because many programs, such as `ls`, use information found in the password file. While the superuser can change anyone's password at any time, we also want individual users to be able to change their own passwords. To accomplish this, the `passwd` program is a setuid program. When the user invokes it, it asks for and confirms the new password information. Then, for a very brief period of time, it changes the user's effective permissions, opens the password file and substitutes in the new information for that user, then changes the effective permissions back.

As an additional level of security, some programs check who is invoking them, and will only let the superuser run them.

Users can use the program `su` to change their effective user ids. This is especially handy if you are logged on under your regular login name, but neeed to do something as superuser. The the `su` command, when invoked without arguments, assumes you want to become root.

**Note** The Finder® does not recognize that you may have changed your effective user id by using the `su` command.

**References** *Local System Administration*



```
$ whoami
student
$ newconfig snd
You must be root to run newconfig
$ su
Password:
# whoami
root
```

## Standard User Privileges

# Notes

# Information Resources

The A/UX system administrator will occasionally have need to reference information. This can be found in several places in the A/UX documentation suite:

- On-line man pages
- A/UX Command Reference
- A/UX Local System Administration
- Setting Up Accounts and Peripherals
- A/UX Local System Administrator's Reference
- A/UX Reference and Index

Additional resources are listed in the Appendices.

# Information Resources

- ❑ On-line man pages
- ❑ Local System Administration
- ❑ Setting Up Accounts and Peripherals
- ❑ System Administrator's Reference
- ❑ A/UX Reference and Index

7

## Notes

# Checkpoints

1 What are the duties of a system administrator?

2 What is the main difference between root and user privileges?

3 What manuals contain system administration information?

# Module 2
# Installing and Configuring A/UX

## Alternate Learning Path

# Table of Contents

# Installing and Configuring A/UX

## Notes

# Installing and Configuring A/UX

**Goal**

The goal for Module 2 is to give you an understanding of setting up and modifying the startup configuration of an A/UX system. You will learn about hardware configurations for A/UX and look at the internals of the startup procedure.

**Objectives**

- describe the hardware requirements for A/UX
- describe the process of A/UX installation
- explain the initial A/UX startup process and options
- describe the purpose and function of A/UX Startup
- briefly describe the boot procedures
- look at the A/UX startup files and examine what they do

**Activities**

Lecture/Discussion/Demonstration/Checkpoints/Exercises

**References**

*A/UX Local System Administration*
*A/UX Installation Guide*

# Objectives

- ❑ List the minimum hardware configuration
- ❑ Describe Macintosh II hardware installation
- ❑ Describe the process of A/UX installation
- ❑ Explain the A/UX Startup options
- ❑ Briefly describe the boot procedures
- ❑ Explain the A/UX login options
- ❑ Describe the concept of run states
- ❑ Explain the purpose of the file /etc/inittab

2

## Notes

# A/UX Hardware Requirements

**CPU**

A/UX requires a Macintosh CPU equipped with a Paged Memory Management Unit (PMMU). The systems currently meeting this requirement are:

Macintosh SE/30
Macintosh II with PMMU
Macintosh IIx
Macintosh IIcx
Macintosh IIci
Macintosh IIfx

**Memory**

A/UX requires a minimum of 4 MB of internal memory. The systems above are internally expandable with 1 MB SIMMs to 8 MB. Using 4 MB SIMMs or a third-party RAM card installed in a NuBus slot provides greater memory expansion opportunities.

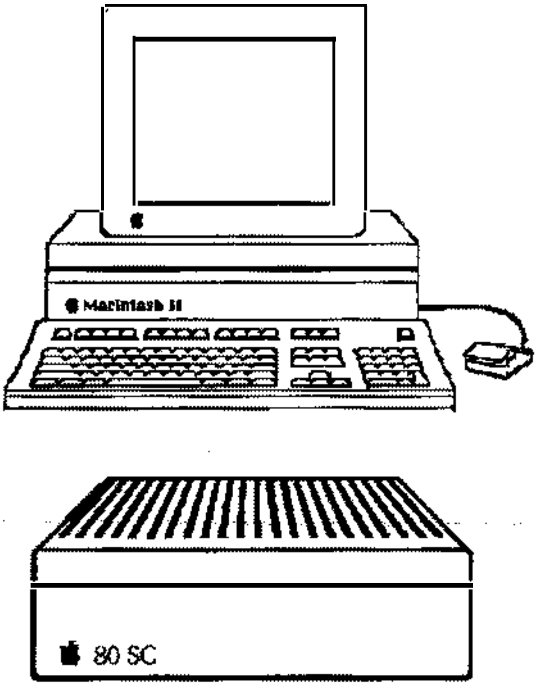
**Hard Disk**

The full A/UX product offering fits on an 80 MB disk drive, leaving approximately 12 MB of free space. Additional drives, of any size, can be connected to the system at any point. Larger drives are available from Apple and third parties.

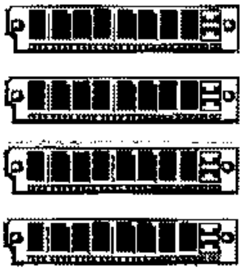
**References**

*A/UX Installation Guide*
*A/UX System Overview*

# Required Hardware



- ❑ Macintosh w/ PMMU
- ❑ 80MB Hard Disk
- ❑ 4 MB RAM



3

## Notes

# Optional Hardware

### Printers

A/UX is shipped configured ready to talk AppleTalk to printers on the printer port, and serially to ImageWriters on the modem port. If desired, reconfiguring either of these ports is simple.

### EtherTalk

EtherTalk cards allow A/UX systems to communicate and share resources over Ethernet® networks. We will be using these during this course. The EtherTalk card is made for Apple by 3Com and uses National Semiconductor's LAN (local area network) chip set. When EtherTalk boards are added to a system, a new kernel is automatically built to include them.

### CD-ROM

A/UX already has kernel driver support for the Apple CD-SC ROM drive. Three file systems are supported: HFS (Macintosh), UFS (Berkeley UNIX®) and System V (AT&T UNIX.)

### Video cards/ Monitors

A variety of Apple and third-party video cards are supported. No special drivers are needed, as long as the vendor has complied with the NuBus standard.

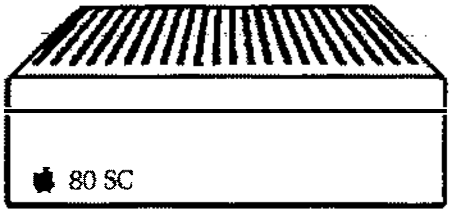
### 40SC Tape Drive

Tape drive support is pre-configured into the A/UX kernel. Four backup programs are shipped with A/UX, covering a wide spectrum of user needs. Additionally, the Macintosh application Tape Backup 40SC version 2.01 supports the backup of A/UX partitions.

### References

*A/UX Installation Guide,*
*A/UX System Overview*

# Optional Hardware



- ❑ Printers
- ❑ EtherTalk card
- ❑ Hard Disks
- ❑ CD-ROM
- ❑ Video Cards/Monitors
- ❑ 40SC Tape Drive

4

## Notes

# A/UX Installation Options

A/UX is shipped on several kinds of media, to suit customer's needs.

**Hard Disk**

The easiest option to install, A/UX is delivered already installed on a hard disk, either external or internal within a system bundle. Only a hardware connection is needed before A/UX is ready to use.

**CD-ROM**

With this option, A/UX is delivered on a CD-ROM disk. The installation is relatively easy because once it is started you can leave it unattended for the 50 minutes it takes to transfer the files. This option is also convenient to use for maintenance, as the disk can be added to the system to appear as a read-only directory.

**Tape**

A/UX is delivered on tape cartridge with this option. Like the CD option, installation can be unattended once it is started, though it takes longer to install. This option is less convenient to use for maintenance, as files must be taken off the tape using a tape transfer program.

**Floppy disk**

This option is the least convenient to install or maintain. A/UX installation cannot be left unattended, and the number of floppy disks is fairly large.

**References**

*A/UX Installation Guide*

# A/UX Installation Options

- ❑ Hard disk
- ❑ CD-ROM
- ❑ Tape
- ❑ Floppy disk

5

## Notes

# A/UX Installation Process

For all the installation options, except hard disk, A/UX must be transferred form the selected media to a hard disk.

**Initialize Disk**

The disk is initialized with HD SC Setup 2.01 (or later.) This software is delivered with A/UX.

**Partition Disk**

The disk is partitioned with HD SC Setup 2.01 (or later.) Partitioning sets up separate sections of the disk for the A/UX and Macintosh operating systems (both must be present to boot a disk.) Partitioning is covered in more detail in a later module.

**Install Macintosh Programs**

A Macintosh System Folder is installed on the appropriate partition using floppy disks. Additionally, two disks of programs necessary to launch A/UX are also installed on the Macintosh partition.

**Boot A/UX Kernel**

A set of three floppy disks are used to boot an A/UX kernel into the system in preparation for loading the A/UX files.

**Install A/UX Files**

The A/UX files are transferred from the delivered media to the hard disk. Special options can be used during installation to create non-standard file layouts, or if you are updating a previous version of A/UX.

**References**

*A/UX Installation Guide*
*Setting Up accounts and Peripherals for A/UX*

# A/UX Installation Process

- ❑ Initialize disk
- ❑ Partition disk
- ❑ Install Macintosh programs
- ❑ Boot A/UX kernel
- ❑ Install A/UX from media

6

## Notes

# A/UX Startup Application

**Bootstrapper**

As you know, A/UX system startup is very quick and easy. The system is configured at the factory to boot into a partition of the disk holding the Macintosh Operating System. Contained in this partition is the A/UX Startup utility. Double-clicking on its icon will launch A/UX. A/UX Startup can also be made the startup application, so that when the system is turned on, A/UX will boot up without intervention.

**StartupShell**

However, A/UX Startup is much more powerful than simply a bootstrapper for the A/UX kernel. Startup actually allows us to interact with the A/UX file system without booting A/UX. This tremendously useful feature means that we can troubleshoot and repair most of the problems we will encounter, without having to restore the entire system from backup, as some other UNIX systems require.

The commands available from Startup are:

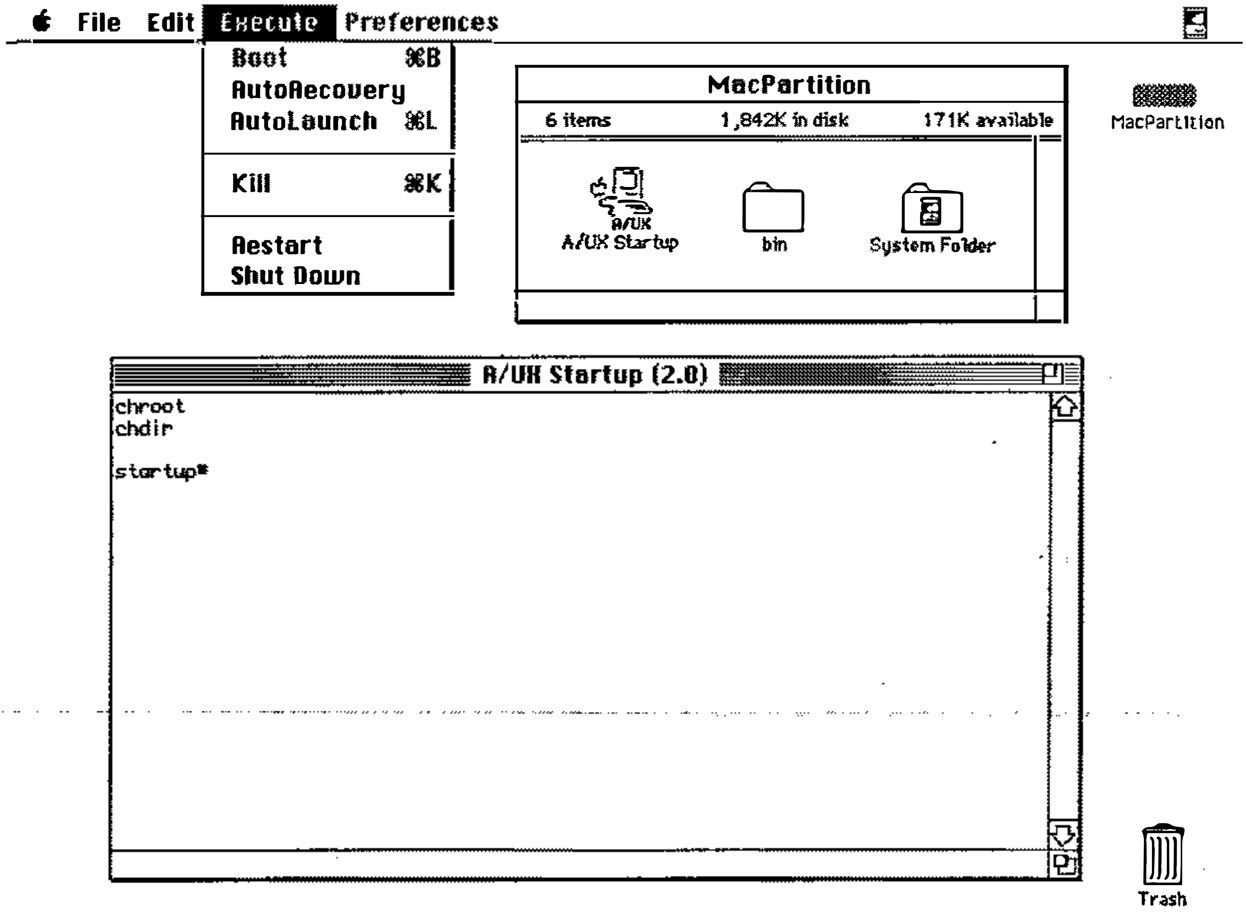
```
cat          chgrp       chmod      chown
cp           cpio        date       dd
dp           ed          esch       fsck
fsdb         kconfig     launch     ln
ls           mkdir       mkfs       mknod
mv           newfs       od         pname
read_disk    rm          stty       tar
```

**Autorecovery**

A/UX's autorecovery feature keeps copies of key system files on a separate disk partition, in the unlikely case the root partition becomes corrupted. Should a problem such as an unreadable kernel file occur, a current version of the kernel can be copied from the reserved partition, and the system will be functional once again. Autorecovery is invoked from Startup by the `esch` command.

**References**

*A/UX Local System Administration*
*StartupShell(8)*



```
chroot
chdir

startup*
```

# Notes

# Preferences - General

A/UX Startup has several options available in the General menu selection that can assist us in maintaining, repairing, and troubleshooting the system.

**Root directory**

This text window displays the value of the built-in root variable. It is initially set to boot A/UX from the same hard disk as A/UX Startup. To boot A/UX from a different disk, replace the current value with `(n, 0, 0)`, where *n* is the SCSI ID of the disk from which you wish to boot.

**Home directory**

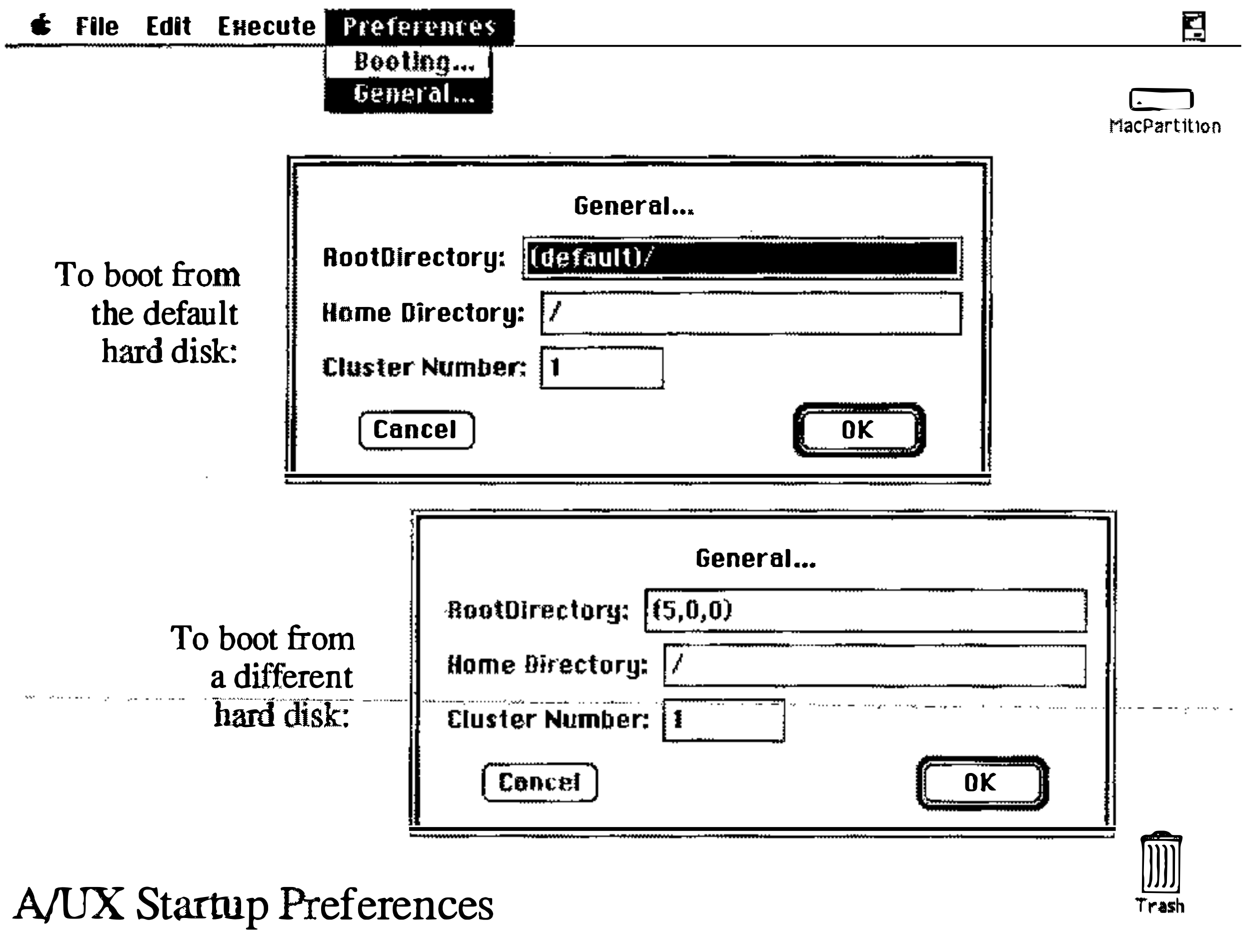
This text window displays the value of the built-in root variable. To change the value, select this box and edit the text.

**Cluster Number**

This points to the number of the Eschatology partition, where backup copies of important system files are kept.

**References**

*A/UX Local System Administration*
*StartupShell (8)*
*Autorecovery (8)*



To boot from the default hard disk:

General...

RootDirectory: (default)/

Home Directory: /

Cluster Number: 1

Cancel OK

To boot from a different hard disk:

General...

RootDirectory: (5,0,0)

Home Directory: /

Cluster Number: 1

Cancel OK

## A/UX Startup Preferences

## Notes

# Preferences - Booting

A/UX Startup has several options available in the Booting menu that can assist in maintaining, repairing, and troubleshooting the system.

**Eject disks on Launch**

When selected, all floppy disks are ejected when the kernel is launched.

**Automatic Boot at startup**

When selected, the boot command is run when Startup is launched, causing A/UX to boot as part of launching Startup.

**Check root file system**

This helps maintain the system by executing a command to automatically check the files on the disk for consistency, and to verify the amount of unused storage on the disk. If either of these are askew, it will repair the problem before booting A/UX.

**Full autorecovery**

This attempts to repair a severely mangled root directory by copying files from a reserved section of the disk.

**Custom command**

This allows an administrator to modify the startup options to suit their own needs.

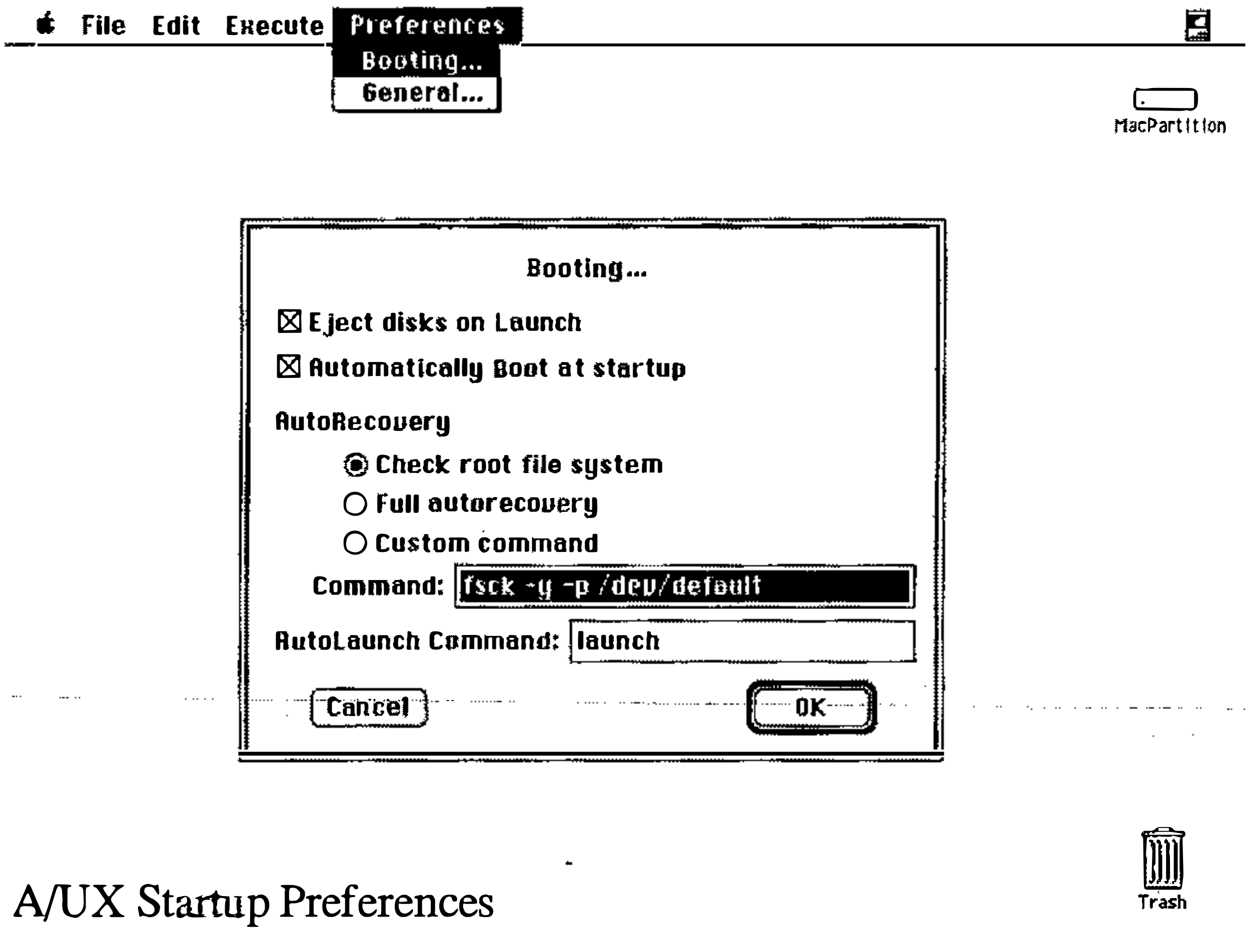
**AutoLaunch command**

This option can be changed to allow an administrator to see the output messages as the system is booting up. Normally these are not of interest, but if a system is failing to boot, these messages can give an indication of where the problem might be. To see these messages change the command to read:

```
launch -v
```

**References**

*A/UX Local System Administration*



Booting...

☒ Eject disks on Launch

☒ Automatically Boot at startup

AutoRecovery

- ● Check root file system
- ○ Full autorecovery
- ○ Custom command

Command: fsck -y -p /dev/default

AutoLaunch Command: launch

A/UX Startup Preferences

## Notes

# Boot Process

As A/UX is booting, it goes through several phases before it is ready for users to log on.

**Phase 1**

Checking

A/UX Startup executes a check to see if any files in the root directory have been damaged. If so, it attempts to repair them.

**Phase 2**

Loading

A/UX Startup loads the kernel.

**Phase 3**

Launching

Control is transferred to the A/UX kernel. At this point, an administrator can view the systems messages output during the startup process by opening the CommandShell console window.

**Phase 4**

Checking file systems at <mount point>

A check performed to see if any files in directories other than the root directory have been damaged. If so, it attempts to repair them.

**Phase 5**

Initializing device drivers

The program `autoconfig` is called to verify the hardware configuration matches the software configuration. If not, a new kernel is made and the system is rebooted.

**Phase 6**

Starting background processes

Various directories have temporary files removed. The program `init` is invoked and through it the system is made ready to accept multi-user logins.

**References**

*A/UX Local System Administration*

# Boot process

- ❑ Phase 1 — Checking
- ❑ Phase 2 — Loading
- ❑ Phase 3 — Launching
- ❑ Phase 4 — Checking file systems at <mountpoint>
- ❑ Phase 5 — Initializing device drivers
- ❑ Phase 6 — Starting background processes

10

## Notes

# Run States

A/UX usually allows multiple users to log into the system via EtherTalk, serial lines, or a combination of these (in addition to the user using the desktop on the system console.)

There are rare times, however, when it is preferable to only allow one user to access the system. An example of this is when making backups of data. If there are several users on the system, it is difficult to guarantee that the data being backed up is valid from one minute to the next. UNIX allows the system to restrict access to the system console, in this case the system is said to be in **single user mode**. Likewise, when multiple people can log into the system it is in **multi-user mode.**

A/UX Release 2.0 was designed so that the system almost never needs to be put into single user mode. If, for some reason, you feel this is necessary, use the following instructions:

**To put the system into single user mode:**

**Step 1** Open a CommandShell window.

**Step 2** Enter the command:

```
shutdown
```

**Note** This command is very different from the menu selection Shut Down, which actually powers down the system.

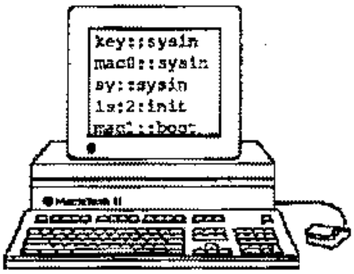
**To put the system back into multi-user mode:**

**Step 1** Enter the command:

```
init 2
```

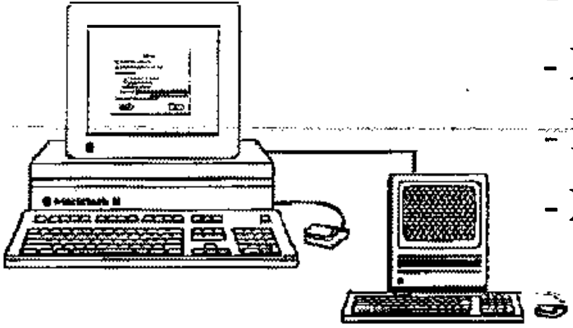
**References** *A/UX Local System Administration*

# Run States



□ Single User
  - Command-line environment
  - Console is terminal emulator
  - Local file systems available

□ Multi-user
  - Others can log in
  - Daemons running
  - Network file systems available
  - Macintosh environment



11

## Notes

# Initialization Table

When A/UX boots, a number of startup processes are executed automatically that initialize the system so you can use it. The initial run state, for example, is determined by information contained in the file /etc/inittab. In Release 2.0 the initial run state is 2, multi-user. This file also controls the background programs, called **daemons**, which allow different activities to take place. Daemons control the following functions:

- opening up a port for another terminal
- allowing network access to take place
- allowing other users to access the system
- print spooling
- electronic mailing

You can learn a lot about operating states simply by reading the inittab file.

### File Structure

The file has the form: *title:state:action:program*, where *title* is just a unique line identifier; *state* is the run state for which this line will be active; and *action* is one of several ways the following *program* is to be executed.

### `init` command

The `init` command is a general process spawner, with the primary role of creating processes as determined by the file /etc/inittab, which is the text file that contains all the relevant information about run levels. The command `who -r` lets you know what `init` state is active.`getty` is the program that sets the serial port speed and invokes the login prompt on terminals (other than the console) connected to the system.

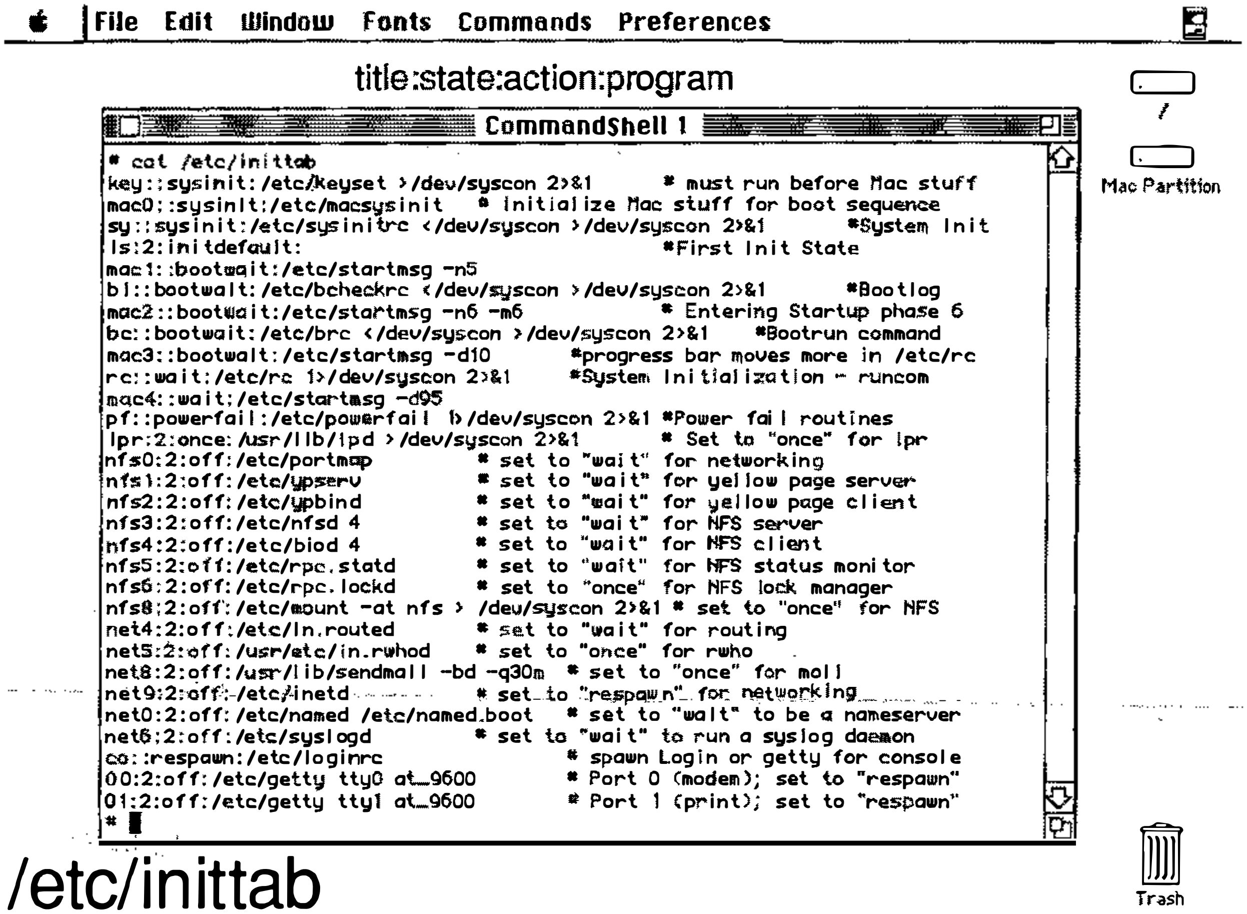
### Modifying Startup Files

/etc/inittab and other startup files can be modified to change various operating parameters. However, improper changes in these files can be catastrophic. Always make a copy of /etc/inittab (for example, to inittab.old) before changing. (Note that you should always copy crucial system files such as inittab before editing.)

### References

*A/UX Command Reference*
*init (1M,) inittab (4,) who (1,) getty (1M)*

title:state:action:program



```
# cat /etc/inittab
key::sysinit:/etc/keyset >/dev/syscon 2>&1        # must run before Mac stuff
mac0::sysinit:/etc/macsysinit   # initialize Mac stuff for boot sequence
sy::sysinit:/etc/sysinitrc </dev/syscon >/dev/syscon 2>&1       #System Init
ls:2:initdefault:                                 #First Init State
mac1::bootwait:/etc/startmsg -n5
b1::bootwait:/etc/bcheckrc </dev/syscon >/dev/syscon 2>&1       #Bootlog
mac2::bootwait:/etc/startmsg -n6 -m6              # Entering Startup phase 6
bc::bootwait:/etc/brc </dev/syscon >/dev/syscon 2>&1    #Bootrun command
mac3::bootwait:/etc/startmsg -d10       #progress bar moves more in /etc/rc
rc::wait:/etc/rc 1>/dev/syscon 2>&1     #System Initialization - runcom
mac4::wait:/etc/startmsg -d95
pf::powerfail:/etc/powerfail 1>/dev/syscon 2>&1 #Power fail routines
lpr:2:once:/usr/lib/lpd >/dev/syscon 2>&1         # Set to "once" for lpr
nfs0:2:off:/etc/portmap         # set to "wait" for networking
nfs1:2:off:/etc/ypserv          # set to "wait" for yellow page server
nfs2:2:off:/etc/ypbind          # set to "wait" for yellow page client
nfs3:2:off:/etc/nfsd 4          # set to "wait" for NFS server
nfs4:2:off:/etc/biod 4          # set to "wait" for NFS client
nfs5:2:off:/etc/rpc.statd       # set to "wait" for NFS status monitor
nfs6:2:off:/etc/rpc.lockd       # set to "once" for NFS lock manager
nfs8:2:off:/etc/mount -at nfs > /dev/syscon 2>&1 # set to "once" for NFS
net4:2:off:/etc/in.routed       # set to "wait" for routing
net5:2:off:/usr/etc/in.rwhod    # set to "once" for rwho
net8:2:off:/usr/lib/sendmail -bd -q30m  # set to "once" for mail
net9:2:off:/etc/inetd           # set to "respawn" for networking
net0:2:off:/etc/named /etc/named.boot   # set to "wait" to be a nameserver
net6:2:off:/etc/syslogd         # set to "wait" to run a syslog daemon
co::respawn:/etc/loginrc                # spawn Login or getty for console
00:2:off:/etc/getty tty0 at_9600        # Port 0 (modem); set to "respawn"
01:2:off:/etc/getty tty1 at_9600        # Port 1 (print); set to "respawn"
# █
```

/etc/inittab

## Notes

# A/UX Startup Files

Many of the programs listed in inittab in turn call other programs. The following is a general description of the boot sequence; at the end of this module are samples of the files involved in the boot process.

- After power on, `/etc/init` reads /etc/inittab to determine how to proceed (which programs are to be run, when , how, and in what order.)
- `init` calls `/etc/macsysinitrc` to start the Desktop and CommandShell environments.
- `init` then calls `/etc/sysinitrc` before the initial run state is established. `autoconfig` is run to make sure the hardware matches the kemel. If not, a new kemel is made and the system is rebooted.
- Control then goes back to `init`, which executes `/etc/bcheckrc`, which runs `fsck` to check file systems other than root.
- `init` also calls `/etc/brc`, which confirms partition names and changes ownership of devices to root.
- `init` calls `/etc/rc`, which mounts file systems (if any are present other than root), and sets up and initiates accounting for multi-user state (as specified by level 2 in inittab.)
- Finally, `init` calls `/etc/loginrc`, which creates the Login dialog box.

Of all these files, inittab is the file accessed and modified most frequently by administrators and can be viewed as a master file for determining the system's operating states.

Unlike previous releases, administrators rarely have to edit this file "by hand." Almost all modifications are done by other programs like `newconfig` and `setport`.

**References** *init (1M,) inittab (4)*

# A/UX Startup Files

**/etc/inittab**
initialization table

- ❑ **/etc/macsysinitrc**
  start Macintosh environment
- ❑ **/etc/sysinitrc**
  system initialization commands, verify hardware matches kernel
- ❑ **/etc/bcheckrc**
  check file system, date, other parameters before mounting file systems
- ❑ **/etc/brc**
  confirm partition names, change ownership of devices to root
- ❑ **/etc/rc**
  mount file systems other than root, start system accounting
- ❑ **/etc/loginrc**
  create Login dialog box

13

## Notes

# Checkpoints

1 What are the minimum hardware requirements for A/UX?

2 What is the primary system initialization file? What does the name of this file stand for?

3 How do you go from multi-user to single user mode?

4 How do you go from single user to multi-user mode?

# Exercises

**Exercise 1** In this exercise you will go through the verbose form of the initial A/UX startup procedure.

**Step 1** Power up your A/UX system. Double-click on the A/UX startup icon.

**Step 2** Set A/UX Startup to be the startup application.

**Step 3** Cancel the Startup utility by pressing ⌘. (COMMAND-PERIOD) while the •A/UX Release 2.0• copyright notification is showing.

**Step 4** Choose Booting from the preferences menu. Once the dialog box opens, change the AutoLaunch command to:

```
launch -v
```

Close the dialog box.

**Step 5** Enter:

```
boot
```

at the startup prompt.

**Step 6** When the Login dialog box appears, log on as root. The password is just a RETURN.

**Step 7** Set passwords for the following users:

root
start
guest

These are login names that have easily guessed passwords when A/UX is delivered, which could possibly cause security problems. Adding passwords is one of the first steps to take after installing A/UX

**Step 8** Shut the system down.

**Exercise 2** In this exercise you will you will delete the root password to gain access to the system. You would need to do this to rescue a user who has forgotten their root password. The subject of passwords is covered more in depth in the next module; for now we will emphasize the ability to make changes in the A/UX file system without running A/UX.

**Step 1** Power up the A/UX system. Double-click on the A/UX startup icon.

**Step 2** Cancel the Startup utility by pressing ⌘. (COMMAND-PERIOD) while the A/UX Release 2.0 copyright notification is showing.

**Step 3** Edit the password file. (In the following script, the computer's responses are in `Courier` font. Your commands are in **`bold Courier`**. Editorial comments (do not type these!) are still in Times font.)

**Note** The thirteen-character encrypted password found in your /etc/passwd file will be different than the one shown in the example below. Use the password found in the file on your system instead of the one shown.

```
startup# ed /etc/passwd                  (start the ed editor)
530
1                                        (select line one)
root:jw9N7xoYnpWve:0:0::/:/bin/sh
s|:jw9N7xoYnpWve:|::|                    (swap the old password)
p                                        (print the line to check it)
root::0:0::/:/bin/sh
w                                        (write the file to disk)
517
q                                        (quit the program)
```

**Step 4** Enter:

**`boot`**

at the startup prompt.

**Step 5** Log on as root when the dialog box reappears. The password is just a RETURN.

**Step 6** Change the root password.

**Step 7** After you have successfully changed the password, shut down the system.

# A/UX System Initialization Files

## /etc/inittab

```
key::sysinit:/etc/keyset >/dev/syscon 2>&1      # must run before Mac stuff
mac0::sysinit:/etc/macsysinit   # initialize Mac stuff for boot sequence
sy::sysinit:/etc/sysinitrc </dev/syscon >/dev/syscon 2>&1      #System Init
is:2:initdefault:                               #First Init State
mac1::bootwait:/etc/startmsg -n5
bl::bootwait:/etc/bcheckrc </dev/syscon >/dev/syscon 2>&1      #Bootlog
mac2::bootwait:/etc/startmsg -n6 -m6            # Entering Startup phase 6
bc::bootwait:/etc/brc </dev/syscon >/dev/syscon 2>&1    #Bootrun command
mac3::bootwait:/etc/startmsg -d10       #progress bar moves more in /etc/rc
rc::wait:/etc/rc 1>/dev/syscon 2>&1     #System initialization - runcom
mac4::wait:/etc/startmsg -d95
sl::wait:(rm -f /dev/syscon;ln /dev/systty /dev/syscon;) 1>/dev/systty 2>&1
mac5::wait:/etc/startmsg -d100
pf::powerfail:/etc/powerfail 1>/dev/syscon 2>&1 #Power fail routines
mac6::wait:/etc/startmsg -q             # Exit Startup
er:2:wait:/usr/lib/errdemon
cr:2:wait:/etc/cron </dev/syscon >/dev/syscon 2>&1
lp:2:off:/usr/lib/lpsched >/dev/syscon 2>&1     # Set to "wait" for lp
lpr:2:once:/usr/lib/lpd >/dev/syscon 2>&1       # Set to "once" for lpr
nfs0:2:off:/etc/portmap         # set to "wait" for networking
nfs1:2:off:/etc/ypserv          # set to "wait" for yellow page server
nfs2:2:off:/etc/ypbind          # set to "wait" for yellow page client
nfs3:2:off:/etc/nfsd 4          # set to "wait" for NFS server
nfs4:2:off:/etc/biod 4          # set to "wait" for NFS client
nfs5:2:off:/etc/rpc.statd       # set to "wait" for NFS status monitor
nfs6:2:off:/etc/rpc.lockd       # set to "once" for NFS lock manager
nfs8:2:off:/etc/mount -at nfs > /dev/syscon 2>&1 # set to "once" for NFS
net4:2:off:/etc/in.routed       # set to "wait" for routing
net5:2:off:/usr/etc/in.rwhod    # set to "once" for rwho
net8:2:off:/usr/lib/sendmail -bd -q30m  # set to "once" for mail
net9:2:off:/etc/inetd           # set to "respawn" for networking
net0:2:off:/etc/named /etc/named.boot   # set to "wait" to be a nameserver
net6:2:off:/etc/syslogd         # set to "wait" to run a syslog daemon
co::respawn:/etc/loginrc                # spawn Login or getty for console
00:2:off:/etc/getty tty0 at_9600        # Port 0 (modem); set to "respawn"
01:2:off:/etc/getty tty1 at_9600        # Port 1 (print); set to "respawn"
```

# /etc/macsysinitrc

```
#!/bin/sh
:
: macsysinitrc
:
#       @(#)Copyright Apple Computer 1989      Version 1.8 of
macsysinitrc.sh o
n 90/01/09 16:12:03
#
# Commands to fire up the Macintosh environment to monitor progress during
# system bootup.  This is invoked by /etc/macsysinit, only if the -v
(verbose)
# flag was NOT given to sash.  startmac and CommandShell must continue to
# live after this process exits, so nohup is used.

PARENT_PID=$1   # used to send synchronization signal back to
/etc/macsysinit

nohup /mac/bin/startmac -s '/mac/sys/Startup System Folder' -f StartMonitor
&
/etc/startmsg -p6 -n3 -m2
```

# /etc/sysinitrc

```
#!/bin/sh
:
: sysinitrc
:
#       @(#)Copyright Apple Computer 1987       Version 1.50 of sysinitrc.sh
on
90/03/30 16:00:28
# System Initialization Commands - executed once only, before INIT
# starts up its' initial level (i.e., this precedes the `initdefault'
level).

/etc/line_sane
# /etc/keyset           # this is now done by the first entry in
/etc/inittab

/bin/stty susp '^Z' kill '^u' erase '^?' intr '^c' echoe ixon
echo "/etc/sysinitrc: system initialization"

umask 022
PATH=/bin:/usr/bin:/usr/ucb:/mac/bin:/etc:/usr/etc:. ; export PATH
HOME=/; export HOME
LOGDIR=/; export LOGDIR
TERM=mac2; export TERM

/bin/cat /etc/RELEASE_ID                # system release ID

echo "
Today is: `/bin/date`
"

trap "" 2

set `/etc/devnm /`''
fs=$1
trap "echo Interrupt" 2

# The root file system should be checked from the A/UX Startup application
# in Macintosh OS, to ensure that the file system is not being accessed.
# If the Macintosh environment is not running, it is OK to run fsck here,
# since the A/UX toolbox's file manager will not have written its cache
files.

/etc/fsstat $fs
if [ $? -ne 0 ] ; then
    /etc/macquery -c /etc/fsck 131 $fs
    if [ $? -ne 2 ] ; then      # Macintosh environment is running
        /bin/sync
        /bin/sync
        /etc/reboot
    else                        # Macintosh environment is not running
        /etc/fsck $fs
    fi
fi

trap "" 2
```

```
/etc/startmsg -n4 -m5

> /etc/mtab                                # create file
/etc/mount -f `/etc/devnm /`               # "mount" root

# Copy the IOP code resources into the kernel.  If /dev/iop fails to
#  open (i.e. no IOPs on this system), we accept that situation silently.

/bin/cp /etc/iop/iopc_0  /dev/iop0 >/dev/null 2>&1
/bin/cp /etc/iop/SERD_60 /dev/iop0 >/dev/null 2>&1
/bin/cp /etc/iop/SERD_61 /dev/iop0 >/dev/null 2>&1

/etc/startmsg -d10

if [ -f /etc/HOSTNAME ]; then             # set domain and hostname
  read host domain < /etc/HOSTNAME
else                                       # use defaults if none specified
  host="localhost"
  domain="localdomain"
fi
/bin/domainname $domain
/bin/hostname $host
/etc/chgnod $host

/etc/startmsg -d20
/etc/autoconfig -t0 -a -o /unix -S /etc/startup -M /etc/master # system
configur
ation
/etc/startmsg -d60
/etc/startup
/etc/startmsg -d90


# If we're going to be entering single-user mode, or if init will
# ask the user what mode to enter, then quit the Macintosh environment,
# sleep to allow CommandShell to relinquish the console, and display
# the console emulator bitmap.

set `/bin/grep initdefault /etc/inittab | /bin/sed 's/:/ /g'` null null
case $2 in
    s|S|null)   /etc/startmsg -q; /bin/sleep 10; /bin/screenrestore ;;
esac


# Add local code below this line...
# ==========================================================================
```

# /etc/sethost

```
:
: /etc/sethost
:
#
#       @(#)Copyright Apple Computer 1987       Version 1.7 of sethost on 
89/12/
11 17:48:13
#
# set hostname and domain name
# called from /etc/sysinitrc

if [ -f /etc/HOSTNAME ]; then
  read host domain < /etc/HOSTNAME
fi

warning='should consist of digits, dashes (-), and lower case letters only.'
hostwarn='It should begin with a letter, and end with a letter or a digit.'
domwarn='Fully qualified domain names may contain dot (.).  Each section of 
the domain name should begin with a letter, and end with a letter or a 
digit.'

T='[a-z][a-z0-9\-]*[a-z0-9]'

while [ "$host" = "" ]; do
  echo "Please enter a host name (it must be unique): \c"
  read host
  while [ `expr "$host" : "$T"` = 0 ]; do
    echo "The host name $warning"
    echo $hostwarn
    echo
    echo "Please enter a host name [$host]: \c"
    read nuhost
    if [ -z "$nuhost" -o "$nuhost" = "$host" ]; then
      break
    else
      host=$nuhost
    fi
  done
  nuhost="yes"
done
while [ "$domain" = "" ]; do
  echo "Please enter a Yellow Pages domain name [none]: \c"
  read domain
  case "$domain" in
  ""|'none') domain='none';;
  *) while [ "`echo $domain | sed '
             s@%@#@g
             s@'$T'@%@g
             s@^[a-z]$@%@
             s@^[a-z]\.@%.@
             s@\.[a-z]$@.%@
             s@\.[a-z]\.@.%.@
             : loop
               s@%\.%@%@
             t loop
```

```
                            '`" != "%" ]; do
      echo "The domain name $warning"
      echo $domwarn
      echo
      echo "Please enter a domain name [$domain]: \c"
      read nudomain
      if [ -z "$nudomain" -o "$nudomain" = "$domain" ]; then
        break
      else
        domain=$nudomain
      fi
    done
    ;;
  esac
  nuhost="yes"
done

if [ -n "$nuhost" ]; then
  echo "$host   $domain" > /etc/HOSTNAME
fi
```

# /etc/bcheckrc

```
#!/bin/sh
:
: bcheckrc
:
#       @(#)Copyright Apple Computer 1987       Version 1.20 of bcheckrc.sh
on 8
9/10/26 13:16:25 (Motorola 2.1.1.1)

# ***** This file has those commands necessary to check the file
# system, date, and anything else that should be done before mounting
# the file systems.

# Push line discipline/set tty sane
/etc/line_sane

trap "" 2

# ***** confirm the partitions (any that are not slice 0, 1, 2, or 31)
pname -a

# Automatically check the non-root file systems, if necessary, using a
# Macintosh interface. To automatically time out with a "Repair"
# answer from a "Do you want to repair?" alert, put the number of
# seconds (>= 1) immediately after the -m (no space).

/etc/fsck -m -p2
```

# /etc/brc

```
#!/bin/sh
:
: brc
:
#       @(#)Copyright Apple Computer 1987       Version 1.7 of brc.sh on
89/06/2
9 15:36:53 (Motorola 2.1)



# Push line discipline/set tty sane
/etc/line_sane

# This file may be used for multi-user boot-up commands.

/bin/chown root /dev/tty[pqrs]* /dev/pty[pqrs]*
/bin/chmod 666 /dev/tty[pqrs]* /dev/pty[pqrs]*
```

# /etc/rc

```
#!/bin/sh
:
: rc
:
#       @(#)Copyright Apple Computer 1987       Version 1.14 of rc.sh on
89/11/2
2 12:15:19 (ATT 1.12)

#       Push line discipline/set the device so it will print
/etc/line_sane 1

set `/bin/who -r`
if [ "$7" = 2 ]
then
        /etc/startmsg -d15

        # put mounts here (/usr, etc.)
        /etc/mount -at 5.2
        /etc/startmsg -d35
        /etc/mount -at 4.2
        /etc/startmsg -d60

        /usr/lib/ex3.9preserve -
        /etc/startmsg -d75

        /bin/rm -f /tmp/* >/dev/null 2>&1    # don't complain about
directories
        /bin/rm -f /usr/spool/uucp/LCK*
        /bin/rm -f /usr/spool/lp/SCHEDLOCK
        /bin/rm -f /usr/adm/acct/nite/lock*
        /etc/startmsg -d80

        if [ -f /usr/adm/sulog ]
        then
                /bin/mv /usr/adm/sulog /usr/adm/OLDsulog
        fi

        if [ -f /usr/lib/cron/log ]
        then
                /bin/mv /usr/lib/cron/log /usr/lib/cron/OLDlog
        fi

        if [ ! -f /etc/wtmp ]
        then
                >/etc/wtmp
                /bin/chmod 666 /etc/wtmp
                /bin/chgrp adm /etc/wtmp
                /bin/chown adm /etc/wtmp
        fi
        /etc/startmsg -d85

        /bin/chmod 666 /etc/utmp
#       /bin/su adm -c /usr/lib/acct/startup
#       echo process accounting started
fi
```

```
# Enable these lines for dial-out UUCP lines - change the tty port
# to match the one you select as the dial-out port.
#       /bin/chmod 666 /dev/tty0
#       /bin/chown daemon /dev/tty0
#       /bin/chgrp daemon /dev/tty0
```

# /etc/loginrc

```
#!/bin/sh
:
: loginrc
:
#       @(#)Copyright Apple Computer 1989      Version 1.7 of loginrc.sh on
90/
02/06 15:45:19

# ***** Verify that /mac/bin/Login exists and is executable.  If it is not,
# spawn /etc/getty for the console instead.  /mac/bin/Login itself also
# execs getty if it is unable to post its dialog.  In this way, the user
# should be able to get logged in even if some of the files needed for
# the Macintosh environment are missing or corrupted.

# Testing for execute permission always returns true for root, so su to
# sys (the group of /mac/bin/Login) before making this test.
#
/bin/su sys -c "/bin/test -x /mac/bin/Login"

if [ $? -eq 0 ]; then
    exec /mac/bin/Login -m2m -s '/mac/sys/Login System Folder' >/dev/console
2>&
1
else
    /bin/screenrestore
    if [ -f /mac/bin/Login ]; then
        echo "/mac/bin/Login is not executable;  launching getty instead." \
                        >/dev/console
    else
        echo "/mac/bin/Login is missing;  launching getty instead." \
                        >/dev/console
    fi
    exec /etc/getty console co_9600
```

# Module 3
# User Administration

# Table of Contents

# User Administration

A/UX® 2.0

## Notes

# Introduction to User Administration

One of the primary responsibilities of a system administrator is maintaining access to the system by designated users. This includes setting up accounts so users can log on to the system, as well as creating and modifying certain files to allow them to customize their working environment.

**Goal**

The goal of Module 3 is to acquaint you with adding users under A/UX and how to create an adjustable work environment.

**Objectives**

By the end of this module, participants will be able to:

- describe the purpose of user accounts and groups
- describe the format of the /etc/passwd and /etc/group files
- add a user to the system
- define the parameters set in a user's .profile file

**Activities**

Lecture/Discussion/Checkpoints/Exercises

**References**

*A/UX Local System Administration*
*Setting up Accounts and Peripherals*
*A/UX Essentials*

# Objectives

❑ Describe the purpose of user accounts and groups

❑ Describe the format of the /etc/passwd and /etc/group files

❑ Add a user to the system

❑ Define the parameters set in a user's .profile file

❑ Change environment variables

2

## Notes

# Why Have User Accounts?

User accounts provide a secure and customized work environment. In the world of the Macintosh Operating System, anyone who turns the computer on can work with all the files on the computer. In the multitasking, multi-user world of the UNIX® operating system there are typically many users on each system. Password and file access security was originally implemented to keep each user's files safe from unauthorized tampering.

In the A/UX world there may still be several people using the system, though perhaps not at the same time. The same need for secure files is present in this situation.

Of course, there may be only one person using the system, but in this case it is likely the system will be connected to a network. The need for secure files still exists.

A system that only allowed access to one's own files would be very restrictive, thus A/UX has different file permissions for each of three sets of users. In addition to the user who created the file, permissions can be set for "groups" and "others."

**Note** File permissions were covered in A/UX Fundamentals.

A group is a set of people with which you may want to share your files. One group may be able to update an address database. Another group may be able to edit the same text file.

The designation "others" is given to those people not in your group. This allows, for example, file writing by your group, but only reading by others.

Access to files is controlled through two files, /etc/passwd and /etc/group. /etc/passwd determines a user's uid (user identification) and a user's gid (group identification).

**References** *A/UX Local System Administration*
*Setting up Accounts and Peripherals*
*A/UX Essentials*

# Why have user accounts?

❏ File protection

❏ Workgroup access

3

## Notes

# /etc/passwd

The system keeps track of all users through the /etc/passwd file. Each user's information is kept on one line of the file. Each line consists of seven fields:

- login name
- encrypted password
- numerical user id
- numerical group id
- user data
- home directory
- shell

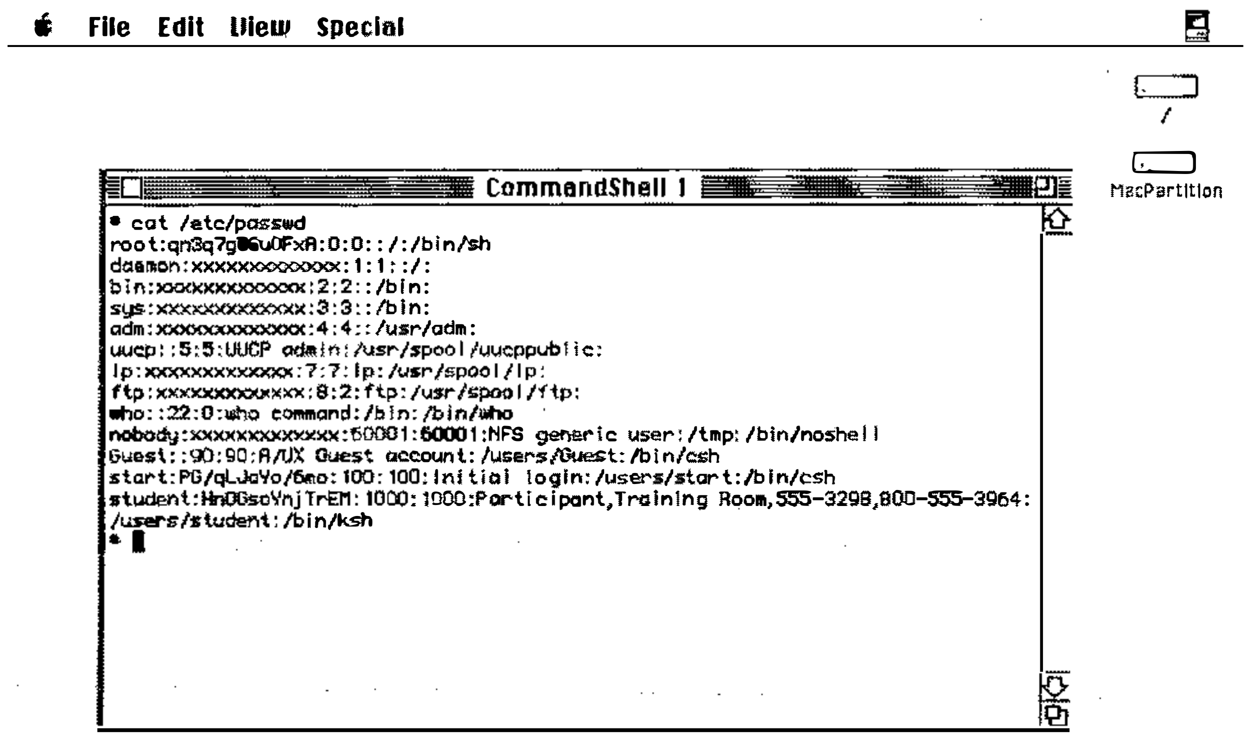
These seven fields are delimited by colons ( : ). The various pieces of user data are delimited by commas.

**Example**

```
# cat /etc/passwd
root:qn3q7gESu0FxA:0:0::/:/bin/sh
daemon:xxxxxxxxxxxxx:1:1::/:
bin:xxxxxxxxxxxxx:2:2::/bin:
sys:xxxxxxxxxxxxx:3:3::/bin:
adm:xxxxxxxxxxxxx:4:4::/usr/adm:
uucp::5:5:UUCP admin:/usr/spool/uucppublic:
lp:xxxxxxxxxxxxx:7:7:lp:/usr/spool/lp:
ftp:xxxxxxxxxxxxx:8:2:ftp:/usr/spool/ftp:
who::22:0:who command:/bin:/bin/who
nobody:xxxxxxxxxxxxx:60001:60001:NFS generic user:/tmp:/bin/noshell
Guest::90:90:A/UX Guest account:/users/Guest:/bin/csh
start:PG/qLJaYo/6mo:100:100:Initial login:/users/start:/bin/csh
student:HnOGsoYnjTrEM:1000:1000:Participant,Training Room,555-3298,
800-555-3964:/users/student:/bin/ksh
```

**Note** The shell is normally one of the UNIX shells (`ksh`, `csh`, or `sh`), though it can be an application program. If this is the case and a user exits the application, they will be logged out.

**References** *A/UX Local System Administration*
*passwd (4)*



```
• cat /etc/passwd
root:qn3q7g06u0FxA:0:0::/:/bin/sh
daemon:xxxxxxxxxxxxx:1:1::/:
bin:xxxxxxxxxxxxx:2:2::/bin:
sys:xxxxxxxxxxxxx:3:3::/bin:
adm:xxxxxxxxxxxxx:4:4::/usr/adm:
uucp::5:5:UUCP admin:/usr/spool/uucppublic:
lp:xxxxxxxxxxxxx:7:7:lp:/usr/spool/lp:
ftp:xxxxxxxxxxxxx:8:2:ftp:/usr/spool/ftp:
who::22:0:who command:/bin:/bin/who
nobody:xxxxxxxxxxxxx:60001:60001:NFS generic user:/tmp:/bin/noshell
Guest::90:90:A/UX Guest account:/users/Guest:/bin/csh
start:PG/qLJaYo/6mo:100:100:Initial login:/users/start:/bin/csh
student:HnOGsoYnjTrEM:1000:1000:Participant,Training Room,555-3298,800-555-3964:
/users/student:/bin/ksh
• █
```

```
login:password:uid:gid:data:homedir:shell
```

# /etc/passwd

## Notes

# /etc/group

The system keeps track of group memberships through the /etc/group file. Users can belong to as many as eight groups at once. Each line of the group file consists of four fields:

- group name
- encrypted password
- numerical group id
- members

These four fields are delimited by colons ( : ). The various members within a group are delimited by commas ( , ).

### Example

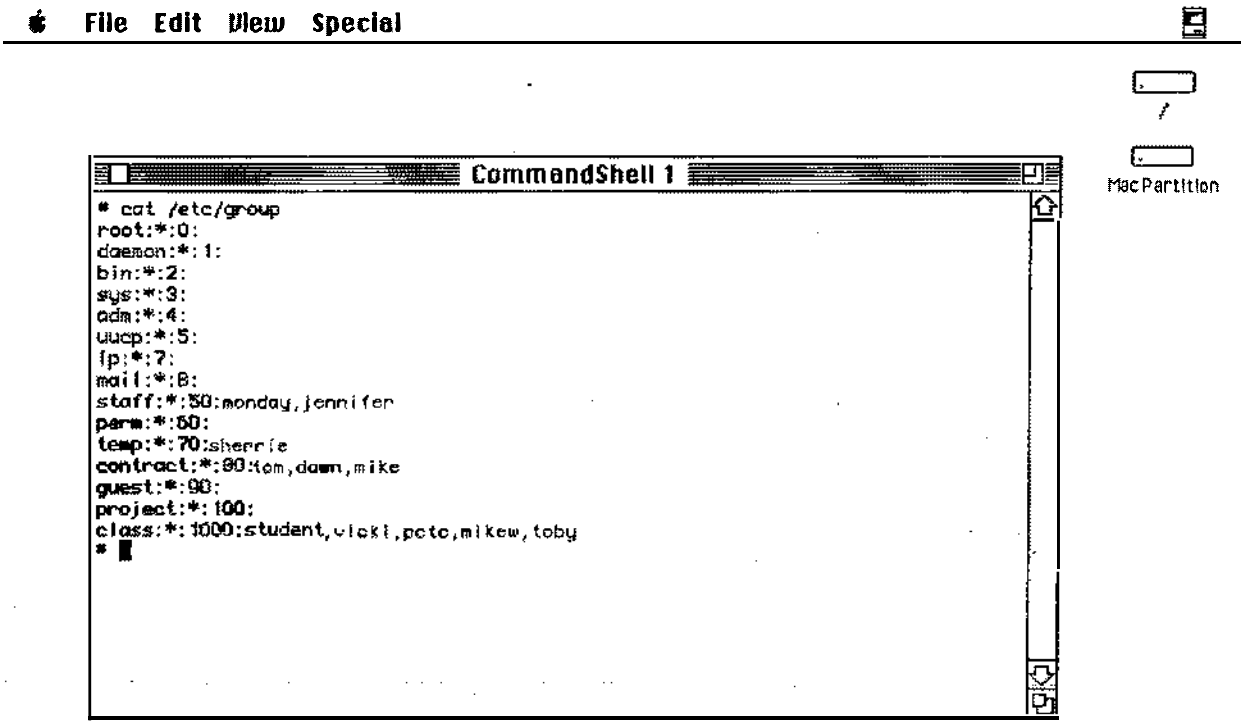
```
# cat /etc/group
root:*:0:
daemon:*:1:
bin:*:2:
sys:*:3:
adm:*:4:
uucp:*:5:
lp:*:7:
mail:*:8:
staff:*:50:monday,jennifer
perm:*:60:
temp:*:70:sherrie
contract:*:80:tom,dawn,mike
guest:*:90:
project:*:100:
class:*:1000:student,vicki,pete,mike,toby
```

### Note

Because there is no easy way to manipulate group passwords and they are inherently insecure, it is not advisable to use them. Generally, restricting memberships to groups provides sufficient security.

### References

*A/UX Local System Administration*
*group (4)*



```
# cat /etc/group
root:*:0:
daemon:*:1:
bin:*:2:
sys:*:3:
adm:*:4:
uucp:*:5:
lp:*:7:
mail:*:8:
staff:*:50:monday,jennifer
perm:*:60:
temp:*:70:sherrie
contract:*:80:tom,dawn,mike
guest:*:90:
project:*:100:
class:*:1000:student,vicki,pete,mikew,toby
# 
```

```
group name:password:gid:members
```

# /etc/group

## Notes

# Adding a User

The adding of a user is a multistep process. The general steps to add a new user are:

**Step 1** A new line is added in /etc/passwd, containing relevant user information.

**Step 2** A home directory is made for the user. The permissions on, and ownership of, the directory are changed to match the new user.

**Step 3** Standard initialization files are copied from /usr/lib/skel into the home directory of the new user. These files are:

.profile
.kshrc
.cshrc
.login
.logout
README

The ownership of the files is changed to be that of the new user.

**Step 4** If the new user is to have a Useful Commands directory, create the directory, change its permissions, and link the various commands to the /Useful Commands directory.

**Step 5** If the new user is to have their own System Folder, it is created in the new home directory. The contents of the folder are copied or linked (as appropriate) into the new folder.

Most of these steps are done for you when you invoke the `adduser` script.

**References** *A/UX Local System Administration*

# Adding a User

- ❑ Add a new line in /etc/passwd
- ❑ Make a home directory
  - change permissions and ownership
- ❑ Copy initialization files
  - change ownership
- ❑ Link Useful Commands directory
- ❑ Add a personal System Folder

6

## Notes

# adduser - First Dialog

Invoking Commando on the adduser shell script gives us a series of dialogs and allows us to add a user, or users, to the system quite easily.

**Operation** The choices here allow the administrator to add one or a series of users.

**Login name** This field contains the user's login name. By convention names are all lower case. The names must be unique on any given system.

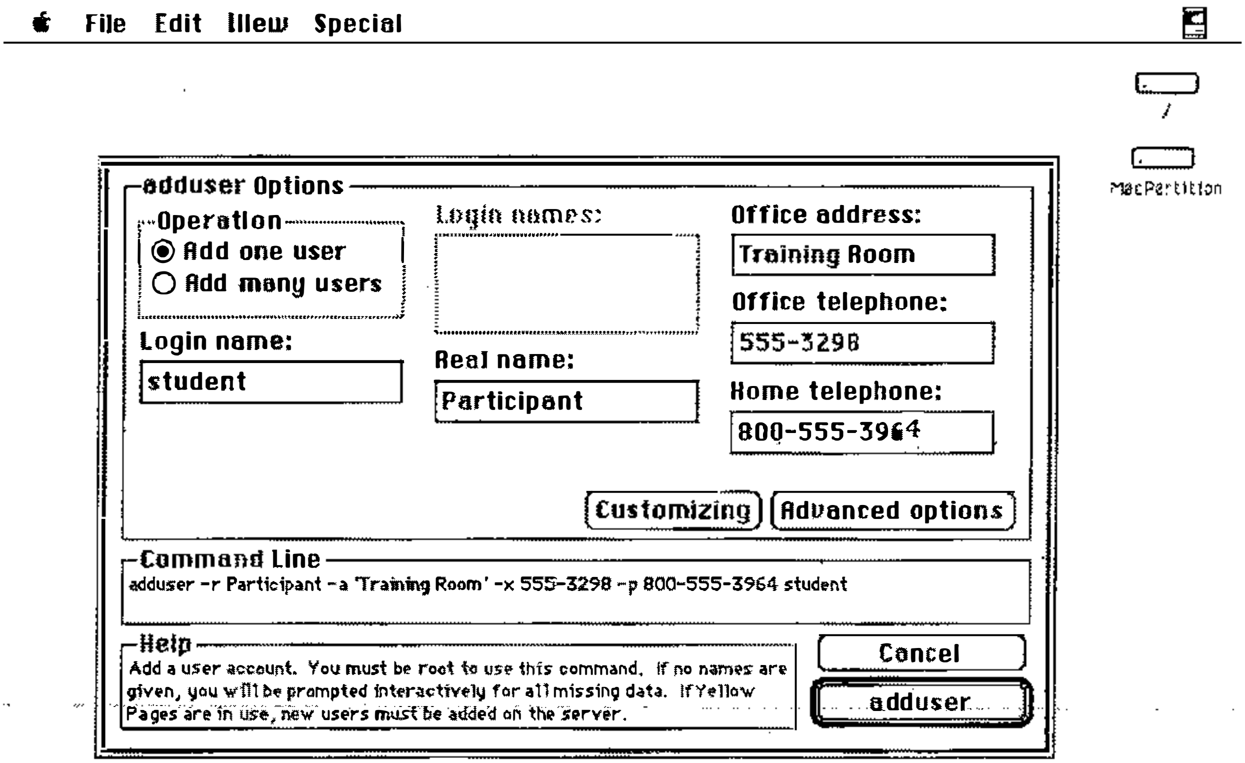
**Miscellaneous** User information choices—Real name, Office address, Office telephone, and Home telephone—are put into the passwd file in a single field.

**Customizing** This leads to another dialog (covered soon) that allows the users additional selections.

**Advanced Options** This leads to another dialog (covered soon) that allows the users additional advanced selections.

**Note** In interactive mode, the administrator is prompted for the above information. Additionally, a password for the new user can be set from within the program in interactive mode.

**References** *adduser (1M)*



adduser Options

Operation
● Add one user
○ Add many users

Login name: student

Login names:

Real name: Participant

Office address: Training Room

Office telephone: 555-3298

Home telephone: 800-555-3964

Command Line

```
adduser -r Participant -a 'Training Room' -x 555-3298 -p 800-555-3964 student
```

Help

Add a user account. You must be root to use this command. If no names are given, you will be prompted interactively for all missing data. If Yellow Pages are in use, new users must be added on the server.

# adduser - First Dialog

Trash

## Notes

# `adduser`- Second Dialog

The Customizing dialog gives the administrator additional choices for the new user.

**Shell**

The choices here allow the administrator to choose a start up program for the new user. C shell is the default. The startup program does not have to be a shell, it can be an application.

**Login group**

This field contains the user's default login group. If the group does not exist, it is created.

**Home directory**

This allows either the creation of a new directory or the use of an existing directory for the new user's login directory.

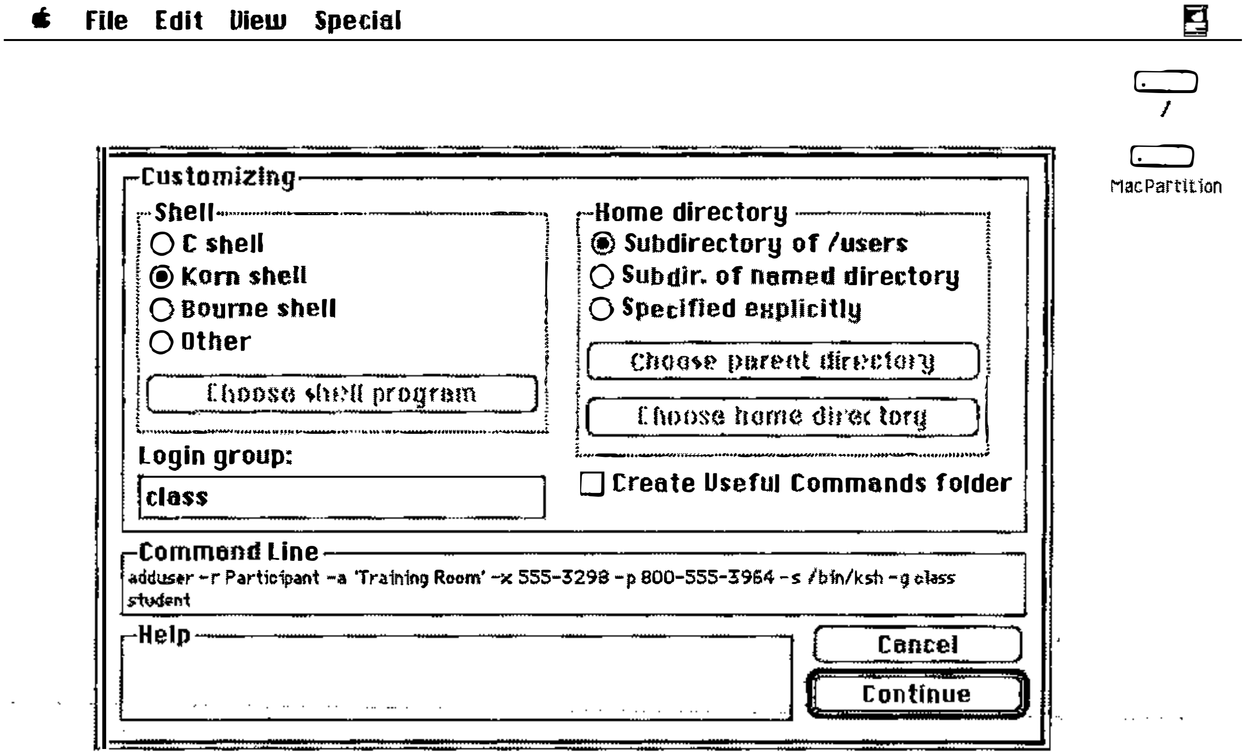
**Create Useful Commands folder**

Selecting this option creates a folder of often-used commands in the new user's home directory.

After the selections are made, click on Continue.

**References**

*adduser (1M)*



Customizing

Shell

○ C shell
● Korn shell
○ Bourne shell
○ Other

Choose shell program

Home directory

● Subdirectory of /users
○ Subdir. of named directory
○ Specified explicitly

Choose parent directory

Choose home directory

Login group:

class

□ Create Useful Commands folder

Command Line

```
adduser -r Participant -a 'Training Room' -x 555-3298 -p 800-555-3964 -s /bin/ksh -g class
student
```

Help

Cancel

Continue

# `adduser` - Second Dialog

## Notes

# adduser- Third Dialog

The Advanced options dialog gives the administrator further choices in setting up a new user.

**Numeric User ID**

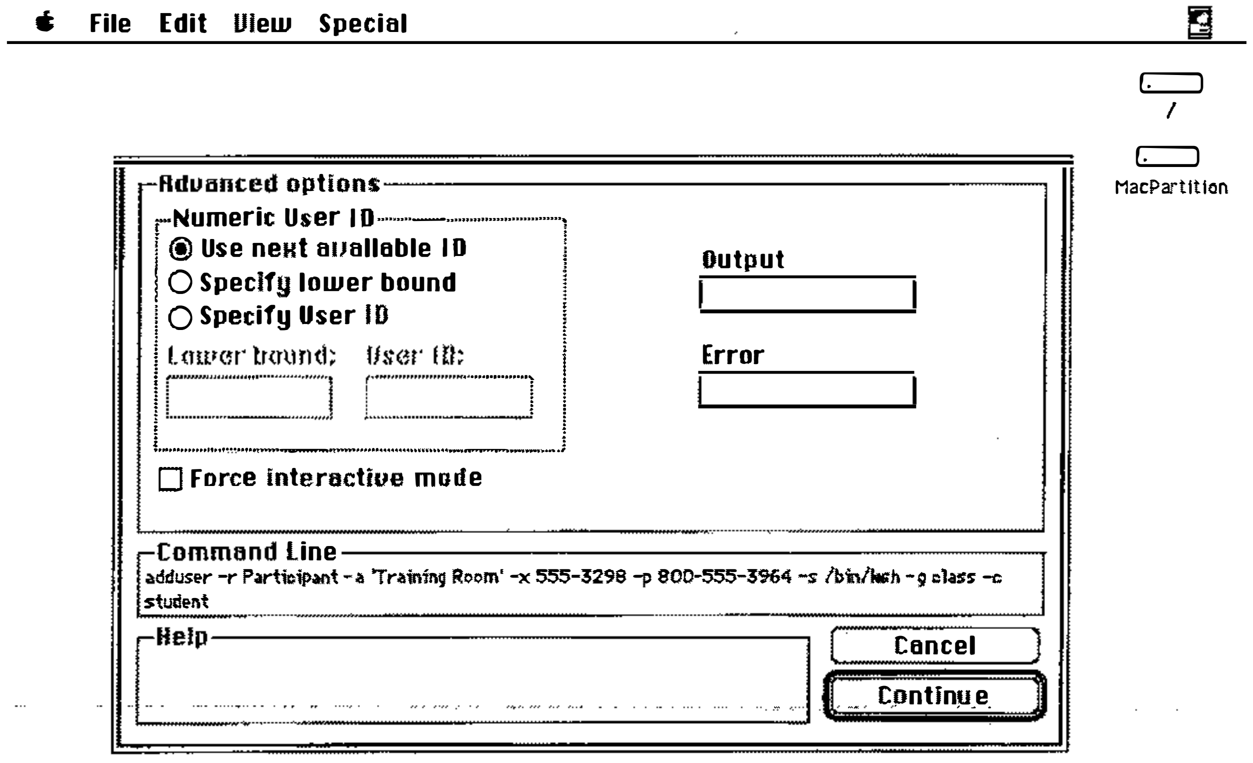
The choices here allow the administrator to choose the next available user id, a specific user id, or the lower bound of a range of user ids (for those installations that segregate groups by user id).

**Force Interactive mode**

Selecting this option causes the shell script to prompt the administrator for information which has not been specified in the first dialog. The administrator is also prompted for a user password.

**References**

*adduser (1M)*



Numeric User ID

● Use next available ID
○ Specify lower bound
○ Specify User ID

Lower bound: User ID:

□ Force interactive mode

Command Line

```
adduser -r Participant -a 'Training Room' -x 555-3298 -p 800-555-3964 -s /bin/ksh -g class -c
student
```

Help

## `adduser` - Third Dialog

## Notes

# Initialization Files

Each time a user logs on, there are certain actions the shell will automatically perform. One of these actions is the execution of some initialization files, assuming of course these files exist. The name of these files is different for each shell.

**Bourne Shell**

When a user logs on to the Bourne shell, the file $HOME/.profile is executed. Whenever subshells are run, the current environment is inherited by the new shell.

**Korn Shell**

Like the C shell, there are two files involved in initializing the Korn shell. When a user first logs on, the $HOME/.profile file is read, followed by the file $HOME/.kshrc. Whenever subshells are run inside the same login session, the $HOME/.kshrc file is re-executed.

**C Shell**

When a user logs on to the C shell, the file `$home`/.cshrc is executed. Immediately thereafter the file `$home`/.login is executed. Whenever subshells are run, the current environment is inherited by the new shell and the file `$home`/.cshrc is executed again. When this happens, the settings in the .cshrc file override the definitions which may have been inherited by the parent shell.

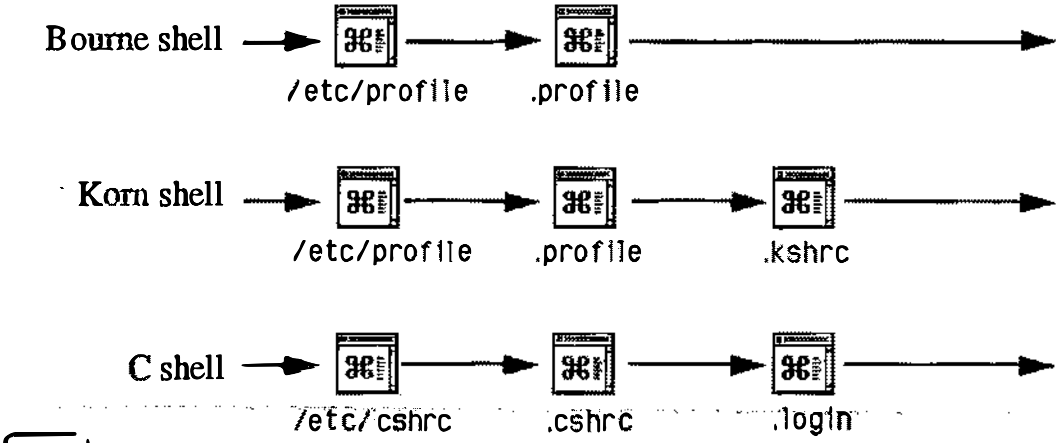
**Note**

Standard copies of the initialization files are in /usr/lib/skel.

**References**

*A/UX Local System Administration*

# Initialization Files



Bourne shell → /etc/profile → .profile →

Korn shell → /etc/profile → .profile → .kshrc →

C shell → /etc/cshrc → .cshrc → .login →

A/UX

10

## Notes

# Environment Variables

Several variables are commonly used to customize a user's work environment. These variables are commonly set in a user's initialization files.

**PATH** This defines the directories to be searched for commands. It also defines the order in which the directories are to be searched.

**umask** This sets the default permissions on files a user creates. Its value is the complement of the desired permissions. For example, if you want your files to be created with permission 754, set the umask to 023.

**TBMEMORY** This sets the size of a user's virtual memory. The maximum is 8MB for a 24-bit login session and 256MB for a 32-bit login session. Perfomance tends to degrade rapidly if virtual memory size is set to more than twice the size of the physical memory in the system. For example, to set the virtual memory size to 8MB use the line:

```
TBMEMORY=8m; export TBMEMORY
```

**Note** MultiFinder® has access to the virtual memory you set with this variable. The maximum memory space MultiFinder can access, however, is 16MB.

**PS1** The defines your primary prompt. For example, to change the prompt use the line:

```
PS1='what now?'
```

**TERM** This defines the characteristics the system will user to communicate with a terminal. It is primarily important when logging on over serial lines, as many different kinds of terminals could be used.

**FINDER_EDITOR** This defines the editor that is invoked when a text file is double-clicked.

**References** *A/UX Local System Administration*

# Environment Variables

□ PATH

□ TBMEMORY

□ umask

□ PS1

□ TERM

□ FINDER_EDITOR

11

## Notes

# Adding a System Folder

A/UX is delivered with its System Folder as a public directory; any user can change files in it. Changes made in this folder, for example added fonts, will be seen by each user of the system when they work on the console.

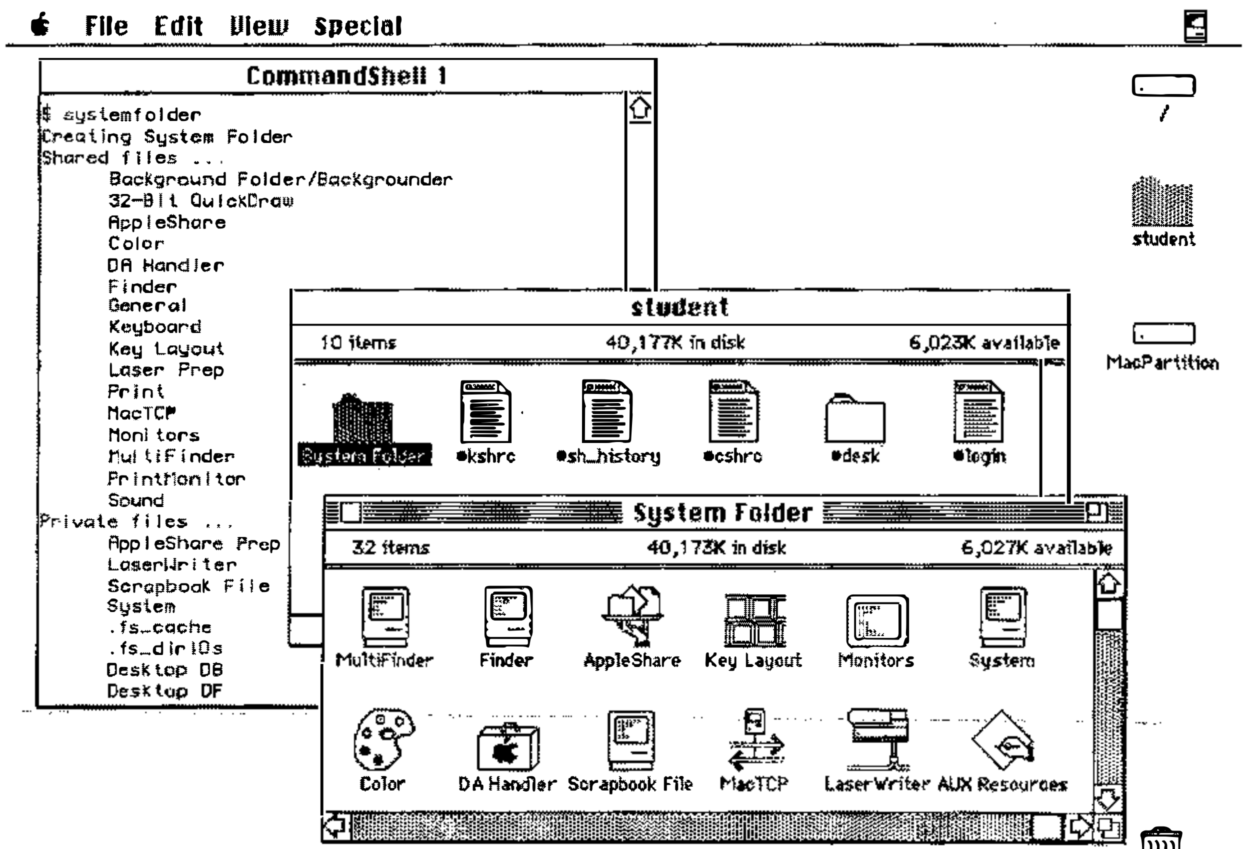
Each user can have their own System Folder so they can customize the way their Desktop works. A user can create a personal System Folder by using the `systemfolder` command.

**Note** The easiest way to add a System Folder for a new user is to log on as that user and run the `systemfolder` command.

Once a user has their own System Folder, changes made in the global System Folder (/mac/sys/System Folder) will not affect them.

Each personal System Folder takes about 1MB of disk space when it is first installed. The size of the folder will grow as users add their own fonts, inits, and cdevs.

**References** *A/UX Essentials*



```
$ systemfolder
Creating System Folder
Shared files ...
        Background Folder/Backgrounder
        32-Bit QuickDraw
        AppleShare
        Color
        DA Handler
        Finder
        General
        Keyboard
        Key Layout
        Laser Prep
        Print
        MacTCP
        Monitors
        MultiFinder
        PrintMonitor
        Sound
Private files ...
        AppleShare Prep
        LaserWriter
        Scrapbook File
        System
        .fs_cache
        .fs_dirIDs
        Desktop DB
        Desktop DF
```

# Personal System Folder

## Notes

# Removing a User

This can be a short process or a lengthy one, depending on the number of files a user has created. Irrespective of length, the procedure itself is straightforward.

**Step 1** Replace the user's password in /etc/passwd with some short character string, such as "x" or "void". Having a regular string in the file prevents anyone from logging in as that user. The editor `vipw` can be used, or if you prefer to use `TextEditor` change the permissions on the file before (and after!) editing.

**Step 2** Backup the all user's files onto tape or disk. A user's files can be identified by the command line:

```
find / -user login_name -print
```

**Step 3** Verify the backup.

**Step 4** Examine all the files owned by the user to determine who else might work with or need them (usually members of the same group). Reassign ownership of the appropriate files.

**Step 5** Remove the remainder of the user's files, including the initialization files in their home directory.

**Step 6** Remove the user from the /etc/group file.

**Step 7** Remove the user from the /etc/passwd file.

**Note** A user should not be removed from the /etc/passwd file until all of that user's files have been reassigned or removed. Several programs use the password file to provide translation between user names and user ids.

**References** *Local System Administration*

# Removing a User

- ❑ Replace user password in /etc/passwd
- ❑ Back up the user files
- ❑ Verify the backup.
- ❑ Reassign ownership of user files
- ❑ Remove user files
- ❑ Remove user from /etc/group
- ❑ Remove user from /etc/passwd

13

## Notes

# Checkpoints

1 What is the purpose of having individual user accounts?

2 What is the purpose of having groups?

3 What is the structure of the /etc/passwd file?

4 What is the structure of the /etc/group file?

5 What steps are there to adding a user to an A/UX system?

6 What steps are there to removing a user from an A/UX system?

# Exercises

**Step 1** Log on to your A/UX system.

**Step 2** Add the user "john" to your system. Use the information below:

| | |
|---|---|
| Full name: | `John Awkhacker` |
| Login name: | `john` |
| Password: | `yoyodyne` |
| User's Identification number: | `200` |
| Login group: | `johns` |
| Full pathname of user's home directory: | `/users/john` |
| Full pathname of user's startup shell: | `/bin/ksh` |

**Step 3** Log off the system, then log back on as john.

**Step 4** Check who owns the files in your home directory.

**Step 5** Log off the system, then log back on as root.

**Step 6** Remove the user john by deleting the line containing that login name from the file /etc/passwd.

**Step 7** Check who owns the files in john's home directory.

**Step 8** Remove the member john from the file /etc/group.

**Step 9** Check what happens to the files in john's home directory.

**Step 10** Shut down your system.

# Module 4
# Disk Drive Administration

## Alternate Learning Path

# Table of Contents

# Disk Drive Administration

## Notes

# Disk Drive Administration

**Goal**

In this module you will learn how to perform a variety of disk administration tasks for A/UX, including making and checking an A/UX file system, repairing damaged file systems, adding auxiliary disk space, and changing disk partitions.

**Objectives**

By the end of this module, you will be able to:

- describe the basic tasks of A/UX disk drive administration
- check and repair errors and inconsistencies in A/UX file systems
- describe the basic A/UX disk partitioning scheme
- partition an A/UX disk drive with HD SC Setup 2.01 and dp
- make a file system, mount, and use an external A/UX disk drive

**Activities**

Lecture/Discussion/Checkpoints/Exercises

**References**

*Setting Up Accounts and Peripherals*
*A/UX Local System Administration*

# Objectives

- ❑ Describe the basic tasks of A/UX disk drive administration
- ❑ Check and repair errors in A/UX file systems
- ❑ Describe the basic A/UX partitioning scheme
- ❑ Partition an A/UX disk drive with HD SC Setup and `dp`
- ❑ Explain what a file system is
- ❑ Create a file system on, mount, and use an external A/UX disk drive

2

## Notes

# What Is Disk Drive Administration?

One of the main responsibilities for administrators of any UNIX®-based system is disk drive administration. A number of disk administration issues must be dealt with, depending on the configuration of your system:

- basic care and maintenance of your internal or external hard drives
- setup and administration of one or more external drives (either Apple or third party)

Within this context, this module will help you become proficient in the following disk administration tasks:

- initial disk setup
- partitioning
- monitoring performance
- checking for errors
- fixing software, if necessary

# What Is Drive Administration?

- ❑ Initial disk setup
- ❑ Partitioning
- ❑ Monitoring performance
- ❑ Checking for errors
- ❑ Fixing software, if necessary

3

## Notes

# Disk Overview

A/UX hard disks are divided into logical sections, called partitions. Partitions allow various information to be put in a well-defined place, distinct and separate from other information on the disk. You have already worked with two partitions on the A/UX disk. The A/UX file system ( / ) and a Macintosh file system (MacPartition) are on the same disk, but are logically separate.

**Partition** A partition is a portion of physical disk space consisting of contiguous disk sectors. Each partition is given a "type" that indicates how it can be used. For example, Macintosh and A/UX partitions are of different types. In addition to their partition number, partitions may also have two kinds of other identifiers: names and slice numbers (more on these soon)

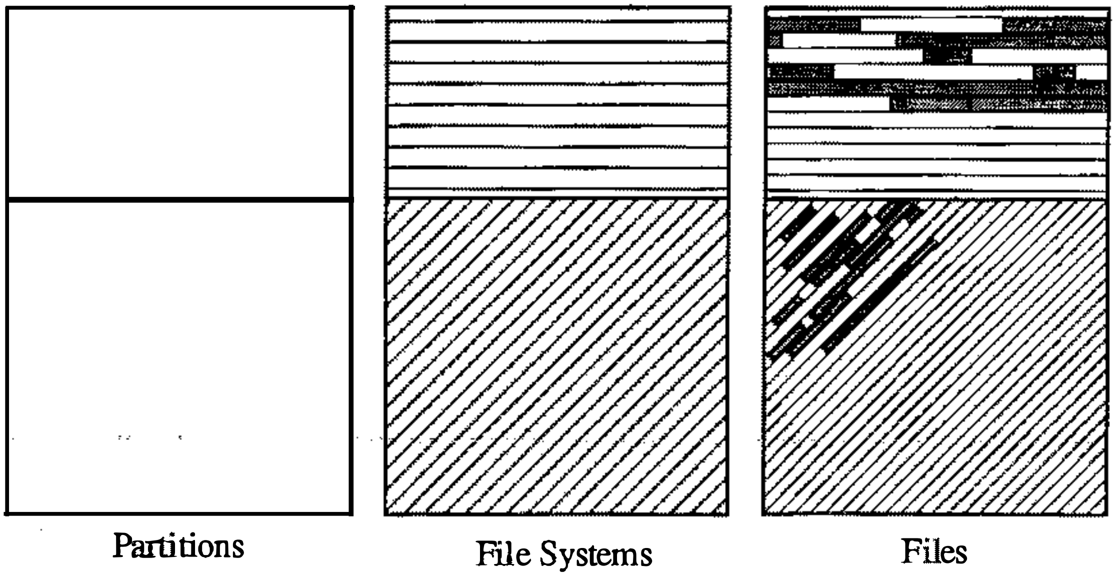
**File System** A file system is a logical framework imposed on a partition that allows access to blocks of disk sectors in a single operation. There are different frameworks that can be used within a single partition, for instance the Berkeley UFS or the AT&T SVFS. File systems do not extend past partition boundaries.

**File** An array of bytes, typically stored as logical blocks within a file system.

There are several advantages to separating disks into partitions:

- Obviously, you can have entirely separate operating systems on the same disk, and boot between them as needed.
- Backups (the subject of the next module) can be made less costly and time-consuming. Since several of the backup packages back up entire partitions, you can spend less money on backup media if you have the frequently changing files in one partition, and the more static files in another. Also, the fewer files you back up, the less time it takes.
- You can selectively grant access to individual file systems to users, especially users on a network.

# Disk Overview



4

## Notes

# How A/UX Names Disks

A/UX designates disk drives by a special file name, which is of the form `/dev/dsk/cndmsy`. The name refers to a special file in the `/dev/dsk` directory; here's what the letters mean:

**c*n*** Controller number, or the number (*n*) of the disk drive. This number is the same as the SCSI ID number of the disk drive.

**d*m*** Drive number. Some disk drive controllers allow you to use more than one disk drive at the same SCSI ID, so the drive number (*m*) is used to distinguish between the drives at the same SCSI ID. This number is 0 in nearly all cases.

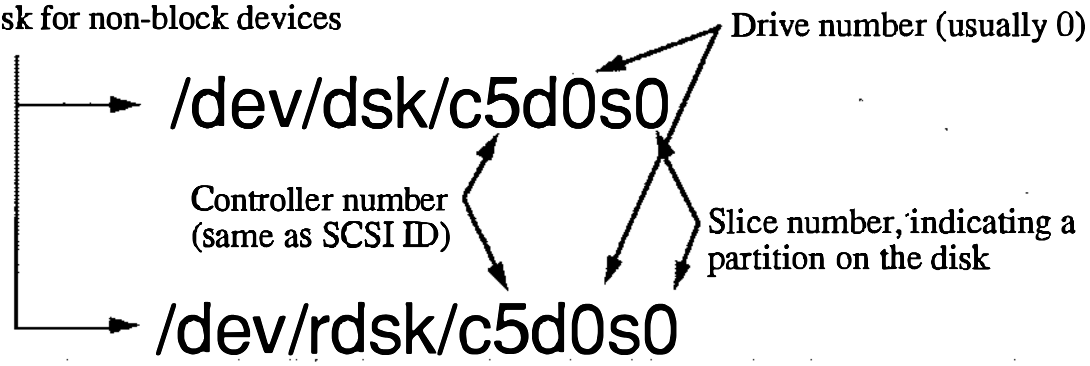
**s*y*** Slice number. The slice number is another name for a particular partition. In A/UX, the root partition is slice 0, swap is 1, and 31 refers to the entire disk. Other A/UX partitions created with HD SC Setup (2.01 or later) are named with slice numbers automatically. The commands `dp` and `pname` can also be used for naming partitions with slice numbers.

# How A/UX Names Disks

Location of the special file:

/dev/dsk for block devices

/dev/rdsk for non-block devices



Drive number (usually 0)

/dev/dsk/c5d0s0

Controller number
(same as SCSI ID)

Slice number, indicating a
partition on the disk

/dev/rdsk/c5d0s0

5

## Notes

# The Default A/UX Disk Partitioning Scheme

The table on the facing page shows the eight partitions on the A/UX Master volume. Here's what everything means:

The number in the slice column is used to specify the A/UX partition using the form `/dev/dsk/c3d0sy`, where*y* is the slice number.

The partition number is used in the partition map (you don't need to know much about this one.)

The partition name is how some commands, such as `pname`, refer to the partition.

The number in the size column is the size of the partition in 512-byte blocks.

**Note** By convention, almost all UNIX programs that report disk blocks use 512-byte blocks for the report.

The number in the start column is the number of the first block in the partition (again in 512-byte blocks.)

The A/UX Root partition is the one from which A/UX runs, and the MacOS partition is the one from which the system boots (via A/UX Startup.) The A/UX system swaps programs out to.the Swap partition.

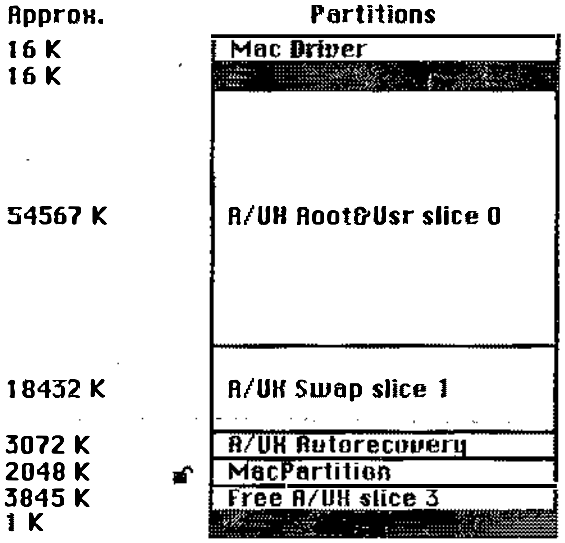
The Eschatology partition is used with the `esch` program (available from A/UX Startup) to help recover the root file system in case of a crash. It is at a different physical area of the disk, so (hopefully) a physical head crash in the root will leave it intact.

The Free A/UX partition is leftover space on the disk that can be used for a (small) A/UX filesystem.

**Note** It is not advisable to repartition the A/UX system disk.

**Reference** *A/UX Local System Administration*

# Default A/UX Partitioning



| Approx. | Partitions |
|---|---|
| 16 K | Mac Driver |
| 16 K | ▨ |
| 54567 K | A/UX Root&Usr slice 0 |
| 18432 K | A/UX Swap slice 1 |
| 3072 K | A/UX Autorecovery |
| 2048 K | MacPartition |
| 3845 K | Free A/UX slice 3 |
| 1 K | ▨ |

80 MB HD SC

| slice | partition | name | size | start |
|---|---|---|---|---|
| - | 1 | Apple | 126 | 1 |
| - | 2 | Macintosh | 32 | 64 |
| - | 3 | Extra | 32 | 96 |
| 0 | 4 | A/UX Root | 109134 | 128 |
| 1 | 5 | Swap | 36864 | 109262 |
| - | 6 | Eschatology 1 | 6144 | 146126 |
| - | 0 | MacOS | 4096 | 152270 |
| 3 | 7 | Free A/UX | 7690 | 156366 |
| - | 8 | Extra | 2 | 164056 |

6

## Notes

# A/UX File Systems

A/UX can interact with four file system structures: HFS, UFS, SVFS, and NFS.

**HFS** This is the standard Macintosh file system.

- filename lengths are 31 characters

**SVFS** This is the standard AT&T UNIX file system. Among its features are:

- files are multiple of 512 byte physical disk blocks
- filename lengths are 14 characters
- single superblock, free block list represents free data blocks

**UFS** This is the UNIX file system devised at the University of California at Berkeley. It has several advantages over the AT&T System V file system, among them:

- supports fragmented blocks, which is more efficient in terms of file space
- fragmented blocks are defragmented dynamically
- symbolic links (links can be made across file systems)
- the last five to ten percent of the file system should not be used
- filename lengths are 255 characters
- multiple superblocks, one per cylinder group; bitmap represents free data blocks in each group

**NFS** The Network File System allows you to mount a file system into your directory structure even though the file system is not physically connected to your system, but is instead logically connected across a network.

**Note** A/UX supports both SVFS and UFS features. Since UFS has several advantages over SVFS, A/UX uses it as the default file system.

# A/UX File Systems

| Attribute | Macintosh HFS | Berkeley UFS | AT&T SVFS | NFS |
|---|---|---|---|---|
| file name length | 31 char | 255 char | 14 char | var. |
| logical block size (bytes) | 512 | 4096 | 1024 | var. |
| physical per logical blocks | 1 | 8 | 2 | var. |
| make file system | HD SC Setup | newfs | mkfs | n/a |
| file permissions | | | | |
| read | network only | yes | yes | yes |
| write | network only | yes | yes | yes |
| execute | no | yes | yes | yes |
| access permissions | | | | |
| user | network only | yes | yes | yes |
| group | network only | yes | yes | yes |
| other | network only | yes | yes | yes |

7

## Notes

# Preparing an Auxiliary A/UX Disk

Because A/UX occupies nearly all of an HD SC 80 disk drive, at some point you will probably need to prepare an external disk drive to serve as auxiliary disk space for A/UX. User files, programs, and applications can be stored on such a drive.

Preparing an auxiliary disk drive involves the following steps:

**Step 1** Initializing the drive

**Step 2** Creating one or more A/UX partitions

**Step 3** Making file systems on partitions

**Step 4** Checking the new partition and file system

**Step 5** Mounting to your A/UX root file system

**Step 6** Testing

# Preparing an Auxiliary A/UX Disk

- ❑ Initializing
- ❑ Partitioning
- ❑ Creating file system
- ❑ Checking
- ❑ Mounting
- ❑ Testing

8

## Notes

# Disk Initialization

**Hard disks**

The first step in preparing an A/UX disk for use is to initialize it with HD SC Setup, version 2.0.1 or higher.

When you initialize the disk, it is first tested by writing and verifying various data patterns to all locations on the disk. The testing process erases any data previously on the disk. Additionally, a bad block table is built, flagging any defective areas found on the disk so they won't be used.

Next, HD SC Setup builds a partition map that contains information about the starting block, length, and partition name. There must be a disk partition map entry (DPME) for each partition on disk. If HD SC Setup does not find a valid partition map at the outset, your only option is to initialize the disk.

You are prompted to name the Macintosh partition of your disk. At this point you can create the A/UX partitions or exit the HD SC program, thus creating a single Macintosh partition.

**Floppy disks**

Generally, when you insert a floppy disk into a drive, you will get a dialog box asking how you want to use the disk (Macintosh or A/UX.) If the desired type doesn't match the disk, it will be formatted appropriately.

**`diskformat`**

You can also use the A/UX `diskformat` command to initialize disks (both hard and floppy,) but this command has limitations. It does no automatic partitioning, as HD SC Setup can do, so it cannot create a Macintosh partition. Thus, a disk prepared with `diskformat` is unusable as a Macintosh volume, unless you go through some complicated partitioning (using the `dp` command.) In general, when creating hard disks for use with A/UX, it's better to initialize with HD SC Setup.

**References**

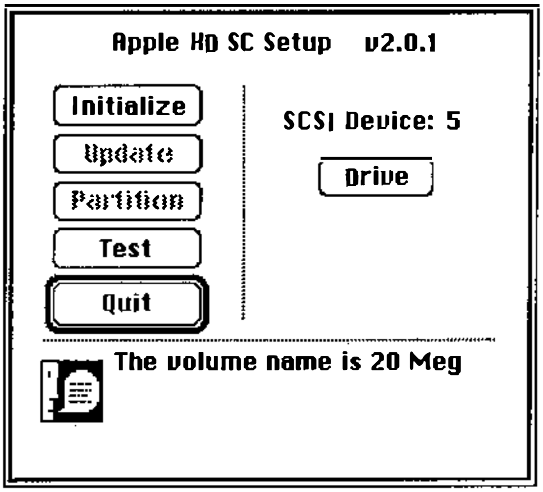
*diskformat (1M)*

# Disk Initialization

MacPartition

20 Meg

Open HD SC Setup
and select Initialize

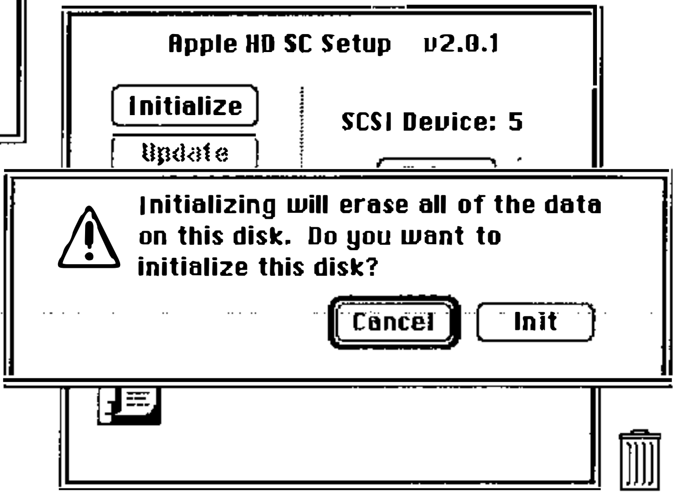


Apple HD SC Setup v2.0.1

SCSI Device: 5

The volume name is 20 Meg

... making sure it's ok
to erase the disk!



Apple HD SC Setup v2.0.1

SCSI Device: 5

Initializing will erase all of the data on this disk. Do you want to initialize this disk?

Trash

## Notes

# Disk Partitioning

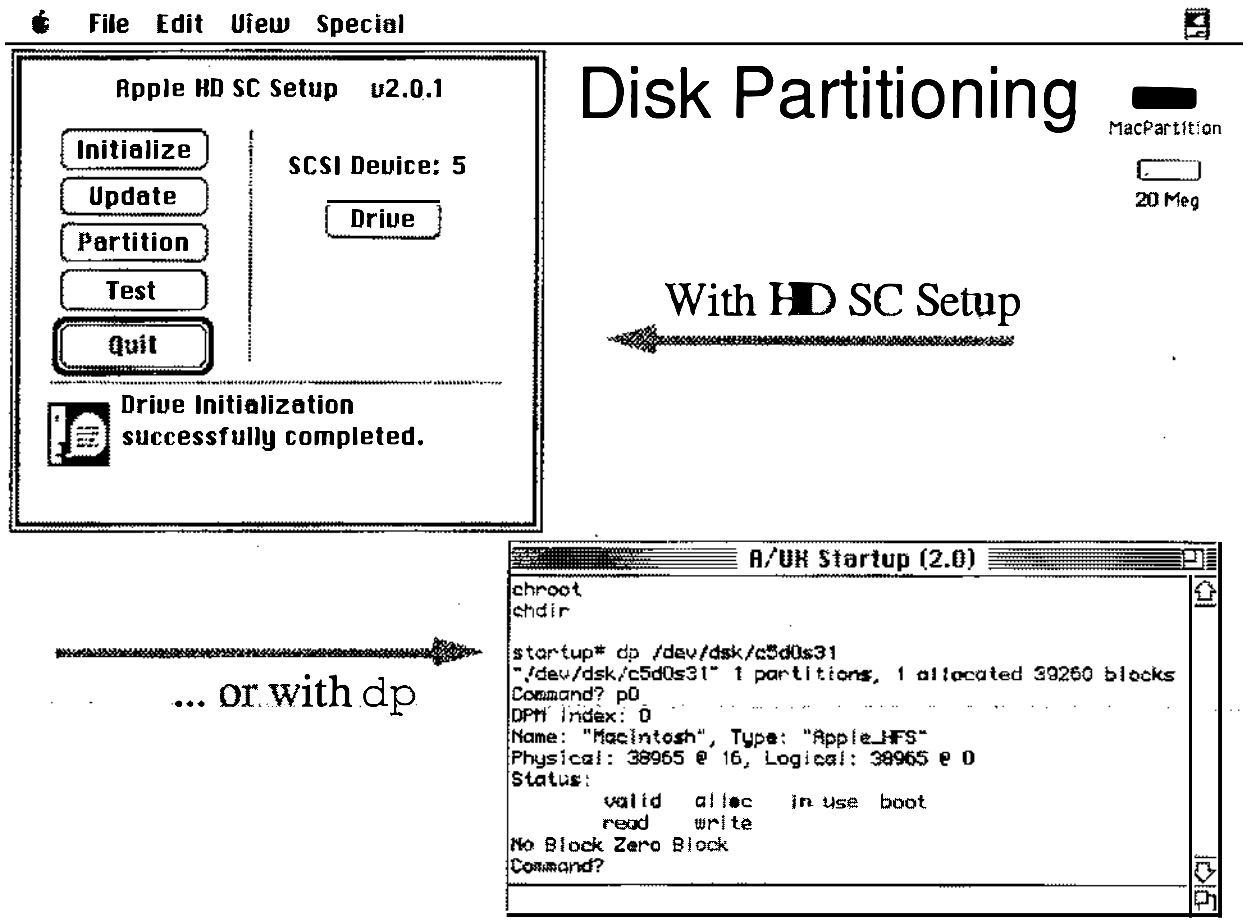
Once initialized, you can create an A/UX partition. The A/UX partition can be further subdivided into other partitions, or you can create new partitions on external drives. This allows a single disk to accommodate multiple file systems and even multiple operating systems.

You can have multiple A/UX partitions on your hard disks. The files that comprise A/UX are already on one partition, the root partition. With an additional hard disk, your own files can be kept on a separate partition on the external disk.

Partitioning in this way simplifies certain operations. For example, maintaining a separate partition for your root and user files speeds backing up your files; you can routinely back up your user partition and not your root partition, which is less likely to change over time.

When you partition, the existing partitions are not accessed at all. Partitions are reconfigured simply by altering the partition map. You can lose data if you change the size of an existing partition.

In this module, you will create a single A/UX partition on a 20 MB external disk drive.



# Disk Partitioning

With HD SC Setup

Drive Initialization successfully completed.

... or with dp

```
chroot
chdir

startup# dp /dev/dsk/c5d0s31
"/dev/dsk/c5d0s31" 1 partitions, 1 allocated 39260 blocks
Command? p0
DPM Index: 0
Name: "Macintosh", Type: "Apple_HFS"
Physical: 38965 @ 16, Logical: 38965 @ 0
Status:
        valid   alloc   in use  boot
        read    write
No Block Zero Block
Command?
```

## Notes

# Partitioning Choices

When you partition with HD SC Setup, you have the following choices:

**Maximum/Minimum Macintosh:**
These configurations create a single Macintosh partition, as indicated. The rest of the disk is unassigned free space, which you can later use to create one or more A/UX partitions.

**50% Macintosh:**
This configuration creates a single Macintosh partition that occupies 50 percent of the disk capacity; the rest is simply unassigned free space, which you can then partition for A/UX.

**Standard A/UX**
The standard A/UX System partitions with a 2 MB Macintosh partition. All remaining space is used for an A/UX partition.

**A/UX Sys, 40 MB Macintosh, Free A/UX**
The standard A/UX System partitions with a 40 MB Macintosh partition. All remaining space is used for an A/UX partition. (Requires 80 MB or larger.)

**A/UX System, Maximum Macintosh:**
The standard A/UX System partitions with all remaining space used for a Macintosh partition.

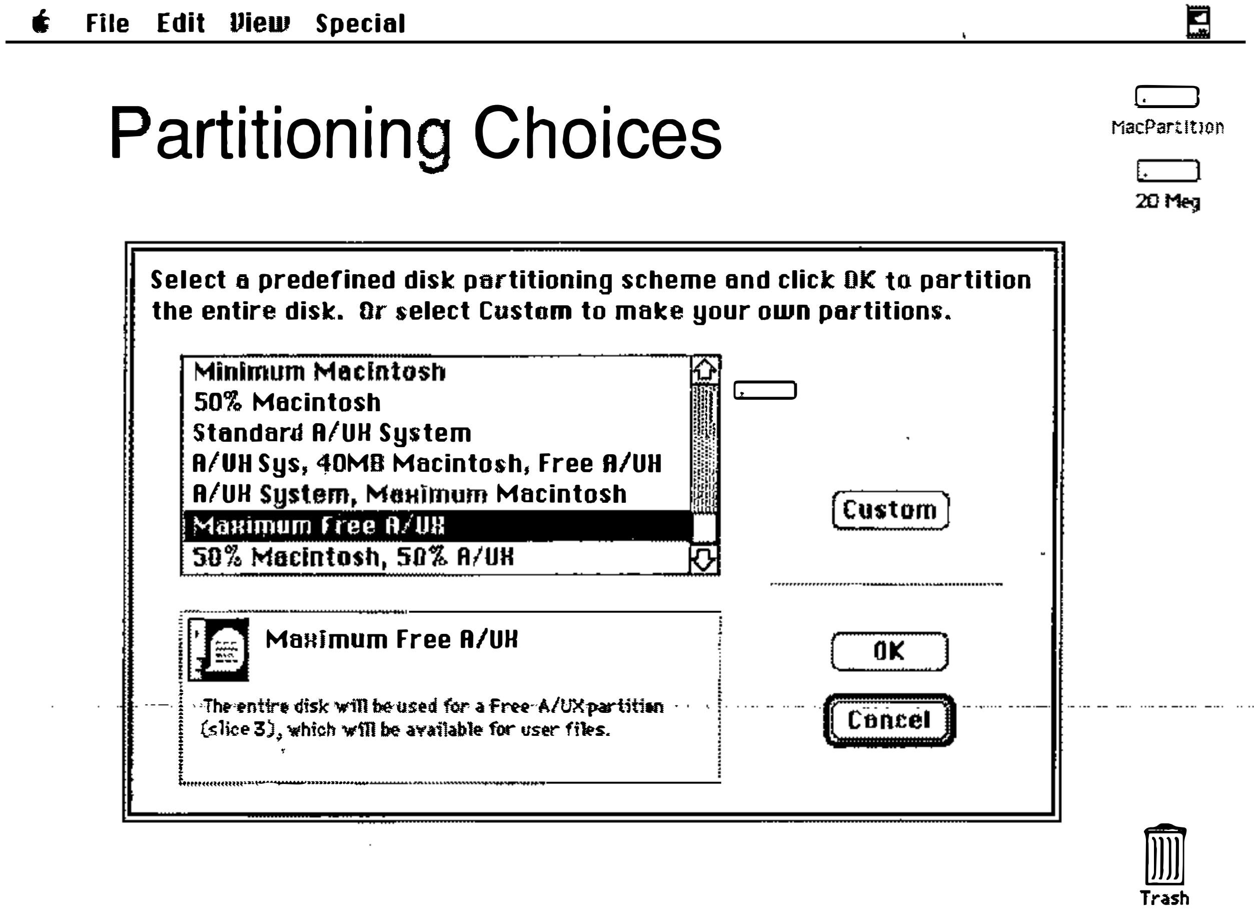
**Maximum Free A/UX**
The entire disk is used for an A/UX partition.

**50% Macintosh, 50% A/UX:**
The entire disk is available for user files. Half the disk is used for a Macintosh partition, half for an A/UX partition.

**References**

*Setting Up Accounts and Peripherals*
*A/UX Local System Administration*

# Partitioning Choices



Select a predefined disk partitioning scheme and click OK to partition the entire disk. Or select Custom to make your own partitions.

- Minimum Macintosh
- 50% Macintosh
- Standard A/UX System
- A/UX Sys, 40MB Macintosh, Free A/UX
- A/UX System, Maximum Macintosh
- Maximum Free A/UX
- 50% Macintosh, 50% A/UX

Maximum Free A/UX

The entire disk will be used for a Free A/UX partition (slice 3), which will be available for user files.

## Notes

# Custom A/UX Partitions

If you prefer, you can create a custom partition on the disk. In the example on the facing page, the entire free space is selected to be an A/UX partition.

**Step 1** Select Custom from the partitioning choice dialog box.

**Step 2** Click in any white areas of the partition map, then select Remove. This de-allocates the initial partitioning installed by the initialization. (NOTE: Do not remove the Mac Driver; you will probably not be able to access the disk if you do.)

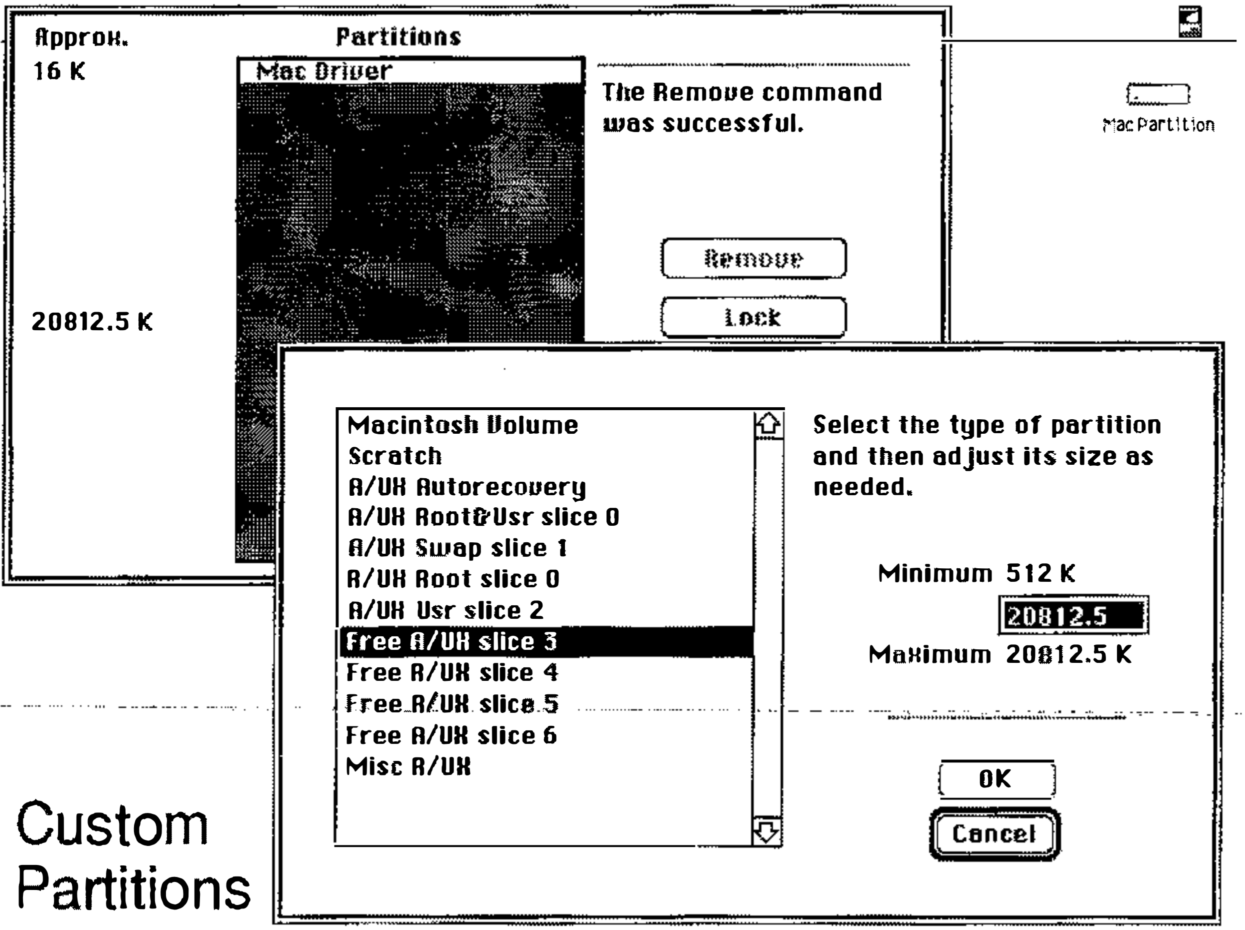
**Step 3** Click in the gray area to select it; a dialog box will appear asking the type of partition you want.

**Step 4** Select a partition type.

**Step 5** Enter a partition size.

**Step 6** Click on the OK Button.

You can subdivide the gray area into any number of A/UX partitions. The partitions can be of any type you specify.



Approx.
16 K

20812.5 K

Partitions
Mac Driver

The Remove command was successful.

Macintosh Volume
Scratch
A/UX Autorecovery
A/UX Root&Usr slice 0
A/UX Swap slice 1
A/UX Root slice 0
A/UX Usr slice 2
Free A/UX slice 3
Free A/UX slice 4
Free A/UX slice 5
Free A/UX slice 6
Misc A/UX

Select the type of partition and then adjust its size as needed.

Minimum 512 K
20812.5
Maximum 20812.5 K

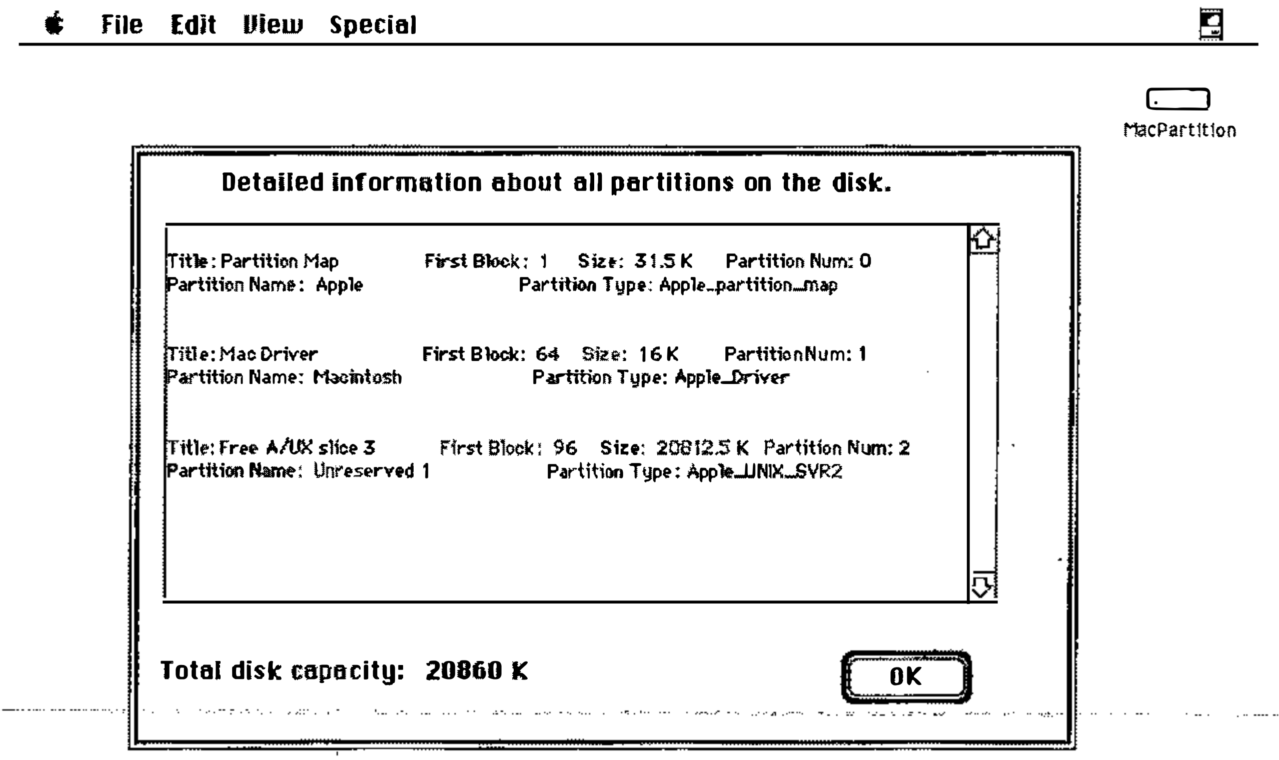
# Custom Partitions

## Notes

# Examining Details

Selecting Details (from the Custom Partitions dialog box) shows you a detailed summary of how your disk is organized.

You should make a special note of the partition number. When you need to use the `dp` command (discussed shortly), you will need to identify the partition by its number (or its name, but *not* its title.)



Detailed information about all partitions on the disk.

```
Title: Partition Map          First Block: 1     Size: 31.5 K    Partition Num: 0
Partition Name: Apple                  Partition Type: Apple_partition_map

Title: Mac Driver             First Block: 64    Size: 16 K      PartitionNum: 1
Partition Name: Macintosh               Partition Type: Apple_Driver

Title: Free A/UX slice 3      First Block: 96    Size: 20812.5 K  Partition Num: 2
Partition Name: Unreserved 1             Partition Type: Apple_UNIX_SVR2
```

Total disk capacity: 20860 K

# Examining Details

## Notes

# Creating a File System

Before you can use the disk space for A/UX, you need to make a file system on the new partition. The `newfs` command makes a UFS (Berkeley) file system.

**Syntax**

```
    newfs [-v] [options] device type
```

`newfs` creates file systems on *device*, which is the device on which the new file system is to be created.

**Note** *device* has to be a character device, that is, of the form `/dev/rdsk/cndmsy`.

The argument *type* indicates the disk type; *type* is used to find the appropriate disk name entry in /etc/disktab. The `newfs` command consults the disk label for disk partition information and /etc/disktab for disk architecture information, calculates the appropriate parameters to use in calling `mkfs`, and then builds the file system by invoking the program `/etc/fs/ufs/mkfs`. Parameters for third-party drives can be edited into /etc/disktab; thereafter they can be specified by *type*.

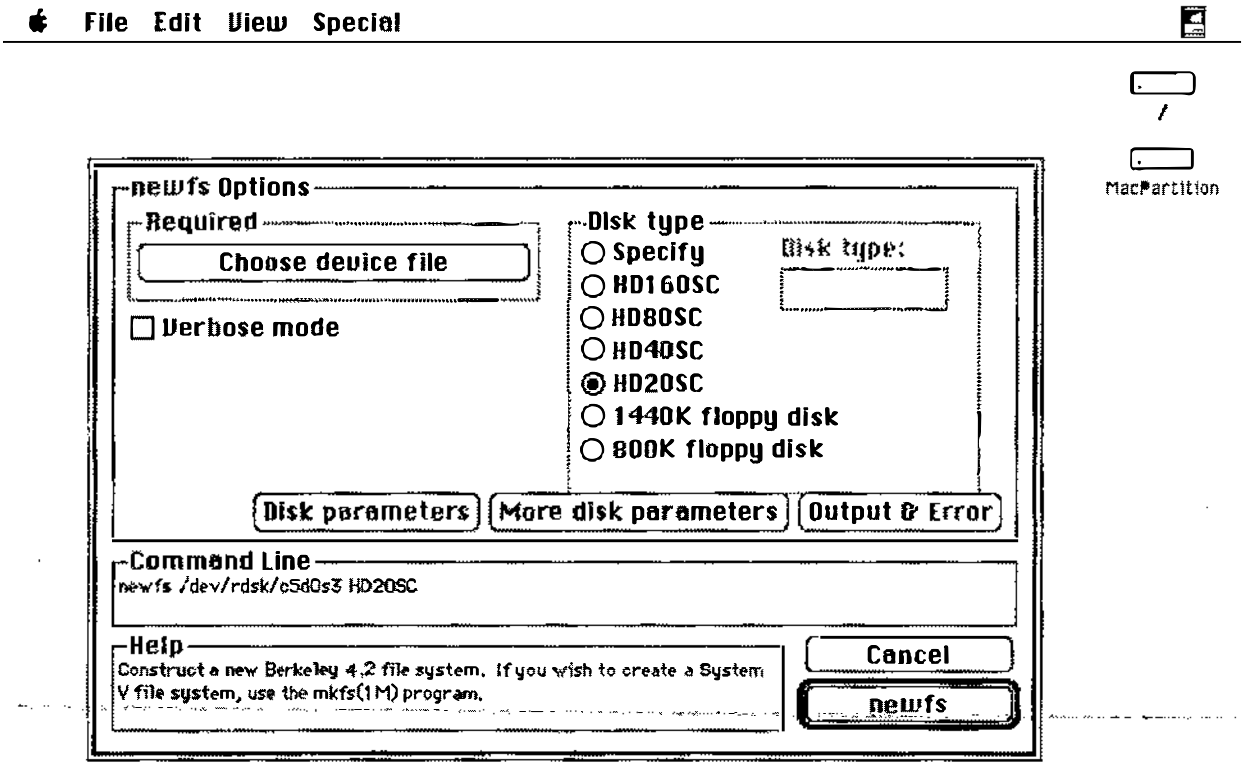
If the –v option is supplied, `newfs` prints out its actions, including the parameters passed to `/etc/fs/ufs/mkfs`.

`newfs` waits 10 seconds before creating the file system. During that time you can enter CONTROL–C to abort the command.

Additional options can be supplied to `newfs` to override the default parameters provided. Unless one has an intimate knowledge of Berkeley file system structure, this is heartily discouraged.

**Note** To make a SVFS file system, use the command `/etc/mkfs`.

**References**
*Setting Up Accounts and Peripherals*
*A/UX Local System Administration*
*newfs (1), disktab (4)*



newfs Options

Required

Choose device file

□ Verbose mode

Disk type

○ Specify

○ HD160SC

○ HD80SC

○ HD40SC

● HD20SC

○ 1440K floppy disk

○ 800K floppy disk

Disk type:

Command Line

```
newfs /dev/rdsk/c5d0s3 HD20SC
```

Help

Construct a new Berkeley 4.2 file system. If you wish to create a System V file system, use the mkfs(1M) program.

# Creating a File System

## Notes

# Checking a File System

Before mounting any file system, it is advisable to check it for inconsistencies. These can arise on UNIX systems because the system keeps track of various kinds of information about all open files in RAM. The files themselves are periodically written to disk, though not at the same time as their status information. If something unforeseen occurs, there is a discrepancy between the state of the file and its description. The program `fsck` compares several pieces of information to identify, and correct where possible, such discrepancies. Some of the checks made are:

1) blocks claimed by more than one inode
2) incorrect link counts
3) incorrect numbers of blocks in file system
4) bad inode format
5) unclaimed blocks
6) incorrect total free block and free inode counts
7) bad free-block list format

To verify the consistency of file systems, the program `fsck` is run as a part of every A/UX boot. Additionally, you can invoke `fsck` from the CommandShell, A/UX Startup, or through Commando.

**Note** File systems being checked, other than the root, should be unmounted. As the root file system cannot be unmounted while in A/UX, it is checked from A/UX Startup.

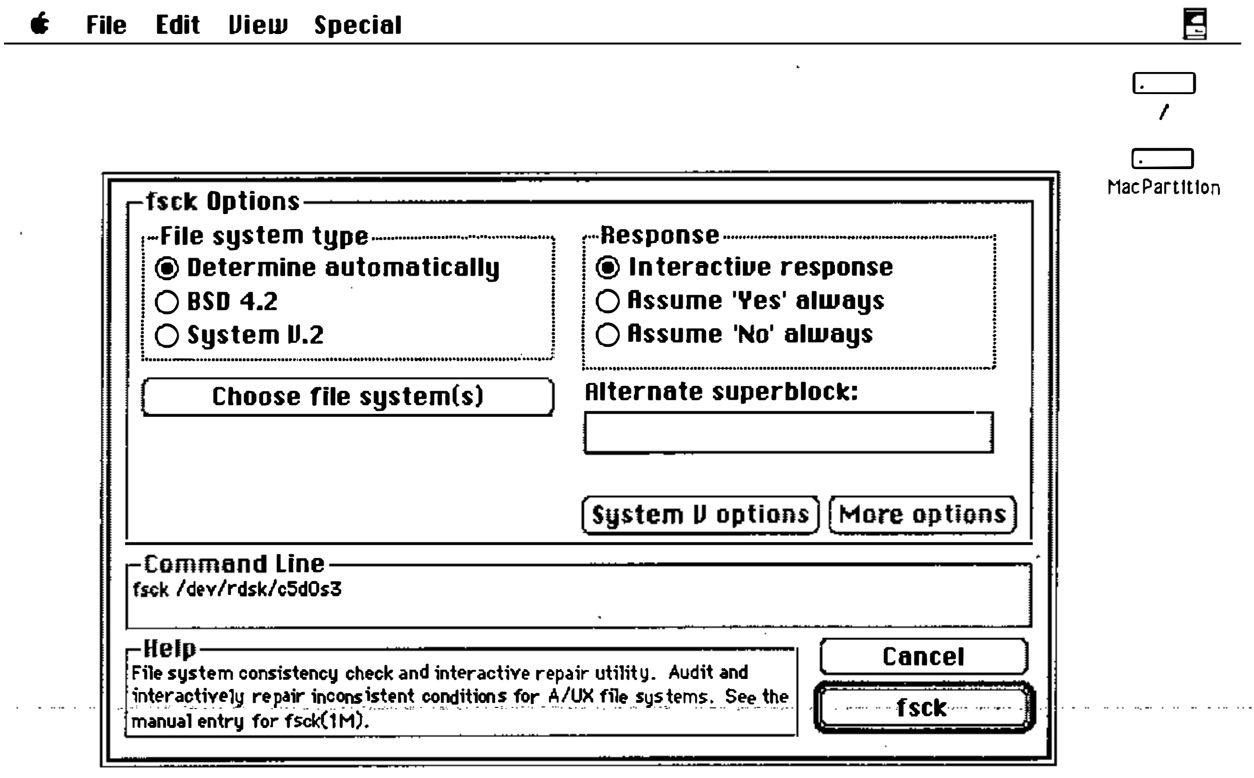
If a file system on disk is corrupted, you may have to run `fsck` a number of times until all the problems are corrected.

While there are several options to `fsck`, few of them are actually required. For example, to check the file system on slice 3 of the external disk drive at SCSI address 5, you would enter the following command:

**Example**

```
fsck -y /dev/dsk/c5d0s3
```

**References** *A/UX Local System Administration*
*fsck (1M), fstab (4)*



fsck Options

File system type
● Determine automatically
○ BSD 4.2
○ System V.2

Response
● Interactive response
○ Assume 'Yes' always
○ Assume 'No' always

Alternate superblock:

Command Line
```
fsck /dev/rdsk/c5d0s3
```

Help
File system consistency check and interactive repair utility. Audit and interactively repair inconsistent conditions for A/UX file systems. See the manual entry for fsck(1M).

# Checking a File System

## Notes

# Mounting a File System

The next-to-last step in adding a new disk is to attach it, via software, to the directory structure already existing in your A/UX file system. This is called mounting the file system, and is simply a way of providing a mapping from a directory name to a disk slice. The fact that you may suddenly be accessing an entirely different disk is transparent to you.

The syntax of the mount command is:

**Syntax**

```
mount [-frv] [-t type] [-T type]
[-o options] [devicefile mountpoint]

umount [-a] [frv] [-t type] [-T type]
[devicefile] [mountpoint]
```

`mount` without arguments simply reports all mounted file systems. The first syntax is the general case of mounting a specific device file to a specific directory. Options are available to limit access to read-only (for devices such as CD-ROM) and to enforce usage quotas, as well as other limits. These options can be included in the fstab file as well. The software mapping is dissolved by using the `umount` command.

**Example**

The following example creates a descriptive mount point and mounts a file system to it.

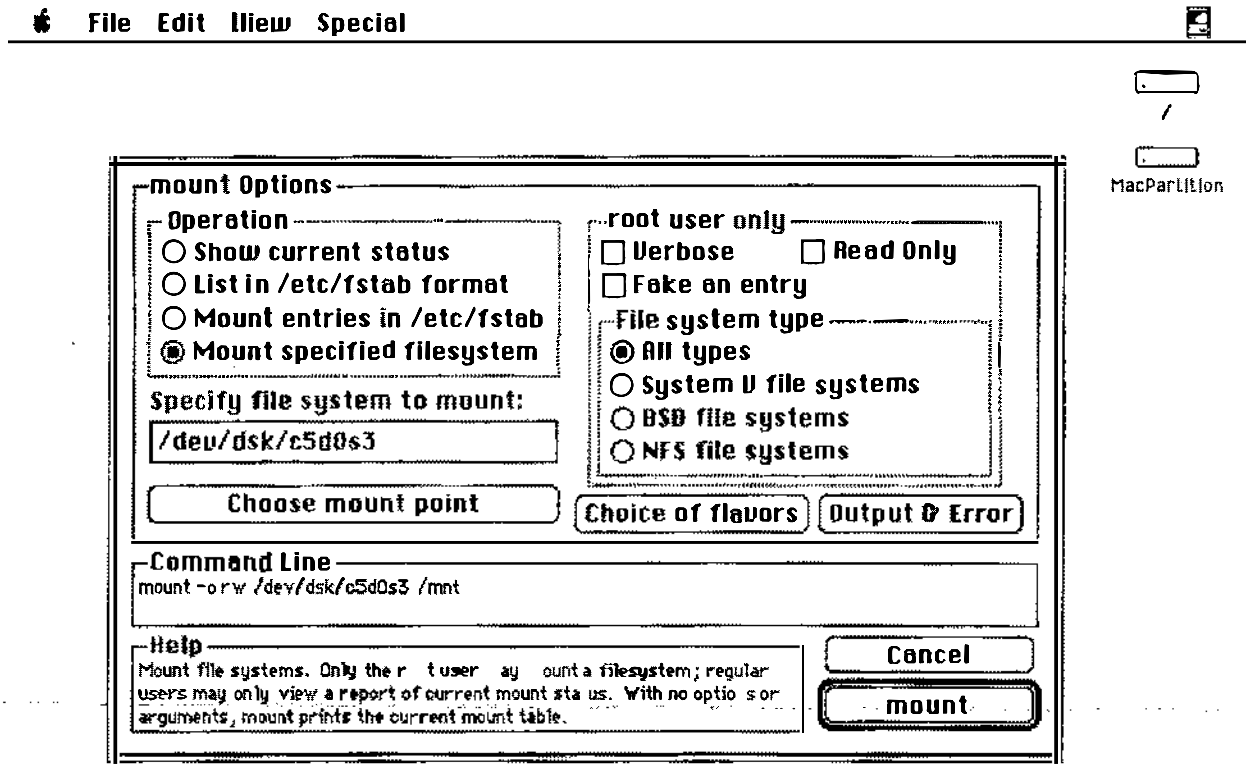
```
mkdir /scsi5
mount -o rw /dev/dsk/c5d0s3 /scsi5
```

**Note**

The mount point can be any directory. If the mount point you use is a directory that already has files in it, those files will be inaccessible until the filesystem is unmounted.

**References**

*A/UX Local System Administration*
*mount (1M), fstab (4)*



mount Options

Operation
○ Show current status
○ List in /etc/fstab format
○ Mount entries in /etc/fstab
● Mount specified filesystem

Specify file system to mount:
/dev/dsk/c5d0s3

root user only
□ Verbose □ Read Only
□ Fake an entry

File system type
● All types
○ System V file systems
○ BSD file systems
○ NFS file systems

Command Line

```
mount -o rw /dev/dsk/c5d0s3 /mnt
```

Help
Mount file systems. Only the r t user ay ount a filesystem; regular users may only view a report of current mount sta us. With no optio s or arguments, mount prints the current mount table.

# Mounting a File System

## Notes

# Testing a File System

The last step in adding a new disk is to test that we can communicate with the disk. This is a simple process, involving only the creation of a file and verifying its presence.

The steps involved in this are:

**Step 1** Mount the file system you are testing.

**Step 2** Verify the file system is mounted with the `mount` command.

**Step 3** Create a file in the directory to which the file system is mounted

**Step 4** Verify the file is in the correct directory.

**Step 5** Unmount the file system.

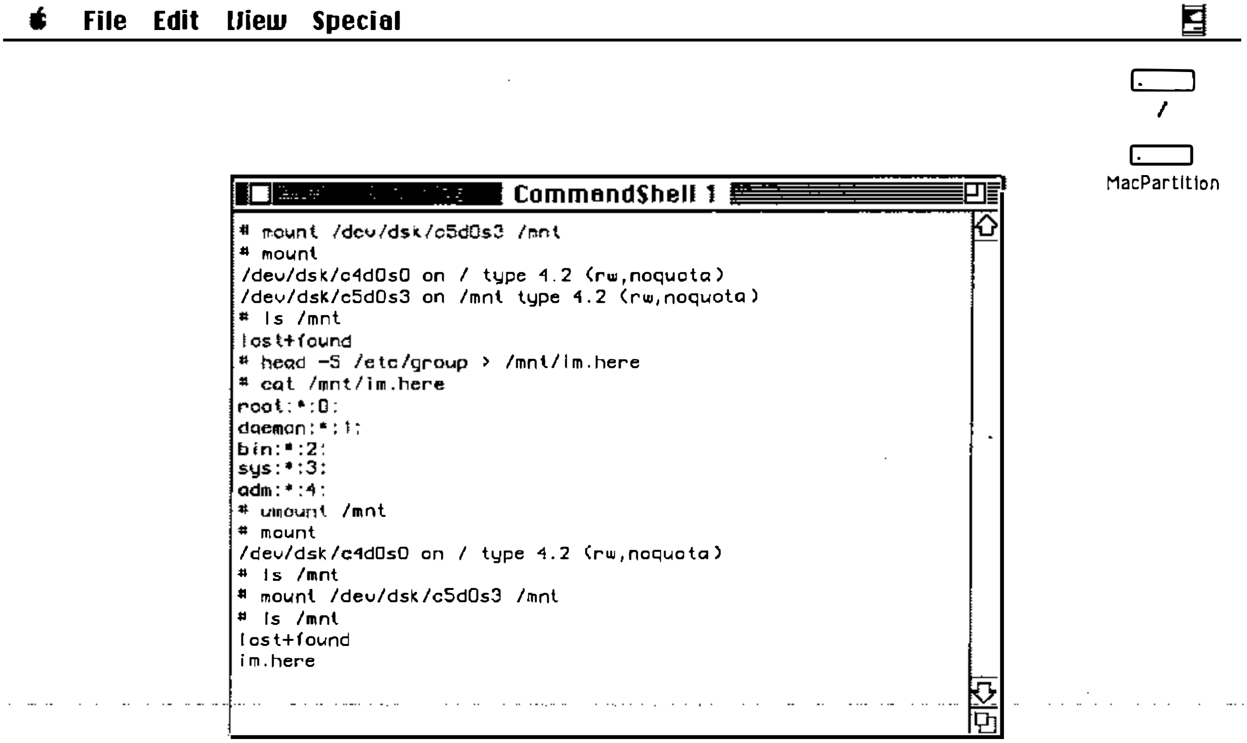
**Step 6** Verify the file system is unmounted with the `mount` command.

**Step 7** Verify the file is no longer accessible.

**Step 8** Remount the file system.

**Step 9** Verify the file is again accessible.

**References** *A/UX Local System Administration*
*mount (1M), touch (1)*



```
# mount /dev/dsk/c5d0s3 /mnt
# mount
/dev/dsk/c4d0s0 on / type 4.2 (rw,noquota)
/dev/dsk/c5d0s3 on /mnt type 4.2 (rw,noquota)
# ls /mnt
lost+found
# head -5 /etc/group > /mnt/im.here
# cat /mnt/im.here
root:*:0:
daemon:*:1:
bin:*:2:
sys:*:3:
adm:*:4:
# umount /mnt
# mount
/dev/dsk/c4d0s0 on / type 4.2 (rw,noquota)
# ls /mnt
# mount /dev/dsk/c5d0s3 /mnt
# ls /mnt
lost+found
im.here
```

# Testing a File System

## Notes

# Automatic Mounts

If you want a file system to mount automatically, put an entry in the file /etc/fstab (fstab stands for File System Table.) The Commando dialog `/etc/fsentry` automates this procedure.

**Format** The format of the file is lines of six fields, delimited by white space:

```
fsname dir type options frequency passno
```

**fsname** This is the device name of the file system to be mounted.

**dir** This is the mount point (directory) at which the file system is to be mounted.

**type** This is the type of file system to be mounted. Valid types are: 4.2 (Berkeley UFS,) 5.2 (AT&T SVFS,) NFS, SWAP, ignore. Ignore is used to show disk partitions not currently in use.

**options** This lists the options for the file system to be mounted. Typical options include: re (read-write,) ro (read-only,) quota (enforce size quotas,) noquota (ignore size quotas.)

**frequency** This is optionally used by `dump.bsd` to determine which file systems need to be dumped.

**passno** This can be used by `fsck` to select which file systems to check.

**Example** The default /etc/fstab file contains the following examples:

```
/dev/dsk/c0d0s0      /      ignore   rw   1 1
rhost_1:filesystem   /mnt   ignore   rw   0 0
```

**Note** For network mounts it is strongly recommended to specify "soft" instead of "hard" mounts, unless the mounted file system is to be changed. If you specify a hard mount and the file system is unavailable for any reason, the system will hang until it becomes available. With a soft mount, your system will continue to run, though the remote file system will not be available.

**References** *A/UX Local System Administration*
*fsentry (4,) fstab (4,) mount (1M,) fsck (1M,) dump.bsd (4)*

# Automatic Mounting

/etc/fstab format:

| | | | | | |
|---|---|---|---|---|---|
| | mount-point | | rw<br>ro<br>quota<br>noquota<br>noauto<br>hard<br>soft | | number |
| fsname | dir | type | opts | freq | passno |
| file system name | | 4.2<br>5.2<br>nfs<br>swap<br>ignore | | number, in days | |

A/UX

18

## Notes

# The dp Command

The HD SC Setup utility will suffice for nearly all disk initialization and partitioning operations. However, HD SC Setup will not suffice for everything. For example, HD SC Setup will not recognize most third-party disk drives. To partition a third-party drive, and in certain other circumstances, you may have to use the A/UX command `dp` (disk partition.) You can execute the `dp` command from within A/UX or from A/UX Startup.

Like HD SC Setup, `dp` allows you to create, change, or delete partitions. The user interface for `dp`, however, is completely different and more difficult than HD SC Setup. It provides all the functionality of HD SC Setup, in addition offering functionality that HD SC Setup does not have:

- `dp` can access disk drives that HD SC Setup will not access.
- `dp` can map slice numbers to partitions.
- `dp` provides a means of changing specific partition attributes (such as name, file system type, starting block, length, etc..)
- It allows you to create custom partitions that are not available in HD SC Setup.

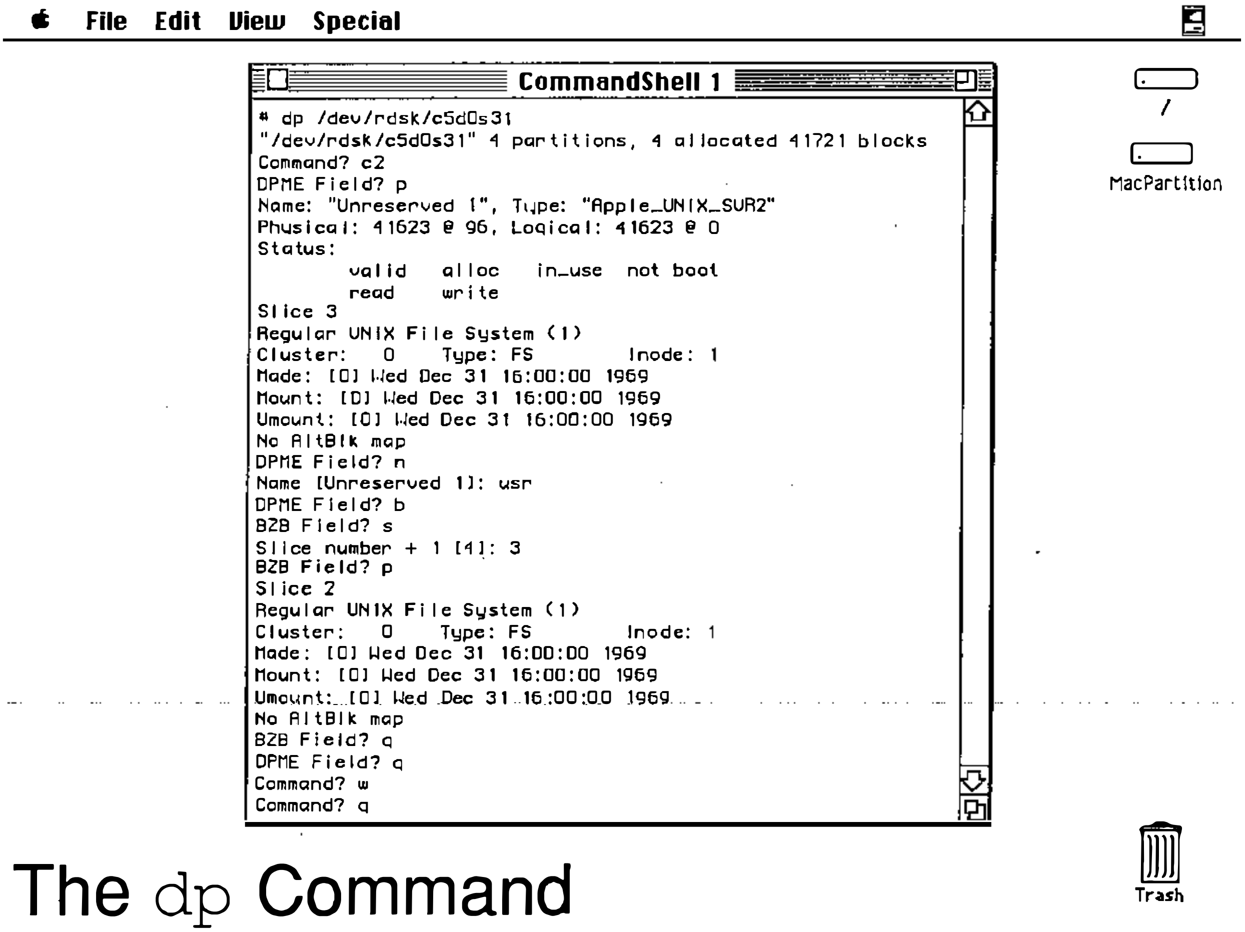
One very useful form of the `dp` command allows you to examine all of the partitions on a disk:

```
echo P | dp -q /dev/rdsk/c5d0s31 | more
```

For more detailed infomation on `dp`, see *A/UX Local System Administration.*

**References**

*A/UX Local System Administration*
*dp (1M,) dpme(4,) bzb(4)*



```
# dp /dev/rdsk/c5d0s31
"/dev/rdsk/c5d0s31" 4 partitions, 4 allocated 41721 blocks
Command? c2
DPME Field? p
Name: "Unreserved 1", Type: "Apple_UNIX_SVR2"
Physical: 41623 @ 96, Logical: 41623 @ 0
Status:
        valid   alloc   in_use  not boot
        read    write
Slice 3
Regular UNIX File System (1)
Cluster:   0    Type: FS        Inode: 1
Made: [0] Wed Dec 31 16:00:00 1969
Mount: [0] Wed Dec 31 16:00:00 1969
Umount: [0] Wed Dec 31 16:00:00 1969
No AltBlk map
DPME Field? n
Name [Unreserved 1]: usr
DPME Field? b
BZB Field? s
Slice number + 1 [4]: 3
BZB Field? p
Slice 2
Regular UNIX File System (1)
Cluster:   0    Type: FS        Inode: 1
Made: [0] Wed Dec 31 16:00:00 1969
Mount: [0] Wed Dec 31 16:00:00 1969
Umount: [0] Wed Dec 31 16:00:00 1969
No AltBlk map
BZB Field? q
DPME Field? q
Command? w
Command? q
```

# The dp Command

## Notes

# File System Size Problems

Files have a way of growing to fill all the available space on a disk. Sooner or later, you will run across the `file system full` error message (or it will run across you.) You will have several relatively simple options at this point, given that you have taken one simple precaution:

**Work as a regular user**

It is important to work as a regular user rather than the superuser. The error message is issued as a warning to regular users when the file system is 95 percent full (90% fir other file systems.) If you are the superuser you get the message when the file system is 100 percent full. By then, solving the problem is significantly more complex.

**Note**

At any point you can issue the `df` command, which tells you the amount of free space left on any given file system.

Once you or one of your users get the `file system full` message, there are several things that can be done:

**Identify archivable files**

`du` can give you a report of the number of blocks occupied by each directory to help you locate "disk hogs," then `ls -l` can be used to identify individual files to be archived. `find` can search for files older than a certain time, or larger than a certain size. Once likely files have been identified, they can be backed up, then removed or compressed.

**File removal**

Usually files named "core" can be removed immediately. Any file in /dev that is not a special file is probably there from a mistaken redirection (these can be quite large.) Similarly, any files appearing in /mnt after a `umount /mnt` command are likely there by mistake. Log files, such as those created by the system accounting procedures, must be pruned regularly.

**File compression**

Infrequently used files can be compressed and easily expanded when the need arises. The command `compress` does this.

**References**

*A/UX Local System Administration*
*df (1,) du (1,) find (1,) pack (1,) compress (1)*

# File System Size Problems

- ❑ 90% of space in UFS is available to a regular user
  - superuser can access 100%
- ❑ File identification
  - the `du` command
  - the `ls -l` command
  - the `find` command
- ❑ File removal
  - extraneous files
  - system accounting files
- ❑ File compression
  - infrequently used files
  - the `compress` command

20

## Notes

# Checkpoints

1 What two utilities are available for initializing (formatting) a disk drive?

2 Which initialization/formatting utility should you use if you want a Macintosh partition of any size?

3 What two utilities are available for partitioning a disk drive?

4 What command would you use to make a Berkeley (UFS) file system on `/dev/dsk/c5d0s0` if the size of the partition is 14816.5 blocks?

5 How would you make a new file system available for use in A/UX?

6 List three advantages and one disadvantage of using `dp` instead of HD SC Setup to create or change partitions on an A/UX disk.

# Exercises

In these exercises, you will use HD SC Setup to initialize and partition a 20 MB HD SC disk drive. Then you will use A/UX commands to change the partition's file system type, to make and check a file system, then mount and use the auxiliary A/UX file space.

## Exercise 1

**Step 1** Power up the A/UX system.

**Step 2** Double-click on the HD SC Setup icon located in your MacPartition folder.

**Step 3** Click the Drive button to select your external 20 MB drive (SCSI Device 5 in this example.) Then click the Partition button:

**Step 4** Select Maximum Free A/UX from the list of partitioning choices.

**Step 5** Click on the OK button when you see the warning dialog box.

**Step 6** Click on the Partition button again after you see the message "Partitioning successfully completed."

**Step 7** Click on the Custom button.

**Step 8** Examine the partition map. Notice that the disk has been allocated in its entirety to A/UX.

**Step 9** Click on the Details button. Notice that the A/UX partition is already allocated to slice 3.

**Step 10** Click on the OK button when you are done examining the details, then quit from the HD SC Setup application.

**Exercise 2** The main purpose of this exercise is to familiarize you to the dp command. dp offers a lot more power and flexibility than can be demonstrated in a single exercise; refer to the A/UX documentation for more information on dp.

**Step 1** Launch A/UX by double-clicking on the A/UX Startup icon.

**Step 2** Log on as root at the dialog box.

**Step 3** Enter the command to use the dp utility, specifying the disk you just formatted. dp responds with a prompt: `Command?`, telling you it is ready for your input.

(continued on the next page)

**Step 4**

You will now enter the series of commands below to the “Unreserved 1” partition into a “usr” partition.

Enter the commands below exactly as shown in bold in the left column below. The Notes column will explain what's happening. If you make a mistake, you can quit from dp by entering CONTROL-C and begin again from Step 1.

### dp Command Prompts and Your Responses (in bold)     Notes

```
Command? p2                          Examine partition 2
DPM Index: 2
Name: "Unreserved 1", Type: "Apple_UNIX_SVR2"
Physical: 41623 @ 96, Logical: 41623 @ 0
Status:
        valid   alloc   in_use  not boot
        read    write
Slice 3
Regular UNIX File System (1)
Cluster:    0    Type: FS        Inode: 1
Made: [0] Wed Dec 31 16:00:00 1969
Mount: [0] Wed Dec 31 16:00:00 1969
Umount: [0] Wed Dec 31 16:00:00 1969
No AltBlk map
Command? c2                          Change partition 2.
DPME Field? n                        Change name field...
Name [Unreserved 1]: usr             ...to make it easily recognizable.
DPME Field? b                        Next, adjust the block zero block.
BZB Field? s                         Change the slice number.
Slice number + 1 [4]: 3              Since we want slice 2, we enter 3.
BZB Field? u                         Specify this is a usr file system...
Usr file system? y                   ...and confirm it..
BZB Field? q                         Quit from the block zero field sequence.
DPME Field? q                        Quit from the DPME field sequence.
Command? p2                          Print info about the new partition 0
DPM Index: 2
Name: "usr", Type: "Apple_UNIX_SVR2"
Physical: 41623 @ 96, Logical: 41623 @ 0
Status:
        valid   alloc   in_use  not boot
        read    write
Slice 2
Regular UNIX File System (1)
Cluster:    0    Type: UFS        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
Command? wq                          Write changes and quit dp.
```

**Exercise 3** This exercise takes you through the steps of making, checking, and mounting the file system.

**Step 1** Enter:the following command to make a new file system on the external disk:

```
cmdo newfs
```

Once the dialog box comes up, select the Disk Type matching the drive you are adding. Click on the Choose device file button. Select the file /dev/rdsk/c5d0s2. Click on the OK button. When the original dialog box comes back, click on the verbose mode check box, then click on the newfs button.

**Step 2** Check the file system by entering the command:

```
cmdo fsck
```

Once the dialog box comes up, click on the Choose file system(s) button. Select the file /dev/rdsk/c5d0s2. Click on the Add button, then click on the Done button. When the original dialog box comes back, click on the fsck button.

**Step 3** Make a directory to use as a mount point.

**Step 4** Mount the external file system to your new directory.

**Exercise 4** In this exercise you test that the file system is accessible.

**Step 1** Check that you have successfully mounted the new file system on the external drive.

**Step 2** Create a test file on your new file system.

**Step 3** Verify you can read back the test file.

**Step 4** Unmount the new file system.

**Step 5** Verify you cannot access the test file.

**Exercise 5** In this exercise you set the system to automatically mount the new file system and again test its accessibility.

**Step 1** Enter the command:

```
cmdo fsentry
```

to have A/UX automatically mount the file system. Click on the Choose device file button once the dialog box comes up. Select the file /dev/dsk/c5d0s3, then click on the OK button. When the original dialog box comes back, click on the Choose mount point button. Select the file /newusr, then click on the Directory button. When the original dialog box comes back, click on the fsentry button.

**Note** The `fsentry` program will create the mount-point directory uf ut diesn't already exist, and will try to mount the new firle system (unless the -n option has been used.)

**Step 2** Verify you can access the test file.

**Step 3** Remove the HD SC Setup program from your MacPartition.

**Step 4** Shut down your system.

# Module 5
# Back Up and Restore

# Table of Contents

# Backups

A/UX® 2.0

## Notes

# Backup and Restore

**Goal**

In this module, you will learn how to back up crucial system files to floppy disk and restore them.

**Objectives**

By the end of this module, you will be able to:

- describe the media and utilities available for backing up and restoring data and programs
- develop a backup strategy
- explain how A/UX addresses various backup devices
- use the `tar`, `cpio`, and `pax` commands to back up and restore data to floppy disk and tape
- set up A/UX for the Apple 40 SC Tape Drive
- use `dump.bsd` and `restore` for backups to floppy disk and tape
- use the Apple 40 SC Tape Backup program to perform partition backups to tape

**Activities**

Lecture/Discussion/Checkpoints/Exercises

# Objectives

- ❑ Describe the media and utilities for backup and restore
- ❑ Develop a backup strategy
- ❑ Explain how A/UX addresses various backup devices
- ❑ Use `tar`, `cpio`, and `pax`
- ❑ Set up A/UX for Apple Tape Drive
- ❑ Use `dump.bsd`and `restore`
- ❑ Use the Apple 40 SC Tape Backup

2

## Notes

# Backing Up Your System

Backing up files and file systems is one of the most important aspects of A/UX system administration. You can back up data and programs to the following types of media:

- 3.5-inch floppy disks
- Hard disks
- Apple Tape Drive SC cartridge tapes
- 9-track tape

This module will cover backing up to floppy disks and Apple cartridge tapes. The commands for backing up to a 9-track tape are identical to those used for floppy disks and tape cartridges; only some of the options are different.

The following commands will be covered in this module:

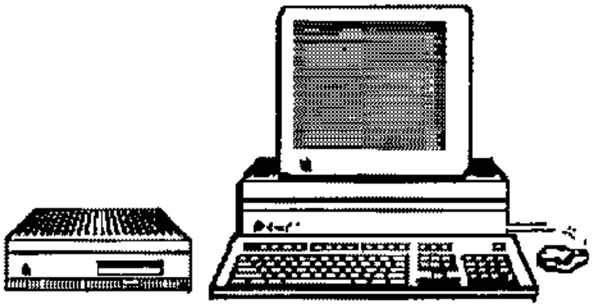
| | |
|---|---|
| `cpio` | Copy Input to Output |
| `tar` | Tape archiver |
| `pax` | Archive in IEEE format |
| `dump.bsd` | Incremental or full network system archive. (This is the same command as `dump` in strict BSD UNIX systems.) |
| `restore` | Restore files or file systems |

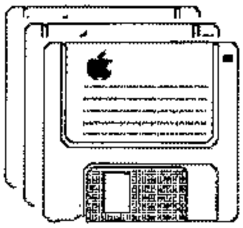
Files archived with `tar` must be restored with `tar`, and `cpio` backups must be restored with `cpio`. The same is true for `dump.bsd` and `restore`. `pax`, however, these can read and write both `tar` and `cpio` formats.

**References** *A/UX Local System Administration*

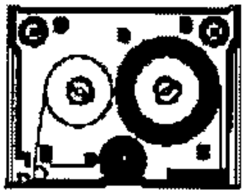
# Backing Up

Backups can be made to:





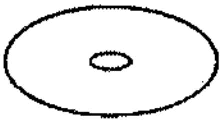
Floppy or
hard disks



Cartridge
or
reel-to-reel
tapes

Backup Utilities:

```
   tar
   cpio
   pax
   dump.bsd
   restore
   Apple 40SC Tape Backup
```



Optical
disks

3

## Notes

# Developing a Backup Strategy

Your backup strategy should include full and incremental system backups. A full backup means copying all the files and programs on your system to a backup medium. An incremental backup means to back up only the files modified since the last backup. The following is a recommended strategy for full and incremental backups:

- Daily backups of user's files, performed by each user.
- Weekly backup of user files by the root user.
- Monthly backups of important system files.
- Quarterly backups of everything.

Always label your backup medium with the date and type of backup. Keep your backups in a safe place, preferably away from the immediate area of your system in case of fire, theft, or vandalism.

**Which medium to use?**

The type of medium—disk, or tape—depends basically on how much material you have to back up and how much time (and money) you want to spend backing up your data, making the choice of media a matter of judgement. For daily incremental backups, using floppy disks may be preferable because of the speed and ease of handling floppy disks. For larger system-wide backups, tape may be the best choice. Note that a full A/UX system backup will require at least two formatted cartridge tapes.

**References**

*A/UX Local System Administration*

# Backup Strategy

□ Daily backup of user's files

□ Weekly backup of user's files

□ Monthly backup of important system files

□ Quarterly backup of everything

4

## Notes

# Specifying the Backup Device

The following sections describe the commands for backing up and retrieving data to floppy disk, disk, or tape: `cpio`, `tar`, `pax`; `dump.bsd`, and`restore`. When you use these commands, you will use the following naming conventions:

| | |
|---|---|
| Internal floppy disk drive | `/dev/rfloppy0` or<br>`/dev/rdsk/c8d0s0` |
| Apple Tape Drive SC | `/dev/rmt/tc`*n* (*n* is the SCSI address of the tape drive) |
| Internal hard disk drive | `/dev/rdsk/c0d0s0` |
| External hard disk drive | `/dev/rdsk/c`*n*`d`*m*`s`*y* (*n* is the SCSI address of the disk drive) |

The "`r`" in `rmt`, `rfloppy0`, and `rdsk` stands for "raw," or character-by-character device, as opposed to a "cooked" or block device. When backing files up it is reasonable to back up to a raw device. Raw devices bypass the buffer cache and tend to work faster.

**Formatting floppy disks**

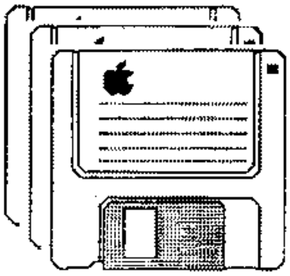
Even though you will use the "raw" device specification when copying to floppy disks, you still need to format floppies before copying to them. A/UX asks you how you want to treat a disk when you first insert it into the machine and formats it appropriately. To eject a disk, enter `eject 0`.

**Reference**

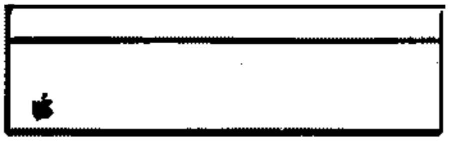
*A/UX Local System Administration*

# Specifying Backup Devices

Apple 40 SC Tape Drive
/dev/rmt/tc*n*



Floppy Disk Drive
/dev/rfloppy0 or
/dev/rdsk/c8d0s0



Hard Disk Drive
/dev/rdsk/c*n*d*m*s*y*

*n* = SCSI ID number

5

## Notes

# mt—Tape Drive Controller Command

The `mt` command allows you to control your external tape drive. In addition to allowing the use of low-level functions, such as spacing forward a certain number of files or rewinding the tape, this command formats tapes and can report the current tape drive status.

### Syntax

```
mt [-f device] command [count]
```

By convention the *device* is referred to by the special device name `/dev/rmt/tc`*n*, where*n* is the SCSI ID of the tape unit.

### Note

In the absence of a *device*, the command looks for an environment variable, TAPE, containing the name of the tape device. The assignment can be placed in .profile or .login as appropriate. If no device is specified and the TAPE variable is not set, the command will fail.

### Formatting a Tape

To format a blank tape, insert the tape and enter:

```
# mt -f /dev/rmt/tc1 format
```

It takes about 37 minutes to format a tape.

### Status

To check the status of a tape drive, enter:

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

### Hint

By assigning an environment variable, TAPE, you can omit the full device specification in certain commands, `mt` included. For example, after making the assignment:

```
TAPE="/dev/rmt/tc1" ; export TAPE
```

the `mt` command to format a tape and check status would be:

```
mt format; mt status
```

# The mt Command

## Syntax

```
mt [-f device] command [count]
```

## Example

```
# mt -f /dev/rmt/tc1 status
Apple tc40 tape drive
total  4836 blocks (39616512 bytes)
avail this cartridge
Last I/O was at blk 0
driver version 1.40
```

6

## Notes

# Tape Capacity

The Apple 40 SC Tape Drive only reads and writes fixed 8 KB blocks (8,192 bytes.) Thus, the driver restricts raw I/O size to a multiple of 8 KB. Reading and writing many 8 KB blocks minimizes user-process overhead and maximizes streaming, but it requires that you think ahead before backing up a large amount of data to a tape with `tar`, `cpio`, or `pax`.

Because some of the backup programs do not handle the end-of-tape (EOT) condition gracefully, you need to make a conversion to find out how much tape capacity will be used in a particular backup. Expressing the output of `du` and `mt status` in terms of 512-byte blocks is a convenient means of determining the tape requirement.

**Example**

Before we back up the files, we want to know how much space we have. To find the amount space on a tape:

**Step 1**

```
# mt status
4836 blocks
```

The tape is blocked in 8 KB blocks, which is equivalent to 16 512-byte blocks. So, multiplying 4836 by 16 gives you the number of 512-byte blocks available on tape: 77,376.

**Step 2**

If, for example, you want to backup the /etc directory, you can find its size with the `du` command:

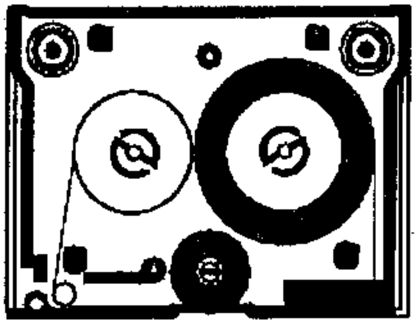
```
# du -s /etc
12702
```

The /etc directory contains 12,702 blocks where each block contains 512 bytes.

**Step 3**

Comparing the two numbers, 12,000 is much less than 77,000. There's plenty of room.

# Tape Capacity



## Will it fit?

Convert results of `mt status` to 512-byte blocks (multiply by 16).

Compare results with `du` output.

```
# mt status
Apple tc40 tape drive
total  4836 blocks ——► 4836*16 = 77376
(16 .5KB blocks per 8KB block)
# du -s /etc
12702   /etc ——————————► 12702 < 77376
(.5KB blocks)
```

7

## Notes

# tar—Tape Archive

The A/UX utility tar stands for tape archive. The tar command is a handy utility to use when you want to move files between BSD and System V based systems. Of course, both environments must have the tar utility.

While tar is very reliable for transferring data between different environments, it is also the slowest way to back up and retrieve data. Further, if there is any chance that the files to be backed up are larger than the backup medium, you must specify the maximum number of blocks the medium can hold. If you do not and a copy runs off the end of the tape or fills up the floppy disk, you will have to start the entire backup again.

You can use tar or pax to recover files archived by tar.

**Syntax**

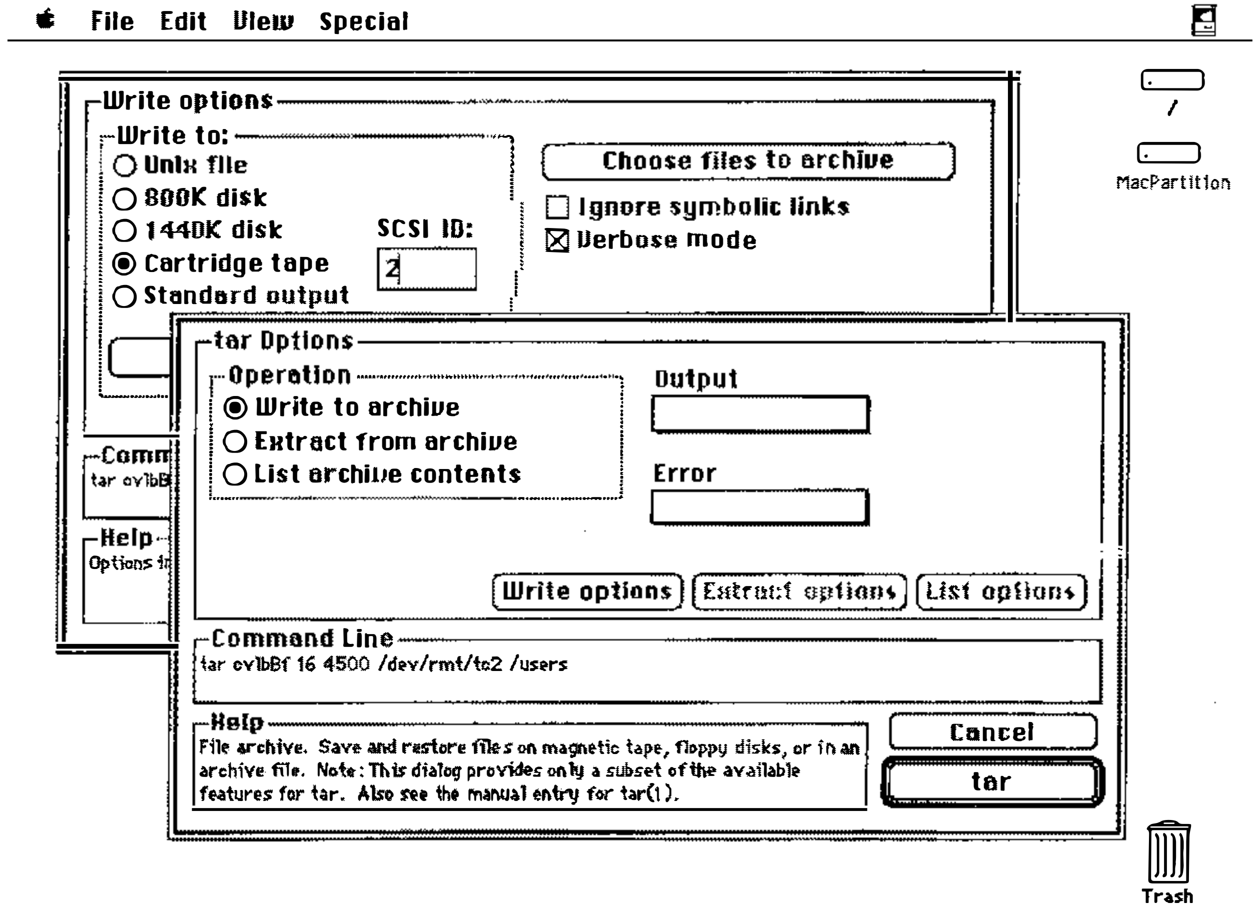
```
tar [key] [file ...]
```

**Note**

tar cannot archive special files or empty directories. The current limit on filenames is 100 characters. The Apple tape drive requires that the blocking factor 16 be used for both reads and writes.

**Reference**

*A/UX Local System Administration*
*tar (1)*



Write options

Write to:

○ Unix file
○ 800K disk
○ 1440K disk
● Cartridge tape
○ Standard output

SCSI ID: 2

□ Ignore symbolic links
☒ Verbose mode

tar Options

Operation

● Write to archive
○ Extract from archive
○ List archive contents

Output

Error

Command Line

```
tar cvlbBf 16 4500 /dev/rmt/tc2 /users
```

Help

File archive. Save and restore files on magnetic tape, floppy disks, or in an archive file. Note: This dialog provides only a subset of the available features for tar. Also see the manual entry for tar(1).

# Notes

# tar—Tape Archive

**Example 1** To copy individual files to floppy disk:

```
# tar cfv /dev/rfloppy0 memo1 memo2 letter*
```

**Example 2** To retrieve files or directories, use the xfv option:

```
# tar xfv /dev/rfloppy0 memo2
```

**Example 3** To copy directories:

```
# tar cbfv 16 /dev/rmt/tc1 /users/start
```

**Example 4** To copy directories where you think it will take more than one tape:

```
# tar cbBfv 16 4500 /dev/rmt/tc1 /
```

**Example 5** To list the contents contents, use the tfv option:

```
# tar tfv /dev/rfloppy0
```

**Example 6** To retrieve the entire contents, use the xfv option without specifying any files:

```
# tar xfv /dev/rfloppy0
```

**Reference** *A/UX Local System Administration*
*tar (1)*

# tar Examples

## Syntax

```
tar [key] [file...]
```

## Example

```
# tar cfv /dev/rfd0 memo1 memo2 letter*
# tar xfv /dev/rfloppy0 memo2
# tar cbfv 16 /dev/rmt/tc1 /users/student
# tar cbBfv 16 4500 /dev/rmt/tc1 /
# tar tfv /dev/rmt/tc1
# tar xfv /dev/rmt/tc1
```

9

## Notes

# cpio—Copy Input to Output

The A/UX utility cpio stands for copy I/O. You do not use file names as arguments to cpio as you do with tar. Instead, cpio usually takes its input from other utilities, such as ls or find, and outputs the data to stdout, which is normally redirected to disk or tape devices.

**Syntax**

```
list_output | cpio -o[vB] > /dev/rfloppy0
list_output | cpio -o[vB] | tcb > /dev/rmt/tcn
```

The cpio command has several advantages over tar. When retrieving directories, it rebuilds the directory for you while maintaining permissions, ownerships, and groups. It can also copy device files, which tar cannot. cpio is usually faster than tar. If more than one disk or tape is required, cpio will prompt for a new one. There is, however, no way to specify a blocking factor in cpio. Consequently, any time we are using the 40SC tape drive, the data stream must go through a filter. The filter tcb, which simply reads stdin and writes 8K blocks to stdout, is provided for use with the tape drive.

The following are some common examples:

**Example 1** To copy an individual file with cpio:

```
# ls letter1 | cpio -ov > /dev/rfloppy0
```

**Example 2** To copy all the files in a directory tree:

```
# find /users -print | cpio -ov > dev/rfloppy0
```

**Example 3** To copy files modified in the last day:

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

**Example 4** To copy files belonging to a particular user:

```
# find / -user bill -print | cpio -ov > dev/rfloppy0
```

**References** *A/UX Local System Administration*
*cpio (1,) find (1,) tar (1)*

# The cpio Output Command

## Syntax

```
cpio -o[options]
list_output | cpio -o[vB] > /dev/rfloppy0
list_output | cpio -o[vB] | tcb > /dev/rmt/tc n
```

## Example

```
# ls letter1 | cpio -ov > /dev/rfloppy0
# find /users -print | cpio -ov > dev/rfloppy0
# find / -mtime -1 -print | cpio -ov > dev/rfloppy0
# find / -user bill -print | cpio -ov > dev/rfloppy0
# find / -print | cpio -ov > | tcb > dev/rmt/tc1
```

10

## Notes

# Retrieving Files with cpio

The syntax for getting files back off of archive media is both more and less convoluted than backing the data up (less in the case of floppies, more in the case of tape). To retrieve files:

**Floppy Syntax**

```
cpio -i[tvdmu] [pattern] < /dev/rfloppy0
```

**Tape Syntax**

```
tcb < /dev/rmt/tcn | cpio -i[tvdmu] [pattern]
```

In the command above, `pattern` is formed as in filename expansion. If you specify a directory, the entire directory (including subdirectories) will be retrieved. If no `pattern` is specified, the entire contents of the backup media will be extracted.

**Example 1**

To view a "table of contents" (a list of files) on the floppy disk:

```
# cpio -it < /dev/rfloppy0
```

**Example 2**

To retrieve a file:

```
# cpio -ivm letter1 < /dev/rfloppy0
```

**Example 3**

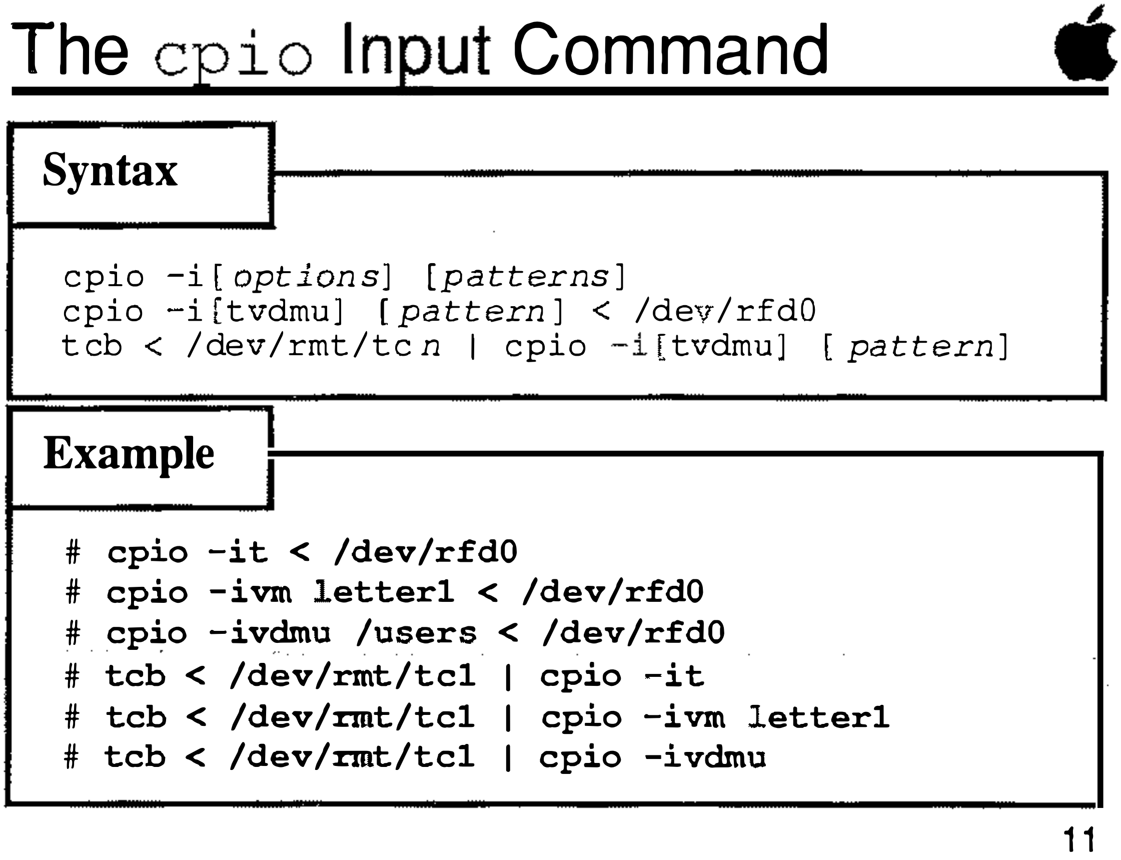
To recover all of the files on a tape:

```
# tcb < /dev/rmt/tc2 | cpio -ivdmu
```

In the commands above, the options mean:

`-i` Extract files from floppy disk (or tape)
`-t` Print a list of files on the floppy disk (or tape)
`-v` Print the file name on your screen after it has been extracted
`-d` Create any directories needed
`-m` Preserve the file modification date
`-u` Extract unconditionally (older versions replace newer)

It is helpful to review the on-line manual pages to learn more about the options available with `cpio`.



# The cpio Input Command

## Syntax

```
cpio -i[options] [patterns]
cpio -i[tvdmu] [pattern] < /dev/rfd0
tcb < /dev/rmt/tcn | cpio -i[tvdmu] [pattern]
```

## Example

```
# cpio -it < /dev/rfd0
# cpio -ivm letter1 < /dev/rfd0
# cpio -ivdmu /users < /dev/rfd0
# tcb < /dev/rmt/tc1 | cpio -it
# tcb < /dev/rmt/tc1 | cpio -ivm letter1
# tcb < /dev/rmt/tc1 | cpio -ivdmu
```

11

## Notes

# pax—POSIX archive extender

pax reads and writes archive files that conform to the "Archive/Interchange File Format" specified in IEEE Standard 1003.1-1988. pax also supports traditional cpio and tar interfaces.

### Syntax

```
pax [-uv] [-t device] [pattern]
pax -r [-uv] [-t device] [pattern]
pax -w [-auv] [-b blocking] [-t device] [-x
format] [pathname]
pax -rw [-uv] [pathname]...  directory
```

This partial list of options have the following meanings:

### Options

| | |
|---|---|
| b | specify blocking factor |
| t | interactive, query about disposition |
| t | specify archive device |
| u | update, new files are not overwritten |
| v | verbose, displays a list of files similar to ls -1 |
| x | specify output format |

### Example 1

To copy individual files to floppy disk:

```
# pax -w -t /dev/rfloppy0 score*
```

### Example 2

To copy directories:

```
# pax -w -b 8k -t /dev/rmt/tc2 /users
```

### Example 3

To examine a list of the contents:

```
# pax -t /dev/rfloppy0
```

### Example 4

To retrieve a file (or directory):

```
# pax -r -b 8k -t /dev/rmt/tc2 ./student/.memo
```

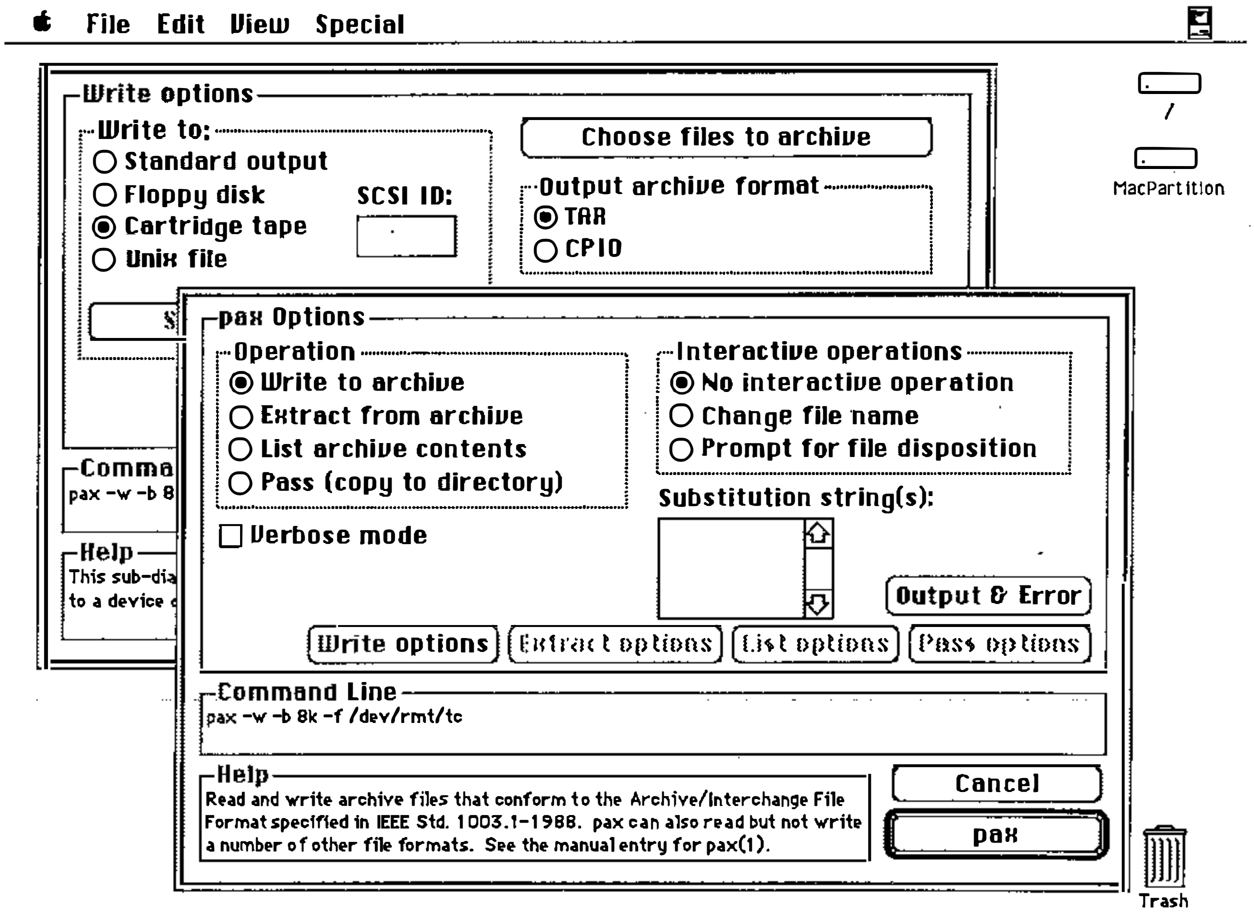
### Example 5

To retrieve the entire contents of an archive:

```
# pax -r -b 8k -t /dev/rmt/tc2
```

### Reference

*pax (1)*



Write options

Write to:
○ Standard output
○ Floppy disk
● Cartridge tape
○ Unix file

SCSI ID:

Output archive format
● TAR
○ CPIO

Comma...
pax -w -b 8...

Help
This sub-dia...
to a device ...

pax Options

Operation
● Write to archive
○ Extract from archive
○ List archive contents
○ Pass (copy to directory)

□ Verbose mode

Interactive operations
● No interactive operation
○ Change file name
○ Prompt for file disposition

Substitution string(s):

Command Line

```
pax -w -b 8k -f /dev/rmt/tc
```

Help
Read and write archive files that conform to the Archive/Interchange File Format specified in IEEE Std. 1003.1-1988. pax can also read but not write a number of other file formats. See the manual entry for pax(1).

# Notes

# dump.bsd and restore

The commands dump.bsd andrestore are particularly useful for doing backups of multi user, multidisk systems. Note that the dump.bsd command is commonly known as dumpfs in other systems. You must use the restore command to retrieve files archived with dump.bsd.

When using dump.bsd, you must specify a dump level from 0 to 9. A dump level of 0 indicates a full system-wide backup. A dump level of 1 to 9 means to back up only those files that were modified since the last backup at a lower dump level. For example, a dump level of 3 would back up all of the files modified since the last level 2 backup was made. The on-line manual page for dump.bsd includes a plan that can serve as a backup plan for your system.

**Note**

Before using dump.bsd, the file/etc/dumpdates must exist. Before running dump.bsd the first time, enter the command:

```
touch /etc/dumpdates
```

This simply creates a zero-length file, or updates the modification time if the file already exists. The first time you run dump.bsd.is the only time you will need to use the touch command.

**References**

*A/UX Local System Administration*
*dump.bsd(1)*

# Dump Levels

0 Do a full system-wide archive

1-9 Archive only those files modified since the last archive of a lower level

13

## Notes

# Backing Up with dump.bsd

The basic format of the `dump.bsd` command is:

### Syntax

```
dump.bsd [ncfuF] [argument ...] [filesystem]
```

The options have the following meanings:

### Options

*n* do a level *n* dump of the system
`c` write to the Apple 40SC tape drive (blocking factor and tape length are set automatically)
`f` write to the specified device
`u` write the date of the beginning of the dump in the file /etc/dumpdates
`F` write to the 800KB 3.5-inch disk drive (blocking factor and maximum number of blocks are set automatically)

When the selected media becomes full, you will be prompted to insert another. As always, backups should be carefully labelled.

If no arguments are given, the options are assumed to be 9u, and a default file system is dumped to the default tape. The default file system is the root file system ( /.) and the default tape is /dev/tape.

### Note

`dump.bsd` must be run on unmounted file systems to insure data integrity.

### Backing up to another disk

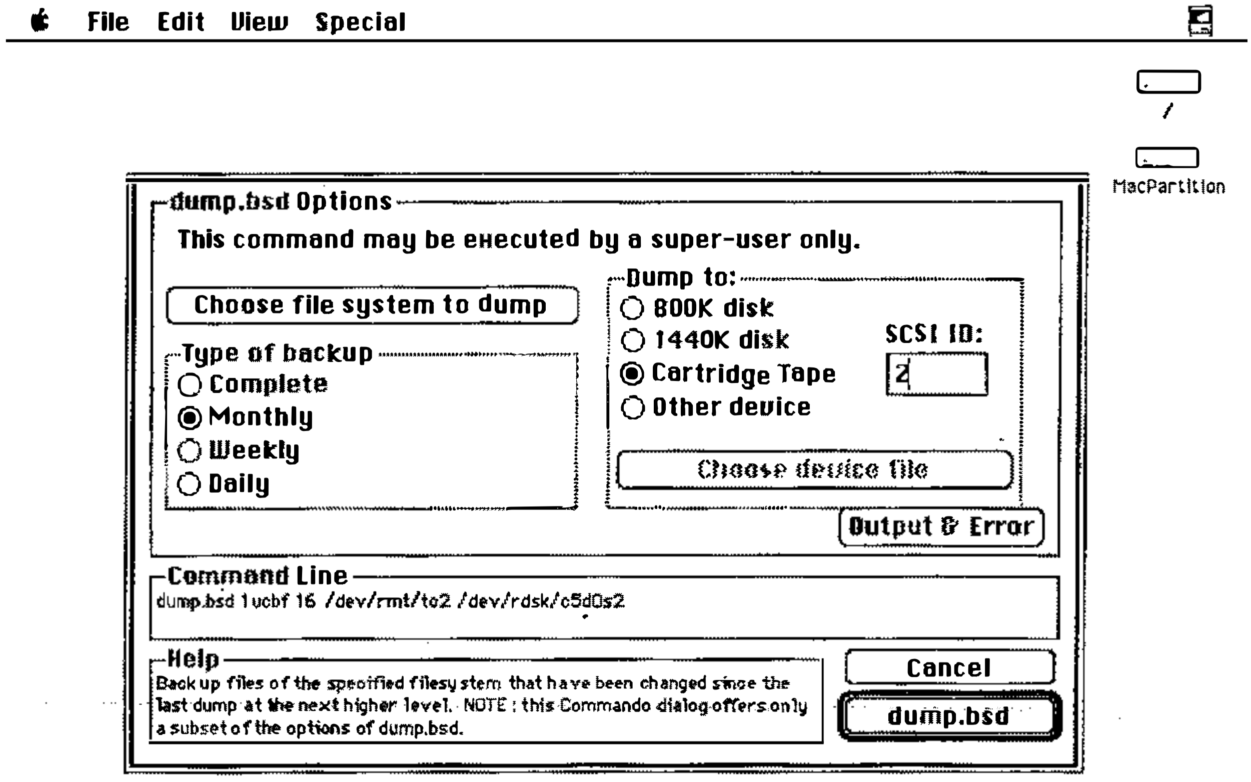
The `f` option allows you to specify another backup device. For example, to back up your root file system (`dev/rdsk/c0d0s0`) to an external disk drive at SCSI ID 0, you would enter the command:

```
# dump.bsd 0uf /dev/rdsk/c5d0s0 /dev/rdsk/c0d0s0
```

Note that you enter the target drive *before* specifying the file system to be backed up.

### References

*A/UX Local System Administration*
*dump.bsd (1)*



dump.bsd Options

This command may be executed by a super-user only.

Type of backup

○ Complete
● Monthly
○ Weekly
○ Daily

Dump to:

○ 800K disk
○ 1440K disk
● Cartridge Tape
○ Other device

SCSI ID: 2

Command Line

```
dump.bsd 1ucbf 16 /dev/rmt/tc2 /dev/rdsk/c5d0s2
```

Help

Back up files of the specified filesystem that have been changed since the last dump at the next higher level. NOTE : this Commando dialog offers only a subset of the options of dump.bsd.

# The `dump.bsd` Command

## Notes

# Using restore

The basic format for `restore` is:

**Syntax**

```
restore [rxtifF] [argument] [filename ...]
```

The options have the following meanings:

**Options**

| | |
|---|---|
| `r` | read and load backup media into current directory, used for full restorations only. |
| `x` | extract the named files |
| `t` | print a list of files on the tape (or floppy disk) |
| `i` | interactively restore files (see below) |
| `f` | read from the specified device |
| `F` | eject the floppy disk when finished |

Certain options, such as `f`, require an *`argument`* to operate. If *`filename`* (which may also be a directory name) is not specified, the entire contents of the media are acted upon.

**Full restoration**

The `r` option causes an entire `dump.bsd` archive to be restored into the current directory. Be careful with the `r` option; if you start `restore r` from the top of the directory hierarchy, you will replace current files with older versions. Use this option only to restore a complete archive onto an empty file system or to restore an incremental archive after a full level 0 restore.

To extract a specific file or directory, use the `x` option:

**Restoring a file from floppy**

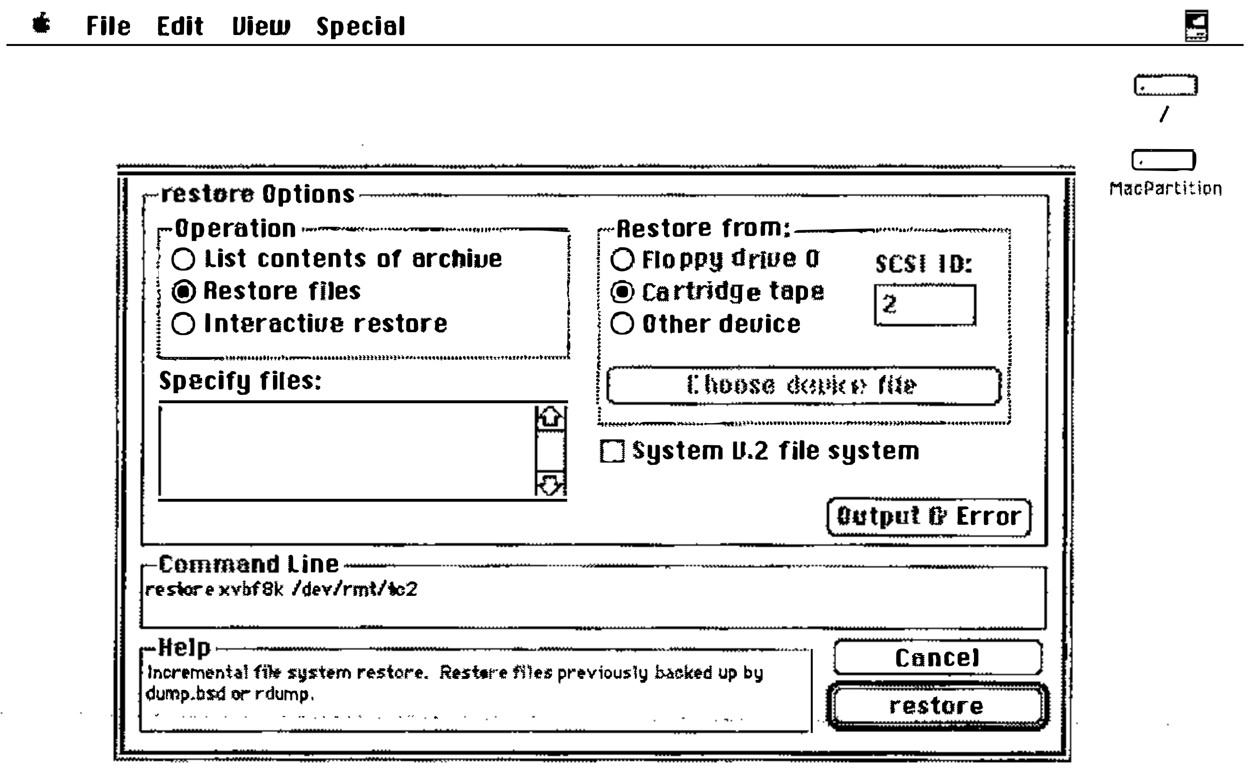
```
# restore xvFf /dev/rfloppy0 letter1
```

If your backup was on floppy disk, you must insert the first floppy disk in the backup series when retrieving files.

**Restoring a directory from tape**

```
# restore xvbf 8k /dev/rmt/tc1 /users
```

If the file is on a multi-volume dump (more than one tape was used to dump the data to,) you will be prompted to enter the volume you want to try and find it on.



restore Options

Operation
○ List contents of archive
● Restore files
○ Interactive restore

Specify files:

Restore from:
○ Floppy drive 0
● Cartridge tape
○ Other device

SCSI ID: 2

□ System V.2 file system

Command Line

```
restore xvbf8k /dev/rmt/tc2
```

Help

Incremental file system restore. Restore files previously backed up by dump.bsd or rdump.

# The `restore` Command

Trash

## Notes

# Interactive restore

### Interactive mode

One of the most useful options of restore is the i option, which allows you to restore files interactively. With this option, restore reads the directory structure stored on the backup media, and lets you select files to restore *as though you had a disk open.*

### Syntax

```
restore i [bfF] [blocking factor] [device]
```

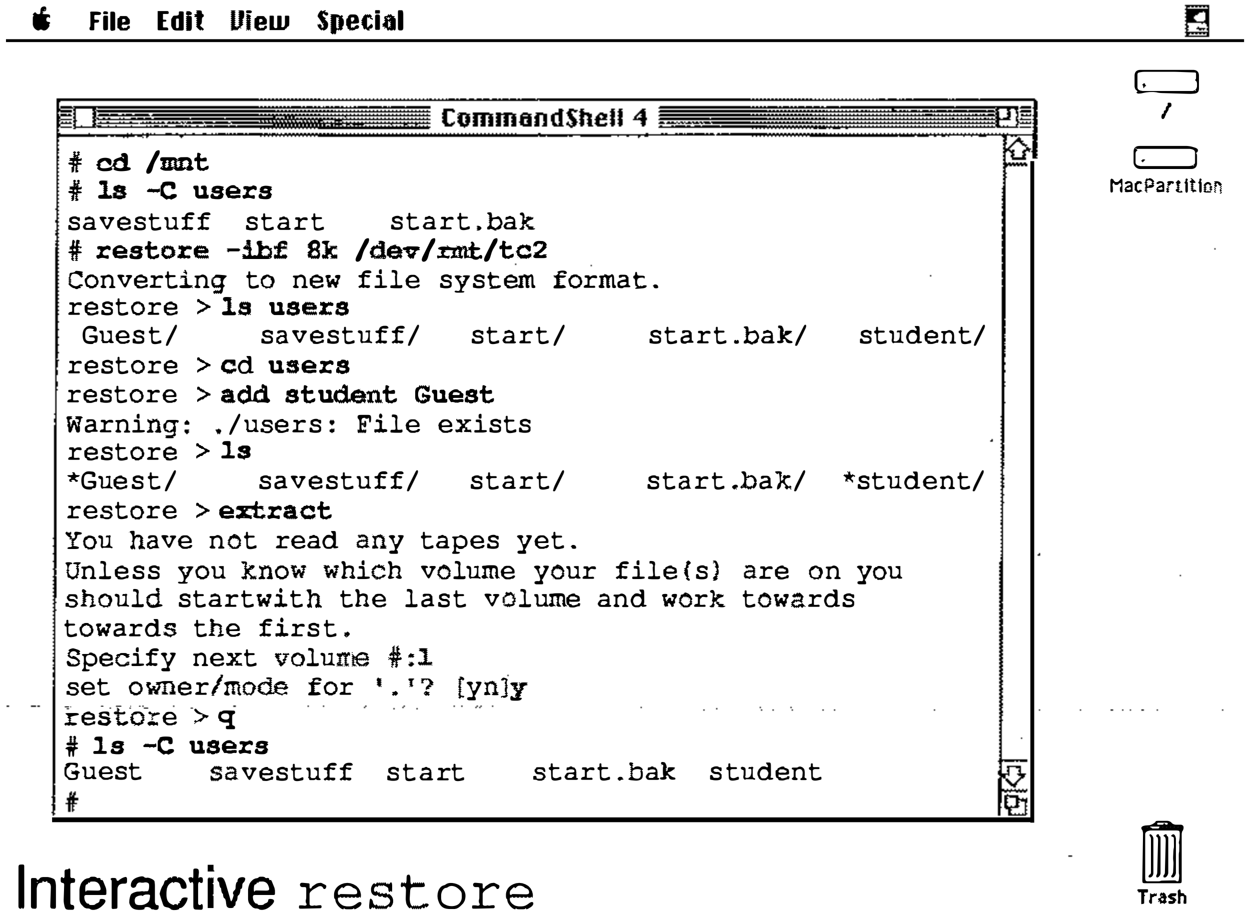
In interactive mode, you will see the prompt `restore >`. The following commands are then available:

| | |
|---|---|
| `ls` | display contents of current media directory. |
| `cd` | changes your current media directory. |
| `pwd` | prints the pathname of the current media directory. |
| `add` | adds entries to the "extraction list." |
| `delete` | deletes entries from the extraction list. |
| `extract` | extracts files. |
| `setmodes` | sets owner, mode and time. |
| `help` | get help with `restore`. |
| `quit` | exit `restore` |

As an example, we'll try and recover the /user/Guest and /user/student directories from a tape backup.

### Example

```
# cd /mnt
# ls -C users
savestuff  start  start.bak              (this ls is of the disk)
# restore -ibf 8k /dev/rmt/tc2
Converting to new file system format.
restore > ls users                       (this ls is of the tape)
 Guest/   savestuff/   start/   start.bak/   student/
restore > cd users                       (this cd is on the tape)
restore > add student Guest
Warning: ./users: File exists
restore > ls                             (this ls is of the tape)
*Guest/   savestuff/   start/   start.bak/  *student/
restore > extract
You have not read any tapes yet.
Unless you know which volume your file(s) are on you
should start with the last volume and work towards
towards the first.
Specify next volume #: 1
set owner/mode for '.'? [yn] y
restore > q
# ls -C users                            (this ls is of the disk)
Guest  savestuff  start  start.bak  student
```



```
# cd /mnt
# ls -C users
savestuff  start    start.bak
# restore -ibf 8k /dev/rmt/tc2
Converting to new file system format.
restore > ls users
 Guest/     savestuff/   start/     start.bak/   student/
restore > cd users
restore > add student Guest
Warning: ./users: File exists
restore > ls
*Guest/     savestuff/   start/     start.bak/  *student/
restore > extract
You have not read any tapes yet.
Unless you know which volume your file(s) are on you
should startwith the last volume and work towards
towards the first.
Specify next volume #:1
set owner/mode for '.'? [yn]y
restore > q
# ls -C users
Guest    savestuff  start    start.bak  student
#
```

# Interactive restore

## Notes

# High Volume Copies

There are three general situations when you want to transfer a lot of information at once.

**Copy a disk**

To copy an entire disk to another of identical size (sometimes called disk cloning) one can use the `dd` utility.

```
# dd if=/dev/rdsk/c0d0s31 of=/dev/rdsk/c3d0s31
> bs=32b
```

**NOTE**

Making copies of A/UX disks, other than for archival purposes, may breach your license agreement.

**Copy a file system**

To copy a file system, first make a new file system, then use `dump.bsd`and `restore` to load it

```
# mkfs /dev/dsk/c5d0s3 blocksize
# mount /dev/dsk/c5d0s3 /mnt
# dump.bsd 0f - /dev/dsk/c0d0s0 | ( cd /mnt;
> restore xf - )
```

**Copy a directory**

To copy a directory (with all its subdirectories) you can use `cpio`:

```
# find directory1 -print | cpio -pdlmu
> /mnt/directory2
```

This command copies the files in *`directory1`* to *`directory2`*, which has been mounted to/`mnt`.

**Note**

The `-p` (pass) option of `cpio` was not discussed earlier.

**References**

*A/UX Local System Administration*
*cpio(1,) dd(1,) dump.bsd (1)*

# High Volume Copies

□ Copy a Disk

```
dd if=/dev/rdsk/c0d0s31
of=/dev/rdsk/c3d0s31 bs=32b
```

□ Copy a File System

```
newfs /dev/rdsk/c5d0s0
mount /dev/dsk/c5d0s0 /mnt
dump.bsd 0f - /usr |
( cd /mnt; restore xf - )
```

□ Copy a Directory

```
find directory1 -print |
cpio -pdlmu /mnt/directory2
```

17

## Notes

# A/UX 40 SC Tape Backup and Restore

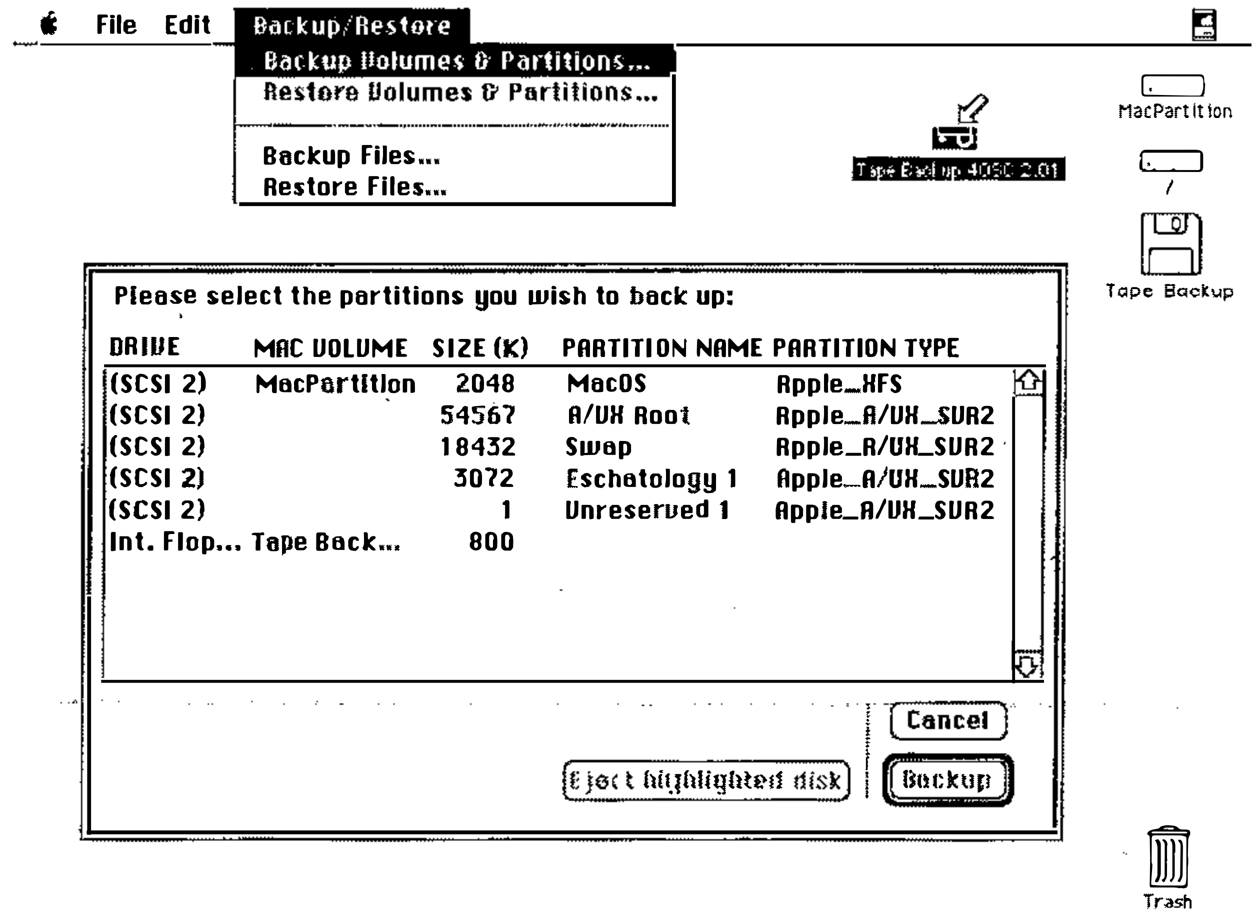
The Macintosh Tape Backup utility will back up and restore A/UX partitions to an Apple tape drive.

It is not possible to use the Tape Backup utility to back up individual files or file systems. You can only back up entire partitions.

A 40 MB tape must first be formatted (Choose the Format option under the File menu.) You can also Clear or Verify a tape cartridge.

You would then select Backup Volumes & Partitions to perform the backup procedure, and Restore Volumes & Partitions to restore a partition to disk.

If you are using an Apple tape drive and an Apple external disk drive, make sure the external devices have different SCSI IDs and that both are set to a number other than 0.



Please select the partitions you wish to back up:

| DRIVE | MAC VOLUME | SIZE (K) | PARTITION NAME | PARTITION TYPE |
|---|---|---|---|---|
| (SCSI 2) | MacPartition | 2048 | MacOS | Apple_HFS |
| (SCSI 2) | | 54567 | A/UX Root | Apple_A/UX_SVR2 |
| (SCSI 2) | | 18432 | Swap | Apple_A/UX_SVR2 |
| (SCSI 2) | | 3072 | Eschatology 1 | Apple_A/UX_SVR2 |
| (SCSI 2) | | 1 | Unreserved 1 | Apple_A/UX_SVR2 |
| Int. Flop... | Tape Back... | 800 | | |

# Notes

# Backup Summary

The the various backup utilities each have their advantages and disadvantages. The choice of which to use will depend on the task at hand.

**`tar`**

`tar` is best used to copy relatively few files. Is must be remembered that `tar` cannot copy special files, and that errors, such as reaching the end of media, are handled ungracefully.

**`cpio`**

`cpio` is better suited copy large numbers of files. `cpio` can copy special files, and when it reaches the end of media it prompts for another. The syntax of `cpio` is somewhat convoluted.

**`pax`**

`pax` is a standard utility, which means that archives stored in this format should be easily transportable. `pax` has the ability to copy symbolic links without copying the source files. However, `pax` is a recent standard, which means it is not universally supported yet.

**`dump.bsd` / `restore`**

`dump.bsd` necessarily can copy large numbers of files, as it can only back up file systems. This can be highly advantageous when file systems have been set up to put the (more rapidly changing) user files in a separate file system. Like `cpio`, `dump.bsd` can copy special files, and when it reaches the end of media it prompts for another. Additionally, `restore` has a unique and powerful interactive restoration capability.

**40SC Tape Backup**

This is a Macintosh utility that simply backs up partitions on Apple disk drives. While it is easy to use, it only supports the 40SC tape drive.

**References**

*A/UX Local System Administration*
*cpio(1), dump.bsd (1), pax(1), tar(1)*

# Backup Summary

| Utility | Use on | Advantages | Disadvantages |
|---|---|---|---|
| `tar` | files, some directories | transportable to other UNIX systems | no special files, single volumes only |
| `cpio` | files, directories | multivolume backups, file traits preserved | convoluted syntax |
| `pax` | files, directories | new "standard" | "new" standard |
| `dump.bsd` | file systems | multivolume backups, selective restore | file systems should be unmounted |
| 40SC Tape Backup | partitions | easy to use | only works with 40SC tape drive |

19

## Notes

# Automating Jobs

A/UX has a program, cron, which automatically performs jobs at a specified time. cron reads the directory /usr/spool/cron/crontabs for files that determine what jobs should be run, and when they should be run.

**Syntax**

```
    crontab [file]
    crontab -l
    crontab -r
```

When invoked with a file name, crontab copies the file into the directory /usr/spool/cron/crontabs, under the user's name. The -l option lists the current crontab file; the -r option removes the the user's (entire) crontab file.

**Note**

Two files are used to control user access to the cron facility, /usr/lib/cron/cron.allow and /usr/lib/cron/cron.deny. These files should not be used at the same time.

The format of a crontab file is lines consisting of six tab-separated fields. These six fields are:

minute (0-59)
hour (0-23)
day of the month (1-31)
month of the year (1-12)
day of the week (0-6, with 0=Sunday)
command

Entries in the fields are separated by commas. Entries are either a number or two numbers separated by a dash (indicating a range.) Putting an asterisk in any of the first five fields means "don't care." Stdout and stderr for the command should be redirected, or output will be mailed to the user.

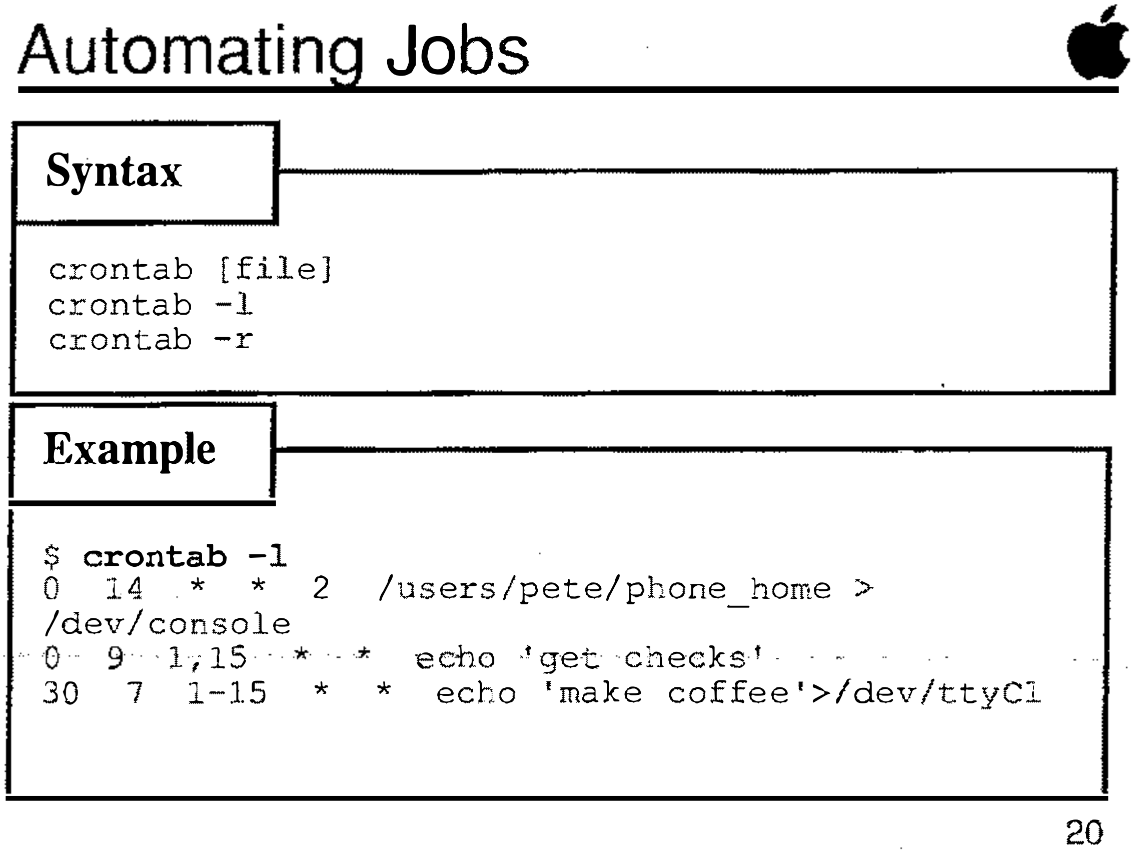
**Example 1**

```
$cat cronfile
0    14    *    *    2    /user/pete/phone_home >
dev/console
```

This example runs the program /user/student/phone_home on Tuesdays, at 2:00 p.m., sending the output to the console.

**References**

*Local System Administration*
*crontab (1,) at (1)*



# Automating Jobs

## Syntax

```
crontab [file]
crontab -l
crontab -r
```

## Example

```
$ crontab -l
0  14  *  *  2  /users/pete/phone_home >
/dev/console
0  9  1,15  *  *  echo 'get checks'
30  7  1-15  *  *  echo 'make coffee'>/dev/ttyC1
```

20

## Notes

# Automatic Backup

`cron` can be used for many kinds of functions. One of these is to remove some of the tedium from backups. By planning disk sizes and scheduling carefully, the majority of backups can be automated.

Given that disk partitions are less than 40 MB, the 40SC tape drive can be used to backup entire partitions, a different partition on each day. (The backups can be run at some hour where there is little likelihood of users being on the system.) In this case the only thing the administrator needs to do is swap tapes once a day.

Alternatively, an incremental backup script could be run. This script can have a provision for sending an error message to the administrator in case the end of the media is reached.

A simple script to back up all the files changed in the last day follows:

### Example Backup Script

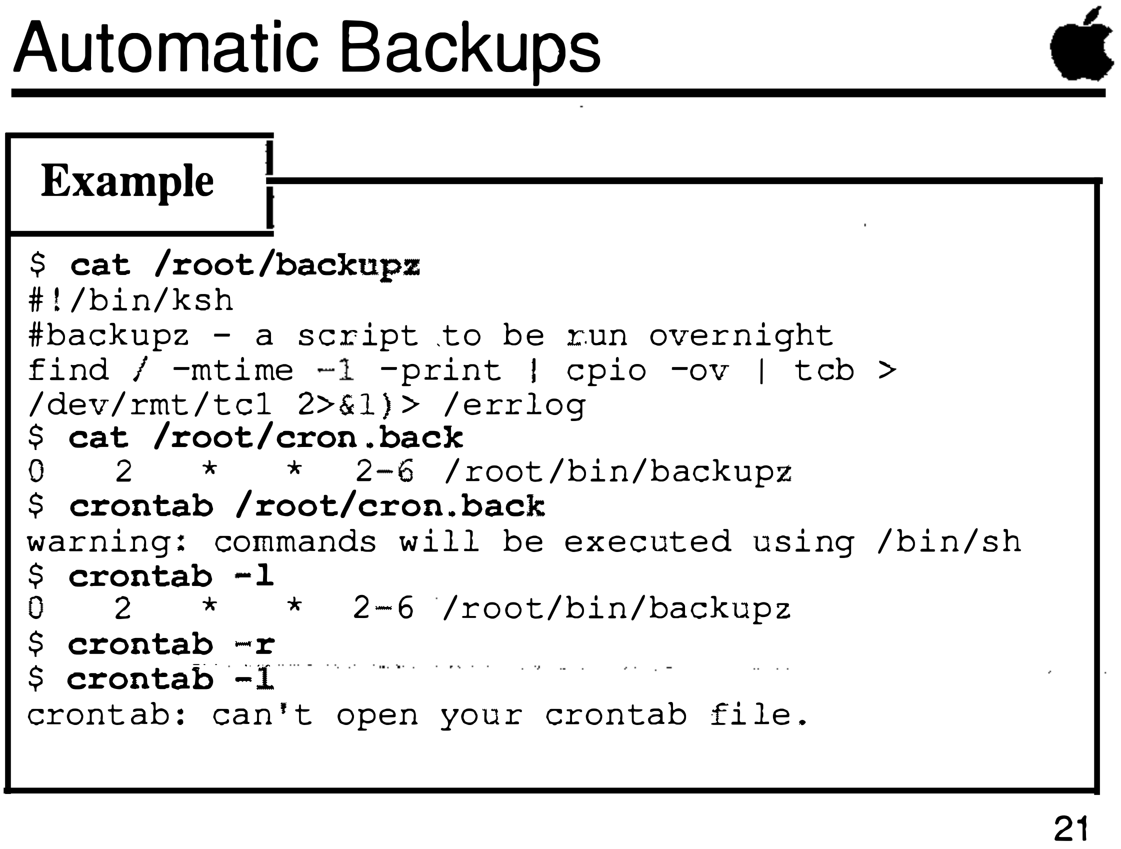
```
#!/bin/ksh
#backupz - a script to be run overnight
find / -mtime -1 -print | cpio -ov | tcb > dev/rmt/tc1 2>&1)> /errlog
```

The crontab file for this might read:

### Example crontab File

```
0 2    *    *    2-6 /root/bin/backupz
```

This runs the script at 2:00 a.m. Tuesday through Saturday mornings. Clearly, in attempting to automate this (or actually, any) function we are making several assumptions. First, that the system will be both running and quiescent at two in the morning on the days we have selected. Second, that the files selected will fit on one tape. Third, that output and error messages being sent to the file /errlog will be preserved in that file.



# Automatic Backups

## Example

```
$ cat /root/backupz
#!/bin/ksh
#backupz - a script to be run overnight
find / -mtime -1 -print | cpio -ov | tcb >
/dev/rmt/tcl 2>&1)> /errlog
$ cat /root/cron.back
0   2    *    *   2-6  /root/bin/backupz
$ crontab /root/cron.back
warning: commands will be executed using /bin/sh
$ crontab -l
0   2    *    *   2-6  /root/bin/backupz
$ crontab -r
$ crontab -l
crontab: can't open your crontab file.
```

21

## Notes

# Checkpoints

1 What do the command names `tar` and `cpio` stand for?

2 What are the general steps required to set up A/UX for the Apple 40 SC Tape Drive?

3 What is a "blocking factor?" Why is it important when backing up to the Apple 40 SC tape drive?

4 When is using `dump.bsd` and `restore` preferable to using `tar` or `cpio` for backing up your system?

5 What is the main limitation in using the Apple 40 SC Tape Backup program?

6 What program is used to invoke backup programs automatically?

# Exercises

Your Apple 40 SC Tape Drive should be connected and turned on, and a formatted tape should be inserted prior to starting the exercises. Refer to the manual *A/UX Local System Administration* and the on-line manual pages for assistance regarding commands with which you are not familiar.

**Exercise 1** In this exercise you will use `tar`, `cpio`, and`pax` to back up and restore files to and from a 3.5-inch floppy disk. In particular, you will back up a copy of /newunix, which you will need to restore later in the course.

**Step 1** Boot A/UX and log on as root. Open a CommandShell window. Insert a floppy disk into your floppy disk drive. The Finder may ask you how you want to use the disk; if so select the A/UX Data button. Format the disk from the command line.

**Step 2** Use the Commando dialog for `tar` to back up the directory /users/start onto the floppy disk. Use the verbose mode.

**Step 3** Open another CommandShell window and verify the backup by listing the contents of the archive. The file listings from the two `tar` commands should be the same, with the exception of the symbolic links. Recall that when a copy is made of a symbolic link the original file, not the link, is copied.

**Step 4** Use `pax` and Commando to back up the directory /users/start

**Step 5** Recover the files from the cpio archive by changing to the new directory and using `cpio` and Commando.

**Step 6** Verify your backup by listing the contents of the `pax` archive. Note that the copies of symbolic links are still symbolic links.

**Step 7** Use `ls` and the Commando dialog for `cpio` in a pipe to back up the file /newunix. (You will need this backup in the module on Autorecovery.)

**Step 8** Verify your backup by listing the contents of the `cpio` archive.

**Step 9** Eject the disk from the command line. Label the disk "/newunix".

**Exercise 2** In this exercise you will use the `tar` command to archive and restore files to and from the Apple 40 SC Tape Drive. You are already familiar with the basic use of these commands; the main difference when using the Apple tape drive is in specifying the blocking factor of 8 Kbytes.

Specifying the tape drive is a little easier if the system can find it through known names. Therefore, the first few steps below involve telling A/UX where to find the drive.

**Step 1** Verify the SCSI address of your tape drive: SCSI Address______

**Step 2** Add the following to the file /etc/profile:

```
TAPE=/dev/rmt/tcN
export TAPE
```

where *N* is the SCSI address.

**Step 3** Link the tape device to the name /dev/tape. Enter:

```
ln /dev/rmt/tcN /dev/tape
```

where *N* is the SCSI address.

**Step 4** Log off, then log back on as root.

**Step 5** Insert a formatted tape and check its status with the `mt` command. This command will tell you how many 8KB blocks are available on the tape:

**Step 6** Make a directory called /tmp/test and copy the /users/start directory into it. (Hint: use the -r option.)

**Step 7** Archive the /tmp/test/start directory onto tape using the `tar` Commando. You will need to specify the device to use.

**Step 8** When the archive is done, verify the archive by listing its contents. Verify the contents match the directory listing.

**Step 9** Remove the /tmp/test directory and its contents. Use `ls` to verify that the directory has been removed.

**Step 10** Restore the archive using the `tar` Commando.

**Step 11** Verify the restoration by listing the files in the start directory.

**Exercise 3** In this exercise you will use `dump.bsd` and `restore` to archive and retrieve a small file system that you will create on floppy disk. You will use this small file system in the interest of saving time.

**Step 1** Insert a blank floppy disk into your floppy drive. *Do not use the backed up copy of /newunix.* Format the disk.

**Step 2** Make a SVFS file system on the formatted floppy disk. The `mkfs` command waits a few seconds before creating the file system—don't disturb it. (We use an SVFS file system because performance isn't important on floppy disks.) Use the following command:

```
mkfs /dev/rfloppy0 1600 1 1
```

**Note** If you want to make a UFS file system on the floppy, you will first need to link the file /dev/rfloppy0 to /dev/rdsk/rfloppy0.

**Step 3** Mount the new file system to your `/mnt` directory. Note that since you can only mount a block device, you must specify the floppy disk drive by `/dev/floppy0`.

**Step 4** Copy the /users/start directory to the new file system. (Hint: use the -r option.)

**Step 5** Check to make sure the copy worked properly by using the `ls` command.

**Step 6** Unmount the `/mnt` directory.

**Step 7** Touch the /etc/dumpdates file to make sure it exists.

**Step 8** Archive the new file system on tape. Make sure your tape unit is on and a formatted tape is inserted, then invoke the `dump.bsd` Commando dialog to build a command line.

**Step 9** Examine the /etc/dumpdates file when the archive is finished.

**Step 10** Mount the new file system to your `/mnt` directory and remove the start directory and its contents. Verify the directory has been removed.

**Step 11** Restore the archive by using the `restore` Commando dialog. Make sure you are in the mounted `/mnt` directory before entering the `restore` command. (During the restoration process, the program will ask you to specify the next volume; enter a 1 at this point.)

**Step 12** Verify the files were restored by using the `ls` command.

**Exercise 4** In this exercise you will use `cpio` to copy your /usr/adm directory over to the file system you have established on the external drive. This is something you would normally do with the entire /usr directory when setting up a network for NFS access. Having your files in a separate file system allows you to limit the availability of your files with NFS.

**Step 1** Verify the external drive is mounted.

**Step 2** Use cpio and the pass option to copy the contents of the directory /usr/adm to the external drive. To do this, change directory to /usr/adm, and start the find portion of the command line from the current directory.

**Step 3** Verify the files have been copied.

(continued on next page)

**Exercise 5** In this exercise you will write a simple program to use the `cron` facility. This trivial example can be translated to have `cron` perform just about any task you can program the system for.

**Step 1** Write a crontab file that sends a message to your console (/dev/console) every 5 minutes.

**Step 2** Install the crontab file.

**Step 3** Verify the program has been installed.

**Step 4** Verify the program runs on schedule.

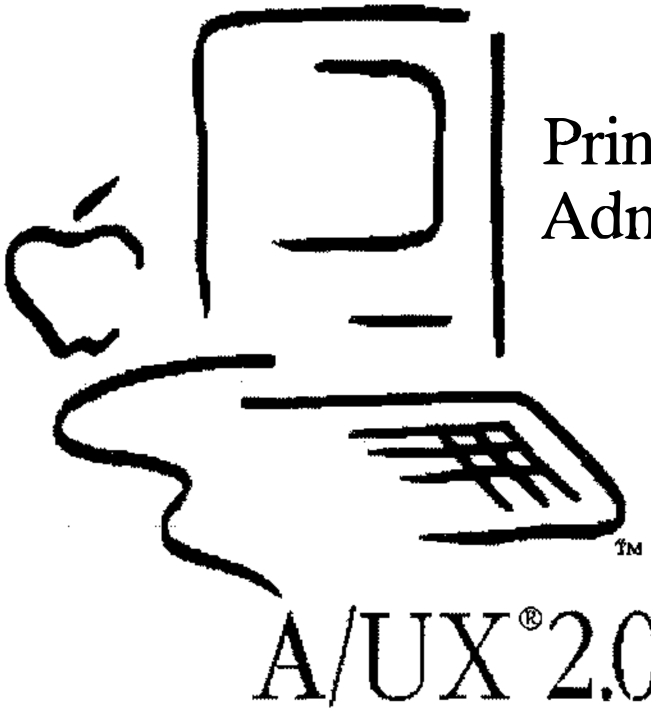
**Step 5** Shut down your system.

# Module 6
# Printer Administration

## Alternate Learning Path

# Table of Contents

# Printer Administration



## Notes

# Introduction to Printer Administration

Printer use in A/UX is a hybrid of UNIX® and Macintosh procedures, but the administration is necessarily more UNIX-like. Printers can still be selected from the Chooser; these may be direct-connected or networked printers. In addition to these, print jobs requested from the command line may also be sent to printers connected to other systems on the network. These can be other A/UX systems, or even other UNIX systems.

**Goal** The goal of Module 6 is to acquaint participants with printing under A/UX, particularly those features that are UNIX implementations. This module necessarily covers some subjects that may be considered network administration, but in the interest of completeness are presented here.

**Objectives** By the end of this module, participants will be able to:

- describe enabling and disabling printers and queues
- explain how to stop and restart the spooling daemon
- add a serial printer to the printer configuration file
- configure the system for remote printing requests
- add an AppleTalk printer to the printer configuration file

**Activities** Lecture/Discussion/Checkpoints/Exercises

**References** *A/UX Local System Administration*
*Setting up Accounts and Peripherals*
*A/UX Essentials*
*A/UX Network Administration*

# Objectives

- ❑ Describe enabling and disabling printers and queues
- ❑ Explain how to stop and restart the spooling daemon
- ❑ Add a serial printer to the printer configuration file
- ❑ Configure the system for remote printing requests
- ❑ Add an AppleTalk printer to the printer configuration file

2

## Notes

# Printing Commands

As a quick review, the print commands used on the A/UX command line are:

To print a file:

**Syntax**

```
lpr [-Pprinter] [-#num] [-m] [filename ...]
```

The various options are:

-P*printer* sent the print job to a *printer*.other than the default.

-#*num* print the specified *num*ber of copies

-m mail is sent to the user when printing is complete

To check the status of a print job:

**Syntax**

```
lpq [-Pprinter] [identifier]
```

The options are:

-P*printer* report on jobs sent to a specific *printer*. Otherwise, the default printer is used.

*-identifier* if *identifier* is a number, report on that job number; if *identifier* is a login name, report on jobs sent by that owner

To remove a print job from the queue:

**Syntax**

```
lprm [-Pprinter] [jobnumber ...]
```

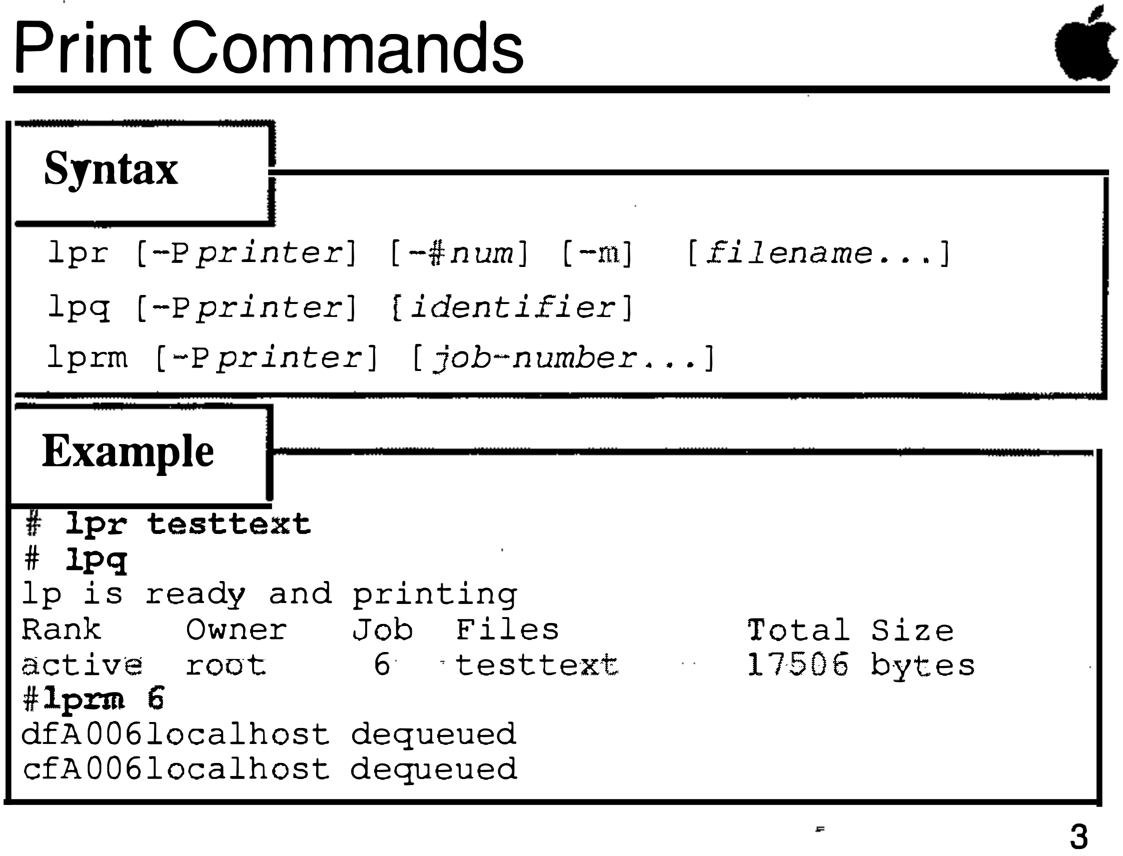
The options are:

-P*printer* remove jobs from the queue of a specific *printer*. Otherwise, the default printer is used.

*jobnumber* remove the specific *jobnumber*. If this is not provided, `lprm` simply removes the first of your jobs in the queue.

**References**

*lpr(1), lpq(1), lprm(1)*
*Setting Up Accounts and Peripherals for A/UX*



# Print Commands

## Syntax

```
lpr [-Pprinter] [-#num] [-m]  [filename...]
lpq [-Pprinter] [identifier]
lprm [-Pprinter] [job-number...]
```

## Example

```
# lpr testtext
# lpq
lp is ready and printing
Rank    Owner   Job  Files          Total Size
active  root     6   testtext       17506 bytes
#lprm 6
dfA006localhost dequeued
cfA006localhost dequeued
```

3

## Notes

# Print Spooling

All print jobs sent to a printer from a CommandShell command line are sent through a **print spooler**. This is done to avoid mixing the output of two people's jobs attempting to print on the same printer at the same time. As each job request is received it is placed in a queue, which is a list of print jobs. Jobs in the queue are then printed in the order received, though the system administrator can rearrange the order of jobs in a queue. Each printer has its own queue.

The system is shipped with the Berkeley `lpr` spooler preconfigured. Each of the printers and queues can be enabled or disabled through software.

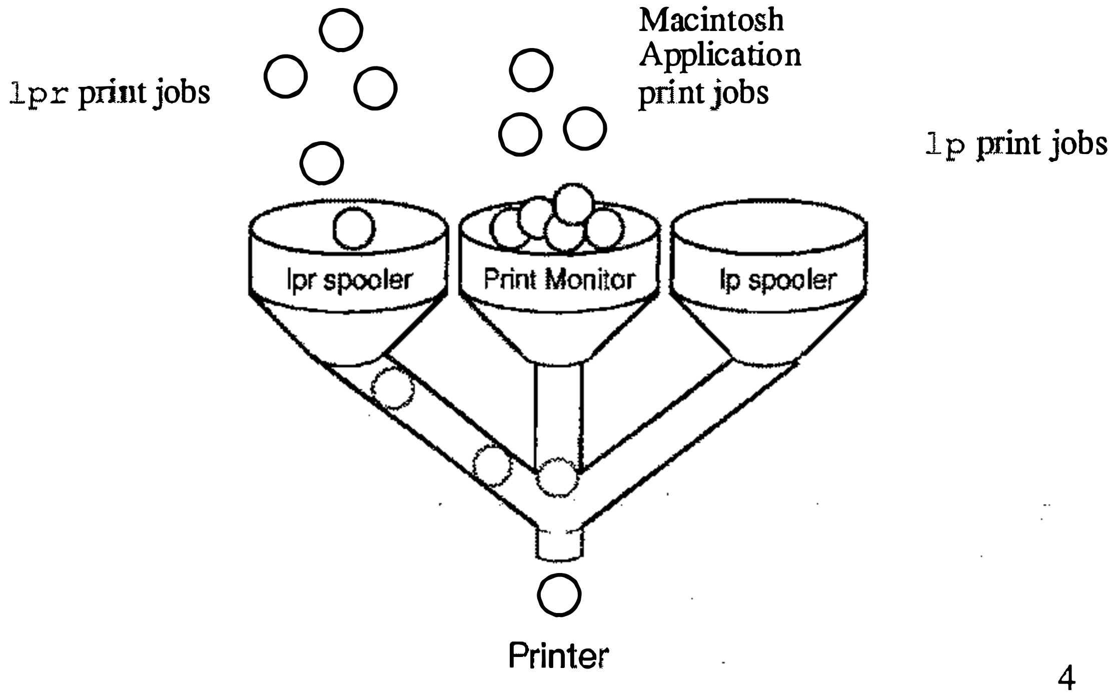
**Note** The AT&T `lp` spooler is also included for backward compatibility, but is not discussed here. Only one of the `lp` or `lpr` spoolers should be active at a time. Refer to *A/UX Local System Administration* for a discussion of this option.

The spooler queues jobs to be printed, waits until the printer is free, then sends the next job in the queue to the printer. This is controlled by a background program called a daemon. The daemon is started when the system first boots up and remains in the background unless stopped. The `lpr` spooler daemon is `/usr/lib/lpd`.

Macintosh applications do not use the A/UX spooler. Instead, they use the Macintosh spooler PrintMonitor. The two spoolers are coordinated because the printer only accepts one job at a time. The spooling of print jobs between Macintosh application requests and A/UX command-line requests is done transparently to the user.

**References** *A/UX Local System Administration*

# Print Spooling



4

## Notes

# Printing under A/UX

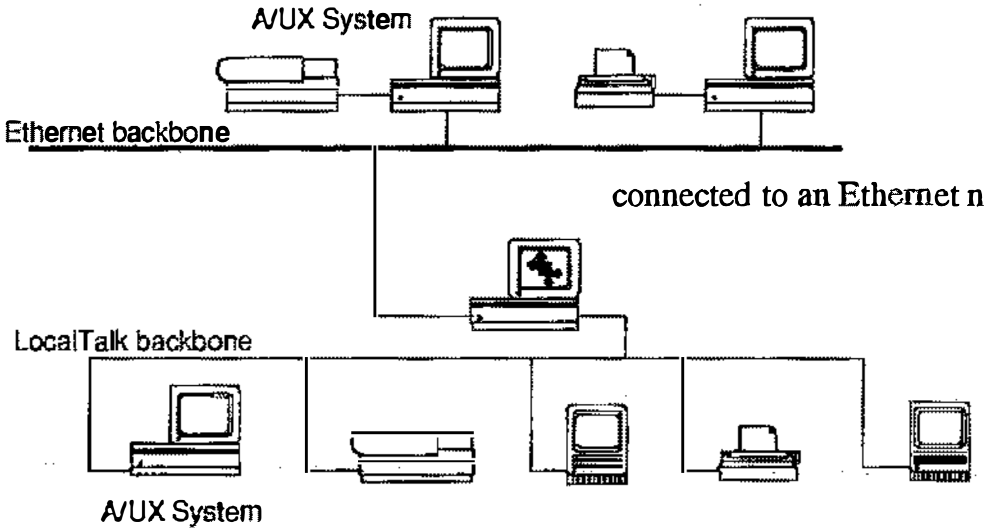
Printing under A/UX can occur either on printers that are connected to the system, called **local printers**, or on printers connected to a network, called **remote printers**. The remote printers may be connected directly to a LocalTalk® network or connected to **host** computers that are in turn connected to an Ethernet network.

Each printer on a LocalTalk network is given a unique name, so jobs can be directed to that printer. Each host computer on an Ethernet network having a remote printer connected to it likewise has a name, so print jobs can be sent to that host for printing. Printer and host names are kept in the printer configuration file, /etc/printcap.

**References** *A/UX Local System Administration*

# Where Does It Go?

Print requests may be sent to printers connected directly to the system,



connected to an Ethernet network,

or to printers connected to LocalTalk networks (possibly through an internet router).

5

## Notes

# Line-Printer Control

Printer administration is done through the line-printer control program `lpc`.

**Syntax**

```
lpc [command [argument ... ]]
```

**Commands**

| | |
|---|---|
| abort | clean |
| disable | down |
| enable | exit |
| help | quit |
| restart | start |
| status | stop |
| topq | up |

**Arguments**

generally a description of which printers are affected, and are usually of the form:

```
{all | printer}
```

or

```
[jobnumber...] [user...]
```

**Note**

This is the first time we have seen the { } notation, which indicates that one of the choices must be used. The vertical bar ( | ) separates the mutually exclusive choices.

Without *commands* or*arguments*, `lpc` prompts for further input with: `lpc>`.

**References**

*A/UX Local System Administration*<br>
*lpc (1M)*

# Line-Printer Control

## Syntax

```
lpc [command [argument ... ]]
lpc [command {all | printer}]
lpc [command [jobnumber...] [user...]]
```

## Example

```
# lpc
lpc> help
Commands may be abbreviated.  Commands are:
abort    enable   disable  help     restart
status   topq     ?        clean    exit
down     quit     start    stop     up
lpc>
```

6

## Notes

# Spooler Status

In order for normal printing to occur, three things need to happen. First, the printer must be enabled. Second, the printer queueing must be enabled. Third, a spooling daemon must be alive to handle the print job.

The system maintains a status on all printers listed in the printer configuration file /etc/printcap.

**Syntax**

```
lpc status [ all | printer ]
```

The status of each of the following is displayed:

**Queue** Reports if a particular queue is enabled (accepting requests) or disabled.

**Printer** Reports if a particular printer is enabled (accepting requests) or disabled.

**Entries** Reports the current number of jobs in a printer's queue.

**Status** Reports the current status of the spooler. (The message `no daemon present` indicates the spooler is idle; if there are still jobs in the queue it indicates the daemon has died prematurely.)

**Example**

```
# lpc status
lp:
        queuing is enabled
        printing is enabled
        1 entry in spool area
        lp is ready and printing
iw:
        queuing is disabled
        printing is disabled
        no entries
        no daemon present
```

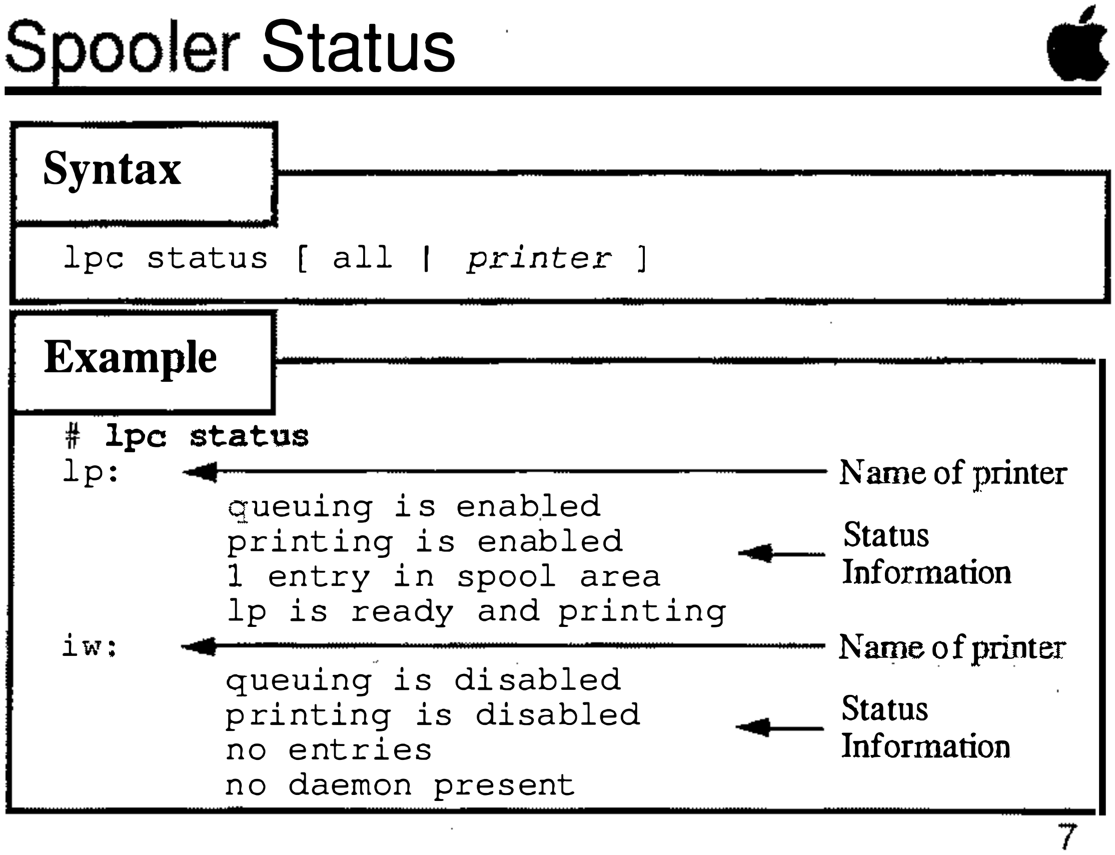
To determine the status on a remote printer, you must examine its queue directly. To do this, issue the command:

**Syntax**

```
lpq -Pprinter
```

where *printer* is the name of the printer.

**References** *lpc (1M,) lpq (1)*



# Spooler Status

## Syntax

```
lpc status [ all | printer ]
```

## Example

```
# lpc status
lp:
        queuing is enabled
        printing is enabled
        1 entry in spool area
        lp is ready and printing
iw:
        queuing is disabled
        printing is disabled
        no entries
        no daemon present
```

Name of printer
Status Information
Name of printer
Status Information

7

## Notes

# Enabling and Disabling Printers

Individual printers can be disabled to prevent printing. This can be beneficial if a printer is to be temporarily unavailable. Jobs can still be queued to the printer, but they won't print until the printer is started.

**Syntax**

```
    lpc abort { all | printer }
```

This command immediately stops the current print job (by killing the spooling daemon) and then disables printing for the specified printers.

**Syntax**

```
    lpc stop { all | printer }
```

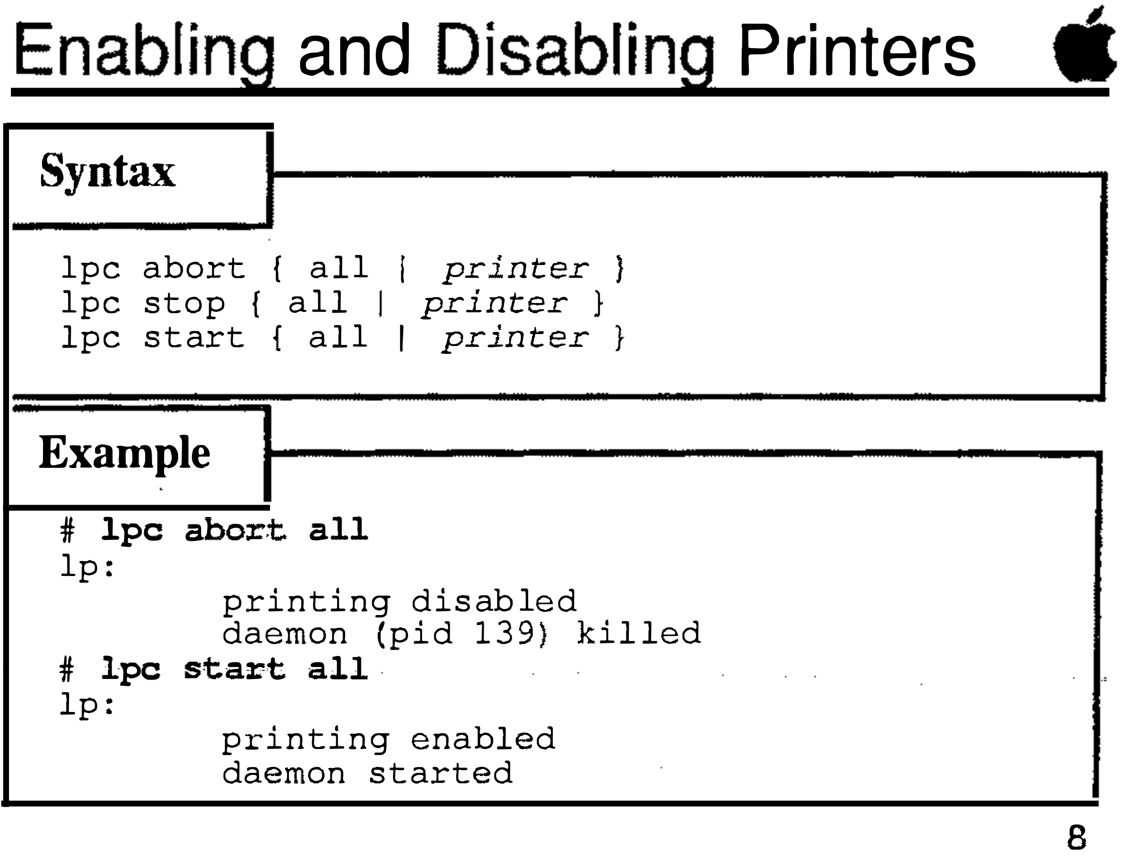
This command disables printing for the specified printers after the current job completes (if there is one).

**Syntax**

```
    lpc start { all | printer }
```

This enables printing and starts a spooling daemon for the listed printers.

**References** *lpc (1M)*



# Enabling and Disabling Printers

## Syntax

```
lpc abort { all | printer }
lpc stop { all | printer }
lpc start { all | printer }
```

## Example

```
# lpc abort all
lp:
        printing disabled
        daemon (pid 139) killed
# lpc start all
lp:
        printing enabled
        daemon started
```

8

## Notes

# Starting and Stopping the Queue

Individual printer queues can be disabled to prevent printing. This can be beneficial if a printer is unavailable, perhaps for maintenance. Jobs will not be queued to the printer, unless they are submitted by the superuser.

**Syntax**

```
    lpc disable { all | printer }
```

This command turns off the spooling queues for the specified printers. This prevents `lpr` from entering new printer jobs in the queue.

**Syntax**

```
    lpc enable { all | printer }
```

This enables spooling on the local queue for the listed printers. This command allows `lpr` to put new jobs in the spool queue.

**Syntax**

```
    lpc down { all | printer } [message]
```

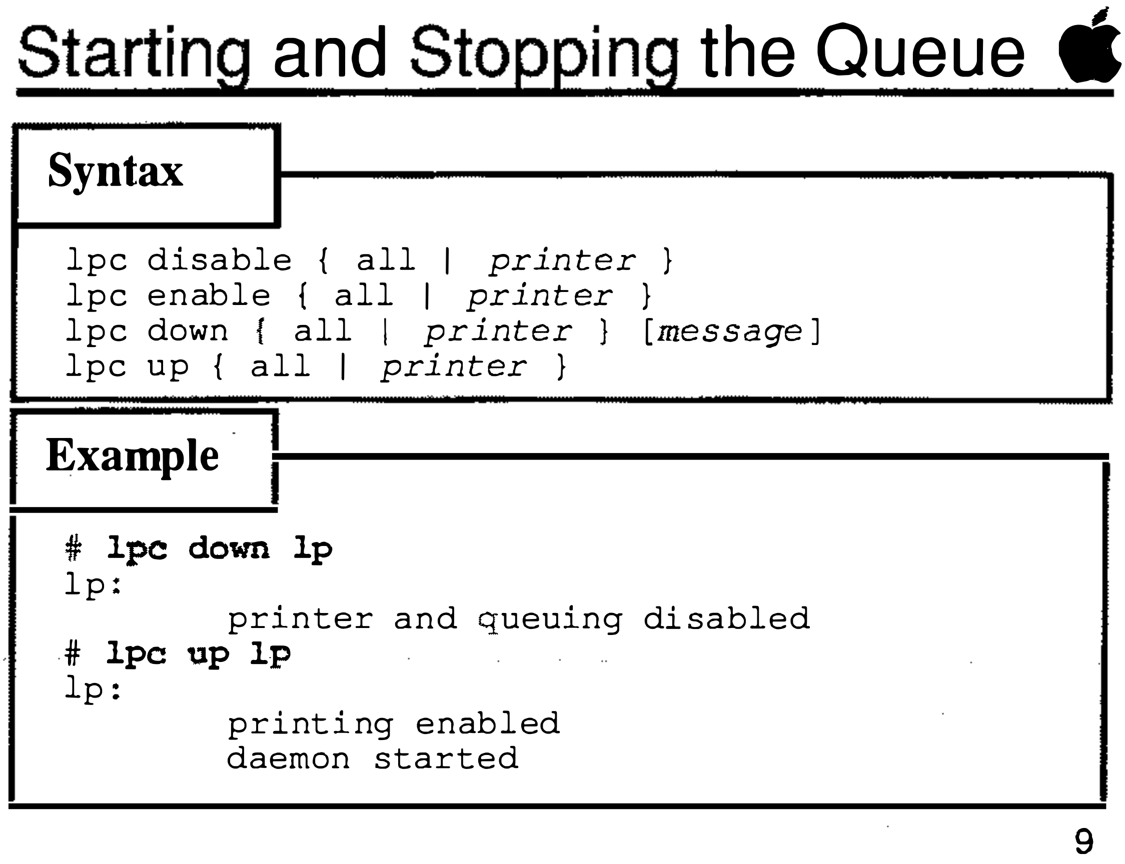
This command turns off the spooling queue for the specified printers, disables printing, and puts a message in the printer status file. The message doesn't need to be quoted. The remaining arguments are treated like *echo(1)*. This is normally used to take a printer down and lets others know why the printer is coming down. `lpq` indicates the printer is down and prints the status message.

**Syntax**

```
    lpc up { all | printer }
```

This enables everything and starts a new printer daemon. Effectively undoes the effects of `down`.

**References** *lpc (1M)*



# Starting and Stopping the Queue

**Syntax**

```
lpc disable { all | printer }
lpc enable { all | printer }
lpc down { all | printer } [message]
lpc up { all | printer }
```

**Example**

```
# lpc down lp
lp:
        printer and queuing disabled
# lpc up lp
lp:
        printing enabled
        daemon started
```

9

## Notes

# Restarting a Print Job

Occasionally some abnormal condition causes the spooling daemon to die unexpectedly, leaving jobs in the queue. `lpc` reports that no daemon is present when this condition occurs.

### Syntax

```
    lpc restart { all | printer }
```

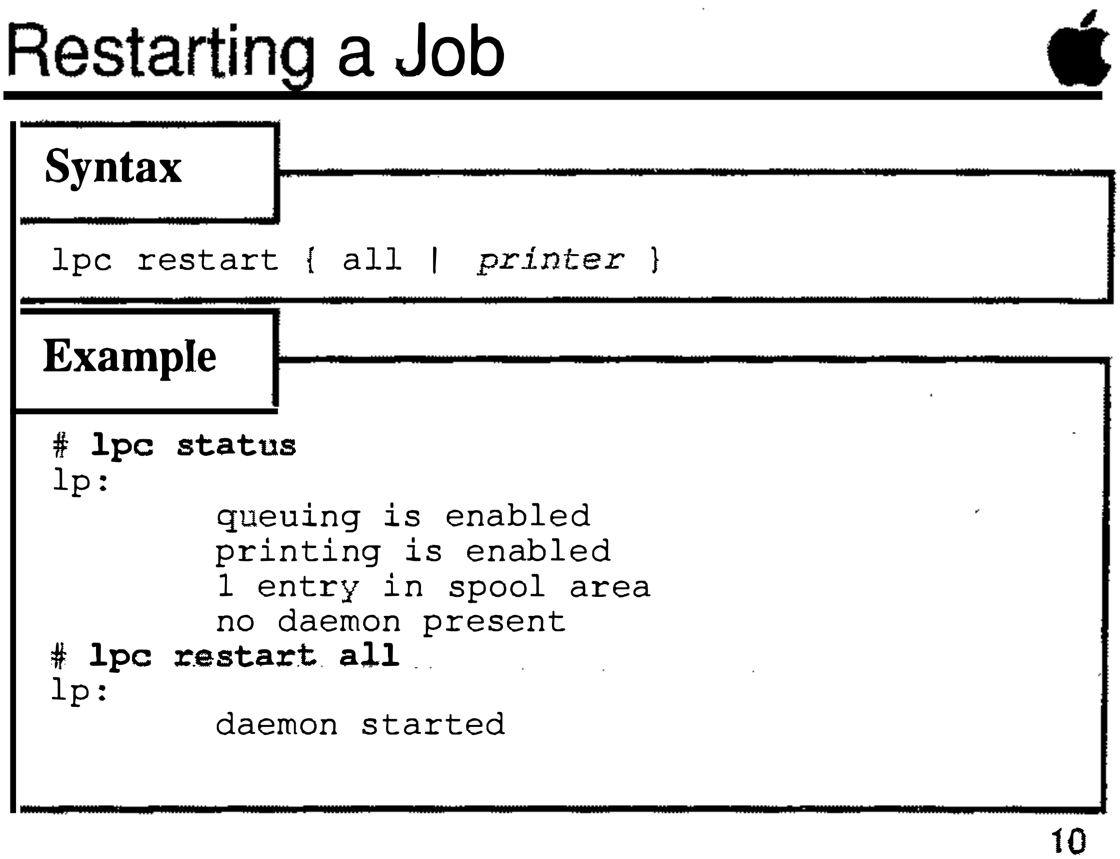
This command attempts to start a new printer daemon. It is needed when the printer and queue are both enabled and there are print jobs waiting to be printed, but no daemon is being spawned.

### Example

```
    # lpc status
    lp:
            queuing is enabled
            printing is enabled
            1 entry in spool area
            no daemon present
    # lpc restart all
    lp:
            daemon started
```

### References

*lpc (1M)*



# Restarting a Job

## Syntax

```
lpc restart { all | printer }
```

## Example

```
# lpc status
lp:
        queuing is enabled
        printing is enabled
        1 entry in spool area
        no daemon present
# lpc restart all
lp:
        daemon started
```

10

## Notes

# Summary of Line Printer Control

The slide to the right summarizes the commands that affect the various aspects of line printer control.

**References** *lpc(1M)*

# Summary: Line Printer Control

| Command | Effect on Printer | Effect on Queue | Effect on Daemon |
|---|---|---|---|
| `abort` | disables | none | kills |
| `stop` | disables | none | none |
| `start` | enables | none | spawns |
| `disable` | none | disables | none |
| `enable` | none | enables | none |
| `down` | disables | disables | none |
| `up` | enables | enables | spawns |
| `restart` | none | none | spawns |

11

## Notes

# Moving Jobs in the Queue

The spooler normally queues print jobs on a "first-in, first-out" basis. While this is normally a reasonably equitable arrangement, there may be times you want to move smaller jobs ahead of larger jobs. Queues can be rearranged with the `topq` command:

**Syntax**

```
topq printer [ jobnum ... ] [ user ... ]
```

This command places print jobs in the order listed at the top of the printer queue. If `user` is specified, all that user's print jobs are moved to the top of the queue.

**References** *lpc (1M)*

# Moving Jobs in the Queue

## Syntax

```
lpc topq printer [ jobnum ... ] [ user ... ]
```

## Example

```
# lpq
Rank    Owner   Job  Files            Total Size
1st     root    6    letter1          681 bytes
2nd     root    7    memo2            3561 bytes
# lpc topq lp 7
lp:
         moved cfA007basilisk
# lpq
Rank    Owner   Job  Files            Total Size
1st     root    7    memo2            3561 bytes
2nd     root    6    letter1          681 bytes
```

12

## Notes

# Network Printing

If you want to be able to print on a networked printer there are several options possible.

If the network is comprised of systems running AppleTalk® and connected via LocalTalk, all networked printers will be available through the Chooser.

If the network is comprised of systems running AppleTalk and connected via EtherTalk®, networked printers will be available through the Chooser if there is an **internet** router between the printer and the network.

Add printers available through the Chooser as AppleTalk printers.

If the network is comprised of systems connected via EtherTalk some networked printers may not be available through the Chooser. This will likely be the case when an A/UX system is connected to a network of non-A/UX systems.

Add printers not available through the Chooser as remote host printers.

Chooser-selectable printers can also be seen (and the default changed) from the command line by using the command:

**Syntax**

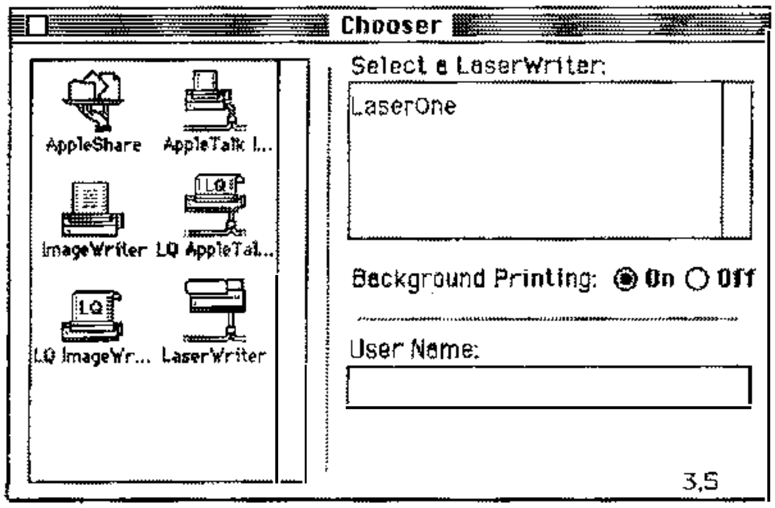
```
at_cho_prn
```

The command displays a list of printer selections and can save the name of the printer that you select. It checks the network to determine which printers are registered on that network. You are prompted to select a printer by entering a number from the printer list display.

**References**

*Local System Administration*
*at_cho_prn (1)*

# Network Printers



Select a LaserWriter:

LaserOne

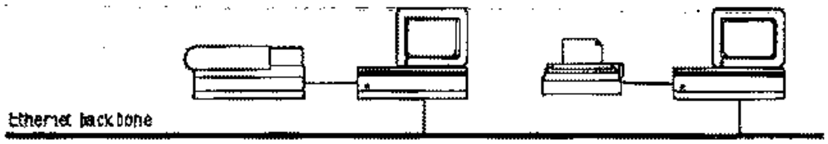
Background Printing: ● On ○ Off

User Name:

3.5

If a printer is visible in the Chooser, add it as an AppleTalk printer.

If a printer is not visible in the Chooser, add it as a remote host printer.



13

## Notes

# Printer Configuration

A/UX is made aware of printers by listing them in the printer configuration file, /etc/printcap.

The file has a simple structure, it is merely a printer name (with aliases) followed by a description of printer characteristics.

The first name of the characteristic list is the name that `lpc` will use to report the printer and queue status. This is followed by optional aliases, separated by vertical bars ( | .) A listing of the printer characteristics comes next. These are fairly cryptic, but are described in *printcap(4.)*

**/etc/printcap**

```
# Sample /etc/printcap file
# Comments (usually printer location, etc)
nm|alias1|alias2:\
 :lp=Output device:\
 :Filter/Printer characteristics:\
 :sd=Spool Directory:
```

**Note**

Entries have their NEWLINES escaped by a backslash.

The changes made to the configuration file will take effect as soon as the file is closed.

**References**

*A/UX Local System Administration*
*printcap (4)*

# Printer Configuration

## Example

```
# Sample /etc/printcap file
# Comments (usually printer location, etc)
nm|alias1|alias2:\
 :lp=Output device:\
 :Filter/Printer characteristics:\
 :sd=Spool Directory:
```

where:

nm is the printer name

alias are alias names

lp defines the output device

filter defines communication parameters. This is usually a set of filters for PostScript printers, and a set of control codes for serial printers.

sd defines the spool directory for this printer

14

## Notes

# Enabling Serial Printing

The /etc/printcap file is set up by default to allow serial printing to an ImageWriter on the printer port of a Macintosh. The portion of the /etc/printcap file controlling this is:

**Example File 1**
**Default**
**/etc/printcap**

```
# Local ImageWriter II printer
iw|iw2|ImageWriter II:\
   :lp=/dev/printer:br#9600:os#0014001:cs#0004060:\
   fd:tr=\f:sd=/usr/spool/lpd/ImageWriter:
```

In this file:

| | |
|---|---|
| `lp` | this is the name of the printer is connected |
| `lp=` | this is the device to which the printer is connected |
| `br#` | this is the baud rate of the printer |
| `os#` | this clears the output flag bits |
| `cs#` | this clears the control flag bits |
| `fd` | use DTR/DCD flow control |
| `tr` | send a form feed (\f) at end-of-file |
| `sd` | this is the spool directory |

**Note**

The system, by default, runs AppleTalk on the printer port. This must be turned off before we can send serial data through that port. Consult the *Terminal Administration* module for instructions on how to disable AppleTalk on the printer port.

With a slight change to the printer configuration file we can allow serial printing to an ImageWriter® on the modem port:

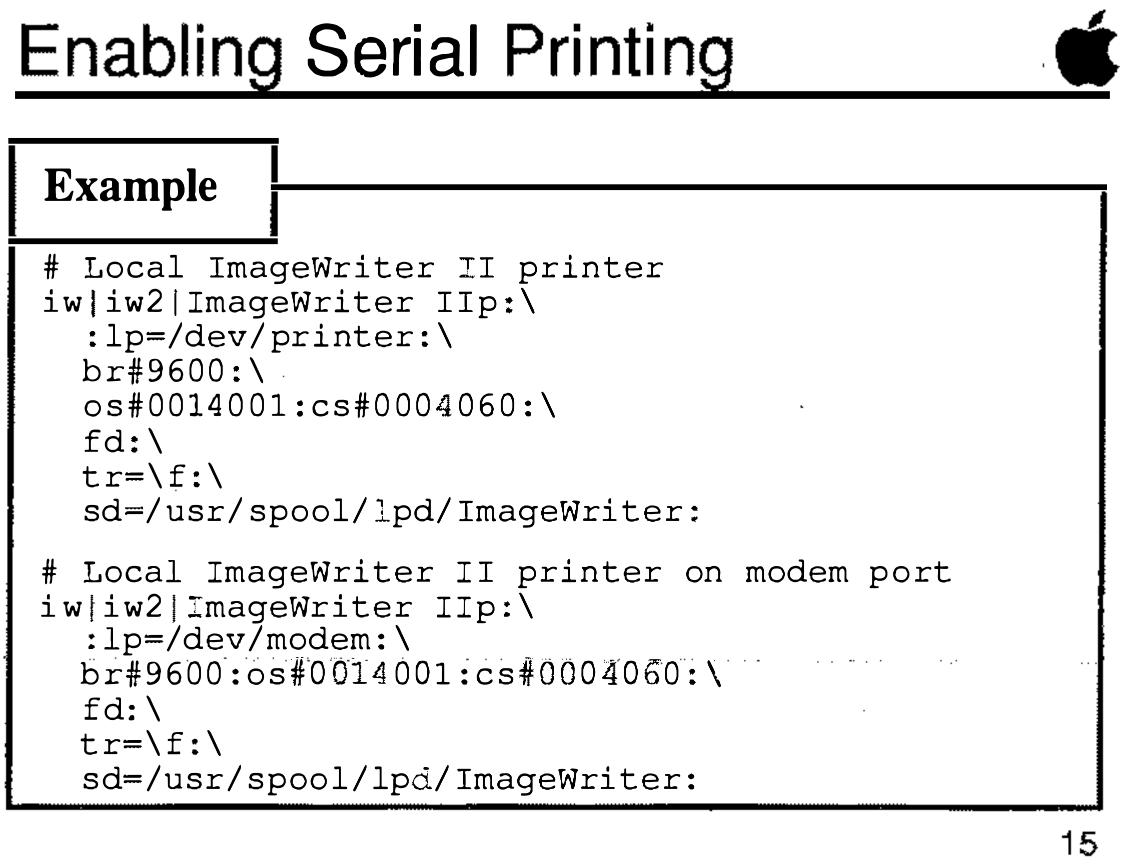
**Example File 2**
**/etc/printcap**

```
# Local ImageWriter II printer on modem port
iw|iw2|ImageWriter IIp:\
   :lp=/dev/modem:br#9600:os#0014001:cs#0004060:\
   fd:tr=\f:sd=/usr/spool/lpd/ImageWriter:
```

All we have done is change the device to which the data is sent (the important change is in bold; `printer` is now`modem`.)

**References**

*A/UX Local System Administration*
*printcap(4)*



# Enabling Serial Printing

## Example

```
# Local ImageWriter II printer
iw|iw2|ImageWriter IIp:\
  :lp=/dev/printer:\
  br#9600:\
  os#0014001:cs#0004060:\
  fd:\
  tr=\f:\
  sd=/usr/spool/lpd/ImageWriter:

# Local ImageWriter II printer on modem port
iw|iw2|ImageWriter IIp:\
  :lp=/dev/modem:\
  br#9600:os#0014001:cs#0004060:\
  fd:\
  tr=\f:\
  sd=/usr/spool/lpd/ImageWriter:
```

15

## Notes

# Adding AppleTalk Printers

In order to print from one of several printers connected to an AppleTalk network, add them as follows:

**Step 1** Get a list of the accessible network printers.

```
# at_cho_prn
```

**Step 2** Create a directory in /usr/spool/lpd with *exactly* the same Object name as that reported by `at_cho_prn`.

```
# mkdir /usr/spool/lpd/printname
```

**Step 3** Change the permissions on the directory to 755. Change both the owner and group of the directory to daemon.

```
# chmod 755 /usr/spool/lpd/printname
# chgrp daemon /usr/spool/lpd/printname
# chown daemon /usr/spool/lpd/printname
```

**Step 4** Symbolically link the files ifilter, ofilter, and nfilter in the new directory to their counterparts in /usr/spool/lpd/AppleTalk.

```
# cd /usr/spool/lpd/printname
# ln -s /usr/spool/AppleTalk/ifilter ifilter
# ln -s /usr/spool/AppleTalk/ofilter ofilter
# ln -s /usr/spool/AppleTalk/nfilter nfilter
```

**Step 5** Make a named pipe to use for spooling. Change its permissions to 660:

```
# mknod pipe p; chmod 660 pipe
```

**Step 6** Change both the owner and group of the new files to daemon.

```
# chgrp daemon ifilter ofilter nfilter pipe
# chown daemon ifilter ofilter nfilter pipe
```

**Step 7** Edit /etc/printcap to add an entry for the new printer:

```
lwl|at1|printname|postscript|PostScript:\
    :lp=/dev/null:\
    :if=/usr/spool/lpd/printname/ifilter:\
    :of=/usr/spool/lpd/printname/ofilter:\
    :nf=/usr/spool/lpd/printname/nfilter:\
    :sd=/usr/spool/lpd/printname:
```

**Step 8** Start the printer, queue, and daemon.

```
# lpc up printname
```

**Note** If the AppleTalk printer is not a PostScript printer, remove the PostScript names from the first line of the printer description.

# Adding AppleTalk Printers

- ❑ Get printer name
- ❑ Make spool directory
- ❑ Adjust permission/ownership of spool directory
- ❑ Link filter files
- ❑ Create named pipe (FIFO)
- ❑ Adjust permission/ownership of filters and pipe
- ❑ Edit /etc/printcap
- ❑ Start the printer, queue, and daemon

16

## Notes

# Adding Remote Host Printers

The section of the default /etc/printcap that allows printing on remote hosts is shown below:

**Example File 1 Default /etc/printcap**

```
# Remote UNIX line printer
# Change 'RemoteHost' to actual name of
# remote UNIX host
remote|remote line printer:\
   :lp=:rp=lp:rm=RemoteHost:sd=/usr/spool/lpd/Remote:
```

To print files on the remote host kite, we simply make the modifications:

**Example File 2 /etc/printcap**

```
# remote A/UX host kite
r2|lwkite|remote line printer:\
   :lp=:rp=lp:rm=kite:sd=/usr/spool/lpd/kite:
```

In this case, we are sending the print jobs to whatever kite has set as its default printer (`rp=lp`).

We can send files to specific printers on hosts all over the network. The steps to do this are:

**Step 1**

Create a directory in /usr/spool/lpd to use as a spooling directory.

```
# mkdir /usr/spool/lpd/printname
```

**Step 2**

Change the permissions on the directory to 755. Change both the owner and group of the directory to daemon.

```
# chmod 755 /usr/spool/lpd/printname
# chgrp daemon /usr/spool/lpd/printname
# chown daemon /usr/spool/lpd/printname
```

**Step 3**

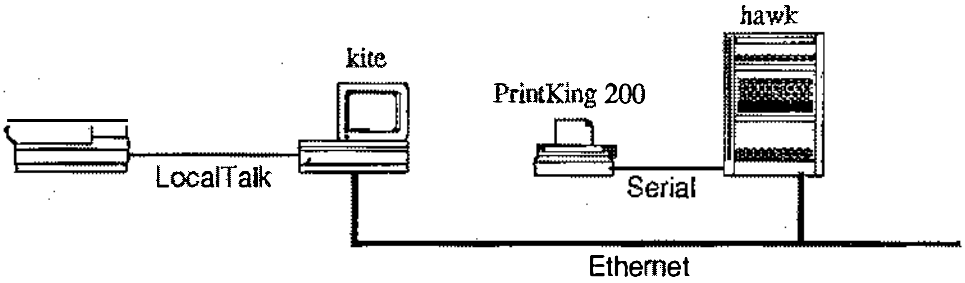
Edit the /etc/printcap file to reflect the desired printer and host names, as well as the new spooling directory.

```
# remote A/UX host hostname
id|printname|remote line printer:\
   :lp=:rp=printid:rm=hostname:\
   sd=/usr/spool/lpd/printname:
```

**Note**

For our print jobs to print on a remote host, we must have access permission to queue print jobs to it.

# Adding Remote Host Printers



## Example

```
# cat /etc/printcap
# Remote UNIX line printers
# remote A/UX host kite
r2|lwkite|remote line printer:\
    :lp=:rp=lp:rm=kite:sd=/usr/spool/lpd/kite:
# remote UNIX host hawk
r3|pkhawk|remote line printer:\
    :lp=:rp=pk:rm=hawk:sd=/usr/spool/lpd/hawk:
```

17

## Notes

# Enabling Others to Print Locally on Your System

In order to allow remote users to print on your system, you must give them permission by including them in the file /etc/hosts.lpd. This file is simply a list of system host names, one per line, of those systems on the network that are allowed to send files for printing through the `lpr` spooler on your system.

**Example File 1 /etc/hosts.lpd**

```
kite
hawk
eagle
```

This file allows any user on the host systems kite, hawk, or eagle to send print jobs via the `lpr` command to the printers listed in your local system's /etc/printcap file.

Access can be restricted to only those remote users who have an account in the local /etc/passwd file by adding the printcap term:

```
:rs:
```

to each printer description in the file /etc/printcap.

It is also possible to restrict printing to local members of certain groups by adding the printcap term:

:rg:*lprgroup:*

Add :rg:*lprgroup*:to the desired printer descriptions in the file /etc/printcap. The group *lprgroup* must be set up on the local system for this to work.

**Example File 2 /etc/printcap**

```
# Local ImageWriter II printer on modem port
# Only members of group legal with accounts
# on this system can use this printer.
il|iwl|ImageWriter IIl:\
   :lp=/dev/printer:\
   br#9600:os#0014001:cs#0004060:\
   rs:rg:legal:\
   fd:tr=\f:sd=/usr/spool/lpd/ImageWriterl:
```

**References** *A/UX Local System Administration*

# Enabling Others to Print Locally

## Example

```
# cat /etc/hosts.lpd
hawk
eagle
# head /etc/printcap
# Local ImageWriter II printer on modem port
iw|iw2|ImageWriter IIp:\
  :lp=/dev/modem:br#9600:os#0014001:cs#0004060:\
  fd:tr=\f:sd=/usr/spool/lpd/ImageWriter:
# Local ImageWriter II printer on printer port
# Only members of group legal with accounts
# on this system can use this printer.
il|iwl|ImageWriter IIl:\
  :lp=/dev/printer:br#9600:os#0014001:cs#0004060:\
  rs:rg:legal:\
  fd:tr=\f:sd=/usr/spool/lpd/ImageWriterl:
```

18

## Notes

# Checkpoints

1 What commands stop print functions? What are the differences in each?

2 What commands start print functions? What are the differences in each?

3 What file controls the printer configurations?

4 What files must be changed, and on which systems, to enable remote printing?

# Exercises

**Exercise 1** Set your system up for remote printing to the serial printer attached to the Leader's system (this assumes the system has been set up with the host name jabberwock.)

To set up remote printing:

**Step 1** Log on to your A/UX system as root.

**Step 2** Edit the file /etc/printcap to add the remote hostname (jabberwock) and printer designation (iw) to the section controlling remote printing.

**Step 3** Print the file /etc/passwd to the remote printer.

**Exercise 2** Set your system up for printing to an additional, non-default, AppleTalk printer.

**Step 1** Check which printer is currently selected. Choose Chooser from the  menu. This is the default printer.

**Step 2** Examine the printers available on the network using the `at_cho_prn` command. Do not change the setting, but do note the printer Object name that was *not* selected in the Chooser.

**Step 2** Create a directory in /usr/spool/lpd with *exactly* the same Object name as the non-default printer reported by `at_cho_prn`.

**Step 3** Change the permissions on the directory to 755. Change both the owner and group of the directory to daemon.

**Step 4** Symbolically link the files ifilter, ofilter, and nfilter in the new directory to their counterparts in /usr/spool/lpd/AppleTalk.

**Step 5** Make a named pipe to use for spooling. Change its permissions to 660.

**Step 6** Change both the owner and group of the new files to daemon.

**Step 7** Edit /etc/printcap to add an entry for the new printer.

**Step 8** Activate the printer, queue, and daemon for the new printer.

**Step 9** Print the file /etc/printcap to the AppleTalk printer.

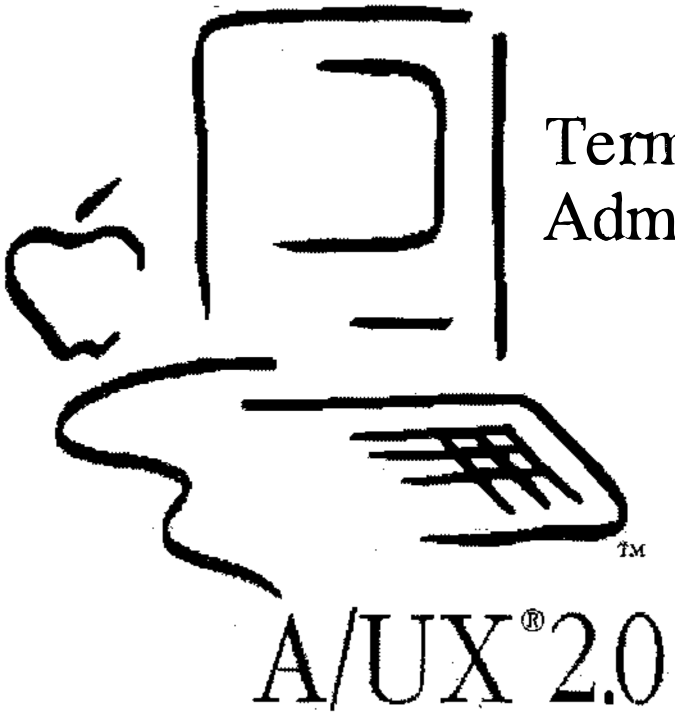
**Step 10** In preparation for the next module, drag the MacTerminal icon from the /usr/tmp folder into the MacPartition disk.

**Step 11** Shut down your system.

# Module 7
# Terminal Administration

# Table of Contents

# Terminal Administration



## Notes

# Introduction to Terminal Administration

Terminals used with A/UX are more UNIX®-like than Macintosh-like. The requirement for large amounts of graphical information restricts the use of the Finder to the system console. However, other users can have access to the command-line interface by connection through the serial ports. Additionally, these ports allow the use of a modem to dial into other systems or to have other users dial into your system.

**Goal**

The goal of Module 7 is to acquaint participants with the mechanics of resetting the system files to allow communication with external terminals and modems.

**Objectives**

By the end of this module, participants will be able to:

- describe the files controlling terminal configuration
- configure the system for modem use
- configure the system for remote terminal use
- set up a Macintosh as a terminal

**Activities**

Lecture/Discussion/Checkpoints/Exercises

**References**

*A/UX Local System Administration*
*Setting up Accounts and Peripherals*
*A/UX Essentials*

# Objectives

- ❑ Describe the files controlling terminal configuration
- ❑ Configure the system for modem use
- ❑ Configure the system for remote terminal use
- ❑ Set up a Macintosh as a terminal

2

## Notes

# Ports

Low-speed (less than 1 MB/sec) communication with the outside world is done through the serial ports, located on the back of Macintosh computers. Though these ports have different icons associated with them, both can be set up through software to function as serial ports.

The port identified by the phone icon is known to the system as /dev/tty0, also as /dev/modem.

The port identified by the printer icon is known to the system as /dev/tty1, also as /dev/printer.

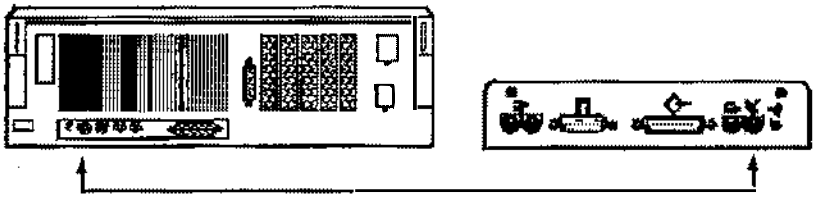
**Note** Despite their device names, one can put printers on /dev/modem, modems on /dev/printer, and terminals on either.

In order to make the hardware connection between the Macintosh and a terminal or modem, simply power down your A/UX system, plug one end of the appropriate serial cable between the two on either the modem or printer ports.

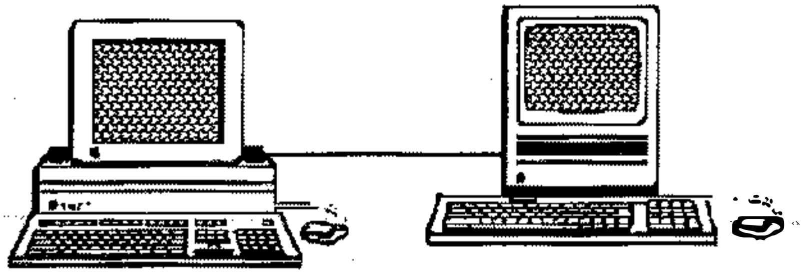
**Note** If you're using a Macintosh as a terminal, use the Apple M0197 cable to the two computers together.

**References** *A/ UX Local System Administration*
*Setting up Accounts and Peripherals*

# Ports



Connect modem or printer ports



Run `setport` (under A/UX)

Run MacTerminal (under Mac OS)

3

## Notes

# /etc/inittab

As mentioned previously, the system decides which processes to run at the beginning of a boot session based on the contents of the file /etc/inittab. The lines in this file that are of interest are:

```
00:2:off:/etc/getty tty0 at_9600 # Port 0 (modem)
01:2:off:/etc/getty tty1 at_9600 # Port 1 (print)
```

We previously discussed this file's structure (in Module 2) so we don't need to spend a lot of time with it now. Suffice it to say that to allow login sessions, change the action (the third field) to "respawn" on the lines with ids (the first field) 00 to 01. To disallow login sessions, change the action to "off."

After editing the file, the `init` command must be run so the system can be identified of the changes:

```
     # init q
```

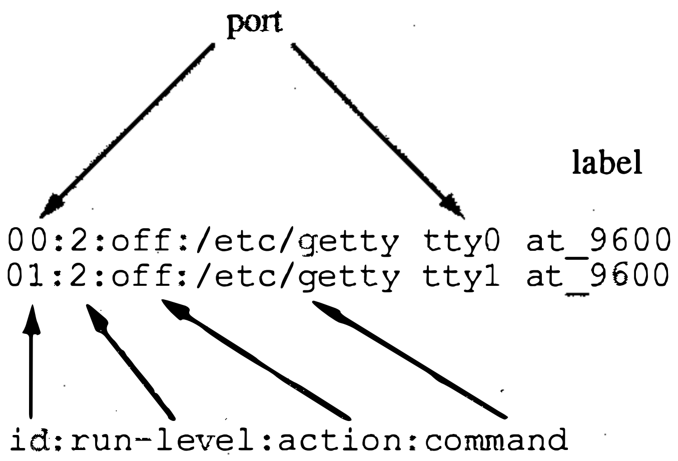
The speed of the port is controlled by its label, in the above cases `at_9600`. This label points to an entry in the /etc/gettydefs file. We will examine this file soon.

We can edit this file to change the parameters. However, A/UX 2.0 has provided a means to easily modify this file, so we need not concern ourselves with editing it. The new command `setport` does this for us.

**References**

*A/UX Local System Administration*
*Setting up Accounts and Peripherals*

# /etc/inittab



port

label

```
00:2:off:/etc/getty tty0 at_9600
01:2:off:/etc/getty tty1 at_9600
```

id:run-level:action:command

4

## Notes

# The `setport` Command

Using the Commando interface to the `setport` script makes setting and resetting ports extremely simple. For those hardy souls who want to use the command-line interface directly, use this syntax:

**Syntax**

```
setport -r [-s speed] device
setport -o [-s speed] device
```

**Options**

`-r` (respawn) this option sets the action of *device* to allow logins

`-o` (off) this option sets the action of *device* to disallow logins

`-s` this option sets the baud rate of *device* to *speed*

The following command edits the inittab file and reinitializes the ports:

**Example**

```
# setport -r -s2400 tty0
```

This command allows login sessions on tty0, the modem port, at 2400 baud.

**Disabling AppleTalk**

The printer port is preconfigured to run AppleTalk when A/UX is installed. Before communication on that port with a terminal, modem, or serial printer can occur, this must be disabled. To do this:

**Step 1**

Take the AppleTalk interface down:

```
appletalk -d
```

**Step 2**

Open the file /etc/appletalkrc with an editor, such as `TextEditor`. Change the word

`localtalk0` to **`ethertalk0`**

Save the file.

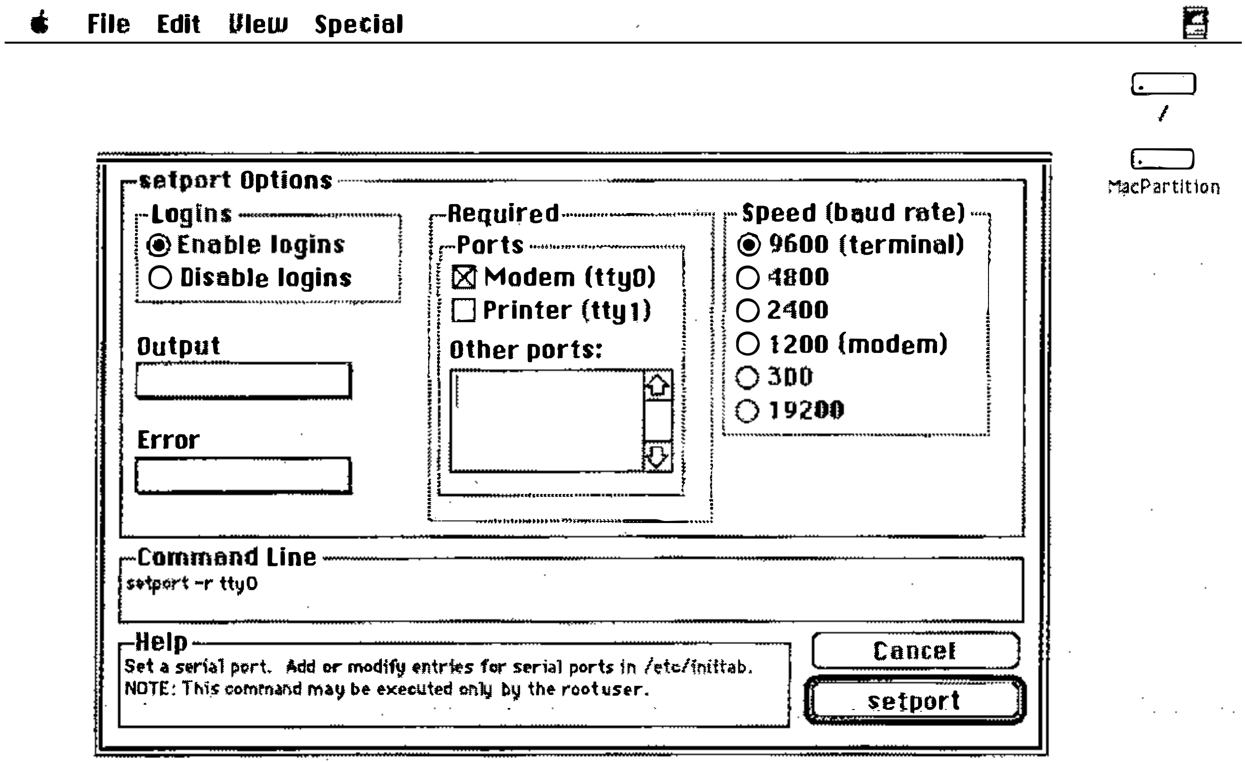
**Step 3**

Bring the AppleTalk interface back up:

```
appletalk -u
```

**Step 4**

Log off, then log back on.



setport Options

Logins
● Enable logins
○ Disable logins

Output

Error

Required
Ports
☒ Modem (tty0)
□ Printer (tty1)
Other ports:

Speed (baud rate)
● 9600 (terminal)
○ 4800
○ 2400
○ 1200 (modem)
○ 300
○ 19200

Command Line

```
setport -r tty0
```

Help
Set a serial port. Add or modify entries for serial ports in /etc/inittab.
NOTE: This command may be executed only by the rootuser.

# The `setport` Command

Trash

## Notes

# /etc/getty

One of the programs that controls the login over serial ports is `/etc/getty`. This command is not invoked from a command line, but is called by `init` from /etc/inittab. `getty` generates the login message shown on a terminal, accepts a user's login name, and sends it to the program `login`. (`login` then calls the appropriate shell.)

`getty` sets the terminal type, speed, and line discipline for serial lines on an A/UX system.

**Syntax**

```
getty [-t timeout] line [speed]
```

**Options**

`-t` if nothing is typed on the line in *`timeout`* seconds after the line is opened, getty exits.

*`line`* the name of the file in /dev to be opened for reading and writing.

*`speed`* a label into the file /etc/gettydefs that defines the initial speed, what the login message looks like, the initial tty settings and the next label to try if the user indicates the speed is wrong.

**Note**

There are other options to `getty` that are not listed here as they are not often used.

**References**

*getty (1M)*

# /etc/getty

- ❑ Monitors serial ports waiting for someone to log on
- ❑ Invoked by `init` from inittab
- ❑ Never invoked directly
- ❑ References /etc/gettydefs to set line discipline

6

## Notes

# /etc/gettydefs

The /etc/gettydefs file contains information used by *getty(1M)* to set up the speed and terminal settings for a line. It supplies information on what the login prompt should look like. It also supplies the speed to try next if the user types an interrupt character (indicating the current speed is not correct.)

Each entry in /etc/gettydefs has the following format:

**Format**

label# initial-flags # final-flags # flow-control # login-prompt #next-label

**Label**

This is the second argument to getty, usually the speed of the line.

**Initial-flags**

This is the speed at which initial communication with the terminal is performed.

**Final-flags**

There are terminal control codes sent to the terminal to define the serial protocol for communication. These are sent just before login.

**Login prompt**

This entire field is printed as the login prompt.

**Next-label**

The next label to try if the user types a BREAK character.

**Example Entry**

```
at_9600# B9600 # B9600 TAB3 # ~MODEM DTR
#\r\n\nApple Computer, Inc. A/UX\r\n\nlogin:
#at_4800
```

**References**

*A/UX Local System Administration*
*Setting up Accounts and Peripherals*
*gettydefs (4)*

# /etc/gettydefs

## Example

```
at_9600# B9600 # B9600 TAB3 # ~MODEM DTR
#\r\n\nApple Computer, Inc. A/UX\r\n\nlogin:
#at_4800


mo_2400# B2400 # B2400 HUPCL TAB3 # MODEM ~DTR
#\r\n\nApple Computer, Inc. A/UX\r\n\nlogin:
#mo_1200
```

label# initial-flags # final-flags # flow-control # login-prompt #next-label

7

## Notes

# Checkpoints

1 Which files are involved with setting up serial communications?

2 What command modifies inittab to set up the serial ports?

3 What file defines the serial line characteristics?

4 What steps are there to setting up the system for serial communication on the Macintosh printer port?

# Exercises

**Note**

The class may be broken into teams during these exercises and the system connected together in pairs. If there is an odd number of systems, three systems can be connected together (one each on the printer and modem port.)

**Exercise 1**

**Step 1**

Decide which computer will run the Macintosh Operating System (hereafter called the Mac system) and which computer will run A/UX (hereafter called the A/UX system.)

**Step 2**

Connect the two system's modem ports together with the appropriate serial cable.

**Exercise 2**

**Step 1**

Boot into the Macintosh Operating System on the Mac system. Launch the application MacTerminal (provided on the Training Setup 1 floppy disk.)

**Step 2**

Configure MacTerminal to have the following characteristics:

```
Terminal type:           VT100
Mode:                    ANSI
Line width:              80 columns
Baud rate:               9600
Bits per character:      8
Parity:                  None
Handshake:               XON/XOFF
Connection:              To another computer
Connection port:         modem
File transfer protocol:  text
```

**Exercise 3**

**Step 1**

Boot into the A/UX system.

**Step 2**

Run `setport` to allow a login session on the modem port. (Remember Commando!)

## Exercise 4

**Step 1** Log into the A/UX system from the Mac system.

**Step 2** Try running the program `/usr/games/worms` from the Mac terminal. What happens?

**Step 3** Echo the file /etc/gettydefs from the A/UX system to the Mac system.

**Step 4** Echo the file /etc/gettydefs from the Mac system to the A/UX console.

**Step 5** List all the files in the A/UX file system on the Mac system screen.

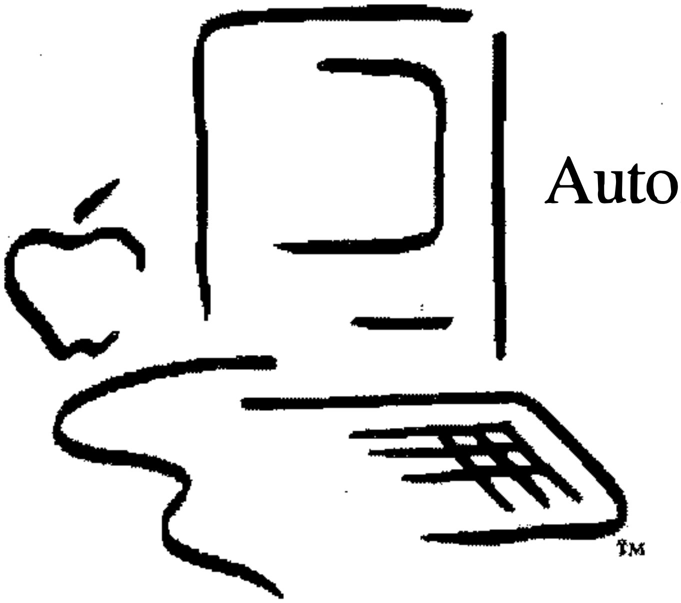
**Step 6** Run `setport` to disallow login sessions on the modem port while the Mac system is still listing files. What happens?

**Step 7** Shut down both systems.

# Module 8
# Autorecovery

## Alternate Learning Path

# Table of Contents



# Autorecovery

A/UX® 2.0

## Notes

# Autorecovery

## Goal

In this module you will explore **autorecovery**, an A/UX feature that can recover damaged file systems in the event of a severe system failure. In addition, the commands that are used to maintain currency in the autorecovery file systems are discussed and put to work in the exercises.

## Objectives

- describe the features and benefits of autorecovery
- explain autorecovery capabilities and limitations
- describe the administrative procedures necessary to maintain autorecovery
- use the autorecovery commands `esch`, `eu`, `eupdate`, and `escher`
- perform basic autorecovery maintenance procedures
- recover a damaged file system with autorecovery

## Activities

Lecture/Discussion/Checkpoints/Exercises
Exercise 1: Autorecovery Overview
Exercise 2: Autorecovery Maintenance
Exercise 3: Using Autorecovery

## References

*A/UX Release Notes*
*A/UX Local System Administration*
*A/UX System Administrator's Reference*

# Objectives

- ❑ Describe the features and benefits of autorecovery
- ❑ Explain autorecovery capabilities and limitations
- ❑ Describe the administrative procedures necessary to maintain autorecovery
- ❑ Use the autorecovery commands
  - `esch`
  - `escher`
  - `eu`
  - `eupdate`
- ❑ Perform basic autorecovery maintenance procedures
- ❑ Recover a damaged file system with autorecovery

2

## Notes

# Autorecovery

Autorecovery refers to built-in A/UX facilities that automatically check and repair or replace critical, specific A/UX system files in the event of a serious system failure.

A/UX (and UNIX® in general) is a very stable and reliable environment. But damage to file systems can occur from a number of causes, such as power failure, hardware problems, operator error, or malicious mischief such as **worm** or **virus** programs. Autorecovery is intended to give the A/UX system administrator a tool to recover the system to an operational state in the event of damage to critical system files. Autorecovery helps ensure that an A/UX system can always be brought up in multi-user mode and be able to operate as part of a network. You may occasionally hear reference to the term *eschatology*. This refers to an old name for autorecovery.

Autorecovery is always run from the A/UX Startup utility. Although you can configure your system to run autorecovery automatically whenever A/UX is launched, this is not recommended. Since much of autorecovery is equivalent of `fsck`, `fsck` alone will often suffice to take care of any inconsistencies.

Depending on the options you specify, autorecovery takes from 10 minutes to nearly an hour, with the routine case being closer to 10 minutes.

Autorecovery does not take the place of regular system backups. It does not recover user files at all, only specific system files.

**References**

*A/UX Local System Administration*
*badblk (1M,) fsck (1M,) autorecovery (1M)*

# Autorecovery

❑ Can help recover damage caused by:

- Power failure
- Hardware problems
- Operator error
- Malicious mischief

3

## Notes

# What Autorecovery Does

A separate area on the A/UX disk is reserved for autorecovery, containing a distinct A/UX file system that contains copies of key system files and other information about A/UX. These files are listed in the Configuration Master List (CML,) which is kept in the file /etc/cml/init2files; a copy of this file is also stored in the autorecovery file system. The CML files can be updated and additional files can be added by the superuser. Other important files in /etc/cml are otherfiles and descriptions. Refer to the man page for cml for specific information on records in the init2files and otherfiles files.

### How autorecovery works

Autorecovery verifies the physical condition of the disk and checks each file in the CML, using rules in the CML to check (and correct, if necessary) attributes such as size, ownership, permissions, type, modification, version, and checksum.

Specifically autorecovery:

- Checks its own file system; if it is damaged, autorecovery does not continue (rarely the case.)
- Checks each A/UX file system to verify that blocks are readable; marks bad blocks.
- Checks the root file system for consistency (with a version of `fsck`) and corrects any errors it finds. If it cannot correct errors, it remakes the file system. In the latter case, all data in the file system is lost and must be restored from backups (this is usually *not* the case.)
- Verifies the key system files listed in CML; corrects inconsistences or replaces files if necessary

### References

*A/UX Local System Administration*
*cml (4)*

# What Autorecovery Does

- ❑ Maintains a (small) separate partition on disk solely for system recovery
- ❑ Checks the Autorecovery file system
- ❑ Checks each A/UX file system
- ❑ Checks and corrects (if necessary) the root file system
- ❑ Verifies files in the CML; fixes or replaces files as necessary

4

## Notes

# What Autorecovery Does Not Do

Autorecovery does not work magic. It is possible that file systems are so damaged that autorecovery will not work. In a catastrophic case (for example, smashing your hard disk with a mallet,) damage could occur to critical system files, the root file system itself, and the autorecovery file system. However, with proper system maintenance and preventative procedures, such severe system failures will indeed be rare. It is also possible that the damaged or missing files are not on the autorecovery list, or that autorecovery is unable to determine that a file is damaged (for example, a text file.)

Autorecovery does not monitor or update itself. It requires regular attention by the system administrator to ensure the viability of the autorecovery system, including running the appropriate update commands on a regular basis, updating the CML, and following generally accepted UNIX security and maintenance procedures.

Interrupting autorecovery while in process can severely damage the autorecovery file systems and the file systems it is intended to protect.

Autorecovery does not update user file systems. Even with autorecovery, it is the responsibility of each user and the system administrator to regularly back up files and file systems.

**References** *A/UX Local System Administration*

# Autorecovery Limitations

- ❑ Won't work if the Autorecovery file system is damaged
- ❑ Won't update user file systems
- ❑ Regular system maintenance still required
- ❑ Interrupting the autorecovery process can damage file systems

5

## Notes

# Autorecovery Administration

It is important to keep the autorecovery file system up to date. Facilities are provided for doing so with minimal effort, and if it is not kept up to date it will be less helpful if and when it is needed.

Autorecovery administration involves two key tasks:

- updating the CML when key system files change
- updating the autorecovery copies of key system files when they change

There are four basic utilities for autorecovery use and administration:

| | |
|---|---|
| `esch` | File system repair (Startup only) |
| `escher` | Notification and update of autorecovery file changes |
| `eu` | Update specific autorecovery files |
| `eupdate` | Special autorecovery administration |

**References**

*A/UX Local System Administration*
*esch(1M,) escher(1M,) eu(1M,) eupdate(1M)*

# Autorecovery Administration

□ Update CML

□ Update autorecovery file system

□ Tools:

- esch
- escher
- eu
- eupdate

A/UX

6

## Notes

# esch Command

The `esch` command invokes autorecovery, and can only be run from the `Startup` utility.

**Syntax**

```
esch [-b] [-cclusternumber] [-f] [-v]
```

If no options are specified, `esch` includes the`badblk` and `fsck` functions by default, which will take at least 45 minutes to run.

**Note**

Once esch is started, it must not be interrupted.

If you include the `-b` option and don't check for bad blocks, `esch` takes about 10 minutes to complete.

**Options:**

`-b` Turn off the `badblk` check of the disk. Each block of the disk can be checked by reading from it. It is useful to run `badblk` on occasion to ensure disk integrity, though it is a time-consuming process.

`-c` Specify a different cluster number. The cluster number is 0 by default, and includes the root, swap, and autorecovery partitions on the system disk.

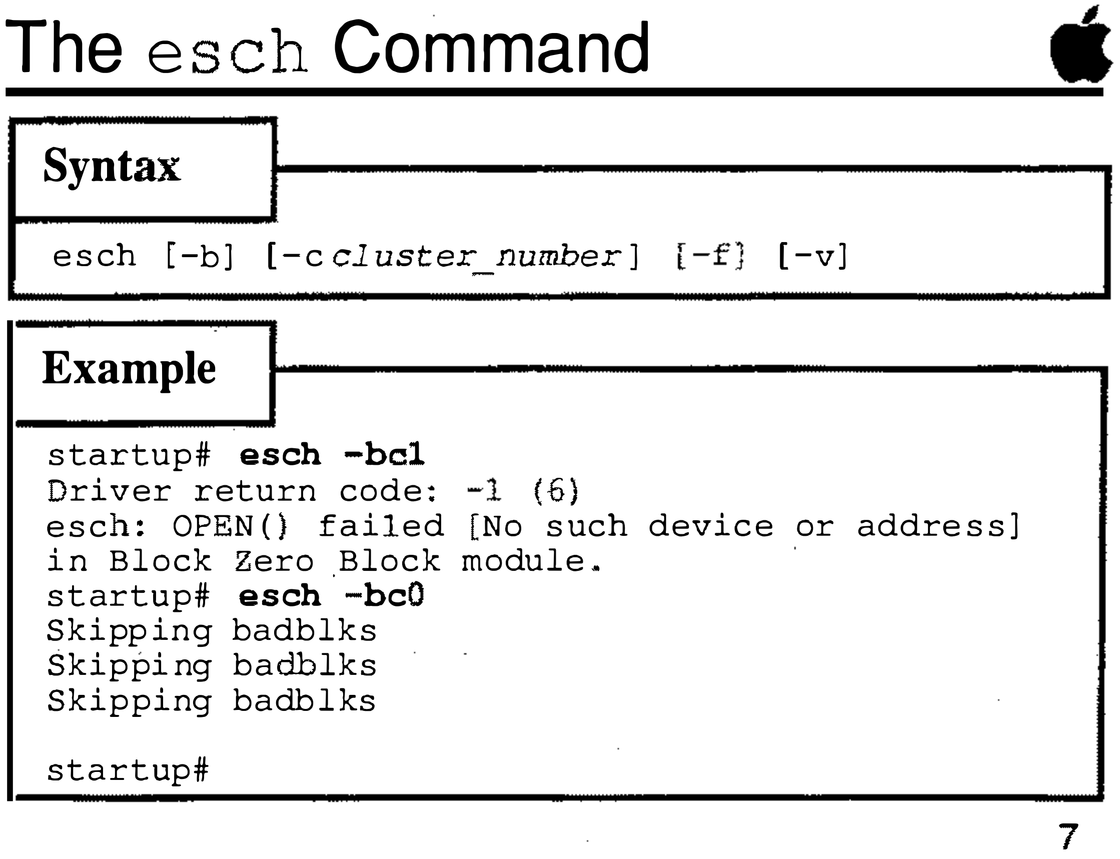
`-f` Turn off `fsck`. Otherwise, a version of `fsck` is run, which is similar to performing an `fsck -y` on all file systems. If `fsck` cannot be performed on the specific file system, a new file system will be built. If you are reluctant to perform an `fsck -y`, `fsck` can be run separately from the `Startup`.

`-s` Save superblock. Never remake the filesystem (which would destroy the previous superblock.)

`-v` Verbose mode. Print information regarding autorecovery activity on the console.

**References**

*A/UX Local System Administration*
*esch(8,)fsck(1M,) badblk(1M,) mkfs(1M)*



# The esch Command

## Syntax

```
esch [-b] [-ccluster_number] [-f] [-v]
```

## Example

```
startup# esch -bc1
Driver return code: -1 (6)
esch: OPEN() failed [No such device or address]
in Block Zero Block module.
startup# esch -bc0
Skipping badblks
Skipping badblks
Skipping badblks

startup#
```

7

## Notes

# escher Command

The escher command is used to mail the superuser a list of files needing updating, though it can also automatically update autorecovery files or prompt for updates. The escher command can also add new entries to the CML and copy the specified files into the autorecovery file systems.

**Syntax**

```
escher [-y] [-m]
escher [filename...]
```

**Options:**

-m Mail the superuser a list of files that need to be updated.

-y Update any files needing updating (not recommended).

*filename* Add the file(s) to the autorecovery system. Note that this option is not used like eu for updating files but only for adding new files. Because of extremely limited disk space, this should not be done without careful forethought.

no options escher prompts for yes (y) or no (n) response to each file that needs to be updated.

**Note**

escher does not use all possible cross-checks when updating an entry, in particular checksum is not used. For this reason, using eu to update the CML is preferred.

**Example**

```
# escher -m
```

This mails root a list of files that need updating.

**References**

*A/UX Local System Administration*
*escher(1M)*

# The `escher` Command

## Syntax

```
escher [-y] [-m]
escher [filename...]
```

## Example

```
# escher -m
# mailx
Mail version 1.5 7/7/89.  Type ? for help.
"/usr/mail/root": 1 messages 1 new
>N  1 root  Mon May  7 20:48   68/3033  escher
```

## Notes

# eu Command

The eu command updates an entry for an autorecovery file in the CML, or adds an entry if the specified file is not currently included in the CML. The list of autorecovery files that comprise the CML is contained in /etc/cml/init2files.

### Note

The autorecovery partition has very little free space. Files should not be added to the CML without careful consideration.

### Syntax

```
eu filename
```

eu takes only one file name at a time. You must specify absolute pathnames when you execute the eu command.

### Examples

```
# eu /etc/inittab
```

### References

*A/UX Local System Administration*
*eu(1M)*

# The eu Command

## Syntax

```
eu filename
```

## Example

```
# eu /unix
# eu /etc/inittab
```

9

## Notes

# eupdate Command

The `eupdate` command invokes a shell script that runs several invocations of the `eu` command to update the CML entries for files typically modified by reconfiguring the kernel.

**Syntax**

```
eupdate
```

This command takes a few minutes to run.

`eupdate` should be run every time after successfully using `newconfig` to build a new kernel (such as a networking kernel,) or after you change any of the other files listed below. Specifically, `eupdate` updates:

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

The A/UX kernel and other files necessary for network operation are copied into the autorecovery file systems, and the CML entries for those files are also updated.

**References**

*A/UX Local System Administration*
*eupdate(1M)*

# The eupdate Command

## Syntax

```
eupdate
```

## Example

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

10

## Notes

# Automatic Autorecovery

You can control whether to have autorecovery run whenever you boot A/UX by editing the Booting... dialog box under the Special menu in the `Startup` utility.

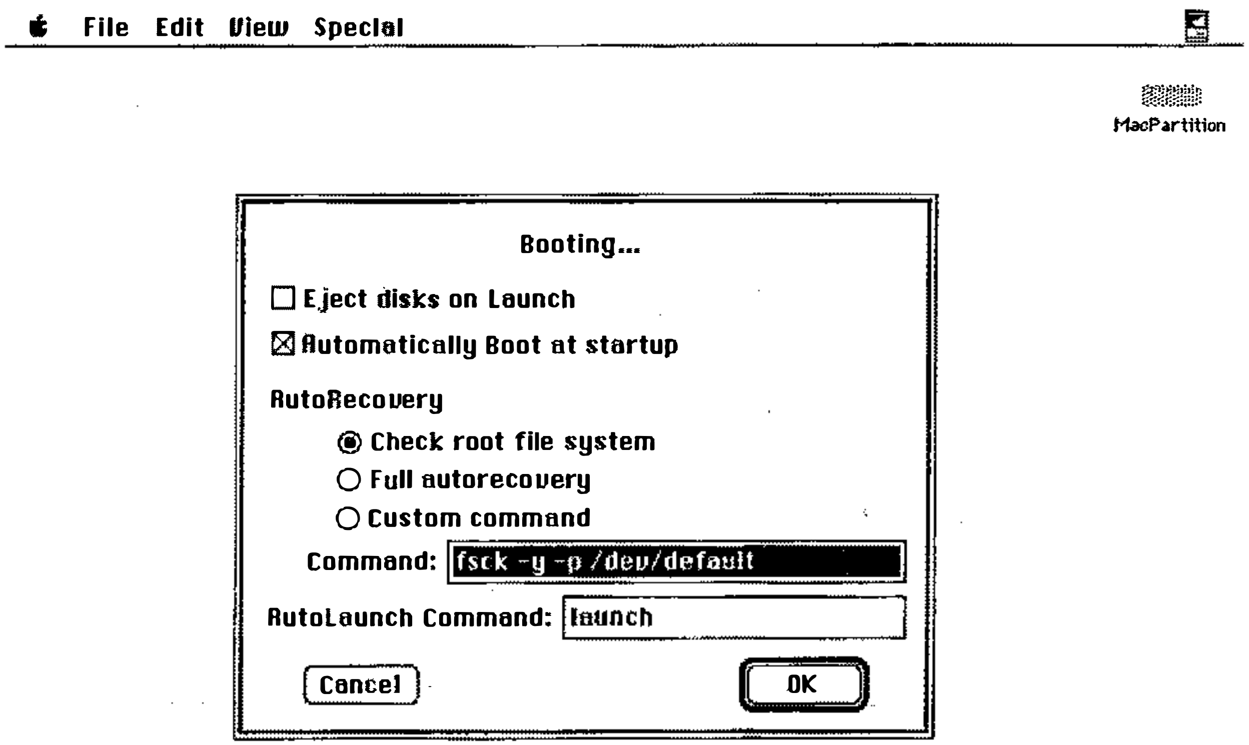
1 Selecting the radio button "Full autorecovery" will cause autorecovery to be performed every time A/UX is launched. Note that the command line invoked is `esch -b`. This is discouraged, as it takes much longer to boot the system using this option.

2 To prevent autorecovery from being run each time you boot A/UX, selecting the radio button labelled "Check root file system." This is the normal option for booting the system.

3 To override autorecovery, manually enter the launch command from A/UX Startup. When entered manually, the `launch` command skips the autorecovery option.

**References**

*A/UX Local System Administration*
*Startup(8)*



Booting...

☐ Eject disks on Launch

☒ Automatically Boot at startup

AutoRecovery

● Check root file system

○ Full autorecovery

○ Custom command

Command: fsck -y -p /dev/default

AutoLaunch Command: launch

# Automatic Autorecovery

## Notes

# Configuration Master List

The three text files in the /etc/cml directory comprise the A/UX Configuration Master List:

**descriptions** Contains a one-line description of all the files shipped with A/UX. This file is linked to /FILES.

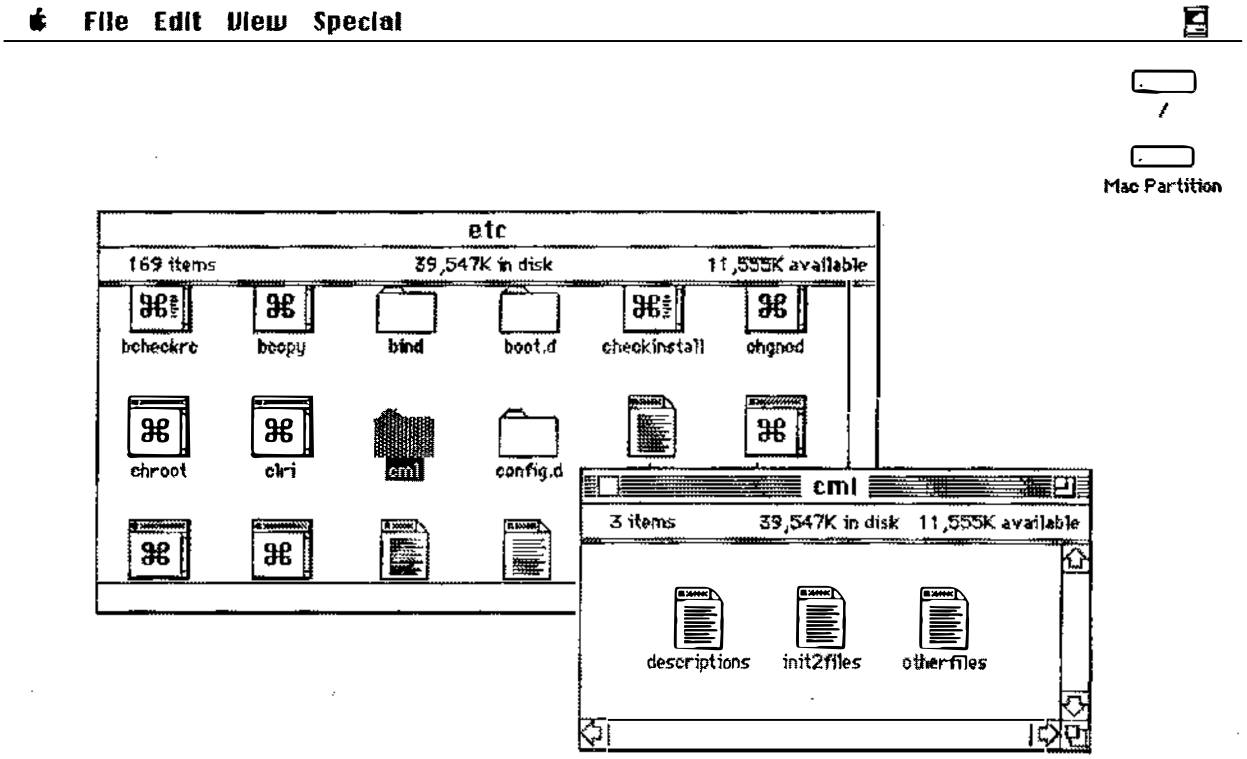
**init2files** Contains a list of each file used by autorecovery, formatted one file per line. This file contains infomation to help autorecovery determine if the files on a system are intact, or in need of replacement by autorecovery.

**otherfiles** Contains a list of all the files shipped with A/UX, formatted one file per line.

The utilities `eu` and `escher` add and maintain entries in the CML. Currently there is no utility available to remove CML entries. Removing CML entries must be done manually. Consult the manual page for *cml (4)* before attempting to remove an entry, or for other information on records in the files init2files and otherfiles.

**Note** The infomation contained in init2files may differ slightly that in otherfiles.

**References** *A/UX Local System Administration*
*cml(4)*



# The Configuration Master List

## Notes

# Checkpoints

1 In general, what does autorecovery do?

2 What are a few limitations of autorecovery?

3 List the four autorecovery commands and briefly state what they do.

4 Where is the master list of autorecovery files located?

5 What is the standard form of the `esch` command (verbose mode on, badblock mode off)?

6 When should you update your autorecovery file systems?

# Exercises

**Exercise 1**

In this exercise you will become acquainted with the autorecovery commands and the files in the autorecovery directory, /etc/cml. At various points you may wish to read the on-line manual pages for the following commands and files: `cml`, `eu`, `escher`, `eupdate`, and `esch`.

**Step 1**

Power on your Macintosh computer and allow A/UX to boot normally. Log on as root.

**Step 2**

Change your working directory to /etc/cml. List the files there. Then examine each of the files. (Since some of these files are both very long and wide, you may want to use the `more` command in a large CommandShell window.)

**Exercise 2**

In this exercise you will run the basic autorecovery maintenance commands, which are necessary before you attempt to run autorecovery itself.

**Step 1**

Run the `escher` command to get a list of those files that may need to be updated.

**Step 2**

Use the `eu` command to update autorecovery for the password file.

**Step 3**

Run the `eupdate` command. This command takes about 2 minutes to complete.

**Step 4**

Read the mail message that has been sent to you from the command you issued in Step 1.

**Exercise 3**

In this exercise you will test the utility of autorecovery. You will render A/UX unbootable by removing both of your bootable kernels (/unix and /newunix) and your inittab file.

**STOP!** **Make sure you have completed all steps in Exercises 1 and 2 before going on.**

**Step 1** Examine your root directory. Notice /unix and /newunix, your two bootable kernels.

**Step 2** Remove the following files: /unix, /newunix, and /etc/inittab. This removes both of your bootable kernel files and your initialization table. The purpose of this step is to create an unbootable system.

**Step 3** To confirm the deletion, examine your root directory again. The files you removed in Step 2 should be gone.

**Step 4** Restart A/UX. At this point you cannot boot your system; A/UX has been rendered inaccessible. You will receive the message:

```
Open on '/unix' failed
```

**Step 5** Launch the backup kernel. At the `startup#` prompt enter:

```
launch /newunix
```

You will receive the message:

```
Open on '/newunix' failed
```

**Step 6** List your root directory. Notice that /unix and /newunix are missing. Similarly, if you try to cat the file /etc/inittab, you will find it missing also.

**Step 7** Enter the following autorecovery command:

```
esch -bvc0
```

**WARNING** **Do not interrupt the `esch` command. Doing so can cause serious damage to file systems.**

**Note** Depending on how the drive has been partitioned, the cluster number may need to be changed. The correct cluster number can be obtained by using `dp`.

**Step 8** After autorecovery has run, list the files in your root directory to confirm that the missing /unix has been replaced.

**Step 9** Restore the file /newunix from the `cpio` floppy disk backup (labelled /newunix) made in the exercise on backups.

**Step 10** Now launch A/UX. You will be able to enter A/UX normally, with your kernel and /etc/inittab file configured as they were before you removed the files.

**Step 11** Shut down your system once you have confirmed you can use the system.

# Module 9
# Security

## Alternate Learning Path

# Table of Contents

# Security

## Notes

# Security

**Goal**

The goal for this module is to provide the system administrator with the tools necessary to create a "secure" A/UX system in stand-alone and networked environments.

**Objectives**

At the end of this module, you will be able to:

- describe general UNIX® security concepts
- list the major user security issues
- list the major A/UX security issues
- list the files important to user and network security, as well as their purpose
- secure the Startup utility; create a minimal MacPartition
- modify system files to increase security

**Activities**

Lecture/Discussion/Checkpoints/Exercise

**References**

*A/UX Essentials*
*A/UX Local System Administration*
*A/UX User Interface*
*A/UX Command Reference*
*A/UX Network System Administration*
*UNIX System Security*

# Objectives

❑ Describe UNIX security concepts

❑ List user security issues

❑ List A/UX security issues

❑ List files important to security

❑ Secure the A/UX Startup utility

❑ Modify files to increase system security

2

## Notes

# A/UX Security

Security is a major concern of all UNIX system administrators. A/UX system administrators need to be familiar with typical UNIX security in addition to certain security issues unique to A/UX. In general, the A/UX system administrator should be familiar with the following:

**General UNIX security practices**

- User administration; password maintenance
- Administrative procedures aimed at increasing security

**User security**

- File/directory protections
- User education about passwords
- User files that affect security

**A/UX-specific security issues**

- Initial password protection
- Access to global system folder
- Access to A/UX file system through A/UX Startup
- Initial boot to multi-user mode
- Physically restricting disks, SCSI drives, entire system

**Remote access security**

- Protecting the system from unauthorized network access
- UUCP issues

There is generally an inverse relationship between the security of systems and ease of administration. Each administrator must decide for themselves where this trade-off balances.

# A/UX Security

- ❑ General UNIX practices
- ❑ User issues
- ❑ A/UX-specific issues
- ❑ Remote access issues

3

## Notes

# General UNIX Security Issues

The A/UX manuals and listed additional resources provide an excellent resource for information on UNIX security. The following are some general guidelines for implementing a security plan.

**Passwords**

Password/login security: Each user should have a unique login name, and every login name must have a password. Remember to maintain different passwords for accounts on different machines. Discourage users from giving away login names or passwords. Encourage users to use non-trivial passwords which they change frequently. Make sure that users cannot write to /etc/passwd.

**Log Off**

Don't leave the console unattended when you boot the system—follow the A/UX Startup program through to the login dialog box. When you leave after logging on, either log off or secure your terminal with a terminal-locking init or DA. Consider physically restricting your system.

**Permissions**

Keep the integrity of your system files secure. Make sure the permissions of the directories /, /bin, /usr/bin, and /etc do not have write permission on them for users.

**References**

*A/UX Local System Administration*
*su(1M)*

# General UNIX Security Issues

- ❑ Manage passwords, logins
- ❑ Never leave console unattended - log out
- ❑ Secure system file/directories

4

## Notes

# Administrative Procedures

There are several procedures that can check for unusual activity on the system.

**Check usage**

Look out for usage patterns, such as large disk usage, large amounts of CPU time, frequent logins, etc. The A/UX system accounting package can help you determine these statistics. This package is covered in a later module.

**Monitor sulog**

Attempts to use the `su` command (substitute user) to log on as root are written to the file /usr/adm/sulog, whose permission is 600. Monitor this file.

**setuid and setgid programs**

It is possible under UNIX to set up commands that act as if they were being invoked by a specific user or a member of a specific group. The mechanism for this is simple: a setuid command takes on the user ID of its owner (the owner of the file that is being executed.) The setgid commands function similarly but take on the group ID of the executed file.

Use the `file` utility to locate all files with set-user-id (setuid) or set-group-id (setgid) permissions (2XXX, 4XXX, 6XXX). There should only be a few of these files owned by the system logins (root, adm, daemon, etc.) They should not have write permission for anyone but the owner, and they should not change without your knowing about it.

If any user approaches you with a setuid program they claim *must* be owned by root to function correctly, it is more than likely there is a security hole embedded in it somewhere.

These and other procedures are listed in *Tightening Up Your System*, later in this module.

**References**

*A/UX Local System Administration*
*su(1M)*

# Administrative Procedures

- ❑ Monitor usage patterns
- ❑ Monitor /usr/adm/sulog
- ❑ Validate set-uid and set-gid programs

A/UX

5

## Notes

# User Security

The following are some basic security guidelines for A/UX users.

**Set directory permissions**

Set the permissions on your home directory, and all subdirectories, to restrict access to other users. In particular, your directories can be set so that no other user has write permission in it. Individual file permissions are meaningless (from a security standpoint) when one has write permission on the directory containing them. Files are invisible to other users when stored in directories whose permissions are 700.

**Set file permissions**

Set appropriate permissions for files within your directories.

**Set umask**

An environment variable called umask sets the initial permissions on files you create. Set your umask initially (in .profile or .login) to 022. This sets newly-created files to permission 755.

**Password awareness**

Use "good" passwords and keep them safe and secret. Change your password regularly.

**Log off**

Don't leave your terminal unattended. When you are done working, log off and wait for the login dialog box to reappear, or secure your terminal with a terminal-locking init or DA.

**For the paranoid**

If you want files thoroughly secret:

- Use `crypt` to encrypt them.
- Store them in a directory whose permission is 700.

**References**

*A/UX Essentials*
*A/UX User Interface*
*chmod (1,) crypt (1)*

# User Security

- ❑ Set file/directory pennissions
- ❑ Set your password – change it regularly
- ❑ Don't leave terminal unattended
- ❑ Set umask in .profile and .login

6

## Notes

# Important Security Files

The following files are important for user security and should be writable only by their owner.

**/etc/passwd**

The most important security file on the A/UX system. Actual passwords are in encrypted form in this file, so read permission is generally made available to system users. Restricting write permission, however, ensures that passwords cannot be removed or replaced except by individual users using the `passwd` command, or by the root superuser.

**/etc/group**

Write permission is also restricted for this file. By adding group memberships, a user would be able to access otherwise restricted files and directories.

**/etc/profile**
**/etc/login**

Write permission is also restricted for these files. These affect the environment of everyone logging on, including the superuser, thus they could harbor some nasty surprises.

**$HOME**

Each user should consider limiting write permission on their home directory (and subdirectories) to themselves.

**.profile**
**.login**

Write permission on these files is limited to each user, for much the same reason given above.

**References**

*A/UX Local System Administration*

# Important Security Files

These files should be writable only by their owner:

/etc/passwd

/etc/group

/etc/profile

/etc/login

Your home directory
and subdirectories

.profile

.login

7

## Notes

# A/UX Security Issues

A/UX value-added features, such as the desktop, Stand-alone Shell and networking packages (Berkeley networking, NFS) create unique security concerns.

**Wake Up!**

At the risk of being obvious, the following must be stated:

**EVERY TIME YOU GIVE THE ROOT PASSWORD TO ANYONE, YOU ARE TRUSTING THEM WITH ALL THE DATA ON YOUR SYSTEM.** ('nuff said.)

**Initial passwords**

A/UX is delivered without passwords for the users root, start, or Guest. These should be set the first time A/UX is booted.

**Personal system folders**

A/UX, as it is delivered, allows all users access to the global system folder so they can customize their desktop. You, as administrator, may want to create personal system folders for each user and restrict access to the global system folder.

**Secure Startup**

Anyone with access to the MacPartition commands has superuser access to the root partition. Secure the Startup utility by removing "dangerous" binaries. (This topic will be handled in detail next.)

**Network awareness**

If the A/UX system is going on a network under Berkeley networking or NFS, make sure the relevant networking files specify only the desired level of access to the local system. This is covered in *A/UX Network Administration.*

**Physical protection**

Since the Macintosh is a relatively small system, consider physically locking up disks, backup tapes, external disk drives, or the entire system if confidential or proprietary information or programs are being stored and processed.

**References**

*A/UX Local System Administration*
*A/UX Network System Administration*
*StartupShell (8)*

# A/UX Security Issues

- ❑ Initial password protection
- ❑ Create personal system folders
- ❑ Control access to A/UX Startup commands
- ❑ Exercise remote access security
- ❑ Consider physically locking disks, CPU

8

## Notes

# Securing the A/UX Startup Application

If a user can gain access to the commands in the bin folder in the MacPartition, then that user essentially has free access to the entire A/UX partition. Access from the A/UX Startup utility completely bypasses normal A/UX (and UNIX) security. Passwords, file and directory permissions, restricted shells, and other security measures are ineffective against access from A/UX Startup.

For example, from A/UX Startup a user could `cat` a new password file in, or do something fun like `rm /unix`. The possibilities for accidental or malicious mischief on the A/UX system are endless, as long as people have unauthorized access to A/UX Startup.

**Minimal MacPartition**

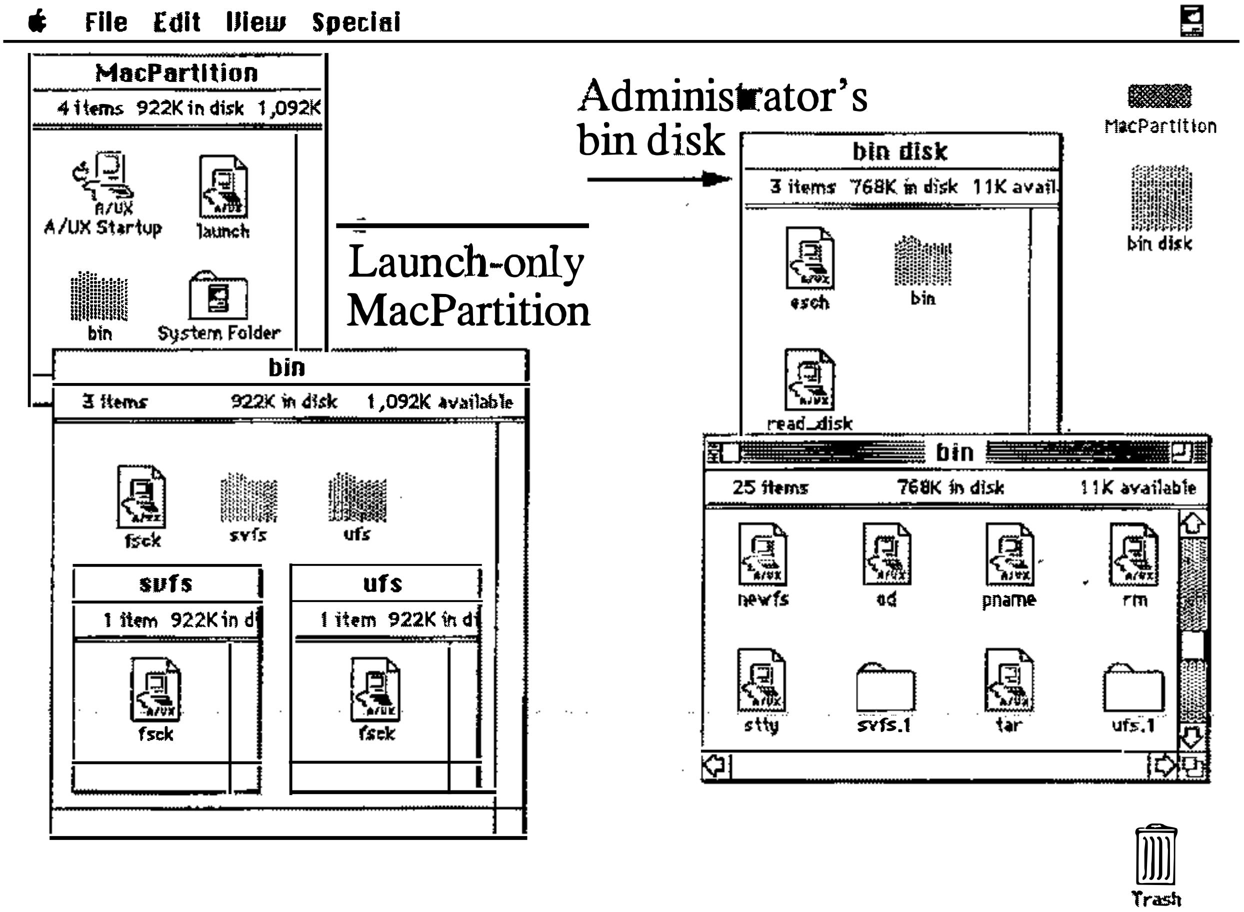
The only commands that need to be in the MacPartition to boot A/UX are A/UX Startup and `launch`. Of all the bin folder commands, A/UX Startup only needs the following:

```
fsck    svfs/fsck    ufs/fsck
```

To secure an A/UX system from unauthorized A/UX Startup access, the system administrator can remove all files from the MacPartition bin folder except the above, thus creating a "launch-only" MacPartition.

**Note**

The `read_disk` and `esch` commands, along with the original bin folder, must be copied to disk (or a hard disk) that unauthorized users would not have access to before commands are removed from the bin folder. Place the bin disk in a secure location. There is not quite enough room on an 800K disk for all the commands, so `svfs/fsck` and `ufs/fsck` are not duplicated on the disk.



Administrator's bin disk

Launch-only MacPartition

# Notes

# Remote Access Security

Two of the most difficult issues of security involve network access to your system and the use of uucp. The Berkeley networking package and NFS are standard with A/UX, as is the serial communications facility uucp (UNIX-to-UNIX Copy.)

**Network access**

When you are establishing network connection with other UNIX systems, it is important to consider security issues that will affect all users.

One of the built-in defaults of the Berkeley networking is that root access over the network is not allowed. If you are also using NFS, you, as system administrator, can allow or deny password-free access to your system by modifying /etc/hosts.equiv. Another important network security file is $HOME/.rhosts, which controls password-free access for individual login names in whose home directory the file resides.

**uucp**

Another troublesome area for some systems is uucp, which links UNIX systems over telephone lines or direct serial lines. Using uucp allows users to transfer files, execute commands on remote machines and send and receive mail. The system administrators on both local and remote computers set permissions regarding user access. These permissions are stored in the file /usr/lib/uucp/USERFILE.

Remote users should only be allowed to copy files to /usr/spool/uucppublic. This file can have write permission for all.

The file /usr/lib/uucp/L.cmds determines what commands the local system will run on behalf of a remote system's request. For tighter system security, limit the commands in L.cmds to rmail, mews and lpr for remote mail, netnews, and remote printing.

**References**

*A/UX Network System Administration*
*A/UX Local System Administration*

# Remote Access Security

- ❑ No root access over Berkeley networks or NFS
- ❑ Password-free login access defined in /etc/hosts.equiv, $HOME/.rhosts
- ❑ Print spooler access defined in /etc/hosts.lpd and /etc/printcap
- ❑ `uucp` permissions defined in /usr/lib/uucp/USERFILE
- ❑ Remote command access defined in /usr/lib/uucp/L.cmds

A/UX

10

## Notes

# Tightening Up your System

The following steps are important for securing your system when it is on a network, such as Berkeley networking or NFS. While this cannot guarantee an impregnable system, it can minimize your risk. For instructions on the correct way to open up access to a networked system, consult the *A/UX Network Administration* course.)

The following should be changed when a new system is installed:

**/etc/passwd** Change both the uid and gid of the user nobody to 65534.

**/.rhosts** `touch` this file to create it, then change the permissions on it to 000.

**/usr/lib/uucp/USERFILE** Change the second line in this file

```
from:    ,     /
to:      ,     /usr/spool/uucppublic
```

**/etc/printcap** Restrict access so that only specific users on remote hosts can send print jobs to your system.

The following should be checked on a regular basis. Many of the steps listed here can be automated by using the `cron` facility and a series of shell scripts.

**/etc/hosts** Periodically verify the address and names of recognized hosts.

**/etc/hosts.equiv** Periodically verify the names of trusted hosts.

**$HOME/.rhosts** Periodically verify that users do not have .rhost files in their home directories.

**/etc/hosts.lpd** Periodically verify the names of remote hosts who are allowed to send print jobs to your system.

**setuid and setgid programs** Periodically verify that the only setuid and setgid programs are those that arrived with the system.

# Tightening Up your System

Modify these files at installation

- /etc/passwd
- /.rhosts
- /usr/lib/uucp/USERFILE
- /etc/printcap

Periodically check these files

- /etc/hosts
- /etc/hosts.equiv
- $HOME/.rhosts
- /etc/hosts.lpd
- setuid and setgid programs

11

## Notes

# Checkpoints

1 List the two unique security issues for A/UX.

2 Describe two methods of physically restricting access to your system.

3 How can you completely prevent users from seeing certain files?

4 How do you make a minimal MacPartition folder?

5 List two networking/communications security issues.

# Exercises

**Exercise 1** In this exercise you will tighten up the security of your system.

**Step 1** Power up your A/UX system. Log on as root.

**Step 2** Edit the password file to change both the uid and gid of the user nobody to 65534.

**Step 3** Create a zero-length file called /.rhosts. Change the permissions on the file to 000.

**Step 4** Edit the file /usr/lib/uucp/USERFILE. Change the second line in this file:

```
from:    ,      /
to:      ,      /usr/spool/uucppublic
```

**Exercise 2** In this exercise you will perform a basic system security routine by creating a “launch-only” MacPartition.

**Step 1** Power up your A/UX system.

**Step 2** Cancel the Startup utility by hitting COMMAND-PERIOD while the •A/UX Release 2.0• copyright notification is showing.

**Step 3** Open the MacPartition disk.

**Step 4** Insert a floppy disk and drag the MacPartition bin folder into its icon.

**Step 5** Open the disk icon, then open the bin folder in it. Remove the fsck icon from both the svfs and ufs folders and drag it to the Trash. Rename the folders svfs.1 and ufs.1 respectively.

**Step 6** Close the bin folder on the disk. Drag the commands `read_disk` and `esch` from the MacPartition window to the disk, then eject the disk. Label the disk MacPartition/bin.

**Step 7** Drag the icons read_disk and esch from the MacPartition window to the Trash.

**Step 8** Open the bin folder in the MacPartition window.

**Step 9** Open the svfs folder in the bin window.

**Step 10** Drag the icons fsdb and mkfs from the svfs window to the Trash. Close the svfs window.

**Step 11** Open the ufs folder in the bin window.

**Step 12** Drag the icons fsdb and mkfs from the ufs window to the Trash. Close the ufs window.

**Step 13** Select all the icons in the bin window. Deselect the svfs and ufs folders. Drag all the selected icons to the Trash. Choose Empty Trash from the Special menu.

**Step 14** Double-click on the A/UX Startup icon.

**Step 15** Cancel the Startup utility by hitting COMMAND-PERIOD while the •A/UX Release 2.0• copyright notification is showing.

**Step 16** Try giving the `ls` command to list the contents of your root directory. The system will respond:

```
Couldn't execute ls
```

**Step 17** Double-click the A/UX icon again and allow A/UX to launch. At the login dialog box, log on as root. This demonstrates it is indeed possible to boot from the minimal MacPartition.

**Step 18** When done with this exercise, shut down your system.

# Important User Security Files: Initial States

## /etc/passwd

```
root::0:0::/:/bin/sh
daemon:xxxxxxxxxxxxx:1:1::/:
bin:xxxxxxxxxxxxx:2:2::/bin:
sys:xxxxxxxxxxxxx:3:3::/bin:
adm:xxxxxxxxxxxxx:4:4::/usr/adm:
uucp::5:5:UUCP admin:/usr/spool/uucppublic:
lp:xxxxxxxxxxxxx:7:7:lp:/usr/spool/lp:
ftp:xxxxxxxxxxxxx:8:2:ftp:/usr/spool/ftp:
who::22:0:who command:/bin:/bin/who
nobody:xxxxxxxxxxxxx:65534:65534:NFS generic user:/tmp:/bin/noshell
Guest::90:90:A/UX Guest account:/users/Guest:/bin/csh
start:PG/qLJaYo/6mo:100:100:Initial login:/users/start:/bin/csh
```

## /etc/group

```
root:*:0:
daemon:*:1:
bin:*:2:
sys:*:3:
adm:*:4:
uucp:*:5:
lp:*:7:
mail:*:8:
staff:*:50:
perm:*:60:
temp:*:70:
contract:*:80:
guest:*:90:
project:*:100:
```

# Module 10
# The Kernel

## Alternate Learning Path

# Table of Contents

# The Kernel

A/UX®2.0

## Notes

# The Kernel

**Goal**

The goal of Module 10 is to discuss the process of adding new device drivers to the kernel, and how the kernel may be adjusted for better performance.

**Objectives**

By the end of this module, participants will be able to:

- explain the process of autoconfiguration
- build a new kernel with `newconfig`
- modify a parameter using `kconfig`
- list three tunable parameters and the conditions under which they should be adjusted

**Activities**

Lecture/Discussion/Checkpoints/Exercises

# Objectives

- ❑ Explain the process of autoconfiguration
- ❑ Build a new kernel with `newconfig`
- ❑ Modify a parameter using `kconfig`
- ❑ List four tunable parameters and the conditions under which they should be adjusted

A/UX

2

## Notes

# Adding Device Drivers

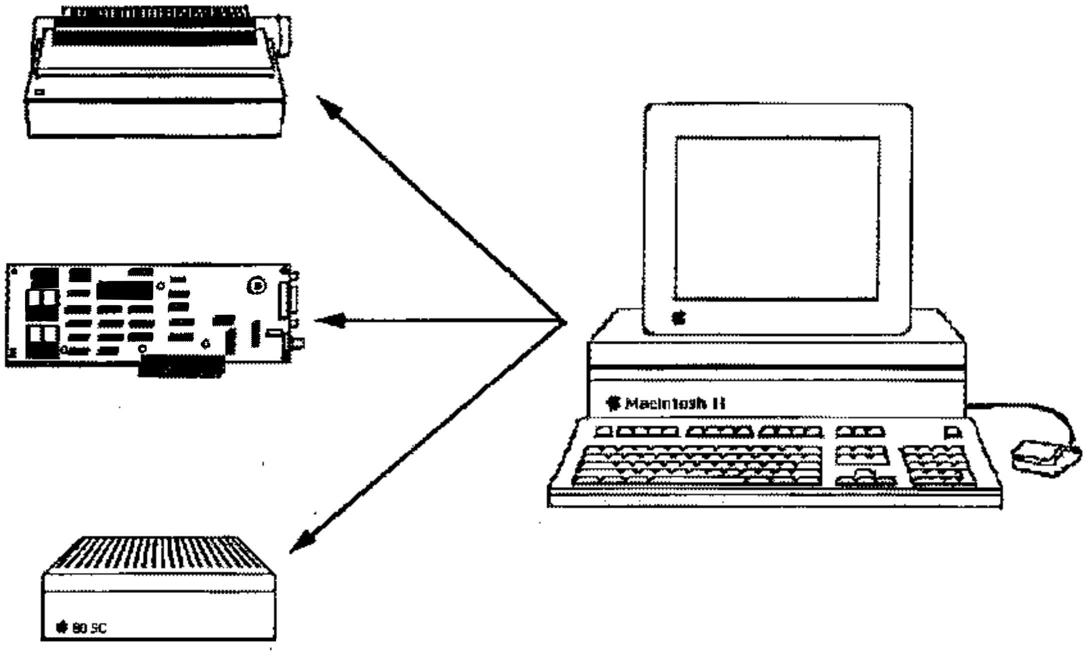
A device driver is software that allows the operating system to access physical devices, such as disk drives and terminals. Since a physical device usually demands attention by interrupting the CPU, the driver code is compiled into the kernel where it can be accessed quickly.

The A/UX kernel, as shipped, contains drivers for printers, terminals, modems, AppleTalk, Apple 40SC Tape Drive, CD-ROM, hard disks, and Apple floppy disks. Included, though not installed, are drivers for various networking facilities, the Apple Sound Chip, and a debugger.

**Note** Device drivers provided for versions of A/UX prior to 2.0 are highly suspect and should not be relied upon. Most third-party vendors will have tested compatible drivers available as of product ship date (June, 1990.)

**References** *A/UX Local System Administration*

# Adding Device Drivers



3

## Notes

# Examining Device Drivers

To see which drivers are in the current kernel, use the command:

**Syntax**

```
module_dump kernel
```

This command dumps information placed in the kernel file by the autoconfiguration process. The following modules can be found:

| | |
|---|---|
| **File System** | ufs, svfs |
| **Serial Line** | tty, scc |
| **Tape Drive** | tc |
| **Sound Chip** | snd |
| **Card Slots** | slots |
| **AppleTalk** | at_atp, at_papd, at_sig, atp, llap, ddp |
| **EtherTalk** | ae6, elap |
| **Networking** | bnet_dr, nfs_dr, slip |
| **User Interface** | toolbox |

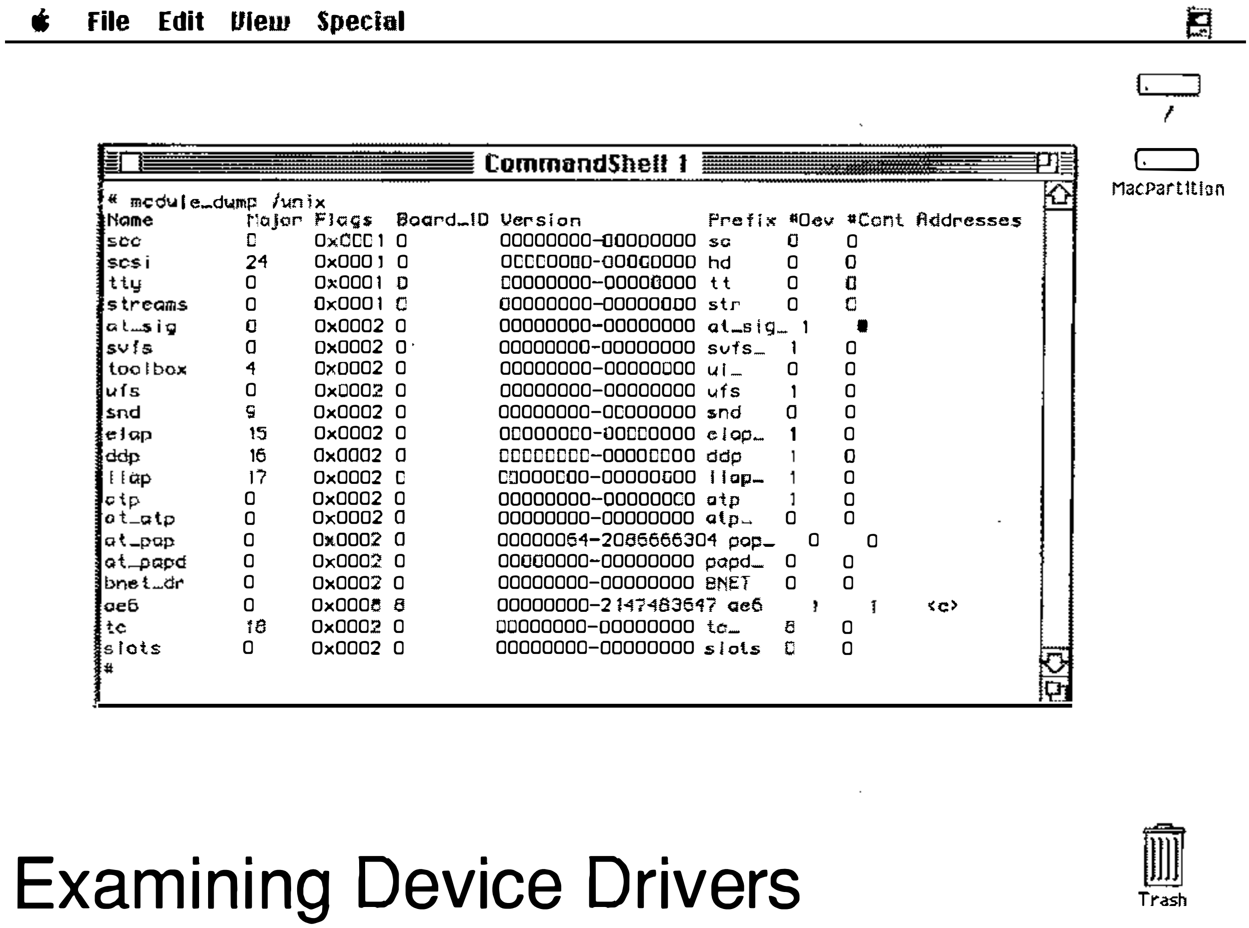
**Example**

To see which drivers are in the current kernel, use the command:

```
module_dump /unix
```

**References**

*A/UX Local System Administration*



```
# module_dump /unix
Name          Major Flags  Board_ID Version            Prefix #Dev #Cont Addresses
scc           0     0x0001 0        00000000-00000000 sc      0    0
scsi          24    0x0001 0        00000000-00000000 hd      0    0
tty           0     0x0001 0        00000000-00000000 tt      0    0
streams       0     0x0001 0        00000000-00000000 str     0    0
at_sig        0     0x0002 0        00000000-00000000 at_sig_ 1    0
svfs          0     0x0002 0        00000000-00000000 svfs_   1    0
toolbox       4     0x0002 0        00000000-00000000 ui_     0    0
ufs           0     0x0002 0        00000000-00000000 ufs     1    0
snd           9     0x0002 0        00000000-00000000 snd     0    0
elap          15    0x0002 0        00000000-00000000 elap_   1    0
ddp           16    0x0002 0        00000000-00000000 ddp     1    0
llap          17    0x0002 0        00000000-00000000 llap_   1    0
atp           0     0x0002 0        00000000-00000000 atp     1    0
at_atp        0     0x0002 0        00000000-00000000 atp_    0    0
at_pap        0     0x0002 0        00000064-2086666304 pap_    0    0
at_papd       0     0x0002 0        00000000-00000000 papd_   0    0
bnet_dr       0     0x0002 0        00000000-00000000 BNET    0    0
ae6           0     0x0008 8        00000000-2147483647 ae6     1    1     <c>
tc            18    0x0002 0        00000000-00000000 tc_     8    0
slots         0     0x0002 0        00000000-00000000 slots   0    0
#
```

# Examining Device Drivers

## Notes

# newconfig

The command `newconfig` is used to add new devices to the kernel. This program is actually a shell script front-end for two other programs: `autoconfig` and `newunix`. While it is certainly possible to run these two programs "by hand," `newconfig` makes it unnecessary.

**Syntax**

```
newconfig [-v] [nonet] [module...] [nomodule...]
```

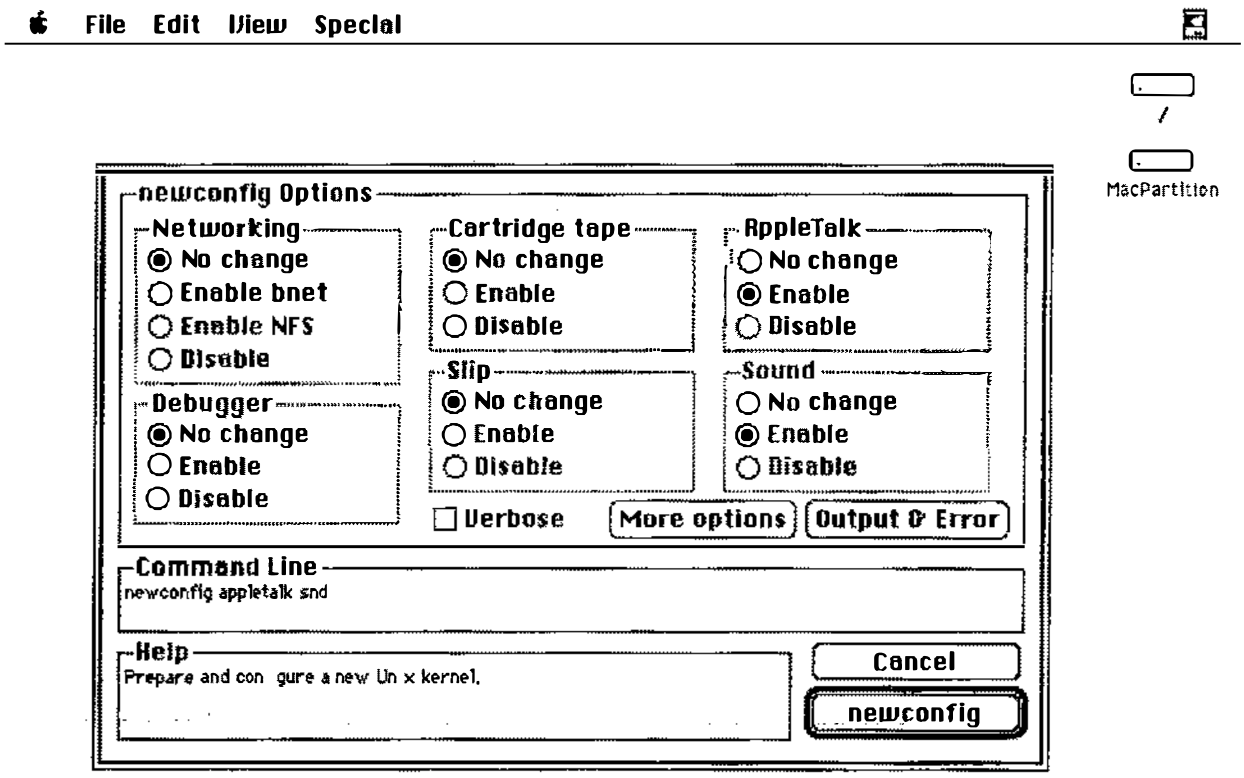
The -v option puts newconfig in verbose mode. The option `nonet` causes the removal of TCP/IP networking capabilities. Specifying a *`module`* causes the named module to be configured in the new kernel; while *`nomodule`* causes the named module to be removed from the new kernel. More than one module can be specified on the command line.

**Note**

When the option `nonet` is used, AppleTalk support is removed also.

**References**

*newconfig(1M)*



**newconfig Options**

Networking
- ● No change
- ○ Enable bnet
- ○ Enable NFS
- ○ Disable

Debugger
- ● No change
- ○ Enable
- ○ Disable

Cartridge tape
- ● No change
- ○ Enable
- ○ Disable

Slip
- ● No change
- ○ Enable
- ○ Disable

AppleTalk
- ○ No change
- ● Enable
- ○ Disable

Sound
- ○ No change
- ● Enable
- ○ Disable

□ Verbose

Command Line

```
newconfig appletalk snd
```

Help

Prepare and configure a new Un x kernel.

newconfig

## Notes

# autoconfig

`autoconfig` is invoked during the startup process to verify the installed driver software matches the installed hardware. If not, autoconfig creates a new kernel to match the hardware.

Drivers for hardware that is not easily checked (for instance, printers, SCSI devices, and floppy drives) are always compiled into the default kernel. Additional drivers may be compiled as well.

**References** *autoconfig(1M)*

# autoconfig

- □ Verifies drivers match hardware during startup
- □ Creates and boots new kernel if match is not found
- □ Standard drivers
  - AppleTalk
  - Hard disk drives
  - Floppy disk drive
  - Cartridge tape drive
  - Modems
  - Printers
  - Network

A/UX

6

## Notes

# Kernel Tuning

The kernel contains several parameters that can be adjusted to provide optimal system performance. On large multi-user systems this can be necessary to preserve adequate throughput.

A/UX is capable of dynamically configuring several of these parameters based on the hardware configuration of the Macintosh. These adjustments cover the majority of user's needs. There are, however, unusual circumstances when these parameters might have to be adjusted by hand.

The parameters that might possibly need to be changed are:

**NBUF** The number of allocated disk I/O buffers. The default value is 0, which tells the system to dynamically allocate 10% of the free memory at boot time.

**NPROC** The total number of processes in the system. The default value is 50, but this is dynamically changed based on the amount of memory in the system. This parameter can be increased if the error message: "proc: table is full" comes up.

**NFILE** The size of the system file-table pool. The default value is 100, but this is dynamically changed based on the amount of memory in the system. This parameter can be increased if the error message: "file: table is full" comes up.

**NINODE** The size of the system inode table. The default value is 100, but this is dynamically changed based on the amount of memory in the system. This parameter can be increased if the error message: "inode: table is full" comes up.

**References** *Local System Administration*
*kconfig(1M)*

# Kernel Tuning

| | |
|---|---|
| NBUF | number of I/O buffers<br>$100 \leq \mathrm{NBUF} \leq 1500$ |
| NPROC | number of proceses<br>$50 \leq \mathrm{NPROC} \leq 200$ |
| NFILE | number of open files<br>$100 \leq \mathrm{NFILE} \leq 400$ |
| NINODE | number of inodes<br>$100 \leq \mathrm{NFILE} \leq \mathrm{NINODE} \leq 400$ |

7

## Notes

# kconfig

The program `kconfig` can read or change kernel parameters on a kernel object file. This does not affect the running kernel; the modified kernel must be booted for changes to take effect.

**Syntax**

```
kconfig [-av] [-nnamelist]
```

The options are:

- `-a` list the current values of the kernel parameters. If the `-a` option is not used, parameters can be changed.
- `-v` used with the `-a` option to produce a commented output.
- `-n` specify that *namelist* is the kernel object file to use. The default is /unix.

If `-a` is not used, standard input is read for a list of changes. You can specify one change per input line using the forms:

```
PARAM  = value    (parameter is set equal to value)
PARAM += value    (parameter is increased by value)
PARAM -= value    (parameter is decreased by value)
```

Where `PARAM` is one of the parameter names (listed on the man pages for `kconfig`) and the`value` is either a decimal constant or a hexadecimal constant preceded by 0x.

**NOTE**

It is recommended that you not change kernel parameters unless you know exactly what you are doing. Incorrect use can cause system failures. (This requires much more knowledge than is covered in this course!)

**References**

*Local System Administration*
*kconfig(1M)*

# kconfig

## Syntax

```
kconfig [-av] [-n namelist]
```

## Example

```
# kconfig -a
NBUF          =       0
NPBUF         =      20
NFILE         =     200
NINODE        =     200
NREGION       =     200
NPROC         =     100
```

8

## Notes

# Checkpoints

1 What command is used to examine the drivers in a kernel?

2 What command is used to build a new kernel?

3 When is `autoconfig` run?

4 What command is used to modify kernel parameters?

5 What are some kernel parameters you might have to modify? How are they usually adjusted?

# Exercises

## Exercise 1

In this exercise we examine the drivers in the current kernel, then build a new kernel incorporating new features and examine the changes.

**Step 1** Power up your A/UX system. Log on as root.

**Step 2** Open a CommandShell window and examine the drivers installed in the current kernel.

**Step 3** Open a second CommandShell window and copy the current kernel to the file /unix.save.

**Step 4** Build a new kernel removing network support.

**Step 5** Compare the drivers installed in the new kernel with those in the old kernel. What capabilities have been removed?

**Step 6** Copy the saved kernel back to the file /unix.

## Exercise 2

In this exercise we examine the kernel parameters in the current kernel, then modify one and examine the changes. The more adventurous can boot the newly modified kernel in the following advanced exercise.

**Step 1** Open a CommandShell window and examine the kernel parameters installed in the current kernel.

**Step 2** Modify the NBUF parameter of the current kernel, setting it to a value of 1500.

**Step 3** Open another CommandShell window and verify the value for NBUF has changed.

## Advanced Exercise 1

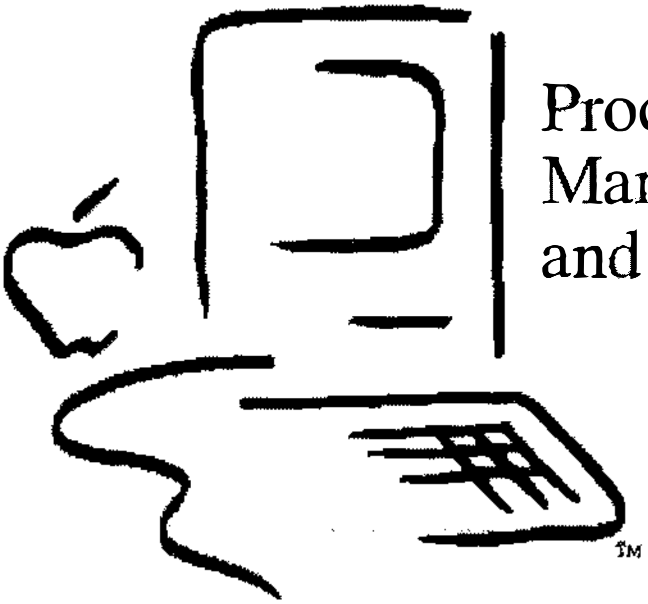
In the last exercise you modified a kernel parameter that, for most systems, probably creates an unbootable kernel. This exercise tests this theory.

**Step 1**

Reboot A/UX. Do not be surprised if the kernel doesn't boot. Recall that you have a saved version of the old kernel that you can move back to its original name by using A/UX Startup. (You will have to drag the `mv` utility from the bin disk created in the Security module back to the MacPartition disk.)

# Module 11
# Process Management and Accounting

# Table of Contents



# Process Management and Accounting

A/UX® 2.0

## Notes

# Process Management and Accounting

**Goal**

The goal of Module 11 is to introduce the participants to the ability of the operating system to run unattended jobs at predefined times.

**Objectives**

By the end of this module, you will be able to:

- define a process
- describe process priority
- list active processes
- explain how to turn on process accounting

**Activities**

Lecture/Discussion/Checkpoints/Exercises

# Objectives

- ❑ Define a process
- ❑ Describe process priority
- ❑ List active processes
- ❑ Kill a wayward process
- ❑ Explain how to turn on process accounting

2

## Notes

# What is a Process?

UNIX® is a multi-tasking system, which means that the operating system can control more than one task at a time. Tasks can be broken down into processes. Each task is completed by the execution of one or more process. While the CPU can only execute one process at a time, several processes can be in memory simultaneously. The CPU can quickly shift between processes, giving the appearance of running them at the same time. When the CPU decides and enforces which program will run next this is known as pre-emptive multi-tasking.

A process is started whenever a program is loaded from the disk into memory. Once in memory the kernel keeps track of the process by assigning a unique process identifier (PID.) We will soon see that we can manipulate our processes via their PID. The kernel then schedules time for the process to be operated on by the systems central processing unit.

**Note** Starting a new process is referred to as spawning a process. When a process creates another process, the new process is called the child. The spawning process is called the parent.

For example, when you log onto your A/UX system on the console you are talking to a program (or process) called Login. It is Login that starts MultiFinder for you. Once you enter your login name and password correctly, the Login process is replaced with the MultiFinder process and from then on you see the Macintosh desktop rather than the Login dialog box.

While you are logged into A/UX you may need to get access to a shell. By opening a CommandShell window we are asking A/UX to start a new shell process for us. The new process does not replace the CommandShell process, but is a child process of CommandShell.

**References** *A/UX Local System Administration*
*Introducing the UNIX System*

# What is a Process?

- ❑ a program in execution

3

## Notes

# Process Priority

Since A/UX is controlling several processes at once, how does it decide which one should be running at which time? This is actually a complex decision made up of several factors. Such things as how long the process has been idle, whether the process has the resources it needs to run, what kind of resources the process may have been waiting for, and the priority of the process are taken into account in **process scheduling**.

We usually only have control over one of the factors, the priority. This is controlled by the `nice` command. The command has this name because regular users can only lower the priority of their processes, thus being "nice." The superuser is the only one allowed to increase a process' priority.

**Syntax**

```
    nice [-increment] command [arguments]
```

The default `nice` increment is 10, the maximum is 19.

**Note** The C shell uses positive instead of negative increments. The result on priority is the same.

**Example**

```
 nice 10 date
```

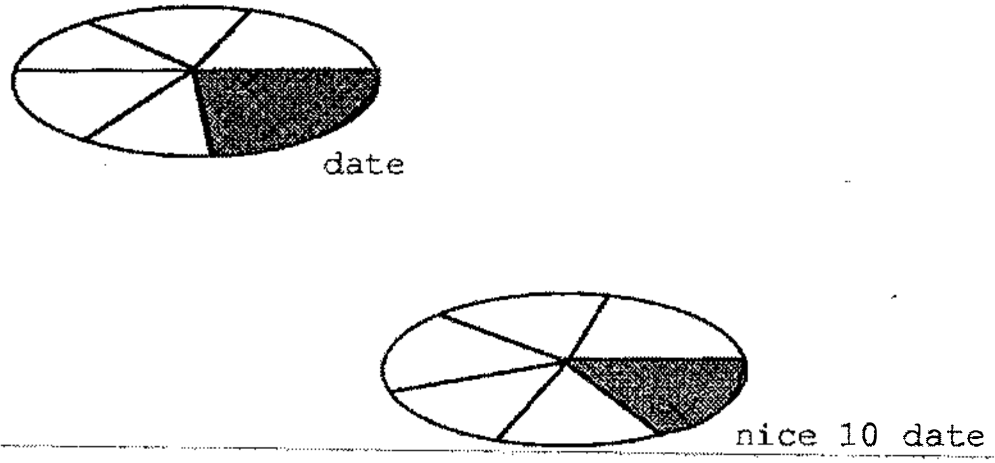
This runs the date command at a lower priority than normal.

Because the MultiFinder can itself be running several programs, and yet is a single process, it is automatically given a higher priority than most processes.

**Note** At times it may appear the system has stopped processing because of a Finder-related event, such as holding the mouse button down. While this does stop screen updates, A/UX continues to run the other A/UX processes.

**References** *A/UX Local System Administration*
*Introducing the UNIX System*

# The `nice` Command



4

## Notes

# Listing the Active Processes

ps is the process status command. It will report information on all current processes. Like most A/UX system commands, ps has many options. Look in the *ps(1)* manual page for more information.

### Syntax

```
ps [-aef] [-u login_name...]
```

Every process has a unique process id (PID). The PID number is used by A/UX to track a process while it is running. The ps command, without any options will list all of the processes that were started in the current window.

### Example 1

```
# ps
   PID TTY        TIME COMMAND
   220 C1         0:01 ksh
   244 C1         0:00 ps
```

The output indicates that two processes, ksh and ps, are running in this window. (C1 is the CommandShell1 window.)

To display a full listing of all of the processes running we would use the options e, l, and f:

### Example

```
$ ps -elf
     UID    PID  PPID TTY        COMMAND
    root      0     0 ?          swapper
    root      1     0 ?          /etc/init
    root      2     0 ?          vhand
    root    104     1 ?          /usr/lib/lpd
    root    101     1 ?          /usr/lib/errdemon
    root    106     1 ?          /etc/portmap
    root    125     1 console    /etc/in.routed
 student    322     1 ?          sh /mac/bin/mac32
 student    340   322 ?          /mac/bin/startmac
 student    341   322 console    /mac/bin/CommandShell -u
 student    342   341 C1         /bin/ksh
 student    343   341 C2         /bin/ksh
 student    427   341 C3         /bin/ksh
    root    345   342 C1         ksh
 student    428   427 C3         /usr/games/worms
 student    429   343 C2         ps -efd
```

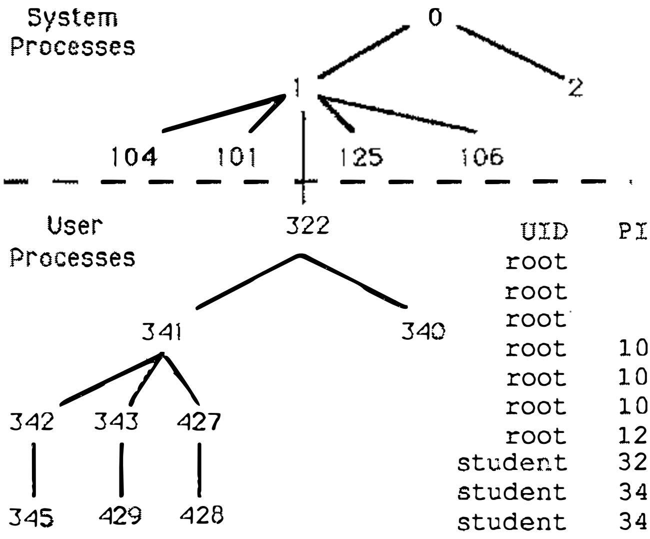
### Note

Columns have been deleted to fit in the space on the page.

### References

*ps (1)*

# Listing the Active Processes



```
     UID   PID  PPID   COMMAND
    root     0     0   swapper
    root     1     0   /etc/init
    root     2     0   vhand
    root   104     1   /usr/lib/lpd
    root   101     1   /usr/lib/errdemon
    root   106     1   /etc/portmap
    root   125     1   /etc/in.routed
 student   322     1   sh /mac/bin/mac32
 student   340   322   /mac/bin/startmac
 student   341   322   /mac/bin/CommandShell -u
 student   342   341   /bin/ksh
 student   343   341   /bin/ksh
 student   427   341   /bin/ksh
    root   345   342   ksh
 student   428   427   /usr/games/worms
 student   429   343   ps -efd
```

5

## Notes

# The kill Command

The output of the `ps -u student` command tells you the process id (PID) of all the commands that you have running. If for any reason you wish to terminate a command forcefully, you use these numbers as arguments to the `kill` command.

**Syntax**

```
kill [-signal] PID
```

Given the output:

**Example**

```
# ps -u student
   PID TTY      TIME COMMAND
   200 console  0:08 sh
   218 console 72:25 startmac
   219 console  6:41 CommandShell
   330 C1       0:01 ksh
   374 C2       0:00 ksh
   376 C1       0:00 ps
```

We have all of the information necessary to `kill` one of our processes. To terminate the `ksh` that is running in the CommandShell2 window we would follow these steps:

**Step 1** Find the name of the `ksh` program under the COMMAND column.

**Step 2** Look at the TTY column to determine which processes are running in CommandShell2 (C2.)

**Step 3** Locate the `ksh` command running on C2, then search back along the line to the PID column to find the process ID. Now you know you want to terminate process number 374.

**Step 4** The command to do this is `kill`, and it is usually executed as follows:

```
kill -9 374
```

**Note** The `kill` command takes signals (special messages to programs) as options. Without the -9 signal, `kill` may not terminate the process.

**References** *ps (1)*

# The `kill` Command

## Syntax

```
kill [-signal] PID [PID...]
```

## Example

```
# kill 374                                # kill -9 374
# ps -u student                           # ps -u student
PID TTY       TIME  COMMAND               PID TTY       TIME  COMMAND
200 console   0:08  sh                    200 console   0:08  sh
218 console  72:25  startmac              218 console  72:26  startmac
219 console   6:42  CommandS              219 console   6:44  CommandS
330 C1        0:01  ksh                   330 C1        0:01  ksh
374 C2        0:00  ksh                   378 C1        0:00  ps
377 C1        0:00  ps
```

6

## Notes

# Process Accounting

In large, multi-user UNIX installations it is common practice to track each person's use of the system for accounting purposes.

A/UX provides a full suite of accounting programs so all the typical functions can be tracked. This can be advantageous even if the costs are not important, because the reports provided by the system can help maintain the security of a system by indicating unusual usage patterns.

The accounting programs track processes; which processes were running, who was running the process, and so on. In addition to this information, the system tracks lower-level activity; how many files were open, percentage of idle time waiting for disk I/O, and so on. While these system activity reports are normally used for benchmarking, they can also help you spot unusual activity.

Initially, process accounting is turned off. To turn the process accounting on:

**Step 1** Edit the file /etc/rc and uncomment (remove the first # character from) the following lines:

```
#    /bin/su adm -c /usr/lib/acct/startup
#    echo process accounting started
```

**Step 2** Edit the file /usr/spool/cron/crontabs/adm and uncomment all the lines in the file. This file instructs `cron` to run the daily accounting.

**Step 3** Reboot your system.

**Note** The file /etc/wtmp, which tracks boot and login data, records information whether process accounting is on or not. This file can eventually grow to a significant size. Even if you are not using the accounting features, this file should be zeroed out periodically using `cron`.

# Process Accounting

- ❑ Accounting startup in /etc/rc
- ❑ Accounting reports started in /usr/spool/cron/crontabs/adm
- ❑ Log files can grow to unreasonable size

7

## Notes

# The acctcom Command: Print System Accounting Information

The acctcom command is the main accounting command, often used interactively. By default, acctcom reads information from the file /usr/adm/pacct.

**Syntax**

```
acctcom [-abiv] [-l chars] [-g group] [-n user]
```

The syntax shown is truncated to file on the page. For the complete syntax for acctcom, read the on-line manual page.

Options are as follows:

| | |
|---|---|
| `-a` | Shows average statistics at end of report |
| `-b` | Output backwards, most recent events first |
| `-g` *group* | Report for the specified group |
| `-i` | Include I/O counts in report |
| `-l` *chars* | Print processes belonging to line *chars* |
| `-u` *user* | Report for the specified user |
| `-v` | Suppress the column headings |

**Note**

acctcom only reports on processes that have terminated.

**Example 1**

To print the processes run by the root:

```
# acctcom -u root
```

**Example 2**

To print the processes run over the modem port:

```
# acctcom -l modem
```

Processes run over a modem port at unusual hours may indicate an attempt to subvert your system.

**References**

*A/UX Local System Administration*
*acctcom(1M)*

# The `acctcom` Command

## Syntax

```
acctcom [-abh] [-1 chars] [-g group] [-n user]
```

## Example

```
# acctcom
COMMAND                               START    END                REAL     CPU      MEAN
NAME         USER      TTYNAME        TIME     TIME              (SECS)   (SECS)   SIZE(K)
#accton      root        .            04:00:24 04:00:24           0.05     0.05      7.17
sh           adm         .            04:00:19 04:00:24           5.50     0.30      8.64
sh           adm         .            04:00:24 04:00:24           0.03     0.03     13.25
mv           adm         .            04:00:24 04:00:24           0.72     0.08      8.10
cp           adm         .            04:00:25 04:00:25           0.55     0.08      8.30
acctwtmp     adm         .            04:00:26 04:00:26           0.38     0.05     11.00
cp           adm         .            04:00:26 04:00:26           0.18     0.08      6.40
chmod        adm         .            04:00:26 04:00:26           0.37     0.03     14.25
chgrp        adm         .            04:00:27 04:00:27           0.87     0.17      6.60
```

8

## Notes

# System Activity Report

sar is the A/UX system activity reporter command. sar will work only if the system accounting functions are turned on (cron must be running.) The sar command has two syntaxes:

**Syntax**

```
sar [options] t [n]
sar [options] [-A]
```

The syntaxes are as follows:

- With t[n], it samples system activity *n* times every t seconds (this syntax is used for getting "live" reports during benchmarking.)

- Without t sar reads /usr/adm/sa/sa*dd* and outputs a report based on other options (see *sar(1)*.)

In addition, the option:

-A Gives the same output as using the options -ubqwcayvm, a detailed multiple subject report.

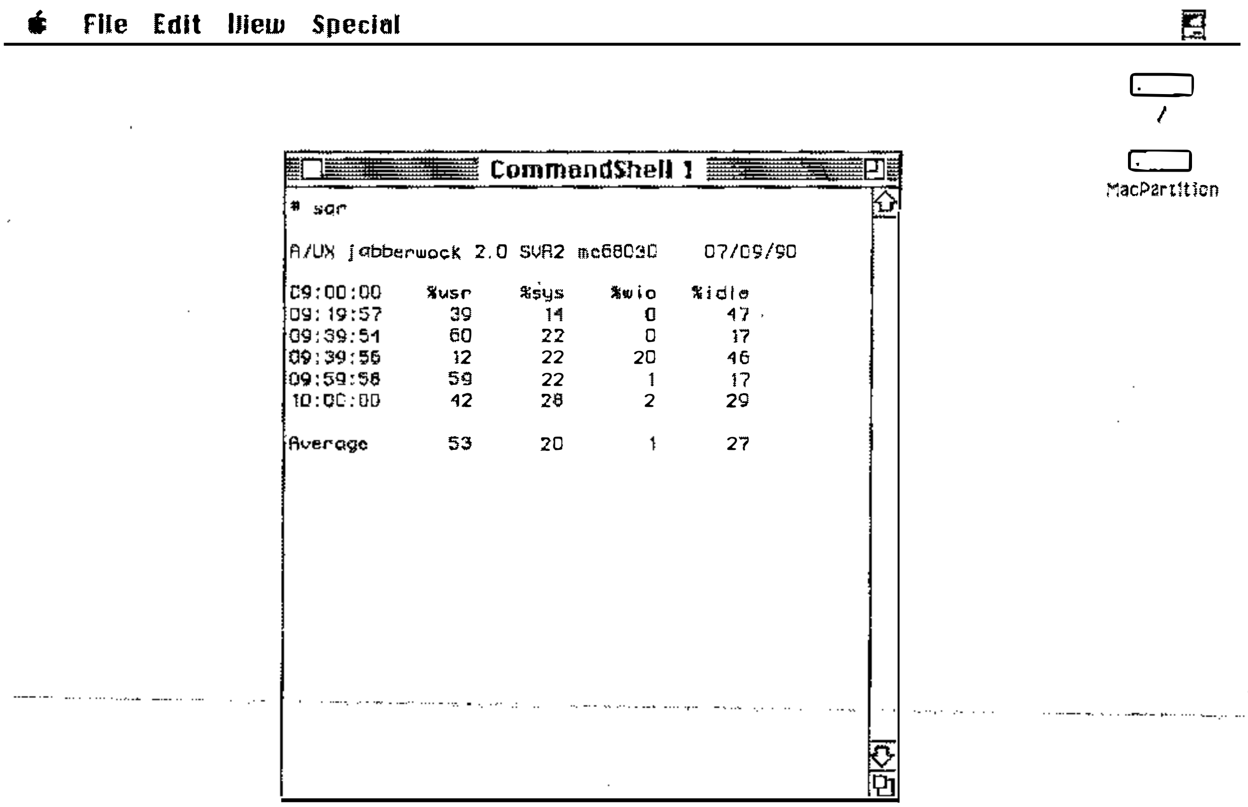
In addition, sar has options to give very detailed reports, for example the transfers per second of data between system buffers and disk (or other block) devices.

With no options, sar reports CPU activity recorded thus far today. Output columns have the following meanings (see example on the facing page):

| | |
|---|---|
| %usr | Percentage of time running in user mode |
| %sys | Percentage of time running in system mode |
| %wio | Idle time with some process waiting for block I/O |
| %idle | Idle time |

**References**

*A/UX Local System Administration*
*sar(1)*



```
# sar

A/UX jabberwock 2.0 SVR2 mc68030     07/09/90

09:00:00     %usr     %sys     %wio    %idle
09:19:57       39       14        0       47
09:39:51       60       22        0       17
09:39:56       12       22       20       46
09:59:56       59       22        1       17
10:00:00       42       28        2       29

Average        53       20        1       27
```

# System Activity Report

## Notes

# The pstat Command: Print System Statistics

The pstat command reads kernel memory in /dev/kmem to obtain detailed information about certain system tables. As with other commands, the syntax below is abbreviated.

**Syntax**

```
pstat [-f] [-i] [-p]
```

The listed options are as follows:

| | |
|---|---|
| -f | print the open file table info |
| -i | print the inode table info |
| -p | print process table for active processes |

Entering pstat with no options summarizes the three listed options.

**Note**

Except in benchmarking and development, this report is of little use.

**References**

*A/UX System Administrator's Reference*
*pstat(1M)*

# The pstat Command

## Syntax

```
pstat [-f] [-i] [-p] ...
```

## Example

```
# pstat
334 buffers: 0 busy, 0 wanted, 223 done, 107 nodev
125 out of 200 UFS inodes active
28 out of 100 processes active
96 out of 200 files active
```

10

## Notes

# The `lav` Command: Print Load Average Statistics

The `lav` command prints the average number of jobs in the run queue over the last one, five, and 15 minutes.

**Syntax**

```
lav
```

From a benchmarking standpoint this command is useful, though it provides little information to assist us otherwise.

**Example**

```
# lav
load average: 0.62 0.65 0.42
```

**References**

*A/UX System Administrator's Reference*
*lav(1)*

# Load Average Statistics

## Syntax

```
lav
```

## Example

```
# lav
load average: 0.62 0.65 0.42
```

11

## Notes

# The w Command: Who's Doing What

The w command lists information summarizing who is on the system and what they are doing. The w command is an example of a BSD command.

**Syntax**

```
w [-options]
```

The fields output are the user name, terminal, login time, the number of minutes the terminal has been idle, the combined CPU time of all processes and children on that terminal (JCPU), the CPU time of currently active processes (PCPU), and the command(s) executing.

Options:

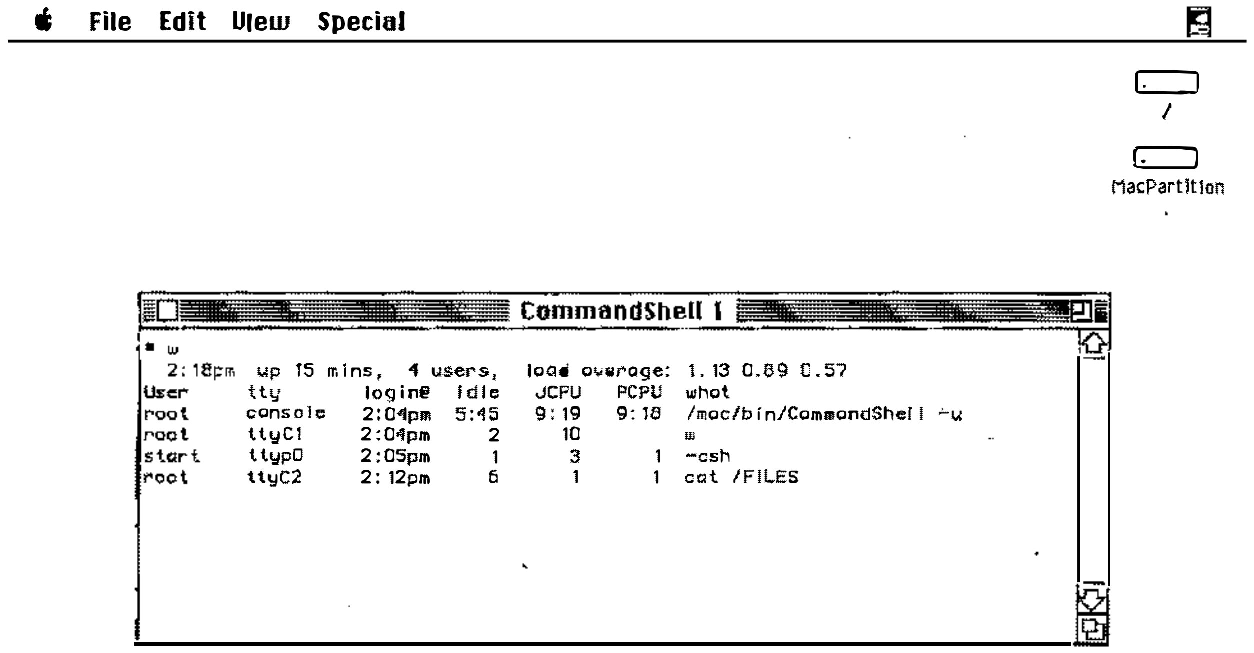
| | |
|---|---|
| `-h` | suppress the heading summary |
| `-u` | only list the heading summary |
| `-s` | short output |
| `-l` | long output (default) |
| *`user`* | list information for user only |

**Note**

w only gives a rough idea of what's going on, as it doesn't list background processes. The user count is always off, because it considers each open CommandShell window to be a separate user.

**References**

*A/UX System Administrator's Reference*
*w(1), uptime(1)*



```
# w
  2:18pm  up 15 mins,  4 users,  load average: 1.13 0.89 0.57
User     tty        login@  idle   JCPU   PCPU  what
root     console   2:04pm  5:45   9:19   9:18  /mac/bin/CommandShell -w
root     ttyC1     2:04pm     2     10         w
start    ttyp0     2:05pm     1      3      1  -csh
root     ttyC2     2:12pm     6      1      1  cat /FILES
```

# Who's Doing What?

Trash

## Notes

# The `timex` Command: Time a Command

The `timex` command times the supplied command, reporting user, system and real time used. `timex` may also summarize accounting data for the command and all of its children, or report total system activity during command execution.

**Syntax**

```
timex [-o] [-p[-options]] [-s] command
```

Some of the options are:

`-o` report on blocks and characters transferred by command

`-p [options]` list process accounting records for the command. The options for `-p` are those supplied to the `acctcom` command

`-s` report on total system activity during the execution of command

The `timex` command can be useful for establishing "benchmarks" for performance of certain tasks. You can time a command when system activity is high and again when it is low, comparing the two marks.

**Example**

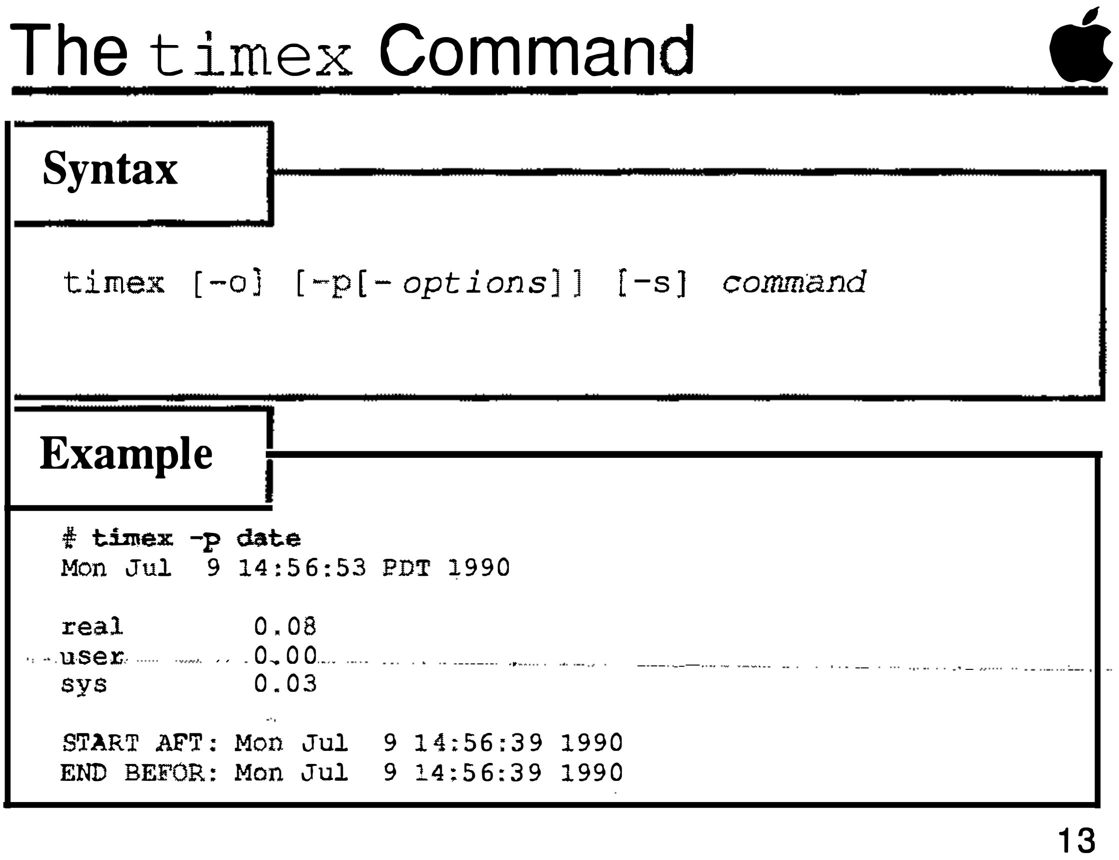
```
# timex -p date
Mon Jul  9 14:56:53 PDT 1990

real        0.08
user        0.00
sys         0.03

START AFT: Mon Jul  9 14:56:39 1990
END BEFOR: Mon Jul  9 14:56:39 1990
```

**References**

*time(1,) timex(1,) acctcom(1M)*



# The `timex` Command

## Syntax

```
timex [-o] [-p[-options]] [-s] command
```

## Example

```
# timex -p date
Mon Jul  9 14:56:53 PDT 1990

real        0.08
user        0.00
sys         0.03

START AFT: Mon Jul  9 14:56:39 1990
END BEFOR: Mon Jul  9 14:56:39 1990
```

13

## Notes

# Checkpoints

1 What is a process?

2 How can you list the active processes?

3 How do you stop a process?

4 What files must be changed to turn on process accounting?

# Exercises

This exercise simply uses some of the commands discussed to create accounting reports, which are printed to your console screen. For each command, you will read the on-line manual page, then execute various forms of the command.

**Exercise 1** In this exercise, you will tum on system accounting.

**Step 1** Power on your Macintosh computer and allow A/UX to boot normally. Log on as root.

**Step 2** Make backup copies the files /etc/rc and /usr/spool/cron/crontabs/adm:

**Step 3** Edit the file /etc/rc. Remove the preceding # character from the lines below:

```
/bin/su adm -c /usr/lib/acct/startup
echo process accounting started
```

**Step 4** Edit the file /usr/spool/cron/crontabs/adm. Remove the first # from each line in the file. When done, it should look like:

```
0 4 * * 1-6 /usr/lib/acct/runacct 2> /usr/adm/acct/nite/fd2log
0 2 * * 4 /usr/lib/acct/dodisk
5 * * * * /usr/lib/acct/ckpacct
15 5 1 * * /usr/lib/acct/monacct
0 * * * 0,6 /usr/lib/sa/sa1
0 18-7 * * 1-5 /usr/lib/sa/sa1
0 8-17 * * 1-5 /usr/lib/sa/sa1 1200 3
# The following is an option to appear in the above directory
0 20 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:00 -i 3600 -uybd
```

**Step 5** Cat the file /usr/adm/.profile and confirm that the following line is present; if it is not, use an editor to add it.

```
PATH=/usr/lib/acct:/bin:/usr/bin
```

**Step 6** Reboot A/UX and log on as root. Take a five minute break while the system collects some accounting data.

**Exercise 2**

In this exercise, you will look at the output of various reports, both process accounting and system activity. To save time for this exercise, you will force report generation by entering the `sal` command a number of times manually, and then reading the reports with the `sar` command. Since you won't be using it for the rest of the course, and to conserve disk space, you will turn the system accounting off at the end the exercise.

**Step 1**

Enter the commands:

```
/usr/lib/sa/sal
sar
```

Repeat this step three or four times.

**Step 2**

Execute the command to run `sar` three times, one second apart.

**Note**

If accounting was recently turned on, `sar` and `acctcom` output may be quite small. If accounting has been on while your system was rebooted, `sar` output may end with the error message "`sar: time change not positive`". In this case, simply remove the file /usr/adm/sa/sa*dd*, where *dd* is the current days date.

**Step 3**

Execute the command to run `sar` with the `-A` option.

```
sar -A
```

**Step 4**

Run the command `acctcom`.

```
acctcom
```

**Step 5**

Run the command `pstat`.

```
pstat
```

**Step 6**

Run the command `lav`.

```
lav
```

**Step 7** Run the command `timex` on the command `ps -el`.

```
timex ps -el
```

**Step 8** Now use the `mv` command to copy the files you saved in exercise 1 back.

This will result in system accounting not being turned on when the system is next rebooted.

**Step 9** Shutdown your system.

# Module 12: Files (Optional)

# Table of Contents

# Notes

# Files

**Goal**

The goal of this module is to introduce you to the concepts and details of files on A/UX systems.

**Objectives**

By the end of this module, you will be able to:

- List the seven types of files in an A/UX system
- List four items stored in an inode
- Describe the difference between a block and a character device files
- Name the information stored in a directory file

**Activities**

Lecture/Discussion/Checkpoints/Exercises

# Objectives

□ List seven types of files in A/UX

□ List four items stored in an inode

□ Describe the difference between block and character devices

□ Name information stored in a directory file

2

## Notes

# A/UX File Facts

As an operating system, one of the main functions of A/UX is the manipulation of files. While the "real work" of a computer is extracting, modifying, and replacing the contents of files, the operating system must keep track of all the files so that information is retrievable. In addition, the operating system maintains several kinds of unique files that allow it to be more efficient in handling resources.

There are seven kinds of files in A/UX:

- Regular
- Directory
- Block Device (Block Special)
- Character Device (Character Special)
- Symbolic Links
- Named Pipes
- Sockets

Each of these is identifiable by a unique designation in the first position of the file permissions (as shown by the `ls -l` command.)

| | |
|---|---|
| Regular | `-rw-r--r--` |
| Directory | `drwxr-xr-x` |
| Block Device | `brw-rw-rw-` |
| Character Device | `crw-rw-rw-` |
| Symbolic Links | `lrwxrwxrwx` |
| Named Pipes | `prw-r--r--` |
| Sockets | `srwxrwxrwx` |

Before we look at each of these kinds of files, let's examine how A/UX locates the files.

**References**

*A/UX Local System Administration*
*Introducing the UNIX System*

# A/UX File Facts

❑ Seven types of files

- Regular
- Directory
- Block Device
- Character Device
- Symbolic Links
- Named Pipes
- Sockets

3

## Notes

# What is an Inode?

An inode (index node) is simply a descriptor of a file that contains information about, and pointers to the location of, a file.

The information stored in an inode for its file is:

- permissions
- owner
- group
- size in bytes
- number of physical blocks on disk
- number of links (number of file names)
- time last accessed (read)
- time last modified (written)
- time inode last modified
- 15 disk block pointers

**Note** The filename is not stored with the information about the file, although the number of names this file has is stored.

UNIX incorporates the space-saving convenience of letting you have different names for the same file. Such names are created by making links to the desired file. The `ln` command is used to create links.

**Note** Hard links cannot be made across file system boundaries.

The inodes are kept in a special location on disk, called the superblock. For open files, A/UX also maintains a copy of this information in memory, and periodically writes it back to disk. From the list of information above, it is easy to guess that special care must be taken to make sure the data in the superblock is safe from harm.

**References** *A/UX Local System Administration*
*A/UX Network System Administration*
*ln (1)*

# What is an inode?

□ permissions

□ owner

□ group

□ size in bytes

□ number of physical blocks

□ number of links

□ time last accessed

□ time last modified

□ time inode last modified

□ 15 pointers to the file's contents

4

## Notes

# Disk Block Address

Stored in the inode are 15 disk block addresses that specify the location of the file on disk.

The first 12 addresses are the locations of the first 12 blocks of the file (each block is 4 KB long.) This is great for files less than about 49,000 bytes in size, since we can find all the data locations without another disk access. However, it is insufficient for all our needs.

The next address specifies the location of a disk block whose contents are a list of the next 256 disk block locations of the file. This block is called an indirect block.

If a file is larger than 268 blocks, the twelfth address is used. This points to a double-indirect block, whose contents are a list of 256 indirect blocks.

For those rare files larger than 65,802 blocks, the thirteenth address could be used. This points to a triple-indirect block, whose contents are a list of 256 double-indirect blocks. In practice, this is never needed.

Using this scheme, we can address extremely large files quite easily.

**References** *Local System Administration*

# 15 Disk Addresses

1-12 point to first 12 blocks

13 points to list of next 256 blocks

14 points to list of next 256 indirect blocks.

15 points to list of next 256 double-indirect blocks

5

## Notes

# Regular Files

Regular files are no more or less than simply a collection of bytes. UNIX does not impose any structure on what a file contains. Executable files (binaries) are treated no differently than text files.

**References** *Local System Administration*

# Regular Files

❑ Just collections of bytes

## Notes

# Directory Files

Directory files in A/UX are where file names are matched up with inodes. The directory files themselves are treated just as files, they have inodes and their disk blocks are scattered around on the disk.

The contents of directory files, however, have special significance. Directories contain lists of inode numbers with corresponding names of files contained in that directory.

When you want to access a file, the operating system first looks up the filename in the appropriate directory. This gives it the inode, and the inode tells the system the disk locations of the data in the file.

Things aren't *quite* this simple, of course. Directory and file permissions are checked against the requesting user's effective id. In order to find the "appropriate directory" the system may have to search through several directory files before locating the directory the file is in.

# Directory Files

□ Just another file, but with specific content

□ Provide the mapping between inodes and names

A/UX

7

## Notes

# Special Files

In the original UNIX, there were only three kinds of files: regular, directory, and special. Special files were to interface to the device drivers, and in that context were special. Today, there are additional file types. So while the files are no longer "special," the term is still used.

Special files reside in the /dev directory, and are the files one selects for doing input and output tasks. The files are actually just names and inodes, thus they do not take up disk space as regular files would. Instead of pointing to disk locations, the inode points to a device driver embedded in the kernel.

When we look at the long listing for files in /dev, the output we get is something like:

```
brw-rw-rw-  1 root sys   5,  0 Apr 10 01:00 floppy0
crw-rw-rw-  1 root sys   5,  0 Apr 10 01:00 rfloppy0
```

A "b" or "c" as the first character identifies these as special files. The "b" stands for block (meaning buffered I/O) and the "c" stands for character (meaning unbuffered I/O.) A block device transfers data in multiple byte blocks (the block size depends on the device,) whereas a character device is read and written to one byte at a time.

Instead of a file size, the listing shows a pair or numbers separated by a comma. This is called a major/minor pair. The major number (in this case 5) is used by the operating system to access the correct device driver(type of device) in the kernel. The minor number (here 0) is usually used to select a physical device (drive 0.)

Notice in the example above that there is both a block and a character device for the floppy drive. This is because some operations, such as reading data, are more efficiently handled on blocks of data. Other operations, such as formatting a disk, are better handled one byte at a time.

# Special (Device) Files

- ❑ directory lists just names and inodes
- ❑ block/character files
  - - block uses i/o buffering
  - - character access can be faster
- ❑ major/minor numbers
  - - major: accesses device driver (built in to kernel)
  - - minor: selects physical device

8

## Notes

# Symbolic Links

A file's inode keeps track of how many links have been made to the file, that is, how many names are associated with that inode. In A/UX parlance these are referred to as "hard" links. As one might expect from this name, A/UX also includes the concept of "soft" links. A soft link is a file whose inode points to another file name instead of a disk location.

This extends the convenience of the whole link concept, because, unlike hard links, symbolic links can be made across file systems.

**Note** When you make a copy of a symbolic link, you don't copy the link, you copy the file to which it points.

When we look at the long listing for symbolic links, the output we get is something like:

```
lrwxrwxrwx 1 root sys  17 Apr 24 15:21 /etc/mkfs -> /etc/fs/svfs/mkfs
```

The "l" as the first character of the file permissions indicates the file is a symbolic link. As can be seen from this example, there is a simple way to tell the regular file to which the link is pointing.

# Symbolic Links

- ❑ implements soft links
- ❑ connection across file systems
- ❑ copying a symbolic link copies the original file, not the link

## Notes

# Named Pipes

In A/UX we sometimes have occasion to redirect the output of a command into another command. This is normally done by piping the two commands together. In some cases it is more convenient to have a standard reference point for referring to pipes; this can facilitate the synchronizing of execution between the two commands. A named pipe is such a reference point. Like a special file, it does not have disk locations associated with it; it exists as a directory entry and is accessed by a pathname. Unlike special files, there is no major/minor pair. Instead the pipe's size is listed as zero.

**Note** When you copy a named pipe with the Finder, it is converted into a regular file.

When we look at the long listing for symbolic links, the output we get is something like:

```
prw-r--r-- 1 lp  lp  0 Mar 22 1989 /usr/spool/lp/FIFO
```

The "p" as the first character of the file permissions indicates the file is a named pipe.

**Note** System administrators rarely have to explicitly use named pipes, usually programs invoked by the administrator use them internally.

# Named Pipes

❑ synchronizes execution

10

## Notes

# Sockets

A socket, according to our dictionary, is an "endpoint of communication." This means that it can be read from and written to much like a regular file. Sockets, however, have a more transient nature.

We can imagine a pipe mechanism where the command we are going to pipe from is ready to send output, and the command we are going to pipe to is ready to receive input. Suppose both these commands then create sockets. The commands search for each other's socket (by name.) Once each socket is found they are connected, and the piping takes place. When the piping is over, the sockets disappear.

Sockets are similar to the above scenario, except communication between sockets is two-way, not limited to the unidirectional flow of pipes. The tremendous advantage of sockets, using the example above, is that the two commands do not have to be on the same system.

Like named pipes, the socket's size is listed as zero.

When we look at the long listing for sockets, the output we get is something like:

```
srwxrwxrwx  1 root sys   0 May 16 13:23 /dev/printer.socket
```

The "s" as the first character of the file permissions indicates the file is a socket.

**Note** System administrators rarely have to explicitly use sockets, usually programs invoked by the administrator use them internally.

# Sockets

- □ endpoints for communication
- □ transient
- □ connection across networks

A/UX

11

## Notes

# Checkpoints

1 What are the seven file types found in A/UX?

2 What information is stored in an inode?

3 What is the difference between a block and character special file?

4 What information is stored in a directory?

# Exercises

**Exercise 1**

Use the `find` command to locate all the following file types in the A/UX file system:

Symbolic Links
Named Pipes
Sockets

**Exercise 2**

Use the `ls` command to identify the inode numbers for the following files:

| File | Inode |
|---|---|
| /unix | |
| /etc/inittab | |
| /etc/passwd | |

# Command Quick Reference Chart

| **Users:** | **Users:** |
|---|---|
| date [*MMDDhhmm*[*YY*]] | display [or set] the date and time |
| passwd [*user*] | change the password [for *user*] |
| who | display who is logged on |

| **Directories:** | **Directories:** |
|---|---|
| cd [directory] | change directory |
| chmod *mode filename* [*filename* ...] | change the directory permissions |
| ls [-alCF] [*directoryname* ...] | list the files [*in directory*] |
| mkdir *directory* [*directory1* ...] | make a directory [*or several*] |
| pwd | print the working directory |
| rmdir [-i] *directory* [*directory1* ...] | remove a directory [*or several*] |
| rm -[i]r *directory* | remove a directory and its contents |

| **Files:** | **Files:** |
|---|---|
| chmod *mode filename* [*filename* ...] | change the file permissions |
| cp *sourcefilename targetfilename* | copy a file |
| find *startdirectory* -name*filename* -print | locate a file |
| ls [-alCF] [*directoryname* ...] | list the files [*in directory*] |
| mv *source* [*source1* ...] *targetdir* | move one [or more] files to a directory |
| mv *sourcename* [*directory/*]*targetname* | rename a file |
| rm [-i] *filename* [*filename1* ...] | remove a file [or several] |

| **File Contents:** | **File Contents:** |
|---|---|
| cat [*filename* ...] | display all the contents |
| cut -*type* [-d*character*] [*filename* ...] | select columns |
| diff [*options*]*file1 file2* | compare two files for diffrences |
| grep [-i] [-v] "*pattern*" [*filename* ...] | search for patterns |
| head [-*number*] [*filename* ...] | display the first [number] lines |
| more [*filename* ...] | display the contents one screen at a time |
| tail [-*number*] [*filename* ...] | display the last [number] lines |
| sort [-rbun] [*filename*...] | sort |
| spell [-*v*][-*b*][+*localdict*] [*filename* ...] | spell check |
| TextEditor[*filename*...] | edit the text in a file |
| wc [-*clw*] [*filename*...] | count words, lines and characters |

# Command Quick Reference Chart

| Commands: | Commands: |
|---|---|
| `alias` [*newcmd*][="*oldcmd*"] | substitute command names |
| `apropos` *keyword [keyword...]* | find commands related to *keyword* |
| `cmdo` | invoke Commando dialog box |
| `find` *startdirectory* -name *filename* -print | locate a file |
| `man` *commandname* | display reference manual page for cmd |
| `whereis` *command* | show pathname *of command* |

| Redirection: | Redirection: |
|---|---|
| command > filename | redirect stdout, overwrite |
| command >> filename | redirect stdout, append |
| command < filename | redirect stdin |
| command1 \| command2 | pipe output from cmd1 to cmd2 |

| Printing: | Printing: |
|---|---|
| `lpr` [-P*printer*] [-#*num*] [-m] [*filename* ...] | print *filename* on *printer* |
| `cat` *filename* \| lpr | print without having permissions |
| `lpc` status | check printer status |
| `lpq` [-P*printer*] [*identifier*] | check spooler status |
| `lprm` [-P*printer*] [*jobnumber* ...] | remove queued print jobs |

| Mail: | Mail: |
|---|---|
| `mailx` *login_name [login_name1 ...]* | mail a message to login_name |
| `mailx` *login_name@host* | mail a message to login_name on internet |
| `mailx` *rhost*[!*rhost*...]!*login_name* | mail a message to login_name via UUCP |
| `mailx` | read mail |
| s[ave] [*messagenumber*] *targetfile* | save message(s) in *targetfile* |
| !*command* | execute *command* from within mailx |
| d[elete] [*messagenumber*] | delete message(s) |
| q[uit] | quit mail |

# ASCII Table Reference Chart

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | nul | 32 | sp | 64 | @ | 96 | ` |
| 1 | soh | 33 | ! | 65 | A | 97 | a |
| 2 | stx | 34 | " | 66 | B | 98 | b |
| 3 | etx | 35 | # | 67 | C | 99 | c |
| 4 | eot | 36 | $ | 68 | D | 100 | d |
| 5 | enq | 37 | % | 69 | E | 101 | e |
| 6 | ack | 38 | & | 70 | F | 102 | f |
| 7 | bel | 39 | ' | 71 | G | 103 | g |
| 8 | bs | 40 | ( | 72 | H | 104 | h |
| 9 | ht | 41 | ) | 73 | I | 105 | i |
| 10 | nl | 42 | * | 74 | J | 106 | j |
| 11 | vt | 43 | + | 75 | K | 107 | k |
| 12 | np | 44 | , | 76 | L | 108 | l |
| 13 | cr | 45 | - | 77 | M | 109 | m |
| 14 | so | 46 | . | 78 | N | 110 | n |
| 15 | si | 47 | / | 79 | O | 111 | o |
| 16 | dle | 48 | 0 | 80 | P | 112 | p |
| 17 | dc1 | 49 | 1 | 81 | Q | 113 | q |
| 18 | dc2 | 50 | 2 | 82 | R | 114 | r |
| 19 | dc3 | 51 | 3 | 83 | S | 115 | s |
| 20 | dc4 | 52 | 4 | 84 | T | 116 | t |
| 21 | nak | 53 | 5 | 85 | U | 117 | u |
| 22 | syn | 54 | 6 | 86 | V | 118 | v |
| 23 | etb | 55 | 7 | 87 | W | 119 | w |
| 24 | can | 56 | 8 | 88 | X | 120 | x |
| 25 | em | 57 | 9 | 89 | Y | 121 | y |
| 26 | sub | 58 | : | 90 | Z | 122 | z |
| 27 | esc | 59 | ; | 91 | [ | 123 | { |
| 28 | fs | 60 | < | 92 | \ | 124 | \| |
| 29 | gs | 61 | = | 93 | ] | 125 | } |
| 30 | rs | 62 | > | 94 | ^ | 126 | ~ |
| 31 | us | 63 | ? | 95 | _ | 127 | del |

# Glossary

**/**

1) main or highest directory; root; 2) separator between parts of pathname

**/bin**

Root subdirectory in which UNIX stores binary or machine executable files: storage container for most commands that users use daily

**/dev/null**

A file used to throw away data not needed; similar to the Macintosh trash can

**/etc**

Root subdirectory in which UNIX stores miscellaneous files important to system administrators; contents include files that control user accounts, such as the file of encrypted passwords

**/tmp**

Root subdirectory in which files are temporarily stored

**/unix**

Executable file that the system executes at boot; this file assigned system parameters; the kernel

**/users**

Root subdirectory in which A/UX stores user login directories

**/usr**

Root subdirectory meaning "user service requests"; stores commands, system spooling, on-line help, etc.

**absolute pathname**

The complete name of a file, given by listing all of the directories leading down to that file, starting from root (/) and concluding with the filename itself. The directories leading to the file are separated from each other and from the filename by slashes. For example, `/etc/passwd` is the absolute pathname of the system password file, `passwd`, located in the `etc` directory beneath the root (/) directory.

**access class**

A designation for access permissions to A/UX files and directories; the three access classes are user, group, and others. The user, or owner, is the person who created the file; the group consists of people, including the owner, who typically work together and need to share files easily; the others class consists of all people using a local system. A file or directory can be set to have different access permissions for each class.

**algorithm**

A step-by-step procedure for accomplishing a task or solving a problem.

**alias**

An alternate name used to invoke or identify a command, a network host, a list of users, or some other applicable entity.

**Apple Desktop Bus (ADB)**

A low-speed, input-only serial bus that connects the keyboard, mouse, and optional input devices to the system bus.

**application**

A program used to perform a particular task, such as computer-aided drawing, document preparation, accounting, or payroll management.

**archive**

(1) A collection of files, plus a table of contents. Archives are used mainly as libraries to be searched by the link editor `ld`. (2) Any collection of files saved simultaneously for backup purposes.

**argument**

A piece of information included on the command line in addition to the command; the shell passes this information to the command, which then modifies its execution in some particular way. Filenames, for example, are often supplied as arguments to commands, so that a command will operate on the named file.

**argument list**

All of the arguments passed to a program.

**ARPANET**

A wide area network that links government, academic, and industrial installations around the world. Primarily connecting research sites, the ARPANET was developed in the 1960s by the Advanced Research Projects Agency of the U.S. Department of Defense. See also **Defense Data Network.**

**assembly code**

A source file written in a low-level programming language that corresponds to a specific computer's binary machine language.

**asynchronous communication**

A method of data transmission in which the receiving and sending devices don't share a timer and no timing data is transmitted. See also **synchronous communication.**

**asynchronous I/O**

The capability to perform an I/O operation while its calling process continues to run. With synchronous I/O, the calling process sleeps until the I/O operation is finished.

**autoconfiguration**

An A/UX facility that automatically configures device drivers into the kernel upon system startup.

**autorecovery**

An A/UX facility that automatically repairs damaged file systems and rebuilds a good system if at all possible.

**A/UX command**

The name of an executable file distributed with the A/UX operating system. For example, `ls` is a binary executable distributed in the `/bin` directory that prints directory information to the terminal; typing `/bin/ls` as a command causes the file to execute. See also **shell program, built-in shell command.**

**A/UX Startup**

The Macintosh application that launches A/UX. If you cancel the automatic launch of A/UX, you are in the **StartupShell** from which you can run some system administration commands.

**A/UX Toolbox** Libraries, subroutines, and utilities that provide access from A/UX to the Macintosh Operating System and to the Macintosh User Interface Toolbox in the Macintosh II family of ROM.

**background job** A process executed by the shell such that the shell is not suspended while waiting for the process to finish. By default, a process starts in the foreground, and the shell waits until the process has finished executing before the shell returns its prompt. You run a process in the background by appending an ampersand character (&) to the end of a command line; the shell prompt reappears instantly, allowing you to run multiple processes simultaneously. See also **foreground job.**

**backslash** The character \, often used as an **escape character.**

**bang** The exclamation point (!), used as a syntactic element by the C shell, by `uucp`, and by other utilities.

**baud rate** The measure of the signal changes per second sent over a transmission line. This usually corresponds to the total number of bits per second sent over the line.

**Berkeley Software Distribution** (BSD) A version of the UNIX operating system developed at the University of California at Berkeley. The A/UX operating system incorporates many of the features of 4.2 BSD.

**binary file** A file, such as a machine language program, whose data is to be interpreted in binary form. See also **text file.**

**bit map** (1) A set of bits that represents the state of a corresponding set of items. (2) In QuickDraw, a pointer to a bit image, the row width of that image, and its boundary rectangle.

**block** A group regarded as a unit; usually refers to data, or to memory in which data is stored.

**block device**

A secondary storage device, such as a disk or tape drive, from which a file system can be mounted.

**block I/O**

The transfer of data that consists of chunks of contiguous information. By default, block I/O consists of 1024-byte chunks under A/UX. See also **character I/O.**

**B-NET**

The A/UX implementation of the **TCP/IP** protocols and utilities.

**boot**

To start an operating system by loading it into the computer's memory.

**boot block**

The first block of a file system. The boot block contains the system's startup instructions.

**boot device**

The peripheral device that reads an operating system's initial startup instructions.

**boot disk**

The disk that contains the initial startup instructions for an operating system.

**Bourne shell**

The standard UNIX System V command interpreter. See also **shell.**

**bridge**

An intelligent link between two or more networks, especially those employing the same media or protocols. See also **gateway.**

**BSD**

See **Berkeley Software Distribution.**

**buffer cache**

A holding area in main memory where read and write information for block I/O is temporarily stored.

**built-in shell command**

A command written into the shell itself rather than in a separate executable file. Built-in shell commands are generally used for writing shell scripts.

**button**
A pushbutton-like image in dialog boxes where you click to designate, confirm, or cancel an action.

**C**
A portable, high-level language that also offers very low-level operations, making it a flexible and efficient language for both application and system programming. A/UX itself is written almost entirely in C.

**canonical**
Adhering to standard, accepted, or authoritative procedures or principles.

**carriage return**
See **newline.**

**central processing unit (CPU)**
The "brain" of the computer; the microprocessor that performs the actual computations in machine language.

**character device**
Any device other than a **block device.** Character devices include terminals, modems, and network interfaces.

**character I/O**
The transfer of data one character at a time, rather than in blocks of characters. See also **block I/O.**

**check boxes**
A small box associated with an option in a dialog box. When you click the check box, you may change the option or affect related options.

**checksum**
The result of an arithmetic operation on a set of data; for example, the sum of bytes in a file. A checksum is used to help verify the integrity of data through stages of processing.

**child process**
A copy of a process, submitted for execution by another process. The original process is called the **parent process,** and the child is created by the system call `fork`.

**choose**
To pick a command by dragging through a menu. You often choose a command after you've selected something for the program to act on; for example,

selecting a disk and choosing the Open command from the file menu.

**click**

(v.) To position the pointer on something, and then press and quickly release the mouse button. (n.) The act of clicking.

**client**

A computer that has access to services on a network. The computers that provide services are called **servers.** A user at a client may request file access, remote login, file transfer, printing, or other available services from servers.

**close box**

The small white box on the left side of the title bar of an active window. Clicking it closes the window.

**coaxial cable**

A cable consisting of two concentric conductors an inner wire and an outer, braided sleeve.

**command interface**

The convention for interacting with A/UX by entering a command line.

**command line**

The entire input string that you enter in response to the shell prompt to issue a command or to start a program. The command line includes the command itself and any **arguments** and **flag options.**

**command mode**

The operating state in which a program (such as a text editor) interprets the characters you type as commands, rather than as data to be entered into a file.

**Commando**

A hybrid application that gives users access to UNIX commands through a dialog-box interface. This feature is unique to A/UX in the UNIX world.

**comment**

Information that is ignored by a program such as a compiler. A comment normally includes instructions, references, or notes for people inspecting a source file.

**Common Object File Format (COFF)**

The output file produced on A/UX systems by the assembler (`as`) and the link editor (`ld`). The term "common" refers to how this format is used on a number of processors and operating systems, including A/UX.

**concurrent processing**

The ability of an operating system to execute multiple programs simultaneously.

**console**

The main terminal (that is, keyboard and screen) of your system. The console must be connected to your system. The console receives log and error messages from the operating system that are not sent to any other terminal.

**control character**

A nonprinting character that orders an action to be performed. For example, the **interrupt character** (by default, entered by holding down CONTROL and pressing C) interrupts a program's execution and returns you to the shell prompt.

**cooked input**

Data that has been processed by a terminal driver's line discipline. Special characters such as the **erase** and **kill characters** cause the line discipline to convert the raw data accordingly.

**cooked mode**

A terminal driver's method of operation that converts data sent from the keyboard or to the terminal screen. This conversion is performed by a line discipline to accommodate interactive use of the system. For example, an *erase* character sent from the keyboard causes the line discipline to delete the preceding character; the terminal driver then sends the converted data to the reading process. See also **raw mode.**

**CPU**

See **central processing unit**

**C shell**

The standard BSD command interpreter. See also **shell.**

**cu**

Call UNIX. This utility allows a user connected to one UNIX system to login in to another

**current directory**

The last directory into which you moved with the `cd` command; this directory is the starting reference point for all **relative pathnames** you enter. Also called the *working directory.*

**cursor**

A symbol on the screen that indicates your position on the command line or inside a file. The cursor is usually a small box or an underscore, and it usually blinks.

**daemon**

A noninteractive process that manages services. Network daemons, for example, automatically handle incoming network connection requests.

**data bits**

Data communications bits that encode transmitted characters or numbers.

**DDN**

See **Defense Data Network.**

**decrement**

In programming, to decrease the value of a variable used as a counter. See also **increment.**

**Defense Data Network**

A single, wide area, packet-switching network that integrates the ARPANET research network and the MILNET defense network.

**demand memory paging**

An A/UX facility that allows the kernel to use secondary storage (usually a hard disk) to store inactive portions of processes while main memory holds active portions.

**demon**

See **daemon.**

**demount**

See **unmount.**

**dereference**

In programming, to obtain a value referenced by a pointer.

**device**

A part of the computer, or a piece of external equipment, that can transfer information.

**device driver**

Kernel-level software that controls the exchange of information between a process and a device.

**device file** A file that represents a device. For example, an A/UX process reading from or writing to the device file `/dev/rfloppy0` is actually reading from or writing to the first 3.5-inch disk drive on the Macintosh II. Also called *special file.*

**device number** A number that contains both the major number and the minor number of a device. See **major number** and **minor number.**

**device switch** A data structure composed of the addresses of routines that manage I/O for a device.

**dialog** In reference to the Macintosh user interface, this is the same as **dialog box.**

**dialog box** (1) A box that contains a message requesting more information from you. Sometimes the message warns you that you're asking your computer to do something it can't do or that you're about to destroy some of your information. In these cases, the message is often accompanied by a beep. (2) A box that an application displays to request information or to report that it is waiting for a process to complete.

**directory** A file that contains a list of other files; similar to a folder in a Macintosh hierarchical file system.

**directory hierarchy** The collection of all files on the currently mounted file systems. See also **hierarchy.**

**Documentor's Workbench (DWB)** A group of utilities used for formatting files. Files formatted by DWB utilities can be printed on a wide variety of output devices.

**double click** (n.) Two clicks in quick succession, interpreted as a single command. The action of a double click is different from that of a single click  for example, clicking an icon selects the icon; double-clicking an icon opens it.

**double-click**

(v.) To position the pointer where you want an action to take place, and then press and release the mouse button twice in quick succession without moving the mouse.

**drag**

To position the pointer on something, press and hold the mouse button, move the mouse, and release the mouse button. When you release the mouse button, you either confirm a selection or move an object to a new location.

**DWB**

See **Documentor's Workbench.**

**ed**

A UNIX line editor

**effective user ID**

One of two user IDs associated by the kernel with a process (see also **real user ID**). When necessary for execution, the effective user ID for a process can be changed by programs to temporarily allow different permissions. After completing the task that required the different permissions, the effective user ID is set back to its original permissions. See also **user ID.**

**end-of-file (EOF)**

Under A/UX, the position of one byte past the last byte in a file; this is equal to the actual number of bytes in the file and is also known as the *logical end-of-file.* If a program calls a routine that uses the physical end-of-file convention, the logical end-of-file is used instead.

**environment**

A list of characteristics that identifies you to the system and influences and constrains your access to it. You can modify many of these characteristics. See **environment variable.**

**environment variable**

A characteristic controlling your use and access of the system that is available to the current shell and all of the child processes invoked from that shell. See also **shell variable.**

**EOF**

See **end-of-file.**

**erase character**

The keyboard character that, when pressed, erases the last character you typed. By default, this character is entered by pressing the DELETE key.

**escape character**

A character that causes a program to interpret the following character or characters in a special way. For the `nroff` and `troff` utilities, for example, the escape character is a backslash (\), a nonprinting character that allows you to insert a command in a line of text.

**Ethernet**

A standard network communications specification generally using a type of coaxial cable to connect computers in a local area network. The Ethernet specification was developed by Digital Equipment Corporation, Intel Corporation, and Xerox Corporation.

**`exec`**

A system call and a built-in shell command that load a program file and execute it by overlaying the address space of the calling process.

**export**

(1) To pass the value of a **shell variable** to a **child process.** (2) To make local file systems available to remote users.

**`export`**

The built-in shell command for the Korn or Bourne shell that passes the values of shell variables to child processes.

**field**

A data item separated from other data by blanks, tabs, or other specific delimiters.

**file**

For UNIX operating systems, an array of bytes; no other structure is implied by UNIX systems, which even treat peripheral devices like files.

**file descriptor**

A number used to identify a file.

**file handling system**

The set of data structures, commands, and subroutines used to manipulate files and data stored on physical devices.

**file mode**

See **permissions.**

**filename**

On UNIX System V operating systems, the name of a file, consisting of up to 14 characters and specified without listing the directory under which the file is located. For example, `passwd` is the filename for the system password file. See also **pathname.**

**filename expansion**

A procedure performed by the shell that derives a list of files from a single, shorthand filename containing metacharacters. Also called *globbing.*

**file system**

A logical device (such as a disk partition) that contains the data structures that implement all or part of the **directory hierarchy.**

**filter**

A utility that transforms its input in some way and writes this transformed data to the standard output. Lines submitted as input to the `sort` command, for example, are reordered so that the lines in the output are arranged alphabetically or numerically.

**Finder**

The application that maintains the Macintosh desktop and starts up other programs at the request of the user. You use it to manage documents and applications, and to get information to and from disks. It's the desktop you see upon starting up your computer, unless you have specified a different startup application.

**FIPS**

Federal Information Processing Standard; federal government's standard for computer purchases; FIPS #151 refines IEEE's POSIX standard

**flag option**

An argument included on the command line that instructs a program to alter its output or to change its mode of execution. A flag option is usually a hyphen followed by one or more characters. For example, the `-l` flag option to the `ls` command makes this utility print extra information, such as the date a file was last saved. Flag options are sometimes referred to as *keyletters*.

**floating-point notation** A method of representing numbers inside the computer by which the decimal point (or more correctly, the binary point) is permitted to "float" to different positions within the number. Some of the bits within the number itself are used to keep track of the point's position. This method is useful for quickly calculating complex mathematical operations.

**folder** (1) For the BSD `mailx` program, a file that you create for saving similar mail messages. (2) A holder of documents and applications on the Macintosh desktop. Macintosh folders, like UNIX file system directories, allow you to organize information in a hierarchical fashion.

**font** A collection of print characters unified by a distinctive look. Times Roman, for example, is the default font for `troff`.

**foreground job** The process attached to the terminal. The shell waits until the foreground job has finished executing before the shell returns its prompt and gives you control again of the terminal. See also **background job.**

**fork** (n.) One of the two parts of a Macintosh file; the data fork contains data accessed via the Macintosh File Manager, and the resource fork contains data used by the application, such as menus, fonts, and icons. (v.) To create a new process with the `fork` system call.

**`fork`** A system call that creates a new process.

**format** (1) To divide a disk into tracks and sectors where information can be stored. Blank disks must be formatted before you can save information on them. (2) To process a text file for output with a utility such as `nroff` or `troff`. See also **formatter.**

**formatter**

A utility that processes text for output to a device. The `nroff` and `troff` utilities, for example, are formatters that justify the margins, center the titles, number the pages, and perform other enhancements that improve the printed appearance of text files.

**Fortran-77**

A high-level programming language especially useful for mathematical and scientific applications.

**frame**

The time elapsed from the start bit to the last stop bit during serial communication.

**full-duplex communication**

A method of data transmission where two devices transmit data simultaneously.

**full pathname**

See **absolute pathname.**

**function**

A subroutine—that is, a preprogrammed calculation—that can be carried out on request from any point in a program.

**gateway**

A computer that connects two or more networks, especially those using different media or protocols. See also **bridge.**

**GID**

See **group ID.**

**globbing**

See **filename expansion.**

**GOSIP**

The government version of OSI standard

**group ID (GID)**

A number that indicates a group to which you belong at login time. As a member of a group, you have access to certain files and directories shared by other members of your group. Each user login name has at least one group ID associated with it.

**header file**

A file whose contents will be included with the source file at compile time—it contains function declarations, macros, types, and defines used by the compiler. Also called *include file.*

**here document** Input to a shell script command that is embedded inside the script itself.

**Hierarchical File System (HFS)** A method of organization in which disk files are grouped together within directories and subdirectories (folders within folders). HFS is used on hard disks and on 800K disks. In a hierarchical file system, a file is specified by its pathname, rather than by a single filename.

**hierarchy** A directory and any files or subdirectories residing under it. See also **directory hierarchy.**

**highlight** To make something visually distinct. For example, when you select a block of text using MacWrite, the selected text is highlighted—it appears as light letters on a dark background, rather than dark-on-light. Highlighting is accomplished by inverting the display.

**home directory** The directory named by your environment variable `$HOME`. The home directory is usually the first directory you enter upon login, as designated in the file `/etc/passwd`. You can tailor your environment by modifying various files in your home directory.

**host** A computer connected to a network.

**icon** An image that graphically represents an object, a concept, or a message.

**include file** See **header file.**

**increment** In programming, to increase the value of a variable used as a counter. See also **decrement.**

**inode** A data structure that defines a file by describing the disk layout of the file data, its permissions, and its access times.

**input mode** See **insert mode.**

**insert mode**

The state whereby a program (such as a text editor) accepts the characters you type as data rather than as commands. Also called *input mode.*

**interactive editor**

A utility for entering and manipulating text while you view the text. The `vi` and `ed` editors are both interactive. See also **stream editor.**

**interactive program**

A program that allows you to enter additional commands and data during its execution instead of making you enter all of your commands and data as flag options and arguments on the command line. The `vi` and `mail` utilities are examples of interactive programs.

**International Standards Organization (ISO)**

A standards organization composed of representatives from the national standards bodies of 63 member countries. This organization has developed standards for pin assignments in data communication plugs, has promulgated the layered model of communications protocols, and has specified and approved protocols for many of the layers in the layered model.

**internet**

(1) A group of networks interconnected by bridges or gateways. (2) The Internet, used as a proper noun, usually refers to the Defense Data Network (DDN), descendent of the DARPA (Defense Advanced Research Projects Agency) Internet (also called the ARPANET). (3) When the proper noun is used as an adjective (for example, Internet domain) this refers to a networking standard used by the DDN.

**internet address**

An address for a computer on a network. The internet address consists of a network number and a host number that is unique for that network.

**interprocess communication**

A mechanism for transmitting information between processes. Interprocess communication mechanisms supported by A/UX include **semaphores, messages, shared memory, signals,** and **sockets.**

**interrupt**

An exception signaled to the processor by a device or by software to notify the processor of a change in condition; for example, an interrupt is signaled at the completion of an I/O request.

**interrupt character**

The keyboard character that, when pressed, interrupts execution of a program and returns you to the shell prompt. By default, CONTROL-C is the A/UX interrupt character (issued by holding down CONTROL while pressing C).

**interrupt handler**

A routine that services interrupts.

**interrupt priority level**

A number identifying the importance of the interrupt. It indicates which device is interrupting and which interrupt handler should be executed.

**interrupt vector**

A pointer to an interrupt handler.

**i-number**

The offset of a particular inode within the i-list.

**I/O redirection**

See **redirection.**

**I/O request**

A request for input from or output to a file or a device.

**IEEE**

Institute of Electrical and Electronic Engineers

**ISO**

See **International Standards Organization.**

**job**

A process that can be stopped, restarted, and moved between foreground and background processing from the C shell.

**job number**

The identification number of a process executed in the background under the C shell. The job number appears next to the command name when you execute the `jobs` command.

**Kermit**

A remote terminal and file-transfer software program used for connecting **microcomputers** and **mainframe computers** across modems and serial lines.

**kernel**

A UNIX program that manages the system hardware. For example, the kernel manages files, communicates with peripherals, and handles other low-level resource management tasks.

**keyletter**

See **flag option.**

**kill character**

The keyboard character that, when pressed, erases the current command line from the shell. After you press the line kill character, the cursor moves to a new line, and you can enter a new command. CONTROL-U is the A/UX default kill character (issued by holding CONTROL down while pressing U).

**Korn shell**

A command interpreter that combines many of the best features found in the standard System V shell (the Bourne shell) and the standard BSD shell (the C shell). See also **shell.**

**LAN**

See **Local Area Network.**

**library**

A collection of related functions or declarations available to a program for linking at compile time.

**line editor**

A utility for entering and manipulating text. The commands to add or change text are entered from a command prompt, they only operate on the lines you specify, and you cannot always see the results of your changes right away. The `ed` and `ex` utilities are line editors. See also **screen editor.**

**link**

(1) To give an alternative name to a file. See also **unlink.** (2) In programming, to collect one or more routines into an executable program.

**list**

To display on a monitor, or print on a printer, the contents of memory or of a file.

**local area network (LAN)**

A group of computers connected for the purpose of sharing resources. The computers on a LAN are typically joined by a single transmission cable, and are located within a small area such as a single building or section of a building.

**local printer** A printer connected directly to the computer that is issuing a print request. See also **remote printer.**

**local system** The computer from which a user originates a network command. See also **remote system.**

**local system administration** Management of a single computer. This includes such functions as starting up and shutting down the system, adding and removing user accounts, and backing up and restoring data. See also **network administration.**

**logical disk** A disk partition that is treated by the operating system as a separate disk. See also **partition.**

**log on** To identify yourself to the system by entering the login name of your account and your account password.

**login name** The name of a user's account. Used for identification purposes.

**login prompt** The prompt (usually `login` on UNIX systems) by which a system tells you that it is ready to accept your login name.

**login shell** The shell that automatically runs after you successfully log in. See also **shell.**

**`lpr` system** A collection of programs and files that are used to manage printer operations. These include the print spooler and a series of maintenance commands.

**Macintosh Operating System** The lowest-level software in the Macintosh. It does basic tasks such as I/O, memory management, and interrupt handling.

**Macintosh User Interface** The standard conventions for interacting with Macintosh computers. The interface ensures users a consistent means of interacting with all Macintosh computers and the applications designed to run on them.

**MacPartition**

The name of the hard disk that contains a Macintosh partition with the `A/UX Startup` application and other partitions with A/UX.

**macro**

A collection of instructions or requests invoked by a single name.

**mail**

Electronic mail

**mainframe computer**

A central processing unit or computer that is larger and more powerful than a minicomputer or a personal computer (microcomputer). Frequently called *mainframe* for short.

**main logic board**

A large circuit board that holds RAM, ROM, the microprocessor, custom-integrated circuits, and other components that make the computer a computer.

**main memory**

The part of a computer's memory whose contents are directly accessible to the microprocessor. Programs and data are usually loaded into main memory, where the computer keeps information while you're working. Secondary memory stores the information when you are not using it. See also **secondary memory.**

**major number**

One of two numbers contained in the inode for a device. The major number identifies a particular device driver (terminal, disk, and so on). See also **minor number.**

**makefile**

A file containing a collection of operations used by the `make` utility to construct related files.

**man page**

Reference manual page accessed on-line

**menu**

A list of choices presented by a program, from which you can select an action. With Macintosh-style programs, menus appear when you use the mouse to point to and press on titles in the menu bar at the top of the screen. Dragging through the menu and releasing the mouse button while a command is highlighted chooses that command.

**message list**

An argument that allows you to specify a group of mail messages by number or name to various `mail` commands.

**message name**

The login name of a user who sent you a message. The message name can be used as an argument to many `mailx` commands.

**messages**

A group of system calls that allow processes to communicate by sending formatted data streams to each other.

**metacharacter**

A character interpreted by a program as standing for other characters or as designating a special function. For example, the ampersand (&) metacharacter at the end of a command line causes the shell to run the command as a background job.

**microcomputer**

A computer, such as any of the Apple II or Macintosh computers, whose processor is a *microprocessor*.

**minor number**

One of two numbers contained in the inode for a device. The minor number provides a unit number for the device. See also **major number.**

**mode**

See **permissions.**

**MOTIF**

OSF's standard windowing system

**mount**

To install a file system onto the directory hierarchy. See also **unmount.**

**mouse**

A small device you move around on a flat surface next to your computer. The mouse controls a pointer on the screen whose movements correspond to those of the mouse. You use the pointer to select operations, to move data, and to draw with in graphics programs.

**mouse button**

The button on the top of the mouse. In general, pressing the mouse button initiates some action on whatever is under the pointer, and releasing the button confirms the action.

**multitasking**

The ability of an operating system like A/UX to execute multiple processes simultaneously by sharing its central processor and peripherals among processes.

**multi-user**

A mode or ability of an operating system to support several people using the same computer simultaneously.

**network**

A collection of interconnected, individually controlled computers, along with the hardware and software used to connect them. A network allows users to share data and peripheral devices and to exchange electronic mail.

**network administration**

Management of the software and hardware that connects computers in a network. This includes such functions as assigning addresses to hosts, maintaining network data files across the network, and setting up internetwork routing. See also **local system administration.**

**Network File System (NFS)**

A protocol suite developed and licensed by Sun Microsystems that allows different makes of computers running different operating systems to easily share files and disk storage.

**newline**

A character that indicates the end of a sequence of bytes. Conventionally, the UNIX operating system interprets the line feed character, the carriage return character, and both together, as a newline character. On the terminal, this character starts a new line by moving the cursor to the first position of the next line. From A/UX, the newline can be entered by pressing RETURN or by holding down CONTROL while pressing J.

**NFS**
See **Network File System.**

**object file**
The form of a routine produced by a language translator such as a compiler or assembler. An object file can be linked to other object files to build a program. See also **source file.**

**OSF**
Open Software Foundation; a standards development consortium of companies including Apollo, DEC, Hewlett Packard and IBM

**Open System Interconnection (OSI)**
A logical structure for network operations standardized within the ISO. OSI provides a network design framework to allow equipment from different vendors to be able to communicate.

**operating system**
Low-level software that controls a computer by performing such basic tasks as I/O, memory management, and interrupt handling.

**OSI**
See **Open System Interconnection.**

**packet**
A unit of data and its control information transmitted across a network. A single message may be carried by one or more packets.

**page**
In A/UX, a 4-kilobyte portion of a program that is defined by the kernel for transfer between main memory and disk storage. See **paging.**

**page fault**
An interrupt that causes the page of data or code needed by a program to be read from disk storage into main memory. See also **page** and **paging.**

**page offset**
The left margin of a printed page. The default page offset for both `nroff` and `troff` is 1 inch.

**page swapping**
See **paging.**

**paging**

A method by which some operating systems (including A/UX) use secondary memory to store inactive portions of processes while active portions are held in main memory. When a process is executing, a portion of its code and data resides in main memory. Other portions, divided into pages, are automatically read in from disk storage as needed. When the system runs low on free main memory, the kernel makes more available by writing unneeded pages back out to disk. The kernel shuffles pages in and out of main memory and disk storage until the process has executed. Also called *page swapping.*

**parent process**

A process that has forked a child process. See also **child process.**

**parity bit**

A data communications bit used to verify that data bits received by one device match the data bits transmitted from another device.

**parity error**

The condition resulting when the **parity bit** received by a device isn't what was expected.

**partition**

A set of contiguous **blocks** on a **physical disk.**

**password**

One of a pair of identifiers that the system verifies against a list maintained on-line; encrypted on the system; invisible on screen when you type it in

**pathname**

A filename prefixed by its directory location. A pathname may contain a list of directories, separated from the filename and from each other by slashes. Each item in a pathname is located in the directory named to its left. For example, /etc/passwd is a pathname for the system password file, passwd, located in the etc directory beneath root (/). See also **absolute pathname** and **relative pathname.**

**peripheral device**

A piece of hardware, such as a disk drive, modem, printer, or terminal, that is connected to a computer and used for reading or writing data.

**permissions**

Authorization to read, write, or execute a file or directory. Under UNIX operating systems, each capability is assigned on an individual, group, and system-wide basis. Also called the *file mode.*

**Permuted Index**

An index to *A/UX Command Reference, A/UX Programmer's Reference,* and *A/UX System Administrator's Reference.* The Permuted Index is found in *A/UX Reference Summary and Index.* The Permuted Index contains three columns. The center column is sorted alphabetically by keywords. When using the Permuted Index, scan the center column for the information you seek. When you find the topic, look to the third column, which shows the reference section that contains more information. If a number from 2 through 5 appears in parentheses after the reference, it is located in *A/UX Programmer's Reference.* If (1M), (7), or (8) follows the reference, then it is listed in *A/UX System Administrator's Reference.* All other citations are listed in *A/UX Command Reference.*

**physical disk**

The entire set of disk **blocks** that exist on the actual disk drive hardware.

**PID**

See **process ID.**

**pipe**

(n.) (1) A command line that connects two or more commands in a series so that the output of one command becomes the input to the next. (2) An intermediate file in which data is passed from one process to another. (v.) To connect two or more commands in a series so that the output of one command becomes the input to the next.

**pipeline**

See **pipe.**

**pointer**

(1) A small shape on the screen that follows the movement of the mouse or shows where your next action will take place. The pointer can be an arrow, an I-beam, a crossbar, or a wristwatch. (2) An item of information consisting of the memory address of

some other item. For example, Applesoft BASIC maintains internal pointers to the most recently stored variable, the most recently typed program line, and the most recently read data item, among other things. The 6502 uses one of its internal registers as a pointer to the top of the stack.

**port**

(n.) (1) A socket on the back panel of a computer where you plug in a cable for connection to a network or a peripheral device. (2) A connection between the central processor unit and main memory or a device (such as a terminal) for transferring data. (v.) To move software from one computer environment to another.

**portability**

The UNIX system and application software written for it can be easily adapted (ported) to run on a variety of machines

**positional parameter**

A variable set on the command line of a shell script and operating as an argument to the script. This variable is called by number, usually 0 through 9, within the shell script. The number refers to the position of the parameter on the command line.

**POSIX**

IEEE proposed standard for UNIX operating system

**postfix notation**

In programming languages, a type of notation in which the variable is followed by the operator (in C, for example, either ++ to increment the variable or – – to decrement the variable). Postfix notation changes the value of the variable after the value has been used. See also **prefix notation.**

**postfix operators**

See **postfix notation.**

**postprocessor**

A utility that accepts as its input the output from another utility. For example, `psdit` is a postprocessor that accepts `troff` output and transforms it into a form suitable for printing on Apple LaserWriters and other PostScript-supported printers.

**prefix notation**

In programming languages, a type of notation in which the variable is preceded by the operator (in C, for example, either ++ to increment the variable or -- to decrement the variable). Prefix notation changes the value of the variable before using its value. See also **postfix notation.**

**preprocessor**

(1) A utility used to transform data that is then written to another utility. For example, `tbl` is a preprocessor that formats tables from properly coded text files; the output of this processor is usually piped to a more general text formatter like `troff`. (2) A function of certain compilers that provides file inclusion, comment deletion, and macro substitution.

**print spooler**

A utility that writes a representation of a document's printed image to disk or to memory, schedules it to print in a queue of other jobs, and then prints it.

**process**

An instance of a program in execution. Usually one copy of a program is stored on a UNIX system like A/UX, but multiple instances of the program—each having its own address space—can be executed simultaneously as separate processes.

**process ID (PID)**

A unique number assigned to each process being executed on the system. The PID is listed with its associated command when you enter the `ps` command. The PID is sometimes called the *process number.*

**process scheduling**

Multitasking process management performed by the kernel. The central processor unit (CPU) can only execute processes one at a time. By equitably scheduling their execution, the kernel lets multiple processes share the CPU efficiently.

**process status**

The names, states, and process numbers of commands submitted for execution; the `ps` command displays this information.

**program**

A file containing coded instructions to the computer. A compiled program is a file created first in source code, then transformed by the compiler into object code. A **shell script** is a program that does not need to be compiled because it is interpreted by the shell.

**prompt**

A character or string of characters displayed on the terminal when a program is waiting for input from you. The Bourne and Kom shells, for example, are set by default to display the dollar sign ($) as their prompt; the C shell is set by default to display the percentage sign (%) as its prompt.

**protocol**

A set of defined communications rules.

**pull-down menu**

A menu that is hidden until you move the pointer to its title and press the mouse button.

**QuickDraw**

The part of the Macintosh User Interface Toolbox that performs all graphic operations on the Macintosh II screen.

**quoting mechanism**

Special syntax in the command line that tells the shell to interpret metacharacters literally, or to control the type of substitution allowed in the command.

**radio button**

A common type of control in dialog boxes. Radio buttons are small circles organized into families–clicking any button on turns off all of the others in the family, like the buttons on a car radio.

**RAM**

An acronym for *random access memory*; memory of the CPU

**raw I/O**

Data transferred directly between a device and user address space. Raw I/O bypasses kemel buffers, resulting in faster data transfer.

**raw mode**

A method of device driver operation that passes data between a terminal and a process without performing any conversions on the data. See also **cooked mode.**

**real user ID**

One of two user IDs associated by the kernel with a process (see also **effective user ID**). The real user ID identifies the user who is responsible for the process.

**record**

A string of data whose structure and interpretation is determined by an application program.

**redirection**

A feature of the shell that allows you to pass the output of a command to a file or device instead of to the terminal screen, and to supply a command with input from a file or device instead of from the keyboard.

**regular expression**

A notation that uses a special set of metacharacters for specifying a text pattern. For example, the `vi` and `ex` editors use the `^` metacharacter at the beginning of a regular expression to stand for the beginning of a line; therefore the regular expression `^A` stands for the set of all lines that begin with an uppercase A.

**relative pathname**

The name of a file, given by listing the directories leading to that file in relation to the current working directory. Directories common to both the working directory and the file are not included in the relative pathname. See also **absolute pathname.**

**remote system**

On a network, any printer other than the **local printer(s).**

**remote system**

On a network, any computer other than the **local system.**

**resource**

(1) Synonymous with **device driver.** A *printing resource* is a system file that lets you print on a corresponding printer attached to the computer. (2) Data or code stored in a resource file and managed by the Resource Manager.

**restricted shell (`rsh`)**
A program that confines a user to a subset of the A/UX system commands.

**ROM**
An acronym for *read-only memory,* which is memory whose contents can be read, but not changed, and is used for storing permanent information. For example, the ROM in the Macintosh II contains the routines for the Macintosh user interface.

**root**
(1) The top directory in a UNIX directory hierarchy. Written as a slash (/), it is the first element in every absolute pathname. (2) The user with unlimited system privileges. Also called the *superuser*.

**root directory**
See **root** (1).

**root file system**
The file system that is always present on a UNIX system; the root file system can never be unmounted.

**root user**
See **root** (2).

**routine**
A part of a program that accomplishes some task subordinate to the overall task of the program.

**SCC**
See **Serial Communications Controller.**

**SCCS**
See **Source Code Control System.**

**screen editor**
A utility for entering and manipulating text. A screen editor displays the contents of a file by a full screen at a time. The commands to add or change text are entered anywhere on the screen, and the screen changes immediately to reflect the changes. The `vi` utility, for example, is a screen editor. See also **line editor.**

**script**
A file containing commands. See also **shell script.**

**scroll bar**
A rectangular bar that may be along the right or bottom of a window. Clicking or dragging in the

scroll bar causes your view of the document to change.

**SCSI**

See **Small Computer System Interface.**

**secondary memory**

Data and program storage for a computer. Secondary memory stores its infonnation on physical media such as disk or tape. Secondary memory offers slower access time to data than main memory, but provides more capacity. See also **main memory.**

**secondary shell prompt**

A character displayed by the shell when the shell expects further input from you. For the Boume and Kom shells, this character is set by default to be the greater-than sign (>), while for the C shell it is set as the question mark (?). The prompt appears, for example, when you use a multiline construct at the initial shell prompt. The secondary shell prompt reappears on each line until the final delimiter or the *interrupt* character is entered.

**select**

(v.) To designate where the next action will take place. To select using a mouse, you click an icon or drag across infonnation. In some applications, you can select items in menus by typing a letter or number at a prompt, by using a combination keypress, or by using arrow keys. (n.) A command to a device such as a printer to place it into a condition to receive data.

**semaphores**

A group of system calls that allow processes to synchronize execution.

**serial communication**

Data communicated over a single-path communication line, one bit at a time.

**Serial Communications Controller** (SCC)

The chip on the Macintosh II main logic board that handles serial I/O through the modem and printer ports.

**serial lines**

Data transmission lines over which infonnation is transmitted sequentially, one bit at a time.

**server**

A computer that provides a particular service across a network. The service may be file access, login access, file transfer, printing, and so on. Computers from which users initiate the service are called **clients.**

**`setuid`**

A subroutine for setting the **real** and **effective nser ID.**

**shared memory**

A mechanism that allows processes to share parts of their virtual address space with each other.

**shell**

A utility that accepts your commands, interprets them, and passes them on to the appropriate programs for execution. A/UX provides three shells: Bourne, C, and Korn. Each can be used as an interpreted programming language. Through **shell variables** and **environment variables,** you can tailor the environment of your shell for your own needs.

**shell command**

See **bnilt-in shell command.**

**shell layer**

An instance of a shell, invoked by the `shl` program. Through this program, you can simultaneously run up to seven shell layers.

**shell program**

A series of commands to be executed by the shell. A shell program may be entered at the shell prompt or stored in a file. Shell programs that are stored in files are referred to as **shell scripts.** Shell programs are sometimes called *user-defined commands.*

**shell prompt**

A character or string of characters displayed on the terminal to show that the shell is waiting for input from the user. The Bourne and Korn shells, for example, are set by default to display the dollar sign ($) as their prompt; the C shell is set by default to display the percentage sign (%) as its prompt.

**shell script**

A shell program contained in a text file. Entering the name of the shell script from the command line executes the commands listed in the shell script.

**shell variable**

A variable local to the shell. A shell variable is available only to the current invocation of the shell, not to any of its subshells or spawned processes. See also **environment variable.**

**Shift-click**

A technique that allows you to extend or shorten a selection by positioning the pointer at the end of what you want to select and holding down the Shift key while clicking the mouse button.

**shntdown permission**

Access granted to the root user to run the shutdown program. The shutdown program brings the system to an almost inactive state before you turn off its power.

**signal**

A software interrupt that causes a program to be temporarily diverted from its normal execution sequence. A/UX uses both System V and BSD signals. Signals can be issued, handled, and otherwise manipulated via a set of **system calls.**

**signal catcher**

A function that detects the interrupt sent from a signal system call. See also **signal.**

**signal handler**

A function that performs the required processing upon receipt of a signal. See also **signal.**

**singletasking**

Allowing performance of only one operation at a time; DOS would be an example of an operating system that is single-tasking

**single user**

Allowing only one person at a time to use the computer; DOS would be an example of an operating system that is single user

**size box**

A box in the lower-right corner of some active windows. Dragging the size box resizes the window.

**SL/IP**

Serial Line/Intemet Protocol; communications protocol that runs on serial cable; similar to TCP/IP

**Small Computer System Interface (SCSI)** A specification of mechanical, electrical, and functional standards for connecting small computers with intelligent peripherals such as hard disks, printers, and optical disks.

**socket** On a network, a communication mechanism originally implemented on the BSD version of the UNIX operating system. Sockets are used as endpoints for sending and receiving data between computers.

**Source Code Control System (SCCS)** A collection of commands used to control changes to text files, such as source code and documentation. SCCS protects files by controlling access and update privileges, and by preventing more than one user at a time from updating a file. SCCS also maintains an audit trail of revisions.

**source file** A text file containing coded instructions to the computer. A source file generally cannot be executed by the computer; instead, the source file must be compiled and linked to produce an executable **program.**

**spawn** To create and execute a new process with the `fork` and `exec` system calls.

**special file** See **device file.**

**spooler** See **print spooler.**

**standard error output** The data stream used for error messages returned by a program. By default, the shell directs error output to your terminal screen.

**standard input** The data stream used for input to a command. By default, the shell accepts as input the characters you type from your keyboard. The less-than sign (<) directs the shell to accept input from a file or device.

**standard output**

The data stream used for output from a command. By default, the shell directs this to the terminal screen. The greater-than sign (>) directs the shell to write the output to a file or device.

**start bit**

A serial communications bit that signals that the next bits transmitted are data bits.

**StartupShell**

The `A/UX Startup` application that launches A/UX. You can run a limited number of system administration commands from the StartupShell.

**stop bit**

A serial communications bit that signals the end of data bits.

**stream editor**

A utility for manipulating text. Rather than allowing you to move back and forth within a file interactively, a stream editor processes the text in a single pass. The `sed` utility, for example, is a stream editor. See also **interactive editor.**

**Streams**

A collection of tools that assist programmers to modularize data transfer between device drivers and processes.

**string option**

A setting specified by a set of characters.

**subdirectory**

A directory that is subordinate to another directory.

**subshell**

A new shell that is created from an existing shell. The subshell, or "child" shell, inherits the environment of its parent.

**superblock**

The block following the boot block on every file system. The superblock contains the main information about the file system, such as its name, its size, and lists of the free blocks and free i-nodes.

**superuser**

The user with unlimited system privileges. Also called *root*.

**suspend character**

From the shell, a character (by default, CONTROL-Z under A/UX) that stops the foreground job and returns you to the shell. Entering the command `fg` brings the suspended job back to the foreground, where it resumes execution.

**SVID**
See **System V Interface Definition.**

**SVVS**
System V Verification Suite; test suite used to confirm conformance of SVID

**swap space**
A disk partition used for temporarily storing unneeded pages of code and data. See also **page** and **paging.**

**synchronous communication**
A method of data transmission that uses a clocking signal on both the sending and the receiving device to ensure an integral number of time intervals between characters. See also **asynchronous communication.**

**system call**
A kernel-level procedure that can be invoked by any application. System calls are documented in Section 2 of *A/UX Programmer's Reference.*

**System V**
The AT&T standard UNIX operating system. System V Release 2 forms the foundation of the A/UX system.

**System V Interface Definition (SVID)**
AT&T's formal specification for compatibility with the UNIX operating system. A/UX adheres fully to the SVID.

**System Folder**
The Macintosh folder containing the System file and related utilities.

**system mailbox**
A file in `/usr/mail` for holding your incoming electronic mail messages until you read them, at which time they are appended by default to the file `mbox` in your **home directory** and deleted from `/usr/mail`.

**TCP/IP**
See **Transmission Control Protocol/Internet Protocol.**

**terminal**
A device through which you interact with the computer; namely, the keyboard, mouse, or other input device and the monitor. See also **console.**

**terminal emulation**
An imitation of a terminal type.

**text box**
The place or places in any dialog box where you can type information.

**text file**
A file containing information expressed in text form and whose contents are interpreted as characters using the American Standard Code for Information Interchange (ASCII) format. See also **binary file.**

**tilde escape**
The tilde character (~), used as an escape character to signal that the next input string is a command.

**title bar**
The horizontal bar at the top of a window that shows the name of the window's contents. You can move the window by dragging the title bar.

**toggle option**
See **toggle variable.**

**toggle variable**
A setting for the shell environment that may be turned ON or OFF with the `set` or `unset` command. For example, the `set noclobber` command entered from the C shell turns on a toggle variable that helps ensure existing files are not accidentally overwritten.

**token**
A sequence of characters delimited so as to be identified by a compiler.

**Toolbox**
The Macintosh toolbox is a group of libraries which allow programmers to create windows, menus and dialog boxes for UNIX applications

**Transmission Control Protocol/Internet Protocol (TCP/IP)** A suite of networking protocols developed initially for the U.S. Department of Defense.

**tree structure**
The layout of a UNIX directory hierarchy. Organized like an inverted tree, the directory hierarchy begins with the root directory at the top. Branching downward from the root are the rest of the directories and files in the system.

**Trusted System**
One that meets federal government's graded approach to security features

**`tty`**
A terminal; `tty` is abbreviated from *teletypewriter,* which was the first terminal device used on the UNIX operating system.

**UI**
UNIX International; standards group formed by AT&T and Sun Microsystems

**`uid`**
See **user ID.**

**`umount`**
A system-administration command that removes a local file system.

**UNIX operating system**
A general-purpose time-sharing system and related set of utilities, originally developed at AT&T Bell Laboratories. A/UX is an enhanced version of the UNIX operating system for the Macintosh II family of computers.

**unlink**
To remove a directory entry for a file. See also **link.**

**unmount**
To remove a file system from the directory hierarchy. See also **mount.**

**user-defined command**
See **shell program.**

**user ID**
A number that identifies a user at the time of login. Often called `uid`.

**user interface**
The rules and conventions by which a computer system communicates with the person operating it.

**user name**
See **login name.**

**utility**
A software tool used for building or maintaining systems or applications. UNIX provides hundreds

of utilities, including compilers, editors, and text formatters.

**uucp**

UNIX to UNIX copy command; utility to enable file transfer between UNIX systems

**vi**

The standard UNIX full screen editor

**virus**

A computer program designed to replicate itself in, and propagate to other computer systems. They can be destructive in nature, erasing disk drives without warning.

**volume**

A piece of storage medium formatted to contain files. A volume can be an entire disk or only part of a disk.

**wide area network (WAN)**

A system of interconnected local area networks that spans a wide geographical area.

**window**

(1) The area that displays information on a desktop; you view a document through a window. You can open or close a window, move it around on the desktop, and sometimes change its size, scroll through it, and edit its contents. (2) The portion of a collection of information (such as a document, picture, or worksheet) that is visible in a viewport on the display screen. Each window is internally represented in a window record.

**word**

(1) The computer's native unit of data. The Macintosh II uses a 32-bit word. (2) For the shell and other programs, a string of nonblank characters bounded by the space character, the tab character, or the beginning or the end of the input line.

**worm**

A computer program designed to replicate itself in a computer system. They are generally benign, except for the amount of disk space they take over.

**working directory**

See **current directory.**

**X Window System**

A device-independent system for windowing and graphics; operates across heterogeneous networks; originally designed by MIT

**X/OPEN**

Standards committee formed in Europe that specifies the function of the UNIX operating system

**Yellow Pages**

Now called Network Information System (NIS), a Network File System facility for sharing a common database of user information across a local area network.

**zoom box**

A small box with a smaller box enclosed in it found on the right side of the **title bar** of some windows. Clicking the zoom box expands the window to its maximum size; clicking it again returns the window to its original size.

# Additional Resources

## Articles:

"The UNIX Time Sharing System", Ritchie, D.M. and Thompson, K., The Bell Systems Technical Journal, Vol.57, No.6, Part 2 (1978), 1903-1929

## Books:

The AWK Programming Language, Aho, A., Kernighan, B. and Weinberger, P.: Addison-Wesley

The Design and Implementaton of the 4.3BSD UNIX Operating System, Leffler, S., McKusick, M., Karels, M. and Quarterman, J.: Addison-Wesley

The Design of the UNIX Operating System, Bach, Maurice J.,: Prentice Hall

Exploring the UNIX System, Kochan, S., and Wood, P.: Hayden Book Company

Internetworking with TCP/IP, Comer, D.: Prentice Hall

Introducing the UNIX System, Morgan, R. and McGilton, H.: McGraw-Hill

Introducing UNIX System V, Morgan, R. and McGilton, H.: McGraw-Hill

The Korn Shell Command and Programming Language, Bolskt, M., and Korn, D.: Prentice Hall

Life with UNIX, A Guide for Everybody, Libes, D. and Ressler, S.: Prentice Hall

Operating Systems: Design and Implementation, Tannenbaum, A.,: Prentice Hall,

Shell Programming, Kochan, S., and Wood, P.: Hayden Book Company

The UNIX Operating System, Christian, K.: Wiley-Interscience

The UNIX Programming Environment, Kernighan, B. and Pike, R.: Prentice Hall

The UNIX System , Bourne, S.: Addison-Wesley

UNIX System Administration, Fielder, D. and Hunter, B.: Hayden Book Company

## Periodicals:

CommUNIXations: UniForum

Computing Systems: USENIX Association

;login;: USENIX Association

UniNews: UniForum

UNIX Review: Miller Freeman Publications Co.

UNIXuser: Marvin Rosenfeld

UNIXWORLD: Tech Valley Publishing

UNIX Bulletins: International Data Corporation

## User Groups:

Association Française des Utilisateurs d'UNIX: Gif-Sur-Yvette, France

Australian UNIX User Group (AUUG), Kensington, N.S.W., Australia

Canadian UNIX Network & International Xchange: St. Catharines, Ontario, Canada

Dansk UNIX-System Burger Gruppe (DKUUG): Copenhagen, Denmark

European UNIX User Group (EUUG), Buntingford, Herfordshire, England

Japan UNIX Society, Tokyo, Japan

National UNIX User Group/Netherlands, Amsterdam, The Netherlands

New Zealand UNIX Systems User Group (NZUSUGI), Hamilton, New Zealand

Svenska Unixanvandares forening, Taby, Sweeden

UNIX Interessengemeinschaft Schweiz, Zurich, Switzerland

USENIX Association: Berkeley, CA, USA

/usr/group: Santa Clara, CA, USA

# Answers to Exercises

## Module 2

**Exercise 1** In this exercise you will go through the verbose form of the initial A/UX startup procedure.

**Step 1** Power up your A/UX system. Double-click on the A/UX startup icon.

**Step 2** Set A/UX Startup to be the startup application.

**Step 3** Cancel the Startup utility by pressing ⌘. (COMMAND-PERIOD) while the •A/UX Release 2.0• copyright notification is showing.

**Step 4** Choose Booting from the preferences menu. Once the dialog box opens, change the AutoLaunch command to:

```
    launch -v
```

Close the dialog box.

**Step 5** Enter:

```
    boot
```

at the startup prompt.

**Step 6** When the Login dialog box appears, log on as root. The password is just a RETURN.

**Step 7** Set passwords for the following users:

```
passwd
passwd start
passwd guest
```

These are login names that have easily guessed passwords when A/UX is delivered, which could possibly cause security problems. Adding passwords is one of the first steps to take after installing A/UX

**Step 8** Shut the system down.

**Exercise 2** In this exercise you will you will delete the root password to gain access to the system. You would need to do this to rescue a user who has forgotten their root password. The subject of passwords is covered more in depth in the next module; for now we will emphasize the ability to make changes in the A/UX file system without running A/UX.

**Step 1** Power up the A/UX system. Double-click on the A/UX startup icon.

**Step 2** Cancel the Startup utility by pressing ⌘. (COMMAND-PERIOD) while the A/UX Release 2.0 copyright notification is showing.

**Step 3** Edit the password file. (In the following script, the computer's responses are in Courier font. Your commands are in **bold Courier**. Editorial comments (do not type these!) are still in Times font.)

**Note** The thirteen-character encrypted password found in your /etc/passwd file will be different than the one shown in the example below. Use the password found in the file on your system instead of the one shown.

```
startup# ed /etc/passwd              (start the ed editor)
530
1                                       (select line one)
root:jw9N7xoYnpWve:0:0::/:/bin/sh
s|:jw9N7xoYnpWve:|::|              (swap the old password)
p                                (print the line to check it)
root::0:0::/:/bin/sh
w                                  (write the file to disk)
517
q                                      (quit the program)
```

**Step 4** Enter:

```
boot
```

at the startup prompt.

**Step 5** Log on as root when the dialog box reappears. The password is just a RETURN.

**Step 6** Change the root password.

**Step 7** After you have successfully changed the password, shut down the system.

## Module 3

**Step 1** Log on to your A/UX system.

**Step 2** Add the user "john" to your system. Use the information below:

| | |
|---|---|
| Full name: | `John Awkhacker` |
| Login name: | `john` |
| Password: | `yoyodyne` |
| User's Identification number: | `200` |
| Login group: | `johns` |
| Full pathname of user's home directory: | `/users/john` |
| Full pathname of user's startup shell: | `/bin/ksh` |

```
adduser -r 'John Awkhacker' -U 200 -i -s /bin/ksh -g johns john
passwd john
```

**Step 3** Log off the system, then log back on as john.

**Step 4** Check who owns the files in your home directory.

Not surprisingly, john owns the files.

**Step 5** Log off the system, then log back on as root.

**Step 6** Remove the user john by deleting the line containing that login name from the file /etc/passwd.

**Step 7** Check who owns the files in john's home directory.

User 200 owns the files, the group is johns.

**Step 8** Remove the member john from the file /etc/group.

**Step 9** Check what happens to the files in john's home directory.

User 200 still owns the files, the group is still johns.

**Step 10** Shut down your system.

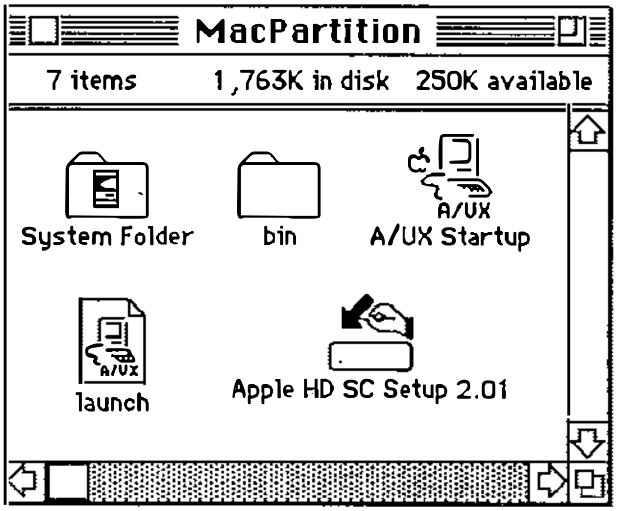
# Module 4

In these exercises, you will use HD SC Setup to initialize and partition a 20 MB HD SC disk drive. Then you will use A/UX commands to change the partition's file system type, to make and check a file system, then mount and use the auxiliary A/UX file space.

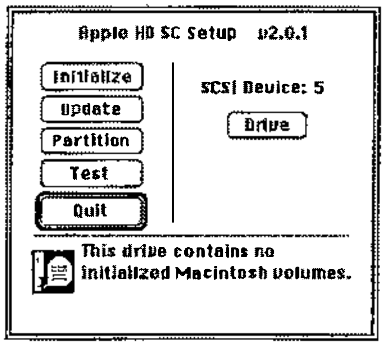
## Exercise 1

**Step 1** Power up the A/UX system.

**Step 2** Double-click on the HD SC Setup icon:



**Step 3** Click the Drive button to select your external 20 MB drive (SCSI Device 5 in this example). Then click the Partition button:

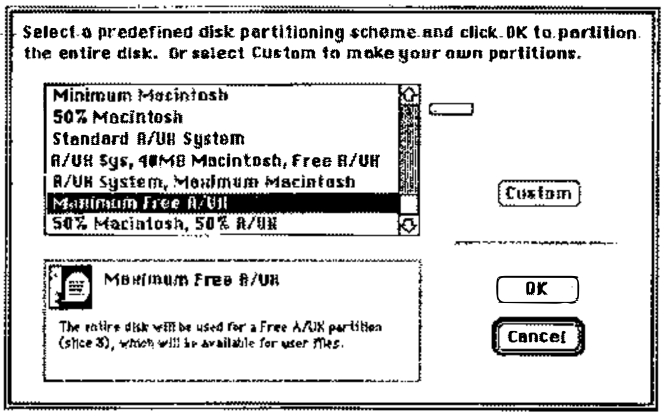


Apple HD SC Setup v2.0.1

SCSI Device: 5

This drive contains no initialized Macintosh volumes.

**Step 4** Select Maximum Free A/UX from the list of partitioning choices.



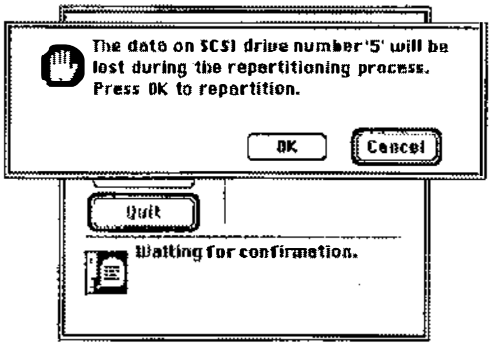
Select a predefined disk partitioning scheme and click OK to partition the entire disk. Or select Custom to make your own partitions.

- Minimum Macintosh
- 50% Macintosh
- Standard A/UX System
- A/UX Sys, 40MB Macintosh, Free A/UX
- A/UX System, Maximum Macintosh
- Maximum Free A/UX
- 50% Macintosh, 50% A/UX

Maximum Free A/UX

The entire disk will be used for a Free A/UX partition (slice 3), which will be available for user files.

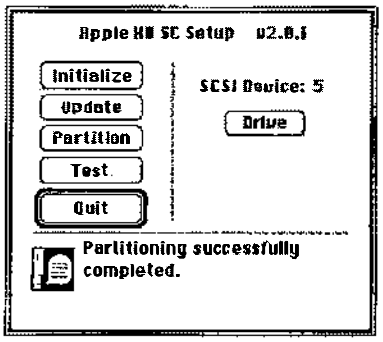
**Step 5** Click on the OK button when you see the warning dialog box.



The data on SCSI drive number '5' will be lost during the repartitioning process. Press OK to repartition.

Waiting for confirmation.

**Step 6** Click on the Partition button again after you see the message "Partitioning successfully completed".

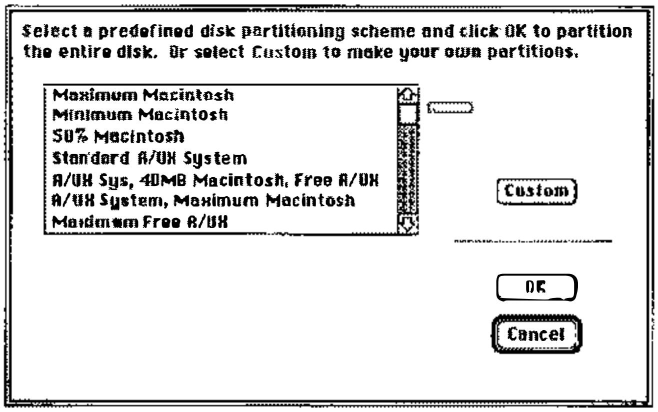


Apple HD SC Setup v2.0.1

SCSI Device: 5

Partitioning successfully completed.

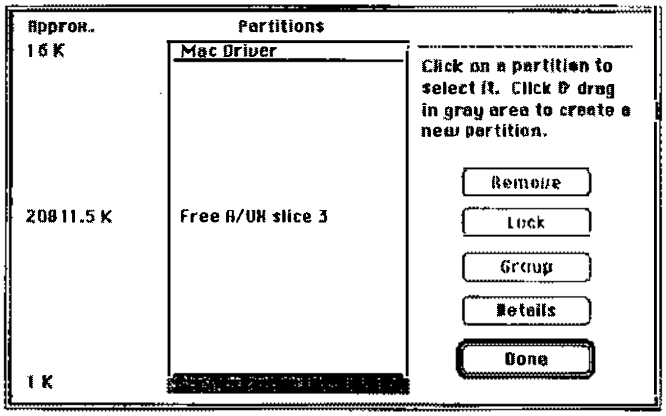
**Step 7** Click on the Custom button.



Select a predefined disk partitioning scheme and click OK to partition the entire disk. Or select Custom to make your own partitions.

Maximum Macintosh
Minimum Macintosh
50% Macintosh
Standard A/UX System
A/UX Sys, 40MB Macintosh, Free A/UX
A/UX System, Maximum Macintosh
Maximum Free A/UX

**Step 8** Examine the partition map:

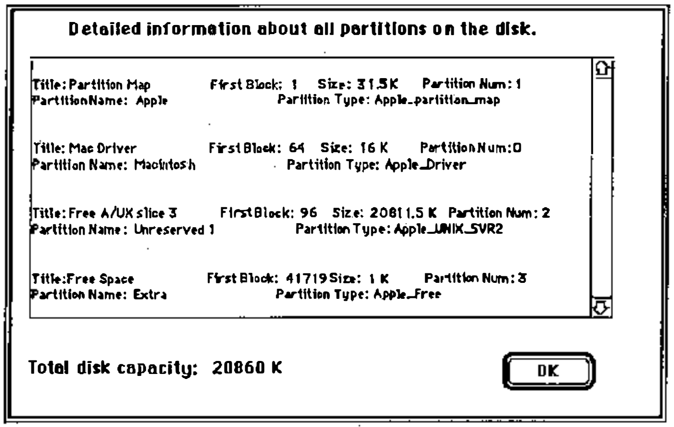


| Approx. | Partitions |
|---|---|
| 16 K | Mac Driver |
| 20811.5 K | Free A/UX slice 3 |
| 1 K | |

Click on a partition to select it. Click & drag in gray area to create a new partition.

Notice the disk has been allocated almost entirely to A/UX.

**Step 9** Click on the Details button. Notice that the A/UX partition is already allocated to slice 3.



Detailed information about all partitions on the disk.

```
Title: Partition Map        First Block: 1    Size: 31.5K      Partition Num: 1
PartitionName: Apple                  Partition Type: Apple_partition_map

Title: Mac Driver           First Block: 64   Size: 16 K       PartitionNum:0
Partition Name: Macintosh             Partition Type: Apple_Driver

Title: Free A/UX slice 3    FirstBlock: 96   Size: 20811.5 K  Partition Num: 2
Partition Name: Unreserved 1          Partition Type: Apple_UNIX_SVR2

Title:Free Space            First Block: 41719Size: 1 K       Partition Num: 3
Partition Name: Extra                 Partition Type: Apple_Free
```

Total disk capacity: 20860 K

**Step 10** Click on the OKbutton when you are done examining the details, then quit from the HD SC Setup application.

## Exercise 2

The main purpose of this exercise is to familiarize you to the dp command. dp offers a lot more power and flexibility than can be demonstrated in a single exercise; refer to the A/UX documentation for more information on dp.

**Step 1** Launch A/UX by double-clicking on the A/UX Startup icon.

**Step 2** Log on as root at the dialog box.

**Step 3** Enter the command to use the dp utility, specifying the disk you just formatted:

```
dp /dev/rdsk/c5d0s31
```

The system will respond:

```
"/dev/dsk/c5d0s31" 4 partitions, 4 allocated
Command?
```

dp responds with a prompt: `Command?`, telling you it is ready for your input.

**Step 4**

You will now enter the series of commands below to the "Unreserved 1" partition into a "usr" partition.

Enter the commands below exactly as shown in bold in the left column below. The Notes column will explain what's happening. If you make a mistake, you can quit from dp by entering CONTROL-C and begin again from Step 1.

### dp Command Prompts and Your Responses (in bold)      Notes

```
Command? p2                        Examine partition 2
DPM Index: 2
Name: "Unreserved 1", Type: "Apple_UNIX_SVR2"
Physical: 41623 @ 96, Logical: 41623 @ 0
Status:
        valid   alloc   in_use  not boot
        read    write
Slice 3
Regular UNIX File System (1)
Cluster:   0    Type: FS        Inode: 1
Made: [0] Wed Dec 31 16:00:00 1969
Mount: [0] Wed Dec 31 16:00:00 1969
Umount: [0] Wed Dec 31 16:00:00 1969
No AltBlk map
Command? c2                        Change partition 2.
DPME Field? n                      Change name field...
Name [Unreserved 1]: usr           ...to make it easily recognizable.
DPME Field? b                      Next, adjust the block zero block.
BZB Field? s                       Change the slice number.
Slice number + 1 [4]: 3            Sincewe want slice 2, we enter 3.
BZB Field? u                       Specify this is a usr file system...
Usr file system? y                 ...and confirm it..
BZB Field? q                       Quit from the block zero field sequence.
DPME Field? q                      Quit from the DPME field sequence.
Command? p2                        Print info about the new partition 0
DPM Index: 2
Name: "usr", Type: "Apple_UNIX_SVR2"
Physical: 41623 @ 96, Logical: 41623 @ 0
Status:
        valid   alloc   in_use  not boot
        read    write
Slice 2
Regular UNIX File System (1)
Cluster:   0    Type: UFS        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
Command? wq                        Write changes and quit dp.
```

**Exercise 3** This exercise takes you through the steps of making, checking, and mounting the file system.

**Step 1** Enter:the following command to make a new file system on the external disk:

```
cmdo newfs
```

Once the dialog box comes up, select the Disk Type matching the drive you are adding. Click on the Choose device file button. Select the file /dev/rdsk/c5d0s2. Click on the OK button. When the original dialog box comes back, click on the verbose mode check box, then click on the newfs button.

The system will respond with something similar to:

```
newfs -v /dev/rdsk/c5d0s2 HD20SC
/etc/fs/ufs/mkfs /dev/rdsk/c5d0s2 41623 17 4 4096 1024 16 10 60 2048 t
Warning: 62 sector(s) in last cylinder unallocated
/dev/rdsk/c5d0s3:     41622 sectors in 613 cylinders of 4 tracks, 17
sectors
        21.3Mb in 39 cyl groups (16 c/g, 0.56Mb/g, 160 i/g)
super-block backups (for fsck -b#) at:
 16, 1128, 2240, 3352, 4368, 5480, 6592, 7704, 8720, 9832,
 10944, 12056, 13072, 14184, 15296, 16408, 17424, 18536, 19648, 20760,
 21776, 22888, 24000, 25112, 26128, 27240, 28352, 29464, 30480, 31592,
 32704, 33816, 34832, 35944, 37056, 38168, 39184, 40296, 41408,
```

**Step 2** Check the file system by entering the command:

```
cmdo fsck
```

Once the dialog box comes up, click on the Choose file system(s) button. Select the file /dev/rdsk/c5d0s2. Click on the Add button, then click on the Done button. When the original dialog box comes back, click on the fsck button.

The system will respond:

```
fsck /dev/rdsk/c5d0s2
** /dev/rdsk/c5d0s2
** Last Mounted on
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
2 files, 5 used, 19705 free (9 frags, 4924
blocks, 0.0% fragmentation)
```

**Step 3** Make a directory to use as a mount point. Enter:

```
mkdir /newusr
```

**Step 4** Mount the external file system to your new directory:

```
mount -v /dev/dsk/c5d0s2 /newusr
```

The system will respond:

```
/dev/dsk/c5d0s2 mounted on /newusr
```

**Exercise 4** In this exercise you test that the file system is accesible.

**Step 1** Check that you have successfully mounted the new file system on the external drive:

```
mount
```

The system will respond with something like:

```
/dev/dsk/c4d0s0 on / type 4.2 (rw,noquota)
/dev/dsk/c5d0s2 on /newusr type 4.2 (rw,noquota)
```

**Step 2** Create a test file on your new file system:

```
cat /etc/group > /newusr/testfile
```

**Step 3** Verify you can read back the test file:

```
cat /newusr/testfile
```

The system will respond with something like:

```
root:*:0:
daemon:*:1:
bin:*:2:
sys:*:3:
adm:*:4:
uucp:*:5:
lp:*:7:
mail:*:8:
staff:*:50:
perm:*:60:
temp:*:70:
```

```
contract:*:80:
guest:*:90:
project:*:100:
class:*:1000:student
```

**Step 4** Unmount the new file system:

```
    umount /newusr
```

**Step 5** Verify you cannot access the test file:

```
    ls /newusr
```

**Exercise 5** In this exercise you set the system to automatically mount the new file system and again test its accessibility.

**Step 1** Enter the command:

```
    cmdo fsentry
```

to have A/UX automatically mount the file system. Click on the Choose device file button once the dialog box comes up. Select the file /dev/dsk/c5d0s3, then click on the OK button. When the original dialog box comes back, click on the Choose mount point button. Select the file /newusr, then click on the Directory button. When the original dialog box comes back, click on the fsentry button.

**Note** The `fsentry` program mounts the new file system.

**Step 2** Verify you can access the test file:

```
    ls /newusr
```

The system will respond with something like:

```
lost+found
testfile
```

indicating the file system is accessible

**Step 3** Remove the HD SC Setup program from your MacPartition.

**Step 4** Shut down your system.

## Module 5

Your Apple 40 SC Tape Drive should be connected and turned on, and a formatted tape should be inserted prior to starting the exercises. Refer to the manual *A/UX Local System Administration* and the on-line manual pages for assistance regarding commands with which you are not familiar.

**Exercise 1** In this exercise you will use `tar`, `cpio`, andpax to back up and restore files to and from a 3.5-inch floppy disk. In particular, you will back up a copy of /newunix, which you will need to restore later in the course.

**Step 1** Boot A/UX and log on as root. Open a CommandShell window. Insert a floppy disk into your floppy disk drive. The Finder may ask you how you want to use the disk; if so select the A/UX Data button. Format the disk from the command line:

```
diskformat /dev/rfloppy0
```

**Step 2** Use the Commando dialog for `tar` to back up the directory /users/start onto the floppy disk. Use the verbose mode.

```
tar cvlfbB /dev/rfloppy0 20 1600 /users/start
```

**Step 3** Open another CommandShell window and verify the backup by listing the contents of the archive.

```
tar tf /dev/rfloppy0
```

The file listings from the two `tar` commands should be the same, with the exception of the symbolic links. Recall that when a copy is made of a symbolic link the original file, not the link, is copied.

**Step 4** Use `pax` and Commando to back up the directory /users/start

**Step 5**

Recover the files from the cpio archive by changing to the new directory and using `cpio` and Commando.

```
pax -w -v -f /dev/rfloppy0 /users/start
```

or

```
pax -w -v -x cpio -f /dev/rfloppy0 /users/start
```

**Step 6**

Verify your backup by listing the contents of the `pax` archive.

```
pax -fv /dev/rfloppy0
```

Note that the copies of symbolic links are still symbolic links.

**Step 7**

Use `ls` and the Commando dialog for `cpio` in a pipe to back up the file /newunix. (You will need this backup in the module on Autorecovery.)

```
ls /newunix | cpio -ov > /dev/rfloppy0
```

**Step 8**

Verify your backup by listing the contents of the `cpio` archive.

```
cpio -it < /dev/rfloppy0
```

**Step 9**

Eject the disk from the command line.

```
eject 0
```

Label the disk "/newunix."

**Exercise 2**

In this exercise you will use the `tar` command to archive and restore files to and from the Apple 40 SC Tape Drive. You are already familiar with the basic use of these commands; the main difference when using the Apple tape drive is in specifying the blocking factor of 8 Kbytes.

Specifying the tape drive is a little easier if the system can find it through known names. Therefore, the first few steps below involve telling A/UX where to find the drive.

**Step 1**

Verify the SCSI address of your tape drive: SCSI Address______

**Step 2** Add the following to the file /etc/profile:

```
    TAPE=/dev/rmt/tcN
export TAPE
```

where *N* is the SCSI address.

**Step 3** Link the tape device to the name /dev/tape. Enter:

```
    ln /dev/rmt/tcN /dev/tape
```

where *N* is the SCSI address.

**Step 4** Log off, then log back on as root.

**Step 5** Insert a formatted tape and check its status with the `mt` command. This command will tell you how many 8KB blocks are available on the tape:

```
    mt status
```

**Step 6** Make a directory called /tmp/test and copy the /users/start directory into it. (Hint: use the -r option.)

```
    cp -r /users/start /tmp/test
```

**Step 7** Archive the /tmp/test/start directory onto tape using the `tar` Commando. You will need to specify the device to use.

```
    tar clbBf 16 4500 /dev/rmt/tc2
    /tmp/test/start
```

**Step 8** When the archive is done, verify the archive by listing its contents. Verify the contents match the directory listing.

```
    tar tbf 16 /dev/rmt/tc2
```

**Step 9** Remove the /tmp/test directory and its contents. Use `ls` to verify that the directory has been removed.

```
    rm -r /tmp/test
    ls /tmp/test
```

**Step 10** Restore the archive using the `tar` Commando.

```
tar xvbf 16 /dev/rmt/tc2
```

**Step 11** Verify the restoration by listing the files in the start directory.

```
ls /tmp/test
```

**Exercise 3** In this exercise you will use `dump.bsd` and `restore` to archive and retrieve a small file system that you will create on floppy disk. You will use this small file system in the interest of saving time.

**Step 1** Insert a blank floppy disk into your floppy drive. *Do not use the backed up copy of /newunix.* Format the disk:

```
diskformat /dev/rfloppy0
```

**Step 2** Make a SVFS file system on the formatted floppy disk. The `mkfs` command waits a few seconds before creating the file system—don't disturb it. (We use an SVFS file system because performance isn't important on floppy disks.) Use the following command:

```
mkfs /dev/rfloppy0 1600 1 1
```

**Note** If you want to make a UFS file system on the floppy, you will first need to link the file /dev/rfloppy0 to /dev/rdsk/rfloppy0

**Step 3** Mount the new file system to your `/mnt` directory. Note that since you can only mount a block device, you must specify the floppy disk drive by `/dev/floppy0`:

```
mount /dev/floppy0 /mnt
```

**Step 4** Copy the /users/start directory to the new file system. (Hint: use the -r option.)

```
cp -r /users/start /mnt
```

**Step 5** Check to make sure the copy worked properly by using the `ls` command:

```
ls /mnt/start
```

**Step 6** Unmount the /mnt directory:

```
umount /mnt
```

**Step 7** Touch the /etc/dumpdates file to make sure it exists:

```
touch /etc/dumpdates
```

**Step 8** Archive the new file system on tape. Make sure your tape unit is on and a formatted tape is inserted, then invoke the `dump.bsd` Commando dialog to build a command line.

```
dump.bsd 0ucbf 16 /dev/rmt/tc2
/dev/rfloppy0
```

**Step 9** Examine the /etc/dumpdates file when the archive is finished:

```
tail /etc/dumpdates
```

**Step 10** Mount the new file system to your /mnt directory and remove the start directory and its contents. Verify the directory has been removed.

```
mount /dev/floppy0 /mnt
rm -r /mnt/start
ls /mnt/start
```

**Step 11** Restore the archive by using the `restore` Commando dialog. Make sure you are in the mounted /mnt directory before entering the `restore` command. (During the restoration process, the program will ask you to specify the next volume; enter a 1 at this point.)

```
cd /mnt
restore xvbf 8k /dev/rmt/tc2
```

**Step 12** Verify the files were restored by using the `ls` command.

```
ls /mnt/start
```

**Exercise 4**

In this exercise you will use cpio to copy your /usr/adm directory over to the file system you have established on the external drive. This is something you would normally do with the entire /usr directory when setting up a network for NFS access. Having your files in a separate file system allows you to limit the availability of your files with NFS.

**Step 1** Verify the external drive is mounted.

```
     mount
```

**Step 2** Use cpio and the pass option to copy. the contents of the directory /usr/adm to the external drive. To do this, change directory to /usr/adm, and start the find portion of the command line from the current directory.

```
     cd /usr/adm
     find . -print | cpio -pdlmuv /newusr
```

**Step 3** Verify the files have been copied.

```
     ls /newusr
```

**Step 4** Shut down your system.

**Exercise 5**

In this exercise you will write a simple program to use the cron facility. This trivial example can be translated to have cron perform just about any task you can program the system for.

**Step 1** Write a crontab file that sends a message to your console (/dev/console) every 5 minutes.

```
cat cronfile
5 * * * * *  echo "Time is up." > /dev/console
CONTROL-D
```

**Step 2** Install the crontab file.

```
crontab cronfile
```

**Step 3** Verify the program has been installed.

```
crontab -l
```

**Step 4** Verify the program runs on schedule.

**Step 5** Shut down your system.

## Module 6

**Exercise 1** Set your system up for remote printing to the serial printer attached to the Leader's system (this assumes the system has been set up with the host name jabberwock).

To set up remote printing:

**Step 1** Log on to your A/UX system as root.

**Step 2** Edit the file /etc/printcap to add the remote hostname (jabberwock) and printer designation (iw) to the section controlling remote printing. The file should look something like:

```
# remote UNIX host
remote|remote line printer:\
    :lp=:rp=iw:rm=jabberwock:sd=/usr/spool/lpd/Remote:
```

**Step 3** Print the file /etc/passwd to the remote printer.

```
# lpr -Premote /etc/passwd
```

**Exercise 2** Set your system up for printing to an additional, non-default, AppleTalk printer.

**Step 1** Check which printer is currently selected. Choose Chooser from the  menu. This is the default printer.

**Step 2**

Examine the printers available on the network using the `at_cho_prn` command:

```
# at_cho_prn
```

Do not change the setting, but do note the printer Object name that was *not* selected in the Chooser.

**Step 2**

Create a directory in /usr/spool/lpd with *exactly* the same Object name as the non-default printer reported by `at_cho_prn`.

```
# mkdir /usr/spool/lpd/printname
```

**Step 3**

Change the permissions on the directory to 755. Change both the owner and group of the directory to daemon.

```
# chmod 755 /usr/spool/lpd/printname
# chgrp daemon /usr/spool/lpd/printname
# chown daemon /usr/spool/lpd/printname
```

**Step 4**

Symbolically link the files ifilter, ofilter, and nfilter in the new directory to their counterparts in /usr/spool/lpd/AppleTalk.

```
# cd /usr/spool/lpd/printname
# ln -s /usr/spool/AppleTalk/ifilter ifilter
# ln -s /usr/spool/AppleTalk/ofilter ofilter
# ln -s /usr/spool/AppleTalk/nfilter nfilter
```

**Step 5**

Make a named pipe to use for spooling. Change its permissions to 660:

```
# mknod pipe p; chmod 660 pipe
```

**Step 6**

Change both the owner and group of the new files to daemon.

```
# chgrp daemon ifilter ofilter nfilter pipe
# chown daemon ifilter ofilter nfilter pipe
```

**Step 7** Edit /etc/printcap to add an entry for the new printer. The file entry should look something like:

```
p2|at2|printname|postscript|PostScript:\
    :lp=/dev/null:\
    :if=/usr/spool/lpd/printname/ifilter:\
    :of=/usr/spool/lpd/printname/ofilter:\
    :nf=/usr/spool/lpd/printname/nfilter:\
    :sd=/usr/spool/lpd/printname:
```

**Step 8** Activate the printer, queue, and daemon for the new printer:

```
# lpc up p2
```

**Step 9** Print the file /etc/printcap to the AppleTalk printer.

```
# lpr -Pp2 /etc/printcap
```

**Step 10** In preparation for the next module, drag the MacTerminal icon from the /usr/tmp folder into the MacPartition disk.

**Step 11** Shut down your system.

## Module 7

**Exercise 1**

**Step 1** Decide which computer will run the Macintosh Operating System (hereafter called the Mac system) and which computer will run A/UX (hereafter called the A/UX system.)

**Step 2** Connect the two system's modem ports together with the appropriate serial cable.

**Exercise 2**

**Step 1** Boot into the Macintosh Operating System on the Mac system. Launch the application MacTerminal (provided on the Training Setup 1 floppy disk.)

**Step 2** Configure MacTerminal to have the following characteristics:

```
Terminal type:            VT100
Mode:                     ANSI
Line width:               80 columns
Baud rate:                9600
Bits per character:       8
Parity:                   None
Handshake:                XON/XOFF
Connection:               To another computer
Connection port:          modem
File transfer protocol:   text
```

**Exercise 3**

**Step 1** Boot into the A/UX system.

**Step 2** Run `setport` to allow a login session on the modem port. (Remember Commando!)

**Exercise 4**

**Step 1** Log into the A/UX system from the Mac system.

**Step 2** Try running the program `/usr/games/worms` from the Mac terminal. What happens?

The game runs just fine, although a bit slower that it does on the console.

**Step 3** Echo the file /etc/gettydefs from the A/UX system to the Mac system.

```
echo /etc/gettydefs > /dev/modem
```

**Step 4** Echo the file /etc/gettydefs from the Mac system to the A/UX console.

```
echo /etc/gettydefs > /dev/console
```

**Step 5** List all the files in the A/UX file system on the Mac system screen.

```
cat /FILES
```

or

```
find / -print
```

**Step 6** Run `setport` to disallow login sessions on the modem port while the Mac system is still listing files. What happens?

At some point the Mac system will (abruptly) stop listing files.

**Step 7** Shut down both systems.

## Module 8

**Exercise 1** In this exercise you will become acquainted with the autorecovery commands and the files in the autorecovery directory, /etc/cml. At various points you may wish to read the on-line manual pages for the following commands and files: `cml`, `eu`, `escher`, `eupdate`, and `esch`.

**Step 1** Power on your Macintosh computer and allow A/UX to boot normally. Log on as root.

**Step 2** Change your working directory to /etc/cml. List the files there. Then examine each of the files. (Since some of these files are both very long and wide, you may want to use the `more` command in a large CommandShell window.)

**Exercise 2** In this exercise you will run the basic autorecovery maintenance commands, which are necessary before you attempt to run autorecovery itself.

**Step 1** Run the `escher` command to get a list of those files that may need to be updated. To do this, enter:

```
escher -m
```

**Step 2** Use the eu command to update autorecovery for the password file. To do this, enter:

```
eu /etc/passwd
```

**Step 3** Run the eupdate command. This command takes about 2 minutes to complete. To do this, enter:

```
eupdate
```

**Step 4** Read the mail message that has been sent to you from the command you issued in Step 1.

**Exercise 3** In this exercise you will test the utility of autorecovery. You will render A/UX unbootable by removing both of your bootable kernels (/unix and /newunix) and your inittab file. Then, with a single Startup command, you will recover your system to a bootable state.

**STOP!** **Make sure you have completed all steps in Exercises 1 and 2 before going on.**

**Step 1** Examine your root directory:

```
ls -aC /
```

Notice /unix and /newunix, your two bootable kernels.

**Step 2** Remove the following files: /unix, /newunix, and /etc/inittab. This removes both of your bootable kernel files and your initialization table. The purpose of this step is to create an unbootable system.

```
rm /unix /newunix /etc/inittab
```

**Step 3** To confirm the deletion, examine your root directory again. The files you removed in Step 2 should be gone.

**Step 4** Restart A/UX. At this point you cannot boot your system; A/UX has been rendered inaccessible. You will receive the message:

```
Open on '/unix' failed
```

**Step 5** Launch the backup kernel. At the `startup#` prompt enter:

```
launch /newunix
```

You will receive the message:

```
Open on '/newunix' failed
```

**Step 6** List your root directory.

```
ls -aC /
```

Notice that /unix and /newunix are missing. Similarly, if you try to cat the file /etc/inittab, you will find it missing also.

**Step 7** Enter the following autorecovery command:

```
esch -bvc0
```

**WARNING** **Do not interrupt the `esch` command. Doing so can cause serious damage to file systems.**

**Note**

Depending on how the drive has been partitioned, the cluster number may need to be changed. The correct cluster number can be obtained by using `dp`.

**Step 8** After autorecovery has run, list the files in your root directory to confirm that the missing /unix has been replaced.

```
ls -aC /
```

**Step 9** Restore the file /newunix from the floppy disk backup (labelled /newunix) made in exercise on backups.

```
cpio -idvu < /dev/rfloppy0
```

**Step 10** Now launch A/UX. You will be able to enter A/UX normally, with your kernel and /etc/inittab file configured as they were before you removed the files. To launch A/UX from the command line, enter:

```
boot
```

**Step 11** Shut down your system once you have confirmed you can use the system.

## Module 9

**Exercise 1** In this exercise you will tighten up the security of your system.

**Step 1** Power up your A/UX system. Log on as root.

**Step 2** Edit the password file to change both the uid and gid of the user nobody to 65534.

```
vipw
```

**Step 3** Create a zero-length file called /.rhosts. Change the permissions on the file to 000.

```
touch /.rhosts
chmod 000 /.rhosts
```

**Step 4** Edit the file /usr/lib/uucp/USERFILE. Change the second line in this file:

```
from:    ,    /
to:      ,    /usr/spool/uucppublic
```

**Exercise 2** In this exercise you will perform a basic system security routineby creating a "launch-only" MacPartition.

**Step 1** Power up your A/UX system.

**Step 2** Cancel the Startup utility by hitting COMMAND-PERIOD while the •A/UX Release 2.0• copyright notification is showing.

**Step 3** Open the MacPartition disk.

**Step 4** Insert a floppy disk and drag the MacPartition bin folder into its icon.

**Step 5** Open the disk icon, then open the bin folder in it. Remove the fsck icon from both the svfs and ufs folders and drag it to the Trash. Rename the folders svfs.1 and ufs.1 respectively.

**Step 6** Close the bin folder on the disk. Drag the commands `read_disk` and `esch` from the MacPartition window to the disk, then eject the disk. Label the disk MacPartition/bin.

**Step 7** Drag the icons read_disk and esch from the MacPartition window to the Trash.

**Step 8** Open the bin folder in the MacPartition window.

**Step 9** Open the svfs folder in the bin window.

**Step 10** Drag the icons fsdb and mkfs from the svfs window to the Trash. Close the svfs window.

**Step 11** Open the ufs folder in the bin window.

**Step 12** Drag the icons fsdb and mkfs from the ufs window to the Trash. Close the ufs window.

**Step 13** Select all the icons in the bin window. Deselect the svfs and ufs folders. Drag all the selected icons to the Trash. Choose Empty Trash from the Special menu.

**Step 14** Double-click on the A/UX Startup icon.

**Step 15** Cancel the Startup utility by hitting COMMAND-PERIOD while the •A/UX Release 2.0• copyright notification is showing.

**Step 16** Try giving the `ls` command to list the contents of your root directory. The system will respond:

```
Couldn't execute ls
```

**Step 17** Double-click the A/UX icon again and allow A/UX to launch. At the login dialog box, log on as root. This demonstrates it is indeed possible to boot from the minimal MacPartition.

**Step 18** When done with this exercise, shut down your system.

## Module 10

**Exercise 1** In this exercise we examine the drivers in the current kernel, then build a new kernel incorporating new features and examine the changes.

**Step 1** Power up your A/UX system. Log on as root.

**Step 2** Open a CommandShell window and examine the drivers installed in the current kernel.

```
module_dump /unix
```

**Step 3** Open a second CommandShell window and copy the current kernel to the file /unix.save.

```
cp /unix /unix.save
```

**Step 4** Build a new kernel removing network support.

```
newconfig nonet
```

**Step 5** Compare the drivers installed in the new kernel with those in the old kernel.

```
module_dump /unix
```

What capabilities have been removed?

Both BNET and AppleTalk support have been removed.

**Step 6** Copy the saved kernel back to the file /unix.

```
cp /unix.save /unix
```

**Exercise 2**

In this exercise we examine the kernel parameters in the current kernel, then modify one and examine the changes. The more adventerous can boot the newly modified kernel in the following advanced exercise.

**Step 1**

Open a CommandShell window and examine the kernel parameters installed in the current kernel.

```
kconfig -a
```

**Step 2**

Modify the NBUF parameter of the current kernel, setting it to a value of 1500.

```
echo "NBUF = 1500" | kconfig
```

or

```
kconfig
NBUF = 1500
CONTROL-D
```

**Step 3**

Open another CommandShell window and verify the value for NBUF has changed.

```
kconfig -a
```

**Advanced Exercise 1**

In the last exercise you modified a kernal parameter that, for most systems, probably creates an unbootable kernel. This exercise tests this theory.

**Step 1**

Reboot A/UX. Do not be surprised if the kernel doesn't boot. Recall that you have a saved version of the old kernel that you can move back to its original name by using A/UX Startup. (You will have to drag the `mv` utility from the bin disk created in the Security module back to the MacPartition disk.)

# Module 11

This exercise simply uses some of the commands discussed to create accounting reports, which are printed to your console screen. For each command, you will read the on-line manual page, then execute various forms of the command.

**Exercise 1** In this exercise, you will turn on system accounting.

**Step 1** Power on your Macintosh computer and allow A/UX to boot normally. Log on as root.

**Step 2** Make backup copies the files /etc/rc and /usr/spool/cron/crontabs/adm:

```
cp /etc/rc /etc/rc.old
cd /usr/spool/cron/crontabs
cp adm adm.old
cd
```

**Step 3** Edit the file /etc/rc. Remove the preceding # character from the lines below:

```
/bin/su adm -c /usr/lib/acct/startup
echo process accounting started
```

**Step 4** Edit the file /usr/spool/cron/crontabs/adm. Remove the first # from each line in the file. When done, it should look like:

```
0 4 * * 1-6 /usr/lib/acct/runacct 2> /usr/adm/acct/nite/fd2log
0 2 * * 4 /usr/lib/acct/dodisk
5 * * * * /usr/lib/acct/ckpacct
15 5 1 * * /usr/lib/acct/monacct
0 * * * 0,6 /usr/lib/sa/sa1
0 18-7 * * 1-5 /usr/lib/sa/sa1
0 8-17 * * 1-5 /usr/lib/sa/sa1 1200 3
# The following is an option to appear in the above directory
0 20 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:00 -i 3600 -uybd
```

**Step 5** Cat the file /usr/adm/.profile and confirm that the following line is present; if it is not, use an editor to add it.

```
PATH=/usr/lib/acct:/bin:/usr/bin
```

**Step 6** Reboot A/UX and log on as root. Take a five minute break while the system collects some accounting data.

## Exercise 2

In this exercise, you will look at the output of various reports, both process accounting and system activity. To save time for this exercise, you will force report generation by entering the `sa1` command a number of times manually, and then reading the reports with the `sar` command. Since you won't be using it for the rest of the course, and to conserve disk space, you will turn the system accounting off at the end the exercise.

**Step 1** Enter the commands:

```
/usr/lib/sa/sa1
sar
```

Repeat this step three or four times.

**Step 2** Execute the command to run `sar` three times, one second apart.

```
sar 1 3
```

**Note** If accounting was recently turned on, `sar` and `acctcom` output may be quite small. If accounting has been on while your system was rebooted, `sar` output may end with the error message "`sar: time change not positive`". In this case, simply remove the file /usr/adm/sa/sa*dd*, where *dd* is the current days date.

**Step 3** Execute the command to run `sar` with the `-A` option.

```
sar -A
```

**Step 4** Run the command `acctcom`.

```
acctcom
```

**Step 5** Run the command `pstat`.

```
pstat
```

**Step 6**

Run the command lav.

```
lav
```

**Step 7**

Run the command timex on the command ps -el.

```
timex ps -el
```

**Step 8**

Now use the mv command to copy the files you saved in exercise 1 back.

```
cp /etc/rc.old /etc/rc
cd /usr/spool/cron/crontabs
cp adm.old adm
cd
```

This will result in system accounting not being turned on when the system is next rebooted.

**Step 9**

Shutdown your system.

# Module 12

**Exercise 1**

Use the find command to locate all the following file types in the A/UX file system:

Symbolic Links
Named Pipes
Sockets

| | |
|---|---|
| Symbolic Links: | **find / -l -print** |
| Named Pipes: | **find / -p -print** |
| Sockets: | **find / -s -print** |

**Exercise 2**

Use the ls command to identify the inode numbers for the following files:

| File | Inode |
|---|---|
| /unix | |
| /etc/inittab | |
| /etc/passwd | |

```
ls -li /unix /etc/inittab /etc/passwd
```