Altos System V™ Series 386 Operations Guide

Document History

EDITION	PART NUMBER	DATE
First Edition	690-21171-001	November 1987
Second Edition	690-21171-002	January 1988
Third Edition	690-21171-003	March 1988
Fourth Edition	690-21171-004	May 1988
Fifth Edition	690-21171-005	October 1988
Sixth Edition	690-21171-006	December 1988
Seventh Edition	690-21171-007	June 1989

Copyright Notice

Trademarks

Limitations

Altos Computer Systems reserves the right to make changes to the product described in this manual at any time and without notice. Neither Altos nor its suppliers make any warranty with respect to the accuracy of the information in this manual.

GUIDE TO YOUR ALTOS SYSTEM V™ SERIES 386 DOCUMENTATION

RUN-TIME SYSTEM

Installation
Part numbers: 690-21170-*nnn*
690-21869-*nnn*

- Installation and upgrade
- Set up Multidrop and UPS

Using the AOM™ Menu System
Part number: 690-18055-*nnn*

- Easy-to-use menus to access programs
- Menu Manager to add, update, remove menus

Operations Guide
Part number: 690-21171-*nnn*

- System administration
- Accounting, file systems
- Backups, port setup
- Communications (UUCP)
- Error messages

Reference (C)
Part number: 690-22869-*nnn*

- Commands (C)

Reference (M)
Part number: 690-22870-*nnn*

- Miscellaneous files (M)

User's Guide
Part number: 690-21178-*nnn*
(Not shipped with the Run-time system)

- Basic concepts and tasks
- Vi, ed, mail, awk, sed
- Shells: sh and csh

TEXT PROCESSING SYSTEM

DOCUMENTER'S WORKBENCH™
Part numbers: 690-15843-*nnn*
690-15844-*nnn*

- Mm macros, reference
- Nroff, troff, tbl, eqn

DEVELOPMENT SYSTEM
Set part number: 690-21585-000

Reference (CP, S, F)

- Programming commands (CP)
- System calls, library routines (S)
- File formats (F)

Programmer's Guide

- Make, SCCS
- Lex, yacc
- Signals, system resources, device drivers
- Adb, sdb
- Shared libraries

C Compiler Library and User's Guide

- I/O functions, pipes
- Curses, terminfo
- Assembly routines
- As, cc, COFF, lint, ld
- Error processing
- Character and string processing

C Compiler Language Reference

- Elements of C
- Program structure
- Declarations, expressions
- Statements, functions
- Preprocessor directives

Macro Assembler User's Guide and Reference

- How to use masm
- Error messages
- Type declarations
- Operands, expressions
- Directives, file control
- Instruction summary

To order the *User's Guide* or any of the above manuals, call 408/434-6688, ext. 3004 and give the manual title and part number.

About This Manual

USING THIS GUIDE

This guide is written for the system administrator. The tasks range from very simple tasks requiring very little knowledge about the operating system to quite complex tasks requiring extensive knowledge about the operating system and your computer.

Each chapter explains the tools and knowledge needed to complete the tasks described in that chapter. In some cases you may be required to seek instruction in another manual, such as the *User's Guide*.

NOTE

The last section of this manual, "Change Information," summarizes the changes that have been made to the manual since the previous version.

Chapter 1, "Introduction," introduces this guide.

Chapter 2, "Starting and Stopping the System," explains how to start and stop the operating system and how to log in as the super-user, the operating system's special system administrator account, and the different modes of operation.

Chapter 3, "Preparing the System for Users," describes how to create accounts for the users who work on your system, how to assign groups, and how to manage user IDs.

Chapter 4, "Using File Systems," tells how to create and mount file systems, how to set permissions, and how to keep the system secure.

Chapter 5, "System Accounting," explains how to maintain free space on the root file system and other file systems.

Chapter 6, "Maintaining File Systems," describes the commands that report how much space is used, locate seldom-used files, and remove or repair damaged files.

Chapter 7, "Backing Up File Systems," describes how to create backup copies of the root file system and other file systems.

Chapter 8, "Using Peripheral Devices," details how to add terminals and other peripheral devices to the system.

Chapter 9, "Performance Management," explains how to reconfigure the system (by modifying the tunable parameters) and how to improve system performance.

Chapter 10, "Solving System Problems," explains how to solve system problems such as a jammed lineprinter or a forgotten password.

Appendix A, "System Directories," lists the most commonly used system directories and files.

Appendix B, "Building a Communication System," explains how to build a communications system using uucp and its related programs.

Appendix C, "The Print Spooler," describes the **lp** and **lpr** print spoolers.

Appendix D, "File Transfer Programs," explains how to transfer ASCII text files from UNIX to UNIX/XENIX, MP/M to UNIX, and UNIX to MP/M on Altos computer systems.

Appendix E, "Operating System Error Messages," lists software and hardware error messages.

Appendix F, "Print Spooler Error Messages," lists error messages generated by the **lp** and **lpr** spoolers.

Appendix G, "Porting Guide," explains how to move your software from an existing XENIX system to UNIX/XENIX System V/386.

MANUAL CONVENTIONS

This section describes documentation conventions used in this manual.

Symbol	Description
boldface type	What you type. For example: Type **tar cv** *files*
boldface type	Used for commands, programs, utilities, functions, and options. For example: Use the **tar** command to back up your files to floppy disk.
italic type	Variables (a value that can change), such as *files*, as shown in the example above. Also for manual titles, such as *Reference (C)*.
reverse type	Key you press. For example: Press Esc. Press Retn.
Ctrl-d	Keys you press simultaneously (separated by a hyphen and shown in reverse type). For example: Ctrl-d means you press and hold the Ctrl key and then press the **d** key.
Esc **c**	Keys you press sequentially.
a, x, n	Variables that can change. They mean: *a* (any letter), *n* (any number), *x* (any letter or number). For example: Version *n.na* Filename *xxxxxx*

ADDITIONAL REFERENCE MATERIALS

For more information on the operating system, see the following list of manuals. To order a manual, call (408) 434-6688, ext. 3004, and give the manual title and part number.

Owner's Guide (part number 690-21264-*nnn* or 690-20351-*nnn* describes how to connect *computer* components and peripherals, turn on power, and use the diagnostic programs.

Altos System V Series 386 Installation (part number 690-21170-*nnn*) explains how to initially install the Run-time system, upgrade existing systems, and set up Multidrop and the Uninterruptible Power Supply (UPS).

Using the AOM Menu System (part number 690-18055-*nnn*) describes how to use the Altos Office Manager (AOM) to install software and manage the operating system.

Altos System V User's Guide (part number 690- 21178-*nnn*) (not shipped with the Run-time system) explains basic operating system concepts and programs, (e.g., vi, ed, sh, csh, mail, sed, and awk).

Altos System V Series 386 Reference (C) (part number 690-22869-*nnn*) and *Altos System V Series 386 Reference (M)* (part number 690-22870-*nnn*) describe the Run-time system commands and extended utilities.

Altos System V Series 386 Development System Set (part number 690-21585-000) contains reference and tutorial materials.

Manuals in this set include:

Altos System V Series 386 C Compiler Library and User's Guide
Altos System V Series 386 C Compiler Language Reference
Altos System V Series 386 Programmer's Guide
Altos System V Series 386 Macro Assembler User's Guide and Reference
Altos System V Series 386 Reference (CP, S, F)

DOCUMENTER'S WORKBENCH (part numbers 690-15843-*nnn* and 690-15844-*nnn*)) describes mm, nroff, troff and typesetting functions and commands.

Contents

APPENDICES

FIGURES

TABLES

TABLES (Cont.)

(BLANK)

Chapter 1
Introduction

(BLANK)

OVERVIEW

The operating system is a powerful system of programs that allow you to accomplish a full spectrum of tasks, from developing programs to creating, editing, and typesetting documents to managing the complete computing needs of a business. The operating system gives the user access to the full resources of your computer. To keep this powerful machine running smoothly, the operating system requires careful control of its operation and a regular schedule of maintenance. This guide explains how to operate and maintain the operating system on your computer, ensuring maximum performance with the least number of system problems.

THE SYSTEM ADMINISTRATOR

Every computer system should have at least one person in charge of system maintenance and operation. That person is called the system administrator (or super-user). It is the system administrator's responsibility to ensure the smooth operation of the system and to perform tasks that require special privileges.

Depending on the size of the system and the number of users on the system, a system administrator's job can be anything from a once-a-week task to a full-time job. Even if the system is small, the system administrator should faithfully perform each required maintenance task.

THE SUPER-USER ACCOUNT

The super-user account is a special account for performing system maintenance tasks. It permits the system administrator unusual privileges that ordinary users do not have, such as accessing all files in the system and executing privileged commands. Many of the tasks presented in this guide require that the system administrator be logged in as the super-user (also sometimes called root). To do this, the system administrator must know the super-user password. Users who are authorized to act as the super-user,

including the system administrator, log in as the super-user only when necessary to perform a system maintenance task. Even if the system administrator is the only person using the system, he should create a user account for himself and use it for day-to-day work, reserving the super-user account for system maintenance tasks only.

The number of individuals who are given the super-user password should be kept to a minimum. Misuse of the super-user functions by naive users can result in lost data, programs, and even the operating system itself.

THE KEYBOARD

Many keys and key combinations have special meanings to the operating system. These keys and key combinations have special names that are unique and may or may not correspond to the keycap labels on your keyboard. To help you find the special keys, the following table shows which keys on an Altos terminal correspond to operating system keys.

Name	Key(s)	Action
BREAK	Break/ Del	Stops current action and returns to the shell. This key is also called the INTERRUPT key.
BACKSPACE	Backspace	Deletes the first character to the left of the cursor.
CTRL-d	Ctrl-d	Signals the end of input from the keyboard; also exits current shell.

Chapter 2
Starting and Stopping the System

(BLANK)

INTRODUCTION

This chapter explains how to start and stop the operating system. It also explains how to log in as the super-user. It assumes you have already installed the operating system.

STARTING THE SYSTEM

Normally, starting the operating system requires nothing more than just turning on the power. Sometimes, however, you may have to perform the following series of steps to prepare the system for operation.

- Cleaning the file system (if the system was improperly stopped)
- Choosing the mode of system operation
- Setting the correct time and date

The following sections describe each of these procedures.

Cleaning the File System

When the system is rebooted, you will see the following message if the system was stopped improperly:

```
The system was not shut down properly.
The root file system will be cleaned.
(Type "no" only if you want to avoid cleaning the file
system)
```

This message indicates that the system was not stopped properly as described in the section "Stopping the System" given later in this chapter. The operating system requires a clean file system to perform its tasks.

To clean the file system, type **y** (for "yes") and press the Retn key. The file system is checked and cleaned; damaged files are repaired and files that can't be repaired are removed. You will see a display of the progress as each step is completed. At some point, you may be asked if you wish to salvage a file. Always answer by typing **y** and pressing the Retn key.

When cleaning is complete, the system usually asks you to choose the mode of operation, but it may also display the following message on the console:

```
** Normal System Shutdown **
```

If it displays this message, the system will automatically reboot.

If the system was shut down correctly, you will see the following message:

```
Type "no" if you do not want to check the file system
```

It is a good idea to answer yes to this, as damage to the file system may occasionally occur even when you shut down correctly or prior to shut down.

LOGGING IN AS THE SUPER-USER

Many system maintenance tasks, when performed during normal operation, require that you log in as the super-user. For example, you must be logged in as the super-user to stop the system.

Before you may log in as the super-user, you will need the super-user password. You also need to see the "login:" message on your terminal's screen.

To log in as the super-user, follow these steps:

1. When you see the "login:" message, type the super-user's login name:

   root

   and press the Retn key. The system then asks for the super-user's password if one has been established.

```
Password:
```

2. Type the super-user's password and press the Retn key. The system does not display the password as you type it, so type each letter carefully.

   The system opens the super-user account and displays the message of the day (if one has been set) and the super-user prompt (#).

   Take reasonable care when you are logged in as the super-user. In particular, you should be very careful when deleting or modifying files or directories. Avoid using wildcard designators (e.g., * or .) in filenames and frequently check your current working directory. Small errors can cause annoying and unwanted changes to the system and user files. Some errors can cause irretrievable damage to a file or the system.

   You can leave the super-user account at any time by pressing Ctrl-d or by typing **logout.**

STOPPING THE SYSTEM

Stopping the operating system takes more than just turning off the computer. You must prepare the system for stopping by using either the **shutdown** or the **haltsys** commands. The following sections describe each command.

Using the Shutdown Command

The **shutdown** command is the usual way to stop the system and should be used when in multiuser mode. Be sure to warn other users that the system is about to be stopped and give them an opportunity to finish their work.

1. Log in as the super-user (see the section "Logging in as Super-User" in this chapter).

2. Type:

   /etc/shutdown -g*nnn*

   where *nnn* is the number of seconds before shutdown (the default is 60), and press the Retn key. The system displays messages that processes are being stopped. Then the following message appears:

```
** Normal System Shutdown **
```

   You may turn off the computer after you see this message.

Using the Haltsys Command

The **haltsys** command may be used to halt the system immediately. In general, it should be used only when no other users are on the system or when in system maintenance mode.

To stop the system with the **haltsys** command, follow these steps:

1. Log in as the super-user (not required when in system maintenance mode). The system opens the super-user account and displays the message of the day and the super-user prompt.

2. Type:

   sync;sync;/etc/haltsys

   and press Retn. The system displays the following message on the console:

```
** Normal System Shutdown **
```

You may now turn off the computer.

CHOOSING THE MODE OF SYSTEM OPERATION

Whenever you turn on or boot up your computer, the system comes up in a multi-user environment, and the following happen:

- The file systems are mounted
- The **cron** daemon is started for scheduled tasks
- The basic networking functions of **uucp** are available for use
- The spooling and scheduling functions of the LP package are available for use
- Users can log in to the ports that are enabled for use.

Operating Levels

This multi-user state is also referred to as "init state 2" because all of the activities of initializing the system are under the control of the **init** process. (See the *Reference (M)* manual for an explanation of **init.**) The "2" refers to entries in the special table /etc/inittab used by **init** to initialize the system to the multi-user state.

Not all activities, however, can be performed in the multi-user state. For example, if you were able to unmount a file system while users were accessing it, you would cause a lot of data to be lost. Hence, for unmounting and other system administration tasks, there is a need for another state, the single-user state (init state s).

The single-user state is an environment in which only the console has access to the system and the root file system alone is mounted. You must be the system administrator

(root) to use this state. As the only user on the system, you are free to perform tasks that affect the file systems and the system configuration.

The state or run level clearly defines the operation of the system. Table 2-1 defines system states.

Table 2-1. System States

Run Level	Description
0	Halt the operating system after terminating all user processes and system daemons, unmounting file systems, and so forth.
1, s, or S	Single-user mode is used to install/remove software utilities, run file system backups/restores, and to check file systems. Though s and 1 are both used to go to single user state, s only kills processes spawned by **init** and does not unmount file systems. State 1 unmounts everything except root and kills all user processes, except those that relate to the console.
2	Multi-user mode is the normal operating mode for the system. The default is that all file systems are mounted in this mode. When the system is powered up, it is put in multi-user mode.
3	Multi-user/Remote File Sharing mode is used to start Remote File Sharing (RFS), connect your computer to an RFS network, mount remote resources, and offer your resources automatically.
4	Initial state for **init** when the system is booted. Allows user to go to single-user mode if root password is entered within 10 seconds.

Table 2-1. System States (Cont.)

Run Level	Description
5	Halt the operating system after terminating all user processes and system daemons, unmounting file systems, and so forth.
6	Halt the operating system without any preliminary cleanup.

How init Controls the System State

The operating system runs in one state or another. The actions that cause the various states to exist are under the control of the **init** process, which is the first general process created by the system at boot time. It reads the file /etc/inittab, which defines exactly which processes exist for which run level.

In the case of the multi-user state (run level 2), **init** scans the file for entries that have a tag for the run level (the tag is a 2) and executes everything after the last colon (:) on the line containing the tag. These tags represent the run levels in Table 2-1.

If you look at your /etc/inittab file, you'll see something that looks like the following. (It is not likely that yours will look exactly like this one; /etc/inittab changes from one configuration to another.)

NOTE

If the /etc/inittab file was removed by mistake and is missing during shutdown, **init** will enter the single-user state (**init s**). While entering single-user state, processes not spawned by **init** will continue to run. You should replace /etc/inittab before changing states again.

```
brc::bootwait:/etc/brc 1>/dev/console 2>&1
ind:4:initdefault:
sulg:4:respawn:/etc/sulogin </dev/console >/dev/console 2>&1
r1:1:wait:/etc/rc1 </dev/console >/dev/console 2>&1
r2:2:wait:/etc/rc2 </dev/console >/dev/console 2>&1
hlt0:5:once:/etc/rc5 </dev/console >/dev/console 2>&1
halt:6:once:/etc/haltsys </dev/console >/dev/console 2>&1
powr:2:powerfail:/etc/powerfail </dev/console >/dev/console 2>&1
conx:346:respawn:/etc/getty console 9600
cons:2:respawn:env - TERM=altos5 /etc/getty console 9600
tt02:2:respawn:env - TERM=altos5 /etc/getty tty02 9600
tt03:2:respawn:env - TERM=altos5 /etc/getty tty03 9600
tt04:2:off:env - TERM=altos5 /etc/getty tty04 9600
tt05:2:respawn:env - TERM=altos5 /etc/getty tty05 9600
tt06:2:respawn:env - TERM=altos5 /etc/getty tty06 9600
tt07:2:respawn:env - TERM=altos5 /etc/getty tty07 9600
tt08:2:respawn:env - TERM=altos5 /etc/getty tty08 9600
tt09:2:off:env - TERM=altos5 /etc/getty tty09 9600
tt10:2:respawn:env - TERM=altos5 /etc/getty tty10 9600
```

The format of each line is:

id:level:action:process

where:

id is one to five characters that uniquely identify an entry.

level is zero or more characters (**0** through **6**, **s**, **a**, **b**, and **c**) that determines what *level*(s) *action* is to take place in. If level is null, the *action* is valid in all levels.

action can be one of the following:

bootwait Start *process* the first time **init** goes from single-user to multi-user state after the system is booted. (If **initdefault** is set to 2, the process will run right after the boot.) **init** starts the process, waits for its termination and, when it dies, does not restart the process.

wait When going to *level*, start *process* and wait until it's finished.

initdefault When **init** starts, it will enter *level*; the *process* field for this *action* has no meaning.

once Run *process* once and don't start it again if it finishes.

powerfail Tells **init** to run *process* whenever a UPS-directed powerfail is detected.

respawn If *process* does not exist, start it, wait for it to finish, and then start another.

restart An Altos-defined action that runs *process* upon recovery from a SHUTSAVE. **Powerfail/shutdown** processes (see **shutdown**(M)) are started before any other processes. This allows re-initialization of certain system functions before resuming normal operation (UPS systems only).

ondemand Synonymous with **respawn**, but used only with level **a**, **b**, or **c**.

off When in *level*, kill process or ignore it.

The **powerfail/shutdown** processes are started before any other processes are started (see **shutype**(M)). This allows re-initialization of certain system functions prior to restarting (UPS systems only).

process
is any executable program, including shell procedures.

can be used to add a comment to the end of a line. Everything after a # on a line will be ignored by **init**.

When changing levels, **init** kills all processes not specified for that level. The following explanation will give you a clearer idea of how the system is controlled by **init**.

Multi-User State

This section explains what happens when you power up and initialize the system, and change the run-level.

Powering Up

When you power up your system, it will enter multi-user state by default. This is the state for normal system operation, where others can log in and use the system. (The *User's Guide* explains tasks and procedures you can do in this state.) You can change the default by modifying the initdefault line in your inittab file. In effect, you go to multi-user state as follows:

1. You turn on the computer.

2. The operating system is loaded and the early system initializations are started by **init.**

3. The run level change is prepared by the /etc/rc2 procedure.

4. Finally the system is made public via the spawning of gettys along the terminal lines.

The **init** sequence looks similar to this:

```
INIT: New run level: 2
      Entering Multi-user Mode...
      WorkNet started.
      System logging started.
      Line printer scheduler started.
      Starting cron.
```

Preparing a Run-Level Change

Now the system must be placed in a particular run level. First, **init** scans the table to find an entry that specifies an *action* of the type **initdefault.** If it finds one, it uses the run level of that entry as the tag it will use

to select the next entries to be executed. In our sample /etc/inittab file, the **initdefault** entry specifies run level 4 (the system maintenance/multi-user state) as the level to select and execute the entry:

```
sulg:4:respawn:/etc/sulogin </dev/console >/dev/console 2>&1
```

The **sulogin** program is executed to determine whether to enter system-maintenance mode or multi-user mode. Unless you have system maintenance functions to perform immediately, you will usually select multi-user mode. The **sulogin** program executes the **init 2** command, which executes /etc/rc2. It executes all files in /etc/rc2.d, which:

- Sets up and mounts the file systems
- Starts the **cron** daemon
- Makes **uucp** available for use, if installed
- Makes line printer (LP) system available for use, if installed

At this moment, the full multi-user environment is established, and your system is available for users to log in.

System-Maintenance (Single-User) State

At times in a given work week, you will need to perform some administrative functions in the single-user mode, such as backing up the hard disk. The normal way to go to single-user mode is through the **/etc/singleuser** command. To do so, become the system administrator (root) and enter:

/etc/singleuser

The **singleuser** script tells you it will terminate all processes and asks if you want to continue. Give the appropriate response for maintenance mode. The program causes the **init** process to scan the inittab file. This procedure executes all the files in /etc/rc1.d directory by calling the /etc/rc1 procedure.

The **init** process enters run level 1 and executes the **r1** entry. The entries for single-user processing in the sample /etc/inittab are:

```
r1:1:wait:/etc/rc1 </dev/console >/dev/console 2>&1
```

When going to init state 1, the system:

- Closes all open files and stops all user processes
- Stops all daemons and services
- Writes all system buffers out to the disk
- Goes to init state s to complete going to single-user mode

With the system in the single-user mode, you can perform the appropriate administrative tasks.

Run Levels

This section describes what happens when there's a run-level change, and explains the different run levels.

Changing Run Levels

In effect, changing run levels follows these broad lines (see Figure 2-1):

1. The system administrator enters a command that directs **init** to execute entries in /etc/inittab for a new run level.
2. Key procedures, such as /etc/shutdown, /etc/rc0, /etc/rc1, /etc/rc2, and /etc/rc3 are run to initialize the new state.
3. The new state is reached. If it is state 1, the system administrator can proceed.

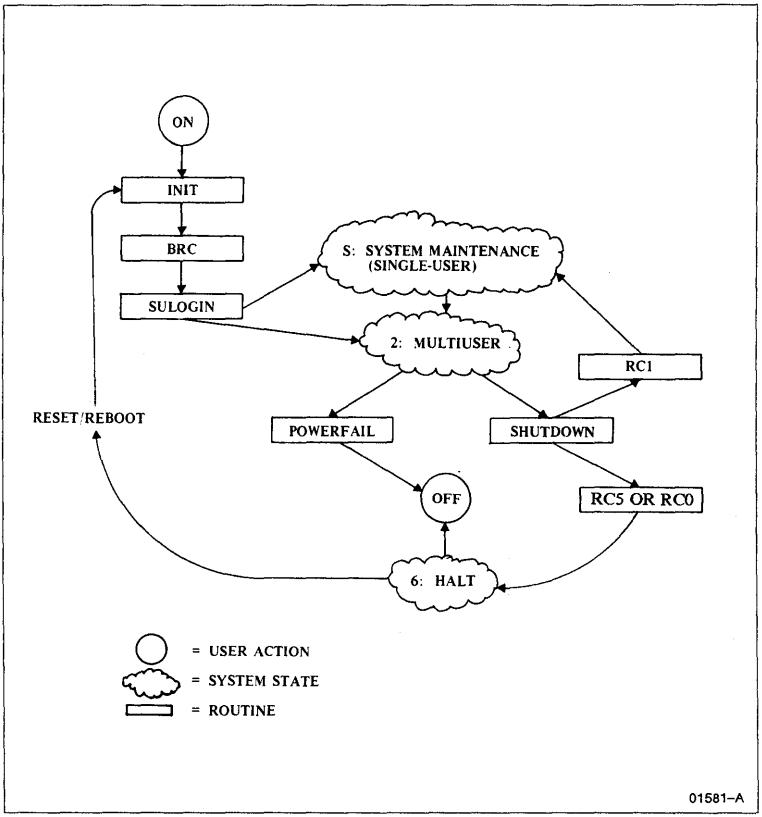


Figure 2-1. A Look at the System Life Cycle

Run-Level Directories

Run levels 0, 1, 2, 3, and 5 each have a directory of files that are executed in transitions to and from that level. These directories are rc0.d, rc1.d, rc2.d, rc3.d, and rc5.d, respectively. All files in these directories are linked to files in /etc/init.d. The run level file name format looks like this:

S*xxname*, K*xxname*, or R*xxname*

where:

S, K or R
: Defines whether the process should be started (**S**), stopped (**K** for killed), or restarted (**R**) upon entering the new run level. The R*xxname* files are executed when power is restored after a power failure on systems with an UPS (Uninterruptible Power Supply).

xx
: Indicate number from 00 to 99. The number indicates the order in which the files will be started (S00, S01, S02) or stopped (K00, K01, K02).

name
: Defines the /etc/init.d file name this file is linked to.

For example, the init.d file cron is linked to the rc2.d file S75cron and rc0.d file K75cron. When you enter init 2, this file is executed with the **start** option: **sh S75cron start.** When you enter init 0, this file is executed with the **stop** option: **sh K75cron stop.** This particular shell script will execute /etc/cron when run with the **start** option and kill the **cron** process when run with the **stop** option. R*xxname* files get the option **restart.**

Because these files are shell scripts, you can read them to see what they do. You can modify the files, though it is preferable to add your own script since the delivered scripts may change in future releases. To create your own scripts follow these rules:

- Place the file in /etc/init.d.
- Link the file to files in appropriate run state directories using the naming convention described previously.
- Have the file accept the **start, stop** and/or **restart** options.

Run Level 0

Run level 0 is used to halt the system to a state where it is safe to reset/reboot or power off the system. When the **shutdown**(M) command is used to directly halt the

system, the **init** process is forced into state 0 where the /etc/rc0 script executes the files in /etc/rc0.d to shut down system and user services.

Run Level 3

Init 3 is used to enter the Remote File Sharing state (run level 3). This procedure executes the files /etc/rc2 and /etc/rc3 to run processes in all directories associated with those two states. On top of the multi-user state (state 2) processes, the processes in the file /etc/rc3.d will:

- Start Remote File Sharing and connect you to the Remote File Sharing network
- Advertise your resources to remote computers
- Mount remote resources on your computer

Details on Remote File Sharing are discussed in the Remote File Sharing manual.

Run Level 4

This state occurs when the system is booted. There is a choice of going to single-user (system maintenance) mode if the root password is entered within 10 seconds, or to multi-user mode so other users can log in.

Run Level 5

Run level 5 is the same as run level 0, and is included to maintain compatibility with previous releases. See run level 0 above.

Run Level 6

This run level is used to halt the operating system without any preliminary cleanup (not recommended as there is no cleanup).

Turning Off the System

To shut down the system from any run level, use the **shutdown**(M) command. *Always* run this command before turning off the power. To shut down the system, enter:

/etc/shutdown *nnn*

where *nnn* is the number of seconds (0 to 999). The /etc/shutdown procedures clean up and stop all user processes, daemons, and other services, unmount the file systems, and shut the system down. **Shutdown** does most of its work by putting **init** in state 0.

Chapter 3
Preparing the System for Users

(BLANK)

INTRODUCTION

User accounts control each person's access to system resources. Each user must have a user account. Each account has a unique login name and may have a password with which the user enters the system, and a home directory where the user works.

It is the system administrator's job to create accounts for all users on the system. It is also the administrator's job to maintain user accounts by deleting or changing user passwords, login groups, and user IDs when necessary.

This chapter explains how to:

- Invoke the User Administration program (**ua**)
- Add user accounts to the system
- Create a group
- Change an account's login group
- Change an account's user ID
- Remove user accounts from the system
- Change an account's password

The following sections describe each in detail.

INVOKING USER ADMINISTRATION

To invoke the user administration program, log in as **root**, then enter:

ua

and press the Retn key. The user administration screen appears (an example screen follows).

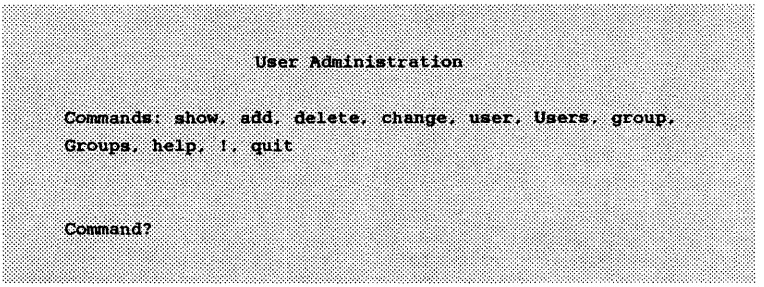


Figure 3-1. User Administration Screen

User Administration Commands

A line of commands displays at the top of the User Administration screen. Table 3-1 contains descriptions of each command.

Table 3-1. User Administration Commands

Command	Description
add **u**ser *username*	Adds a new user to the system
add **g**roup *groupname*	Adds a new group to the system
delete **u**ser *username*	Deletes a user from the system
delete **g**roup *groupname*	Deletes a group from the system
show **u**ser *username*	Displays a user's attributes
show **g**roup *groupname*	Displays a group's attributes
change **u**ser *username*	Changes a user's attributes
change **g**roup *groupname*	Changes a group's attributes
show **U**sers	Shows all current users
show **G**roups	Shows all current groups
! [*shell command*]	Returns to the shell; executes commands
help	Displays the help screen
quit	Returns to the AOM Shell or shell

Creating and Changing a User Account

Enter just the first letter of a command. Guidelines for creating/changing a user account follow.

To add a user:

1. Enter **a** ("add" appears at the bottom of the screen).
2. Enter **u** ("user" appears next to "add").
3. Enter a user name (i.e., rich), and press Retn.

   For example, to add a user named rich, invoke the User Administration program and enter the following:

   a u rich

   and press Retn. The system responds with:

```
Update new user: rich
```

You can only enter one user or group at a time; **add Users** and **add Groups** are not legal commands.

NOTE

If you make a mistake while typing, press Ctrl-x to backspace over an entire user name or group name.

The **ua** program automatically assigns a user ID, group ID, full name, home directory, and default login shell. Initially there is no password, so that the new user can select a password. If your machine is on a network, it is best if the user ID and account name for this user are the same on all machines in the network.

The screen then displays the system settings for rich (see Figure 3-2). If you want to change a setting, select the letter that precedes it.

```
                    User Administration

Commands: show, add, delete, change, user, Users, group,
Groups, help, !, quit


              a. User:       rich
              b. User ID:    28
              c. Group:      other
              d. Group Id:   1
              e. Password:   <NOT SET>
              f. Full Name:  rich
              g. Directory:  /usr/rich
              h. Shell:      /bin/aomlogin

              q. (quit -- return to top level)


Which Field?
```

Figure 3-2. Example of Creating a New User Account

Then type **q** to cause the changes to take effect. The system displays:

```
Installing files from /etc/newuser

Command?
```

You can now enter another User Administration command, or type **q** again to return to the system prompt.

Guidelines

When you are creating or changing a user account, follow these guidelines:

1. Make the user name short (the user will enter it often). A user name can have up to 8 letters or numbers, but it cannot contain a space.

   The user enters the name exactly as created. Use only lower-case letters.

2. Do not use a name with upper-case letters unless that person actually has a terminal with only upper-case letters. If a name is created with all upper-case letters, the operating system assumes that the user has a terminal that supports only upper-case letters. Strange things happen, and the use of the system is hampered.

3. Choose the login shell the users will use. The shells are:

   - /bin/aomlogin -- Altos Office Manager (with menus)
   - /bin/sh -- Bourne Shell
   - /bin/csh -- C Shell

CREATING A GROUP

A group is a collection of users who share a common set of files and directories. The advantage of groups is that users who have a common interest in certain files and directories can share these files and directories while protecting them from other users. Initially, all new users belong to the common system group named "other," but you can create new groups by invoking the User Administration program.

To create a new group, you need to choose a group name and a group identification number (group ID). You also need to make a list of the users in the new group. The group name may be any sequence of letters and numbers up

to eight characters long, and the group ID may be any number in the range 0 to 32768. Both the group name and ID must be unique; that is, they must not be the same as any existing group name or ID.

The following example shows how to create a new group, called accounting:

1. Log in as **root.**

2. Invoke the User Administration program by typing:

   ua

   and press Retn.

3. Type:

   a g accounting

   and press Retn.

The screen displays:

```
a.  Group:  accounting
b.  Group Id:  8
    Members:

c.  (add a new member)
d.  (delete a member)
q.  (quit -- retn to top level)
```

You may add or delete a new member, or quit the program by typing the letter opposite the desired command (e.g., **q** for quit).

You can create any number of new groups. Each group may have any number of members. Furthermore, any user may be a member of any number of groups. Multiple group membership is especially convenient for users who have interests that span a variety of areas.

A user who is a member of several groups can gain access to each group by using the **newgrp** command. See the *User's Guide* for details.

CHANGING A USER'S LOGIN GROUP

At login time, a user is automatically placed in the proper login group. This is the group given by the group ID (a number) in the user's /etc/passwd file entry (see the section "Adding a User Account" in this chapter). You can change the user's login group by changing the group ID. To change the group ID you need the group ID of the new login group.

The following example shows how to change the group ID.

1. Log in as the **super-user.**

2. Invoke the User Administration program (**ua**), and type:

   c u peter

   and press the Retn key. The screen will list all information pertaining to the user.

```
a.  User: peter
b.  User Id:  12
c.  Group:  other
d.  Group Id:  1
e.  Password:  <SET>
f.  Full Name:  Peter C. Adams
g.  Directory:  /usr/peter
h.  Shell:  /bin/sh
q.  (quit -- Retn to top level)
```

You can change any of the information, by typing the letter preceding the data you want to change. Type **d** to modify the Group ID field to any number.

You must not change the group IDs for special system accounts such as "cron" and "root." System accounts have user ID numbers less than 200. The user ID is the third item in the password entry.

Note that changing a user's login group does not change the "group ownership" of that user's files. Group ownership defines which group has access to a user's files. If users in the new group wish to access the user's files, you must change the group ownership with the **chgrp** (for "change group") command. For details, see the section "Changing Group Ownership" in Chapter 4.

CHANGING A USER ID

Sometimes it is necessary to change the user ID in a user's account entry to allow a user to access files and directories transferred from other computers. In particular, if a user has different accounts on different computers and frequently transfers files and directories from one computer to another, then it makes life easier if the user IDs in each of his account entries are the same. In other words, for all users except the super-user, the /etc/passwd file entries for those users should be identical on all machines.

To change a user ID, follow these steps at every computer for which the user has an account:

1. Log in as the **super-user.**

2. Invoke the User Administration program by typing:

   ua

   and pressing the Retn key.

3. Follow the same steps as in the previous section, (Changing a User's Login Group) except type **b** instead of **d** to modify the user ID field.

In most cases, you can change the user ID to the same number as the user's most-used account. But the new number must be unique on every system on which the user

has an account. If there is any conflict (for example, if the number already belongs to another user on one of the systems), you must choose a new number. Just make sure it is unique, and that you copy it to all systems on which the user has an account.

Once a user's ID has been changed, you must change the "user ownership" of the user's files and directories from the old user ID to the new one. You can do this with the **chown** (for "change owner") command described in Chapter 4. For example, to change the ownership of johnd's home directory, type:

chown johnd /usr/johnd

and press Retn. Note that you may use the **find** command, described in Chapter 6, to locate all files and directories with the user's old user ID.

REMOVING A USER ACCOUNT

It is sometimes necessary to remove a user account from the system. Before you can remove the user account, you can remove all files and directories from the user's home directory, or move them to other directories. If you wish to save the files, you may use the **tar** command to copy the files to a floppy disk (see the section "Copying Files to a tar Disk or Tape" in Chapter 7).

For example, to remove a user account, follow these steps:

1. Log in as the **super-user.** Enter:

   ua

   and press Retn.

2. Type:

   d u *username*

   and press Retn.

3. The system then responds with:

```
Delete username's home directory (/usr/username)?
(y/n)
```

4. Type **y** to remove all the files in the user's home directory. Type the letter **n** to only delete the user, and not the files in the user's home directory.

5. If you type **y** to delete the user's home directory, the screen displays:

```
Removing directory /usr/username
User username deleted
```

6. If you choose to only delete the user and not the user's home directory, the screen displays:

```
Directory /usr/username not removed
User username deleted
```

7. You are now back to the command level in the User Administration program.

CHANGING A USER'S PASSWORD

Normally, a user can change the password of his own account with the **passwd** command (see the *User's Guide).*

A password can have a minimum of one character and a maximum of six characters.

Sometimes, however, it may be necessary for the super-user to change a user's password. The super-user may change the password of any user (including himself) with the **passwd** command.

To change a password, follow these steps:

1. Log in as **root.**

2. Type:

   passwd *login-name*

   (where *login-name* is the user's login name) and press Retn. The command displays the message:

```
New password:
```

3. Type the new password and press Retn. The password does not display as you type it, so type carefully. The command asks you to type the password again:

```
Retype new password:
```

4. Type the password again and press Retn.

To see how an ordinary user can change his own password with the **passwd** command, see the *User's Guide.*

(BLANK)

Chapter 4
Using File Systems

(BLANK)

INTRODUCTION

This chapter describes one of the most important responsibilities of a system administrator: controlling and recording users' access to the files and directories on the system. It introduces file systems, permissions, and system security.

FILE SYSTEMS

A file system is a logical arrangement of files, directories, and the information needed to locate and access these items.

Each computer has at least one file system. This main file system is called the root file system and is represented by the slash symbol (/). The root file system is where the Run-time System is installed. It usually contains all the user directories as well.

You create a file system with the **mkfs** command. This command sets the size and format of the file system. You can mount additional file systems with the **mount** command. Once mounted, you may access the files and directories in the new file system as easily as files and directories in the root file system. (The root file system is permanently mounted.) When you no longer need to access a file system (for example, you want to physically disconnect a second hard disk), use the **umount** command to unmount the disk before you disconnect it.

If you add an additional hard disk to your machine, you must create an additional file system on the hard disk with the **add.hd** command and tell the operating system where to find the file system with the **mount** command.

You can create file systems on hard and on floppy disks. A reason for creating new file systems on floppy disks is to establish a collection of application programs and data files that can be easily mounted and used when needed.

FILE SYSTEM REQUIREMENTS

Many of the file system maintenance tasks described in this chapter require the use of special filenames, block sizes, and gap and block numbers.

SPECIAL FILES

A special file is the name of the device special block or character I/O file corresponding to a peripheral device, such as a hard or floppy disk drive. These names are required in such commands as **mfks, mount,** and **df** to specify the device containing the file system to be created, mounted, or searched. For example:

/dev/hd0b (root file system)
/dev/fd0 (floppy device)

BLOCK SIZES

The block size of a disk is the number of sectors of storage space available in one logical block. For example, for Altos System V on the Altos 386 system, the block size is 2K bytes. Many commands require input that defines the number of blocks to be operated on. Other commands report disk space in terms of blocks, while others report physical disk sectors, which are 512 bytes each. The block size of a typical floppy disk depends on the total storage capacity of the disk as given by the manufacturer.

GAP AND BLOCK NUMBERS

The gap and block numbers are used by the **mkfs** command to describe how the blocks are to be arranged on a disk.

Use the following formula to determine the number of blocks (sectors/cylinder):

> number of heads x sectors/track = number of sectors per cylinder

The following table gives the file system rotational gaps currently used on Altos systems.

CPU	Controller		
	ESDI	ST-506	SCSI
Intel 80386	9	5	17

NOTE

These values may change in the future.

To see the number of sectors/track (spt), enter:

drive /dev/*device*.drinfo spt

and press Retn. *Device* is a special device such as hd0.

To see the number of heads, enter:

drive /dev/*device*.drinfo heads

and press Retn.

Creating a File System

You can create a file system on a formatted floppy disk by using the **mkfs** command.

Note that if a file system already exists on the disk, it will be destroyed by this procedure. For this reason, be particularly careful not to create a new file system on top of the root file system. If you destroy the root file system, you will have to reinstall the operating system.

To make a file system on a floppy disk, follow these steps:

1. Log in as **root.**

2. Insert a formatted floppy disk into a floppy disk drive. Make sure there is no write-protect tab on the disk.

3. Type:

/etc/mkfs *specialfile blocks*

where you supply *specialfile* and *blocks* and press Retn. The system creates the file system.

For example, the following command creates a file system on the floppy /dev/fd0 with 1440 blocks (which is the size of a standard Altos floppy).

/etc/mkfs /dev/fd0 1440 Retn

Mounting a File System

Once you have created a file system, you tell the system where to find that file system with the **mount** command. The **mount** command allows a user to associate a file system on floppy or disk with a directory name. After mounting, the file system can be referred to with the directory name.

To mount a file system you need:

- The name of a disk drive (*specialfile* as specified in **mkfs**)
- The name of an empty directory

Disk drive names (*specialfile*) are given in Appendix A. The directory to receive the file system may be any empty directory (contains no files) that is not your current working directory. Note that the directory, /mnt, is specifically reserved for mounted file file systems.

For example, to mount a file system, follow these steps:

1. Log in as the super-user.
2. Insert the disk containing the file system into a floppy disk drive.
3. Type the appropriate **mount** command, and press Retn. The command should have the form:

   /etc/mount *specialfile directoryname*

   where *specialfile* is the name of the disk drive containing the disk and *directoryname* is the name of the directory to receive the file system. If the disk has a write-protect tab, make sure you include the **-r** option.

For example, you may use the following command to mount the floppy disk, /dev/fd0, onto the directory named /account.

/etc/mount /dev/fd0 /account Retn

If the command displays the message:

```
mount: Structure needs cleaning
```

or

```
mount: Possibly damaged file system
```

use the **fsck** command to clean the file system and try to mount it again (see the section "File System Integrity" in Chapter 5). If the **mount** command displays the message:

```
mount: Device busy
```

either the file system has already been mounted and cannot be mounted twice, or a user is currently somewhere in the directory (directory name). If a user is in the directory, the user must leave that directory before you can use it with the **mount** command.

To check that the file system was properly mounted, use the **cd** command to change to the directory containing the mounted system and the **ls** command to list the contents. The command displays the files and directories in the file system. Be sure to use the **cd** command to leave the directory after finishing your work in it.

Figure 4-1 shows a file system tree before and after mounting.

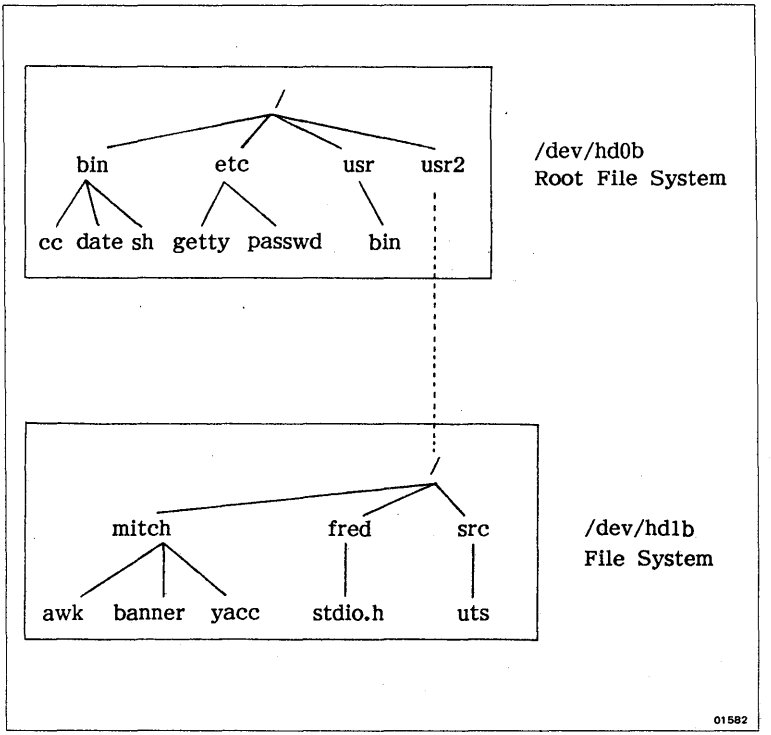


Figure 4-1. File System Tree Before and After Mounting

Note that frequently used file systems can be mounted automatically when starting the system by adding the appropriate mount commands to the /etc/fstab file. See the section "Changing the System Start Up Files" in Chapter 10 for details and the description of this file in the *Reference (M)*.

Unmounting a File System

When you no longer need to access a mounted file system, unmount it with the **umount** command. Unmounting a file system does not destroy its contents. It merely removes access to the files and directories in that file system. If

you don't do this, the file system will be "unclean" when mounting again (e.g., a floppy). To unmount a mounted file system, type:

/etc/umount *specialfile*

and press Retn. There is only one "n" in "umount"; *specialfile* is the name of the special file corresponding to the disk drive containing the disk with the file system. The files and directories on the device are no longer accessible until the device is remounted. The directory is then available for mounting another file system.

For example, the following command unmounts a file system from the second hard disk /dev/hd1b:

/etc/umount /dev/hd1b Retn

Before unmounting a file system, make sure that no files or directories are being accessed by other commands or programs. The **umount** command displays the message:

```
umount: Device busy
```

if someone is currently using the file system.

PERMISSIONS

Permissions control access to all the files and directories in an operating system. A user can access only those files and directories for which he has permission. All other files and directories are inaccessible. The super-user can access all files and directories.

Permissions can be set for three types of users: the user, group, and other (where user is the owner of the file; group specifies all users with the same group ID as the owner; and other specifies all other users).

Displaying Permissions

You can display the permission settings for all the files in a directory with the ls -l (for "list directory") command. This command lists the permissions along with the name of the file's owner, the size (in bytes), and the date and time the file was last changed. The command display has the following format:

```
-rwxrw---- 1 johnd  group  11515 Nov 17  14:21 file1
```

The permissions are shown as a sequence of ten characters at the left of the display. The sequence is divided into four fields. The first field (the "type" field) has a single character, the other fields ("user," "group," and "other") have three characters each. The characters in the fields have the following meanings.

In the "type" field:

d Indicates the item is a directory

- Indicates the item is an ordinary file (the above file is an ordinary file)

b Indicates the item is a device special block I/O file

c Indicates the item is a device special character I/O file

m Indicates a shared memory file

p Indicates a FIFO (named pipe)

s Indicates a semaphore file

In the "user," "group," and "other" fields:

r Indicates read permission. Read permission for a file means you may copy or display the file. Read permission for a directory means you may display the files in that directory.

w Indicates write permission. Write permission for a file means you may change or modify the file. Write permission for a directory means you may create files or subdirectories within that directory.

x Indicates execute permission (for ordinary files) or search permission (for directories). Execute permission for a file means you may invoke the file as you would a program. Execute permission for a directory means you may enter that directory with the **cd** command or list the contents of the directory.

\- Indicates no permission.

s Indicates the set-user-ID or set-group-ID bit is on, and the corresponding user or group execution bit is also on.

S Indicates the set-user-ID or set-group-ID bit is on, and the corresponding execution bit is off.

t Indicates the 1000 (octal) bit, or sticky bit is on, and execution is on.

T Indicates the 1000 (octal) bit, or sticky bit is on, and execution is off.

For example, the permissions

```
-rwxrwxrwx
```

indicate an ordinary file with full read, write, and execute access for everyone (file owner, group, and other).

The permissions

```
-rw-------
```

indicate an ordinary file with read and write access for the file owner.

The permissions

drwxr-x--x

indicate a directory with search access for everyone, read access for the user and group, and write access for only the user.

For example, when you create a file, the operating system normally assigns the following permissions:

-rw-rw-r--

This means that everyone may read the file, but only the user and group may write to it. When you create a directory, the system assigns the permissions:

drwxr-xr-x

This means everyone may search and read the directory, but only the user may create and remove files and directories within it.

Changing Permissions

To change the permissions of a file you must be the owner of that file or the super-user. You can change the permissions of a file or a directory with the **chmod** (for "change mode") command. This command changes the permissions of a specific file or directory. You indicate which levels of permissions you wish to change (user "u", group "g", or move "-"), and which permissions you wish to change (read "r", write "w", or execute "x"). For example, the pattern:

chmod u+x

adds execute permission for the user. The pattern:

chmod go-w

removes write permission for group and other.

The command has the form:

chmod *pattern file ...*

where *file* is the name of a file or directory. If more than one name is given, they must be separated by spaces. For example, to change the permissions of the file "receivables" from "-rw-r--r--" to "-rw-------", type:

chmod go-r receivables

and press Retn.

After using **chmod**, use the **ls -l** command to check the results. If you have made a mistake, use **chmod** again to correct the mistake.

Changing the File Creation Mask

The file creation mask is a special number, kept by the system, that defines the permissions given to every file and directory when it is created by a user. Initially, the mask has the value "022" which means every file receives the permissions:

-rw-r--r--

and every directory receives the permissions

drwxr-xr-x

You can change the mask and the initial permissions your files and directories receive, by using the **umask** command.

The **umask** command has the form:

umask *value*

where *value* is a three-digit number, one digit each for file owner, group, and other, respectively. Each digit can have a value 0-7 as shown by the following table. For directories "execute" means search permission.

Table 4-1. Mask Values and Permissions

Digit	Permission
0	Read, write and execute
1	Read and write
2	Read and execute
3	Read
4	Write and execute
5	Write
6	Execute
7	No permissions

MANAGING FILE OWNERSHIP

Whenever a file is created by a user, the system automatically assigns ownership of that file to that user. This allows the creator to access the file according to the default permissions, discussed in the previous section. The system also assigns a "group" ownership to the file. The group ownership defines which group may access the file according to the "group" permissions. The default group is the one group the user belongs to when creating the file.

Only one user and one group may have ownership of a file at any one time. (These are the owner and group displayed by the **ls -l** command.) However, you may change the ownership of a file by using the **chown** and **chgrp** commands.

Changing User Ownership

You can change the user ownership of a file you own with the **chown** command. Only the super-user can change user ownership of another user's files. The command has the form:

chown *login-name file . . .*

where *login-name* is the name of the new user and *file* is the name of the file or directory to be changed. For example, the command

chown johnd projects.june

changes the current owner of the file projects.june to "johnd."

The **chown** command is especially useful after changing the user ID of a user account (see the section "Changing a User's ID" in Chapter 3).

Changing Group Ownership

You can change the group ownership of a file you own with the **chgrp** command. Only the super-user can change group ownership of another group's files. The command has the form:

chgrp *group-name file . . .*

where *group-name* is the name of a group given in the /etc/group file and file is the name of the file you wish to change. For example, the command

chgrp shipping projects.june

changes the group ownership of the file *projects.june* to the group named shipping.

The **chgrp** command is especially useful if you have changed the login group of a user (see the section "Changing a User's Login Group" in Chapter 3).

SYSTEM SECURITY

Every system, no matter what its size, should have some form of protection from unauthorized access to the computer, disks, and system files. The following sections suggest ways for a system administrator to protect the system.

Physical Security

You can protect the physical components of the computer, especially floppy disks, by taking these steps:

1. Keep unauthorized personnel out of the work area.

2. Organize and lock up all floppy disks when not in use. They should not be stored with the computer itself.

3. Keep disks away from magnetic sources, direct sunlight, smoke, and severe changes in temperature.

4. Do not use ball point pens to write labels on disks.

5. Make backup copies of all floppy disks.

Access Security

You can protect the system from access by unauthorized individuals by taking these steps:

1. Remind users to log out of their accounts before leaving the terminal.

2. Discourage users from choosing passwords that are easy to guess. Passwords should be at least six characters long and include letters, digits, and punctuation marks.

3. Keep the super-user password secret from all but necessary personnel.

Protect Special Files

You can prevent ordinary users from gaining direct access to the data and program files on the system's hard and floppy disks by protecting the system's special files. The special files, in the /dev directory, are used primarily by the system to transfer data to and from the computer's hard and floppy disks as well as other devices, but can also be used by ordinary users to gain direct access to these devices.

Since direct access bypasses the system's normal protection mechanisms and allows ordinary users to examine and change all files in the system, it is wise to protect the special files to ensure system security.

To protect the special files, log in as the super-user and use the **makedevs** command to set appropriate permissions.

Chapter 5
System Accounting

(BLANK)

The operating system accounting provides methods to collect perprocess resource utilization data, record connect sessions, monitor disk utilization, and charge fees to specific logins. A set of C language programs and shell procedures is provided to reduce this accounting data into summary files and reports. This chapter describes the structure, implementation, and management of this accounting system, as well as a discussion of the reports generated and the meaning of the columnar data.

GENERAL

The following list is a synopsis of the actions of the accounting system:

- At process termination the system kernel writes one record per process in /usr/adm/pacct in the form of acct.h.
- The **login** and **init** programs record connect sessions by writing records into /etc/wtmp. Data changes, reboots, and shutdowns (via **acctwtmp**) are also recorded in this file.
- Disk utilization program **acctdusg** and **diskusg** break down disk usage by login.
- Fees for file restores, etc., can be charged to specific logins with the **chargefee** shell procedure.

ACCOUNTING

- Each day the **runacct** shell procedure is executed via **cron** to reduce accounting data and produce summary files and reports.
- The **monacct** procedure can be executed on a monthly or fiscal period basis. It saves and restarts summary files, generates a report, and cleans up the sum directory. These saved summary files could be used to charge users for system usage.

FILES AND DIRECTORIES

The /usr/lib/acct directory contains all of the programs and shell procedures necessary to run the accounting system. The adm login (currently user ID of 4) is used by the accounting system and has the login directory structure shown in Figure 5-1.

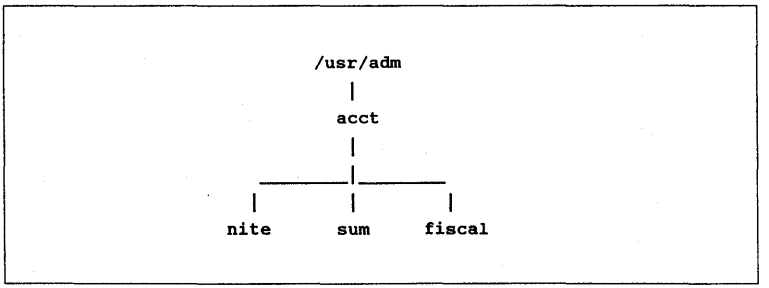


Figure 5-1. Directory Structure of the "adm" Login

The /usr/adm directory contains the active data collection files. (For a complete explanation of the files used by the accounting system, see Figure 5-2 at the end of this section.) The nite directory contains files that are reused daily by the **runacct** procedure. The sum directory contains the cumulative summary files updated by **runacct**. The fiscal directory contains periodic summary files created by **monacct**.

DAILY OPERATION

When the system comes up into multiuser mode and executes the /etc/rc2.d/S??acct script, /usr/lib/acct/startup is executed which does the following:

1. The **acctwtmp** program adds a "boot" record to /etc/wtmp. This record is signified by using the system name as the login name in the wtmp record.

2. Process accounting is started via **turnacct.** Turnacct on executes the **accton** program with the argument /usr/adm/pacct.

3. The **remove** shell procedure is executed to clean up the saved pacct and wtmp files left in the sum directory by **runacct.**

The **ckpacct** procedure is run via **cron** every hour of the day to check the size of /usr/adm/pacct. If the file grows past 1000 blocks (default), **turnacct** switch is executed. The advantage of having several smaller pacct files becomes apparent when trying to restart **runacct** after a failure processing these records.

The **chargefee** program can be used to bill users for file restores, etc. It adds records to /usr/adm/fee which are picked up and processed by the next execution of **runacct** and merged into the total accounting records.

Runacct is executed via **cron** each night. It processes the active accounting files, /usr/adm/pacct, /etc/wtmp, /usr/adm/acct/nite/disktacct, and /usr/adm/fee. It produces command summaries and usage summaries by login.

When the system is shut down using **shutdown,** the **shutacct** shell procedure is executed. It writes a shutdown reason record into /etc/wtmp and turns process accounting off.

Each morning execute the /usr/lib/acct/prdaily command to print the previous day's accounting report.

SETTING UP THE ACCOUNTING SYSTEM

In order to automate the operation of this accounting system several things need to be done:

1. Link the /etc/init.d/acct file to a start file in the /etc/rc2.d directory and to a stop file in the /etc/rc0.d directory.

```
ln /etc/init.d/acct /etc/rc2.d/S??acct
ln /etc/init.d/acct /etc/rc0.d/K??acct
```

2. For most installations, the following screen contains three entries that should be made in /usr/spool/cron/crontabs/adm so that **cron** will automatically run the daily accounting.

```
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
```

3. To facilitate monthly merging of accounting data, the following entry in /usr/spool/cron/crontabs/adm will allow **monacct** to clean up all daily reports and daily total accounting files and deposit one monthly total report and one monthly total accounting file in the fiscal directory.

   15 5 1 * * /usr/lib/acct/monacct

   The above entry takes advantage of the default action of **monacct** that uses the current month's date as the suffix for the file names. Notice that the entry is executed at such a time as to allow **runacct** sufficient time to complete. This will, on the first day of each month, create monthly accounting files with the entire month's data.

4. The PATH shell variable should be set in /usr/adm/.profile to:

   PATH = /usr/lib/acct:/bin:/usr/bin

RUNACCT

Runacct is the main daily accounting shell procedure. It is normally initiated via **cron** during nonprime time hours. **Runacct** processes connect, fee, disk, and process accounting files. It also prepares daily and cumulative summary files for use by **prdaily** or for billing purposes. The following files produced by runacct are of particular interest:

nite/lineuse	Produced by **acctcon,** reads the wtmp file, and produces usage statistics, for each terminal line on the system. This report is especially useful for detecting bad lines. If the ratio between the number of logoffs to logins exceeds about 3/1, there is a good possibility that the line is failing.
nite/daytacct	This file is the total accounting file for the previous day in tacct.h format.
sum/tacct	This file is the accumulation of each day's nite/daytacct and can be used for billing purposes. It is restarted each month or fiscal period by the **monacct** procedure.
sum/daycms	Produced by the **acctcms** program. It contains the daily command summary. The ASCII version of this file is nite/daycms.
sum/cms	The accumulation of each day's command summaries. It is restarted by the execution of **monacct.** The ASCII version is nite/cms.
sun/loginlog	Produced by the lastlogin shell procedure. It maintains a record of the last time each login was used.
sum/rprtMMDD	Each execution of **runacct** saves a copy of the daily report that can be printed by prdaily.

ACCOUNTING

Runacct takes care not to damage files in the event of errors. A series of protection mechanisms are used that attempt to recognize an error, provide intelligent diagnostics, and terminate processing in such a way that **runacct** can be restarted with minimal intervention. It records its progress by writing descriptive messages into

the file active. (Files used by **runacct** are assumed to be in the nite directory unless otherwise noted.) All diagnostics output during the execution of **runacct** is written into fd2log. **Runacct** will complain if the files lock and lock1 exist when invoked. The lastdate file contains the month and day **runacct** was last invoked and is used to prevent more than one execution per day. If **runacct** detects an error, a message is written to /dev/console, mail is sent to root and adm, locks are removed, diagnostic files are saved, and execution is terminated.

In order to allow **runacct** to be restartable, processing is broken down into separate reentrant states. A file is used to remember the last state completed. When each state completes, statefile is updated to reflect the next state. After processing for the state is complete, statefile is read and the next state is processed. When **runacct** reaches the CLEANUP state, it removes the locks and terminates. States are executed as follows:

SETUP	The command **turnacct** switch is executed. The process accounting files, /usr/adm/pacct?, are moved to /usr/adm/Spacct?.MMDD. The /etc/wtmp file is moved to /usr/adm/acct/nite/wtmp.MMDD with the current time added on the end.
WTMPFIX	The wtmp file in the nite directory is checked for correctness by the **wtmpfix** program. Some date changes will cause **acctcon1** to fail, so **wtmpfix** attempts to adjust the time stamps in the wtmp file if a date change record appears.
CONNECT1	Connect session records are written to ctmp in the form of ctmp.h. The lineuse file is created, and the reboots file is created showing all of the boot records found in the wtmp file.
CONNECT2	Ctmp is converted to ctacct.MMDD which are connect accounting records. (Accounting records are in tacct.h format.)

PROCESS	The **acctprc1** and **acctprc2** programs are used to convert the process accounting files, /usr/adm/Spacct?.MMDD, into total accounting records in ptacct?.MMDD. The Spacct and ptacct files are correlated by number so that if **runacct** fails the unnecessary reprocessing of Spacct files will not occur. One precaution should be noted; when restarting **runacct** in this state, remove the last ptacct file because it will not be complete.
MERGE	Merge the process accounting records with the connect accounting records to form daytacct.
FEES	Merge in any ASCII tacct records from the file fee into daytacct.
DISK	On the day after the dodisk procedure runs, merge disktacct with daytacct.
MERGETACCT	Merge daytacct with sum/tacct, the cumulative total accounting file. Each day, daytacct is saved in sum/tacctMMDD, so that sum/tacct can be recreated in the event it becomes corrupted or lost.
CMS	Merge in today's command summary with the cumulative command summary file sum/cms. Produce ASCII and internal format command summary files.
USEREXIT	Any installation dependent (local) accounting programs can be included here.
CLEANUP	Clean up temporary files, run **prdaily** and save its output in sum/rprtMMDD, remove the locks, then exit.

RECOVERING FROM FAILURE

The **runacct** procedure can fail for a variety of reasons; usually due to a system crash, /usr running out of space, or a corrupted wtmp file. If the active MMDD file exists, check it first for error messages. If the active file and lock files exist, check fd2log for any mysterious messages. The following are error messages produced by **runacct** and the recommended recovery actions:

ERROR: locks found, run aborted

The files lock and lock1 were found. These files must be removed before **runacct** can restart.

ERROR: acctg already run for date : check /usr/adm/acct/nite/lastdate

The date in lastdate and today's date are the same. Remove lastdate.

ERROR: turnacct which returned rc=?

Check the integrity of **turnacct** and **accton.** The **accton** program must be owned by root and have the setuid bit set.

ERROR: Spacct?.MMDD already exists

File setups probably already run. Check status of files, then run setups manually.

ERROR: /usr/adm/acct/nite/wtmp.MMDD already exists, run setup manually

Self-explanatory.

ERROR: wtmpfix errors see /usr/adm/acct/nite/wtmperror

Wtmpfix detected a corrupted wtmp file. Use **fwtmp** to correct the corrupted file.

ERROR: connect acctg failed: check /usr/adm/acct/nite/log

The **acctcon1** program encountered a bad wtmp file. Use **fwtmp** to correct the bad file.

ERROR: Invalid state, check /usr/adm/acct/nite/active

The file statefile is probably corrupted. Check statefile and read active before restarting.

RESTARTING RUNACCT

Runacct called without arguments assumes that this is the first invocation of the day. The argument MMDD is necessary if **runacct** is being restarted and specifies the month and day for which **runacct** will rerun the accounting. The entry point for processing is based on the contents of statefile. To override statefile, include the desired state on the command line. For example:

To start runacct:

```
nohup runacct 2> /usr/adm/acct/nite/fd2log&
```

To restart runacct:

```
nohup runacct 0601 2> /usr/adm/acct/nite/fd2log&
```

To restart runacct at a specific state:

```
nohup runacct 0601 WTMPFIX 2> /usr/adm/acct/nite/fd2log&
```

FIXING CORRUPTED FILES

Unfortunately, this accounting system is not entirely foolproof. Occasionally, a file will become corrupted or lost. Some of the files can simply be ignored or restored from the file save backup. However, certain files must be fixed in order to maintain the integrity of the accounting system.

Fixing WTMP Errors

The wtmp files seem to cause the most problems in the day-to-day operation of the accounting system. When the date is changed and the system is in multiuser mode, a set of date change records is written into /etc/wtmp. The **wtmpfix** program is designed to adjust the time stamps in the wtmp records when a date change is encountered. However, some combinations of date changes and reboots will slip through **wtmpfix** and cause **acctcon1** to fail. The following steps show how to patch up a wtmp file.

```
cd /usr/adm/acct/nite
/usr/lib/acct/fwtmp < wtmp.MMDD > xwtmp
vi xwtmp
```

Edit **xwtmp** to delete any corrupted records or delete all records from the beginning up to the date change.

You then type:

```
usr/lib/acct/fwtmp -ic < xwtmp > wtmp.MMDD
```

If the wtmp file is beyond repair, create a null wtmp file. This will prevent any charging of connect time. **Acctprc1** will not be able to determine which login owned a particular process, but it will be charged to the login that is first in the password file for that user id.

Fixing TACCT Errors

If the installation is using the accounting system to charge users for system resources, the integrity of sum/tacct is quite important. Occasionally, mysterious tacct records will appear with negative numbers, dupli-

cate user IDs, or a user ID of 65,535. First check sum/tacctprev with **prtacct**. If it looks all right, the latest sum/tacct.MMDD should be patched up, then sum/tacct recreated. A simple patchup procedure would be:

```
cd /usr/adm/acct/sum
/usr/lib/acct/acctmerg -v < tacct.MMDD > xtacct
vi xtacct
```

Edit **xtacct** to remove any corrupted records and write any duplicate uid records to another file.

You then type:

```
/usr/lib/acct/acctmerg -i < xtacct > tacct.MMDD
/usr/lib/acct/acctmerg tacctprev < tacct.MMDD > tacct
```

Remember that the **monacct** procedure removes all the tacct.MMDD files; therefore, sum/tacct can be recreated by merging these files together.

UPDATING HOLIDAYS

The file /usr/lib/acct/holidays contains the prime/nonprime table for the accounting system. The table should be edited to reflect your location's holiday schedule for the year. The format is composed of three types of entries:

1. Comment Lines: Comment lines may appear anywhere in the file as long as the first character in the line is an asterisk.

2. Year Designation Line: This line should be the first data line (noncomment line) in the file and must appear only once. The line consists of three fields of four digits each (leading white space is ignored). For example, to specify the year as 1982, prime time at 9:00 a.m., and non-prime time at 4:30 p.m., the following entry would be appropriate:

   ```
   1982    0900    1630
   ```

   A special condition allowed for in the time field is that the time 2400 is automatically converted to 0000.

3. Company Holidays Lines: These entries follow the year designation line and have the following general format:

   day-of-year Month Day Description of Holiday

   The day-of-year field is a number in the range of 1 through 366 indicating the day for the corresponding holiday (leading white space is ignored). The other three fields are actually commentary and are not currently used by other programs.

DAILY REPORTS

Runacct generates five basic reports upon each invocation. They cover the areas of connect accounting, usage by person on a daily basis, command usage reported by daily and monthly totals, and a report of the last time users were logged in.

The following paragraphs describe the reports and the meanings of their tabulated data.

Daily Report

In the first part of the report, the from/to banner should alert the administrator to the period reported on. The times are the time the last accounting report was generated until the time the current accounting report was generated. It is followed by a log of system reboots, shutdowns, power fail recoveries, and any other record dumped into /etc/wtmp by the **acctwtmp** program (see **acct**(C) in the *Reference (C)*.

The second part of the report is a breakdown of line utilization. The TOTAL DURATION tells how long the system was in multiuser state (able to be accessed through the terminal lines). The columns are:

LINE The terminal line or access port.

MINUTES
The total number of minutes that line was in use during the accounting period.

PERCENT
The total number of MINUTES the line was in use divided into the TOTAL DURATION.

SESS The number of times this port was accessed for a **login**(C) session.

ON This column does not have much meaning anymore. It used to give the number of times that the port was used to log a user on; but since **login**(C) can no longer be executed explicitly to log in a new user, this column should be identical with SESS.

OFF This column reflects not just the number of times a user logged off but also any interrupts that occur on that line. Generally, interrupts occur on a port when the **getty**(M) is first invoked when the system is brought to multiuser state. Where this column does come into play is when the # OFF exceeds the # ON by a large factor. This usually indicates that the multiplexer, modem, or cable is going bad, or there is a bad connection somewhere. The most common cause of this is an unconnected cable dangling from the multiplexer.

During relay time, /etc/wtmp should be monitored as this is the file that the connect accounting is geared from. If it grows rapidly, execute **acctcon1** to see which tty line is the noisiest. If the interrupting is occurring at a furious rate, general system performance will be affected.

Daily Usage Report

This report gives a by-user breakdown of system resource utilization. Its data consists of:

UID — The user ID.

LOGIN NAME — The login name of the user; there can be more than one login name for a single user ID, this identifies which one.

CPU (MINS) — This represents the amount of time the user's process used the central processing unit. This category is broken down into PRIME and NPRIME (nonprime) utilization. The accounting system's idea of this breakdown is located in the /usr/lib/acct/holidays file. As delivered, prime time is defined to be 0900 through 1700 hours.

KCORE-MINS — This represents a cumulative measure of the amount of memory a process uses while running. The amount shown reflects kilobyte segments of memory used per minute. This measurement is also broken down into PRIME and NPRIME amounts.

CONNECT — (MINS) This identifies "Real Time" used. What this column really identifies is the amount of time that a user was logged into the system. If this time is rather high and the column "# OF PROCS" is low, this user is what is called a "line hog". That is, this person logs in first thing in the morning and does not hardly touch the terminal the rest of the day. Watch out for these kinds of users. This column is also subdivided into PRIME and NPRIME utilization.

DISK — BLOCKS When the disk accounting programs have been run, the output is merged into the total accounting record (tacct.h) and shows up in this column. This disk accounting is accomplished by the program **acctdusg.**

OF PROCS — This column reflects the number of processes that was invoked by the user. This is a good column to watch for large numbers indicating that a user may have a shell procedure that runs amok.

OF SESS — This is how many times the user logged onto the system.

DISK SAMPLES — This indicates how many times the disk accounting was run to obtain the average number of DISK BLOCKS listed earlier.

FEE — An often unused field in the total accounting record, the FEE field represents the total accumulation of widgets charged against the user by the **chargefee** shell procedure. The **chargefee** procedure is used to levy charges against a user for special services performed such as file restores, etc.

Daily Command and Monthly Total Command Summaries

These two reports are virtually the same except that the Daily Command Summary only reports on the current accounting period while the Monthly Total Command Summary tells the story for the start of the fiscal period to the current date. In other words, the monthly report reflects the data accumulated since the last invocation of monacct.

The data included in these reports gives an administrator an idea as to the heaviest used commands and, based on those commands' characteristics of system resource utilization, a hint as to what to weight more heavily when system tuning.

These reports are sorted by TOTAL KCOREMIN, which is an arbitrary yardstick but often a good one for calculating "drain" on a system.

COMMAND NAME — This is the name of the command. Unfortunately, all shell procedures are lumped together under the name sh since only object modules are reported by the process accounting system. The administrator should monitor the frequency of programs called **a.out** or **core** or any other name that does not seem quite right. Often people like to work on their favorite version of backgammor only they do not want everyone to knov about it. **Acctcom** is also a good tool to use for determining who executed a suspiciously named command and also if superuser privileges were used.

NUMBER CMDS — This is the total number of invocations of this particular command.

TOTAL KCOREMIN — The total cumulative measurement of the amount of kilobyte segments of memory used by a process per minute of run time.

TOTAL CPU-MIN — The total processing time this program has accumulated.

TOTAL REAL-MIN — The total real-time (wall-clock) minutes this program has accumulated. This total is the actual "waited for" time as opposed to kicking off a process in the background.

MEAN SIZE-K — This is the mean of the TOTAL KCOREMIN over the number of invocations reflected by NUMBER CMDS.

MEAN CPU-MIN — This is the mean derived between the NUMBER CMDS and TOTAL CPU-MIN.

HOG FACTOR — This is a relative measurement of the ratio of system availability to system utilization. It is computed by the formula:

(total CPU time) / (elapsed time

	This gives a relative measure of the total available CPU time consumed by the process during its execution.
CHARS TRNSFD	This column, which may go negative, is a total count of the number of characters pushed around by the read(S) and write(S) system calls.
BLOCKS READ	A total count of the physical block reads and writes that a process performed.

Last Login

This report simply gives the date when a particular login was last used. This could be a good source for finding likely candidates for the archives or getting rid of unused logins and login directories.

SUMMARY

The system accounting was designed from a system administrator's point of view. Every possible precaution has been taken to ensure that the system will run smoothly and without error. It is important to become familiar with the programs and shell procedures. The manual pages should be studied, and it is advisable to keep a printed copy of the shell procedures handy. The accounting system should be easy to maintain, provide valuable information for the administrator, and provide accurate breakdowns of the usage of system resources for charging purposes.

Table 5-1. Accounting System Files

File	Description
Files in the /usr/adm directory	
diskdiag	diagnostic output during the execution of disk accounting programs
dtmp	output from the **acctdusg** program
FEE	output from the **chargefee** program, ASCII tacct records
pacct	active process accounting file
pacct?	process accounting files switched via **turnacct**
Spacct?.MMDD	process accounting files for MMDD during execution of **runacct**
Files in the /usr/adm/acct/nite directory	
active	used by **runacct** to record progress and print warning and error messages. activeMMDD same as active after **runacct** detects an error
cms	ASCII total command summary used by **prdaily**
ctacct.MMDD	connect accounting records in tacct.h format
ctmp	output of **acctcon1** program, connect session records in ctmp.h format
daycms	ASCII daily command summary used by **prdaily**
daytacct	total accounting records for 1 day in tacct.h format
disktacct	disk accounting records in tacct.h format, created by dodisk procedure

Table 5-1. Accounting System Files (Cont.)

File	Description
Files in the /usr/adm/acct/nite directory (Cont.)	
fd2log	diagnostic output during execution of **runacct** (see **cron** entry)
lastdate	last day **runacct** executed in date +%m%d format
lock	lock1 used to control serial use of **runacct**
lineuse	tty line usage report used by **prdaily**
log	diagnostic output from **acctcon1**
logMMDD	same as log after **runacct** detects an error
reboots	contains beginning and ending dates from wtmp, and a listing of reboots
statefile	used to record current state during execution of **runacct**
tmpwtmp	wtmp file corrected by **wtmpfix**
wtmperror	place for **wtmpfix** error messages
wtmperrorMMDD	same as wtmperror after **runacct** detects an error
wtmp.MMDD	previous day's wtmp file

Table 5-1. Accounting System Files (Cont.)

File	Description
Files in the /usr/adm/acct/sum directory	
cms	total command summary file for current fiscal in internal summary format
cmsprev	command summary file without latest update
daycms	command summary file for yesterday in internal summary format
loginlog	created by lastlogin
pacct.MMDD	concatenated version of all pacct files for MMDD, removed after reboot by **remove** procedure
rprtMMDD	saved output of **prdaily** program
tacct	cumulative total accounting file for current fiscal
tacctprev	same as tacct without latest update
tacctMMDD	total accounting file for MMDD
wtmp.MMDD	saved copy of wtmp file for MMDD, removed after reboot by **remove** procedure
Files in the /usr/adm/acct/fiscal directory	
cms?	total command summary file for fiscal ? in internal summary format
fiscrpt?	report similar to **prdaily** for fiscal ?
tacct?	total accounting file for fiscal ?

Chapter 6
Maintaining File Systems

(BLANK)

INTRODUCTION

File system maintenance, an important task of the system administrator, keeps the system running smoothly, keeps the file systems clean, and ensures adequate space for all users. To maintain the file systems, the system administrator must monitor the free space in each file system, and take corrective action whenever it gets too low.

This chapter explains the file system maintenance commands. These commands report how much space is used, locate seldom-used files, and remove or repair damaged files.

MAINTAINING FREE SPACE

The system operates best when at least 15% of the space in each file system is free. In any system, the amount of free space depends on the size of the disk containing the file system and the number of files on the disk. Since all disks have a fixed amount of space, it is important to carefully control the number of files stored on the disk.

If a file system has less than 15% free space, system operation can become sluggish. If no free space is available, the system stops any attempts to write to the disk. This means that the user's normal work on the computer (creating new files and expanding existing ones) stops.

The only remedy for a file system which has less than 15% free space is to delete one or more files from the file system. The following sections describe strategies for keeping the free space available.

Strategies for Maintaining Free Space

The system administrator should regularly check the amount of free space of all mounted file systems and remind users to keep their directories free of unused files. You can remind users by including a reminder in the message of the day file /etc/motd. (See the section "Changing the /etc/motd File" in Chapter 10.)

If the amount of free space slips below 15%, the system administrator should:

1. Send a system-wide message asking users to remove unused files.

2. Locate exceptionally large directories and files, and send mail asking the owner to remove unnecessary files.

3. Locate and remove temporary files.

4. Clear the contents of system log files (such as /etc/wtmp).

Finally, if the system is chronically short of free space, it may be necessary to add additional or higher capacity disks to your machine.

Displaying Free Space

You can find out how much free space exists in a particular file system with the **df** (for "disk free") command. This command displays the number of "sectors" available on the specific file system. A sector is 512 characters (or bytes) of data.

The **df** command has the form:

df *specialfile*

where *specialfile* can be the name of a special file corresponding to the disk drive containing the file system. If you do not give a special filename, then the free space of all normally mounted file systems is given.

For example, to display the free space of the root file system /dev/root, type:

df /dev/root

and press the Retn key. The command displays the special filename and the number of free blocks and inodes. You may compute the percentage of free space by comparing the displayed value with the total number of blocks in the file system.

Sending a System-Wide Message

If free space is low, the super-user may send a message to all users on the system with the **wall** (for "write all") command. This command copies the messages you type at your terminal to the terminals of all users currently logged in.

To send a message, type:

/etc/wall

and press Retn. Type the message, and if necessary, press Retn to start a new line. After you have typed the message, press Retn, and then Ctrl-d. The command displays the message on all terminals in the system. To leave the **wall** command, press Ctrl-d. This removes the link to other terminals.

Displaying Disk Usage

You can display the number of blocks used within a directory by using the **du** command. This command is useful for finding excessively large directories and files.

The **du** command has the form:

du *directory*

The optional *directory* parameter must be the name of a directory in a mounted file system. If you do not give a directory name, the command displays the number of blocks in the current directory.

For example, to display the number of blocks used in the directory /usr/johnd, type:

du /usr/johnd

and press Retn. The command displays the name of each file and directory in the /usr/johnd directory and the number of blocks used.

Displaying Blocks by Owner

You can display a list of users and the number of blocks they own by using the **quot** (for "quota") command. The command has the form:

quot *specialfile*

The *specialfile* must be the name of the special file corresponding to the disk drive containing the file system.

For example, to display the owners of files in the file system mounted on the main disk drive /dev/hd0b, type:

quot /dev/hd0b

and press Retn. The command displays the users who have files in the file system and the number of blocks in these files.

Mailing a Message to a User

If a particular user has excessively large directories or files, you may send a personal message to the user with the mail command.

To begin sending a message through the mail, type:

mail *login-name*

and press Retn. The *login-name* must be the login name of the recipient. To send a message, type the message, press Retn, and then type Ctrl-d. If the message has more than one line, press Retn at the end of each line. The **mail** command copies the message to the user's mailbox, where he may view it also by using the **mail** comman See the *User's Guide* for details.

Locating Files

You may locate all files with a specified name, size, date, owner, and/or last access date by using the **find** command. The command is useful for locating seldom-used and excessively large files.

The find command has the form:

find *directory parameters*

The *directory* must be the name of the first directory to be searched. (It will also search all directories within that directory.) The *parameters* are special names and values that tell the command what to search for (see **find**(C) in the *Reference (C)* for complete details). The most useful parameters are:

-name *file*

-a time *+number*

-print

The **-name** parameter causes the command to look for the specified file. The **-atime** parameter causes the command to display only files that have not been accessed for *number* days. The **-print** parameter displays the names of files matching the parameters.

For example, to locate all files named core in the directory /usr, type:

find /usr -name core -print

and press Retn. The command displays the locations of all core files it finds.

Locating Temporary Files

You can locate temporary files with the **find** command. A temporary file contains data created as an intermediate step during execution of a program. These files may be left behind if a program contained an error or was prematurely stopped by the user. The name of a temporary file depends on the program that created it.

In most cases, the user has no use for either core or temporary files and they can be safely removed.

When searching for core or temporary files, is a good idea to search for files which have not been accessed for a reasonable period of time. For example, to find all core

files in the /usr directory which have not been accessed for seven days or longer, type:

find /usr -name core -atime +7 -print

and press Retn.

Clearing Log Files

The operating system maintains a number of files, called log files, that contain information about system usage. When new information is generated, the system automatically appends this information to the end of the corresponding file, preserving the file's previous contents. This means the size of each file grows as new information is appended. Since the log files can rapidly become quite large, it is important to periodically clear the files by deleting their contents.

You can clear a log file by typing:

cat /dev/null > *filename*

and press Retn (where *filename* is the full pathname of the log file you wish to clear). A log file normally receives information to be used by one and only one program so its name usually refers to that program. Similarly, the format of a file depends on the program that uses it. See Appendix A, "System Directories," for descriptions of the log files.

In some cases, clearing a file affects the subsequent output of the corresponding program. For example, clearing the file /usr/adm/wtmp causes a loss of accounting information.

Expanding the File System

If free space is chronically low, it may be to your advantage to expand the system's storage capacity by buying a new hard disk. Once mounted, you may use this new file system for your work, or even copy user or system directories to it.

FILE SYSTEM INTEGRITY

Since file systems are normally stored on hard disks, occasional loss of data from the file system through accidental damage to the disks is not unusual. Such damage can be caused by conditions such as an improper system shutdown, hardware errors in the disk drives, or a worn out disk.

Such damage usually affects one or two files, making them inaccessible. In very rare cases, the damage causes the entire file system to become inaccessible.

The operating system provides a way to restore and repair a file system if it has been damaged. The **fsck** (for "file system check") command checks the consistency of file systems and, if necessary, repairs them. The command does its best to restore the information required to access the files, but it cannot restore the contents of a file once they are lost. The only way to restore lost data is to use backup files. For details about backup disks, see Chapter 7, "Backing Up File Systems."

Repairing the File System

You can repair a file system with the **fsck** command. The super-user should use this command, and the system should be in single-user mode. The command has the form:

fsck *specialfile*

The *specialfile* must be the name of the special file corresponding to the disk drive containing the file system (see Appendix A, "System Directories").

NOTE

If the file system is not the root file system, you must run **umount** before running **fsck.**

For example, to check the root file system on the main hard disk (/dev/hd0b), type:

fsck /dev/hd0b

and press the Retn key. The program checks the file system and reports on its progress with the following messages.

```
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Pathnames
** Phase 3 - Connectivity
** Phase 4 - Reference Counts
** Phase 5 - Check Free List
```

If a damaged file is found during any one of these phases, the command asks if it should be repaired or salvaged. Type **y** to repair a damaged file. You should always allow the system to repair damaged files even if you have copies of the files elsewhere or intend to delete the damaged files.

Note that the **fsck** command deletes any file that it considers too damaged to be repaired. If you suspect a file system problem and wish to try to save some of the damaged file or files, check other possible remedies before you invoke the command.

Automatic File System Check

The operating system sometimes requests a check of the file system when you first start it. This usually occurs after an improper shutdown (for example, after a power loss). The **fsck** program repairs any files disrupted during the shutdown. For details, see the section "Cleaning the File System" in Chapter 2.

Chapter 7
Backing Up File Systems

(BLANK)

INTRODUCTION

A file system backup is a copy, on floppy disk or tape, of the files in the root directory or other regularly mounted file systems. A backup allows the system manager to save a copy of the file system as it was at a specific time. The copy may be used later to restore files that are accidentally lost or temporarily removed from the file system to save space.

This chapter explains how to create backups of the root directory and other file systems, and how to restore files from the backups.

STRATEGIES FOR BACKUPS

System administrators should back up the root directory (and any other mounted file systems) on a regular basis. In particular, they should make daily copies of all files modified during the day, and should make periodic (e.g., weekly) copies of the entire root directory and other mounted file systems.

The operating system offers several ways to back up file systems; two are the **archive** and **tar** commands.

The **tar** command is useful on systems with one or two users, or on any system where ordinary users wish to make personal copies of their directories. The command lets the system administrator or user choose the files and directories to be copied.

The **archive/recover** commands are a way to backup the entire disk image very quickly on streaming cartridge tape. The dump.hd and restore.hd scripts, described in this chapter, use the **archive** and **recover** commands. See **archive**(C) and **recover**(C) in the *Reference (C)* for a description of the commands.

A typical backup schedule includes a daily backup once a day and a periodic backup once a week. A daily backup saves only those files modified during that day; a periodic backup saves all files in the file system. The appro-

priate schedule for a system depends on how heavily the system is used and how often files are modified. In all cases, a periodic backup should be done at least once a month.

The system administrator should schedule backups at times when few (if any) users are on the system. This ensures that the most recent version of each file is copied correctly. You can use the **cron**(C) program to automatically back up the system to cartridge tape at any time you specify, as long as the backup will fit on a single tape or floppy disk. See **cron**(C) in the *Reference (C)*.

A regular schedule of backups requires a large number of floppy disks and adequate storage for the disks or tape cartridges. Daily backups should be saved at least two weeks; periodic backups should be saved indefinitely. Disks should be properly labeled with the date of the backup and the names of the files and directories contained in the backup. After a backup has expired, the disk may be re-used to store new backups.

USING THE tar COMMAND

The **tar** command copies specified files and directories to and from floppy disks or tape. On systems with one or two users, it gives the system administrator a direct way to make backup copies of the files modified during a day. On systems with many users, it gives ordinary users a way to make personal copies of their own files and directories.

Copying Files to a tar Disk or Tape

You can copy a small number of files or directories to a floppy disk or tape with the **tar** command. The command has the form:

tar cv *files*

The files are the names of the files or directories you want to copy.

To use the **tar** command, you need a blank cartridge tape or a formatted floppy disk and the names of the files and/or directories you wish to copy. If you give a directory name, the command copies all files in the directory (including subdirectories) to the disk or tape.

Example 1: To copy files a, b, and c to the floppy disk in the disk drive, type

tar cv a b c

and press Retn.

Example 2: To copy files a, b, and c to the tape, type

tar cvbf 1024 /dev/rct a b c

and press Retn.

Use the following modifiers with the **tar** command to do incremental backups:

B Saves all files modified after the date and time of the file you specify (instead of /etc/bkupdate).

i Saves all files modified after *date* and *time.*

I Saves all files modified after the date and time in the file /etc/bkupdate.

See the **tar** command in the *Reference (C)* for more information on how to use these modifiers.

Restoring Files from a tar Disk or Tape

You may also use the **tar** command to restore files from a floppy disk or tape. The command simply copies all files on the disk to your current directory (only if the file on the floppy does not have a full pathname). In this case, the command has the form:

tar xv

Example 3: To restore files from the floppy disk in the drive, type:

tar xv

and press Retn. This command copies files on the floppy disk in the drive to the current directory.

Example 4: To restore files from a tape, type:

tar xvbf 1024 /dev/rct

and press Retn.

Since the **tar** command copies files only to the current directory (if the file name is not a full pathname) make sure you are in the correct directory before you invoke the command. You can change to another directory with the **cd** command. More details on the use of the **tar** command can be found in the *Reference (C)*.

USING THE dump.hd/restore.hd COMMANDS

You can back up and restore from the hard disk to magnetic tape using the **dump.hd** and **restore.hd** commands. These commands save and restore an entire disk image in streaming mode, which may be the quickest way to backup a larg number of files. The **dump.hd** and **restore.hd** scripts use the **archive** and **recover** utilities described in the *Reference (C)*.

NOTE

You should be in single-user mode to use these commands because changes occurring to the file system during the backup will result in a damaged file system. It is also a good idea to run the **fsck** program before you back up the system.

To back up the entire hard disk:

1. Insert the tape.

2. Enter:

   /etc/dump.hd

   and press Retn.

To restore the contents of your hard disk from a tape made with the **dump.hd** command:

1. Boot the system from a copy of your Root File floppy disk.

2. Insert the tape.

3. Select option **c** on the menu.

CAUTION

This procedure will overwrite all existing data on the hard disk with files as they existed at the time of the dump.

This command will re-make your hard disk and restore all the files present at the time of the dump.

To back up/restore all files on an additional hard disk, see the **dump.hd**(C) and **restore.hd**(C) commands in the *Reference (C)*.

(BLANK)

Chapter 8
Using Peripheral Devices

(BLANK)

INTRODUCTION

One important task of the system administrator is to add peripheral devices such as terminals and line printers to the system.

To add a peripheral device, the system administrator must make the physical connection between the device and the computer, then use the correct system commands to enable the device for operation.

Note that all physical connections between a device and the computer are device dependent. For more information about these connections, see the *Owner's Manuals* that came with your computer and the device.

SETTING UP THE PORTS FOR TERMINALS AND PRINTERS

To work properly with a printer or terminal, the operating system needs to know certain things about that equipment.

Your system is already set up so you can connect Altos terminals and standard printers to the ports on the COM (and/or Multidrop COMM) board on your system.

If you connect your system to WorkNet II, the following ports will be disabled for use with terminals, printers, and modems:

- MULTIDROP COMM board - port04
- COM board - port09

NOTE

If you have Multidrop capabilities, the system can support up to 255 ports. However, only the maximum number of users you are licensed for can log in at one time.

In most cases, the port number and Multidrop address number are the same. See the *Owner's Guide* for addressing information.

Table 8-1 shows an example of the factory (default) settings for a system with a single 10-port SIO board.

Table 8-1. Port Configuration

Port Name	Device Type	Terminal Type	Printer Name	Printer *No.	Baud Rate	Action (On/Off)	Modem
console	terminal	altos3			9600	On	No
parallel	printer					off	No
tty01	printer		local	default	9600	off	No
tty02	terminal	altos3		2	9600	On	No
tty03	terminal	altos3		2	9600	On	No
tty04	terminal	altos3		2	9600	On	No
tty05	terminal	altos3		2	9600	On	No
tty06	terminal	altos3		2	9600	On	No
tty07	terminal	altos3		2	9600	On	No
tty08	terminal	altos3		2	9600	On	No
tty09	terminal	altos3		2	9600	off	No

* or Runstate

To access the port configuration program, log in as **root** and type:

pconfig

and you will see the following message that refers to the type of terminal you are using:

```
Your terminal type is "xxxxx".  If correct press RETN.
If not, enter the correct terminal type and press RETN:
```

where *xxxxx* is the name of the terminal you are using.

Type **Retn** if your terminal type is displayed. The Port Configuration screen appears. An example screen follows in Figure 8-1.

```
                    PORT CONFIGURATION UTILITY

Feb 23  15:20:17   1987                              Press ^W for Help

Port       Device     Terminal   Printer   Printer   Baud   Action    Modem?
Name       Type       Type       Name      *No.      Rate   (On/Off)

console    terminal   altos3                         9600   On        No
parallel   printer                                   off    No
tty01      printer               local     default   9600   off       No
tty02      terminal   altos3               2         9600   On        No
tty03      terminal   altos3               2         9600   On        No
tty04      terminal   altos3               2         9600   On        No
tty05      terminal   altos3               2         9600   On        No
tty06      terminal   altos3               2         9600   On        No
tty07      terminal   altos3               2         9600   On        No
tty08      terminal   altos3               2         9600   On        No
tty09      terminal   altos3               2         9600   off       No

To add a new port to the list.
MAIN-MENU:   Add   Change   Delete   Remote-printer   Quit

*
  or Runstate
```

Figure 8-1. The Port Configuration Screen

Use this screen to add a new port to the list, change a port's settings, or remove a port from use. You can set up a port for use with a terminal, printer, or modem.

The top line of the screen displays the current date and tells you to press Ctrl-w to get Help. The center part of the screen lists the following for the ports on your system:

- Port name (e.g., console, parallel, tty01)
- Type of device (terminal or printer)
- Type of terminal
- Printer name (or run state)
- Printer number (or run state)
- Baud rate
- Action (on or off)
- Modem connection

The bottom of the screen contains a Message line and Command line. The Command line currently contains the Main menu. From this menu, you can add, change, or delete a port, set up a remote printer, or quit the program. (Other menus also appear on the Command line.)

Note that the cursor is on the **Add** option, and the Message line explains, "To add a new port to the list." The Message line displays a description of an command. When necessary this line also displays an error or warning message.

As you use the Port Configuration program, you will be selecting commands or items from two types of menus:

- Command line menu
- Center screen menu

The following tables explain how to select a command or item from each of these menus.

Table 8-2. Command Line Menu

Action	Key to Press
Move to another command	First letter of a command
Right (move forward)	Spacebar
Left (move backward)	Back Space

Table 8-3. Center Screen Menu

Action	Key to Press
Select highlighted option (current setting or default)	Retn
Move down in a column	Down Arrow, Ctrl-d, or Spacebar
Move up in a column	Up Arrow or Ctrl-u
Move right in a row	Right Arrow or Ctrl-r
Move left in a row	Left Arrow or Ctrl-l
Go to next screen	Next Scrn or Ctrl-n
Go to previous screen	Prev Scrn or Ctrl-p

There are Help screens to guide you through this program. If you need an explanation of a particular option, press Ctrl-w, which will display help for the current screen.

Adding a Port

To add (enable) a new port,

1. Select **Add** from the Command line. The screen displays a list of valid port names.

2. Type the name of the new port (**tty16,** for example). The default settings for that port are displayed. You can press Retn to select the default settings, select a new setting, or enter information about the port. The Message line displays the following:

```
A video display unit with a screen and keyboard
DEVICE-TYPE:  Terminal  Printer  Modem  Other
```

3. Valid device types are Terminal, Printer, Modem, or Other. Select the type of device you want to add, and press Retn.

4. For terminals, the screen displays a list of all valid terminals. Select the terminal type (or type its name), and press Retn.

   For printers, see "Setting Up a Printer."

5. Next, you are asked if you want to set up a modem on the port. Select **Yes** or **No** and press Retn.

   The difference between selecting "Terminal" with a modem, and "Modem" is that serial lines for "Modem" are configured for both dialing in and dialing out (bidirectional).

6. If you select **No,** the final settings for the port appear on the screen. If you select **Yes,** the screen displays the available baud rates (speeds); the cursor highlights the MODEM300 baud rate. Select a baud rate and press Retn.

7. Then you are asked to select an action for the modem: **Active** (on) or **Inactive** (off). Select the modem action for that port and press Retn.

8. A message then asks if you want an auxiliary printer on the port. Select **Yes** or **No**.

   If you select **No**, the final settings for the port appear on the screen. If you select **Yes**, a list of printers appears on the screen.

NOTE

If you print from an auxiliary printer, during printing, the system will redirect data to the printer using the terminal's transparent print mode. All echoed output to the terminal will be disabled; however, you can stop printing and regain control of the terminal by pressing **Break/Del**.

9. Select a printer from the list and press **Retn**. You are then prompted to type a printer name and printer number. (Valid numbers are 0 to 255, and valid names are up to 14 alphanumeric characters.)

10. Next, you are asked to supply the mode flags for the auxiliary printer. These flags set the protocol for the printer (for example, odd or even parity). Your printer manual explains these flags. Press **Retn** for the default, or enter the flags for your particular printer.

11. The Port Configuration screen reappears, displaying the ports (including the changes you've just made).

    The final settings are displayed for that port. If you are finished changing the settings and want to resume installation, select **Quit** and press **Retn**.

12. You are asked for confirmation that the port assignments are correct. If they are correct, select **Yes** and press **Retn**. The system updates the port configuration information.

    If the changes are not correct, select **No**. You are asked if you want to continue in **pconfig**. If you want to make other changes or corrections, select **Yes**.

Changing a Port

To change the settings for a port,

1. Select **Change** from the Main menu Command line.

2. Select the port you wish to change by moving the cursor to that port (or typing its name) and pressing Retn. You are then asked questions about the device attached to that port.

3. The screen displays the type of device connected to that port; valid types are Terminal, Printer, Modem, or Other.

   The default settings for the port are displayed. You can press Retn to select the default settings, or enter information about the port. Select a new type of device, or press Retn to leave this setting unchanged.

   If you select Printer, see the following section, "Setting Up a Printer."

4. For terminals, the screen displays a list of the valid terminals. The cursor is on the current terminal type (if the device on this port was previously a terminal).

   Select the terminal type (or type its name) and press Retn, or press Retn to leave this setting unchanged.

5. Next, you are asked if you want to set up a modem on that port. Select Yes or No and press Retn, or press Retn to leave this setting unchanged.

The next questions ask for the following information:

- Baud rate (speed) of the modem.
- Action for the modem: Active or Inactive.
- Auxiliary printer for the port and mode flags for the printer.

To answer these questions, you can either select from a list (move the cursor to the item you want), press Retn to leave the setting unchanged, or type the required information. Steps 6 through 12 in the previous section, "Adding a Port," describe responses to these questions.

Setting Up a Printer

Your system is already set up for a serial and parallel printer (see Figure 8-1). Using the Port Configuration program, you can do the following:

- Change the existing serial printer
- Add another serial printer
- Change the printer device assigned to the parallel port
- Set up a remote printer

Serial Printer

To add or change a serial printer port,

1. Select **Add** or **Change** from the Main menu Command line.
2. Select the port you want to change.
3. Select **Printer** when you are prompted for a device type. A list of valid printer types appears on the screen.
4. Select a printer type from the list, or if you are adding a printer, enter the type (e.g., laser) and press Retn. You are then asked to supply a name for the printer.
5. Type a name for the printer and press Retn. Valid names can be up to 14 alphanumeric characters. The next prompt asks for a printer number.

6. Type a number for the printer and press **Retn**. Valid numbers are 0 through 255. The first (default) printer is printer 0, the second printer is 1, and so on.

7. Then the screen displays valid baud rates (speeds) for the printer; the cursor highlights the current baud rate. Select a rate and press **Retn**, or press **Retn** to leave this setting unchanged.

8. Next, a message asks you to enter the printer mode flags, and displays the default values for these flags. These flags set the protocol for the printer (e.g., odd or even parity). Your printer manual explains these flags. For example,

   -parity; cs8 -tabs opost

   Press **Retn** to select the displayed flags, or enter new flags as you are prompted.

Then the main Port Configuration screen reappears.

Parallel Printer

To change the printer device assigned to the parallel printer port,

1. Select **Change** from the Main menu Command line.

2. Select the parallel printer port. A list of valid printer types appears on the screen.

   NOTE

   Use a Centronics interface on the parallel port on the file processor board.

3. Select a printer type from the list, or if you are adding a printer, enter the type and press **Retn**. You are then asked to supply a name for the printer.

4. Type a name for the printer and press Retn. Valid names can be up to 14 alphanumeric characters. The next prompt asks for a printer number.

5. Type a number for the printer and press Retn. Valid numbers are 0 through 255. The first (default) printer is printer 0, the second printer is 1, and so on.

6. Next, a message asks you to enter the printer mode flags, and displays the default values for these flags. These flags set the protocol for the printer (e.g., odd or even parity). Your printer manual should explain these flags. For example,

   -parity; cs8 -tabs opost

   Press Retn to select the displayed flags, or enter new flags as you are prompted.

The Port Configuration screen reappears and displays the settings for the ports.

Remote Printer

A remote printer is one that is attached to a port on a remote machine networked to your system. To set up a remote printer,

1. Select **Remote-printer** from the Command line. You will see a list of valid remote-printer names. You are prompted for a printer name.

2. Type a name for the printer and press Retn. Then you are asked to enter a number for the printer.

3. Type a number for the printer and press Retn. Valid numbers are 0 through 255. The first (default) printer is printer 0, the second printer is 1, and so on. Then you are asked for a printer type.

4. Select or enter the printer type (e.g., laser) and press Retn. The next prompt asks for the name of the remote machine to which the printer is connected.

5. Type the name of the remote machine and press Retn. Then the screen displays valid baud rates (speeds) for the printer; the cursor highlights the current baud rate.

6. Select a baud rate and press Retn, or press Retn to leave this setting unchanged.

   Next, a message asks you to enter the printer mode flags, and displays the default values for these flags. These flags set the protocol for the printer (e.g., odd or even parity). Your printer manual should explain these flags. For example,

   -parity; cs8 -tabs opost

7. Press Retn to select the displayed flags, or enter new flags as you are prompted.

   The Port Configuration screen reappears and displays the settings for the ports.

Testing a Printer

After you set up a port for a printer, it's a good idea to test it using the **lp** program. Do this after you finish the installation procedure. For example, type:

lp *filename* and press Retn

where *filename* is the name of a file (e.g., /etc/passwd). If the printer does not print correctly, refer to your printer manual and check one or more of the following:

- The printer is connected to the correct port and is switched to "ON."
- You have the correct type of printer cable.

- The following settings for the printer are correct:
  - baud rate
  - parity setting
  - linefeed or carriage return settings
  - XON/XOFF protocol
  - word length setting

Removing a Port from Use

To remove a port from use,

1. Select **Delete** from the Main menu Command line. New commands appear on the Command line.
2. Select **Remove** from the Command line. A message asks if you're sure you want to delete the port from use.
3. To confirm the deletion, select **Yes.** If you don't want to remove the port, select **No.** At this point, you can remove another port from use.
4. To return to the Port Configuration Main menu, select **Quit.**

Leaving the Port Configuration Program

When you are finished making changes to the ports,

1. Select **Quit** from the Command line. A message asks:

```
Do you want to save the changes?
CONFIRM:  Yes  No
```

2. To save the changes you have made, select **Yes.** The ports are reconfigured and you are returned to the system prompt.

   If you decide you do not want to save the changes, select **No.** You are asked if you want to continue in **pconfig.** If you select **Yes,** the **pconfig** screen re-appears. If you select **No,** you are returned to the system prompt.

Chapter 9
Performance Management

INTRODUCTION

This chapter describes ways to monitor and enhance the performance of your Altos computer system. It is written for a system administrator who has an in-depth knowledge of the UNIX operating system. Topics discussed include:

- An introduction to performance management and how to find and fix performance problems
- Ways to improve performance: tuning the kernel and disk subsystem, reducing peak load, and estimating capacity
- Performance tools
- Tunable parameters

GENERAL APPROACH TO PERFORMANCE MANAGEMENT

Performance management is an activity that may need your attention when you first set up your Altos computer. When you bring the computer up for the first time, the system is automatically set to a basic configuration that is satisfactory for most applications. This configuration, however, cannot take into account the usage patterns and the behavior of your particular applications. For this reason, the structure of the system allows you to reconfigure it to enhance the performance for your particular application.

It is very possible that you may never have to do any special fine tuning of your system, and that your only experience with reconfiguration is when you add new memory and peripherals.

Finding Problems

The system is automatically configured the first time it is booted. After the system has been running a day or so, you may receive signals that the system needs tuning. In particular, you may see that the response time at the console is frequently slow. Or, you may need to reconfigure the system when you add a device driver. Your job of performance management begins then. To proceed, you will use the tools described later in this chapter.

Fixing Problems

At this point, you need to take some corrective action, such as modifying the tunable configuration parameters. This is usually referred to as tuning the kernel, since you are adjusting the essential control structures at the heart of the system (the kernel). Many of these parameters are described in detail later in this chapter.

When you change any of these parameters you must then reconfigure the system: recreate a new bootable version of the operating system with the new parameter definitions. This includes:

- Uninstalling optional kernel packages not needed by your applications. This procedure makes disk and memory space available for user programs and can improve performance.
- Reconfiguring the operating system after you modify either the hardware or the operating system software.
- Improving disk utilization by controlling how your file systems are organized on the disk and caching frequently used programs in memory.
- Defining best system usage patterns, such as encouraging users to run large non-interactive programs at night.

These topics are discussed in more detail in the remaining sections of the chapter.

IMPROVING PERFORMANCE

This section explains how to improve system performance by:

- Improving disk utilization
- Defining the best system usage patterns
- Investigating performance problems

Improving Disk Utilization

Disk input/output may cause a bottleneck in system performance. Steps in tuning the disk subsystem for better use include:

- Choosing the proper number of buffers
- Organizing the file systems to minimize disk activity

Choosing Buffer Size

Use the number of buffers (NBUF and NHBUF) given in the figure titled "Suggested Paramater Values" as a starter. These values come close to optimum for most system workloads.

The NBUF parameter controls the number of buffers in the system buffer cache used to reduce the need to access the disks. These buffers hold recently-used data on the chance that it will be needed again. NHBUF specifies the number of hashing buckets in the buffer cache. The more buffers, the greater chance that needed data can be found in the buffers without the system having to do a time-consuming disk read. The read and write cache hit ratios listed by the **sar -b** command indicate how effective the system buffers are. If the value for NBUF is left null, the system calculates the values shown in Figure 9-2. The value for NHBUF is a power of 2 that is roughly one-quarter the value of NBUF; it must be specified in the file **/usr/sys/master.d/kernel.**

Increasing NBUF and NHBUF, up to a point, improves system performance. A system with four megabytes of memory can typically devote roughly 500K bytes of memory to buffers (250 buffers at approximately 2K bytes each) while a system with 8 megabytes of memory can devote 1000K bytes of memory to buffers. However, if too many buffers are allocated, there may not be enough memory space for efficient operation of user processes, and the amount of swapping done by the system will increase. The swapping activity usually costs more in system efficiency than is gained by having a large amount of buffer space. If the **sar -w** command shows that your system has **swpot/s** greater than 1.0, adding buffers may not be beneficial.

After the operating system has run for a day or so, you will want to check for excessive swapping activity. If such activity is found, reduce the number of buffers (NBUF and NHBUF) or increase your system memory.

Organizing the File System

This section describes several actions that you can take to reduce the overhead of file access. As file systems are used, the blocks of individual member files tend to become physically scattered around the disk(s) and I/O becomes less efficient. This scattering yields poor ordering of blocks with files and poor directory structure.

Organization of File System Free List

The file systems are set up to allocate free blocks in a manner that allows the files to be read or written with efficiency. A free list array is created when a file system is created with **/etc/mkfs**(M). The free list is set up with the rotational gap specified by **mkfs** options. The gap is given in sectors regardless of file system block size. The difference between successive block numbers in the free list is the rotational gap. For example, a file created on a system with a rotational gap of 13 (with a 2K file system) may consist of blocks 510, 514, and 518. When the file is read, I/O requests are sent to the disk drive to read blocks 510, 514, and 518. As soon as the drive finishes reading block 510 and has started to process the second request, block 514 will be moving over

the read/write head just as the drive is ready to read that request. This method makes for efficient I/O operation.

However, as you start changing files (changing size or removing), the efficiency starts to decrease. When several files are being created at once, they will be contending for blocks from the free list. Some of the blocks allocated to the files will be out of sequence. So, the free list also becomes scattered about the disk as blocks are allocated and freed.

A table of file system rotational gaps appears in the chapter titled "Using File Systems."

Directory Organization

Directory organization also affects input/output performance. The problems show up when files are removed by users. When a file is removed from a directory, the inode number becomes null. This leaves an unused slot for that inode; over time the number of empty slots may become quite large. If you have a directory with 100 files in it and you remove the first 99 files, the directory still contains the 99 empty slots, at 16 bytes per slot, preceding the active slot. In effect, unless a directory is reorganized on the disk, it will retain the largest size it has ever achieved.

Restoring Good File System Organization

There is no automatic way to reorganize the file system, but you can manually rearrange it. There are a variety of ways to do this. Note that in all cases the file system(s) must be unmounted or for the root file system, you must be in single-user mode.

1. To reorganize the free list, run **fsck -s**.

2. To reorganize particular directory structures, use **cpio -pdm** to copy them to a temporary location; remove the original structure; then use **cpio -pdm** to copy them back to their original location.

3. To prevent file system indirection (which causes inefficient directory searching), use the following command to find directories that are larger than one (1) block.

   find / -type d -size +4 -print

   If you find directories of this size, consider breaking them up into smaller directories.

4. If you have more than one disk, balance heavily used file systems across the disks.

Defining Best System Usage Patterns

After the kernel and the system activities are tuned and the file systems organized, it's time to perform some housekeeping activities to reduce prime-time load. You should:

- Give important jobs priority
- Reduce unnecessary activities
- Schedule selected jobs when the system is not so busy
- Make user-defined features, such as **.profile** and $PATH, more efficient

ps

Use the **ps**(C) command to obtain information about active processes. The command gives a picture of what is going on, which is useful when you are trying to identify what processes are loading the system. Look at the TIME (minutes and seconds of CPU time used by processes) and STIME (time when process first started) entries.

When you spot a "runaway" process (one that uses progressively more system resources over a period of time), check with the owner. It is possible that such a process should

be stopped immediately via the **kill -9** command. When you have a real runaway, it continues to eat up system resources until everything grinds to a halt.

When you spot processes that take a very long time to execute, you should consider using **cron**(C) to execute the job during off-hours.

User $PATH Variables

$PATH is searched upon each command execution. Before outputting "not found," the system must search every directory in $PATH. These searches require both processor and disk time. If there is a disk or processor bottleneck, changes here can help performance.

You should check user $PATH variables for:

- Path efficiency. $PATH is read left to right, so the most likely places to find the command should be first in the path (**/bin** and **/usr/bin**). Make sure that a directory is not searched more than once for a command.
- Convenience. Users may prefer to have the current directory listed first in the path (**:/bin**).
- Path length. In general, $PATH should have the least number of required entries.
- Large directory searches. Searches of large directories should be avoided if possible. Put any large directories at the end of $PATH.

SAMPLES OF GENERAL PROCEDURES

This section depicts typical approaches to performance management. First, it describes a general procedure for troubleshooting performance problems. Then, it shows a sample of a typical system reconfiguration.

Investigating Performance Problems

One of the most common symptoms that a problem exists is poor response time. Figure 9-1, which follows, shows a typical troubleshooting procedure. After you identify a problem area, see "Reconfiguring the System" later in this chapter to make the necessary system parameter changes.

Check for Excess Swapping

First, look at swapping activity, since the swapping of pages is costly in both disk and CPU overhead. Generate the **sar -qw** report. Look at the percentage that the swap queue is occupied (%swpocc) for values greater than 5. Then look at the swap-out rate (swpot/s) for values greater than 1.00.

Check whether the "freemem" count (number of pages available to user programs), shown by **sar -r**, is consistently less than the value of the tunable parameter GPGSHI (high-water mark).

If any or all of these are happening frequently, increase your system memory or try to reduce memory allocated for system buffers. If a large number of buffers are configured and the buffer cache hit ratios (**sar -b**) are 90% or more, try decreasing the number of buffers (stored in the tunable parameter NBUF). It's possible that returning memory space occupied by some buffers will solve the swap ping problem (leaves more space for user programs). Additional memory for user programs can also be obtained by uninstalling optional kernel utilities that are not needed by your applications.

Check for Disk Bottleneck

If the value of %wio (from the **sar -u** report) is greater than 10%, or if the %busy for a disk drive (obtained by **sar -d**) is greater than 50%, then the system has a disk bottleneck. Ways to alleviate a disk bottleneck include:

1. Increase the number of buffers.

2. Organize the file system to minimize disk activity. If you have two disks, distribute the file systems evenly.

3. Consider adding more memory if the situation persists. Additional memory reduces swapping/paging traffic and allows an expanded buffer pool (reducing the number of user-level reads and writes that need to go out to disk).

4. Consider adding an additional disk and balancing the most active file systems across the two disks.

Check for Potential Table Overflows

To check for potential table overflows, get the **sar -v** report, which lets you know if overflows have occurred in the process, file, or inode tables. Avoid overflows in these tables by increasing NPROC, NFILE, and NINODE/NS5INODE (in the **/usr/sys/master.d/kernel** file).

Shift Workload to Off-Peak Hours

Examine **/usr/spool/cron/crontabs** to see if jobs are queued for peak periods that might better be run at times when the system is idle. Use the **ps** command to determine what processes are heavily loading the system. Encourage users to run large, non-interactive commands (such as **nroff** or **troff**) at off-peak hours. You may also want to run such commands with a low priority by using the **nice**(C) or **batch**(C) commands.

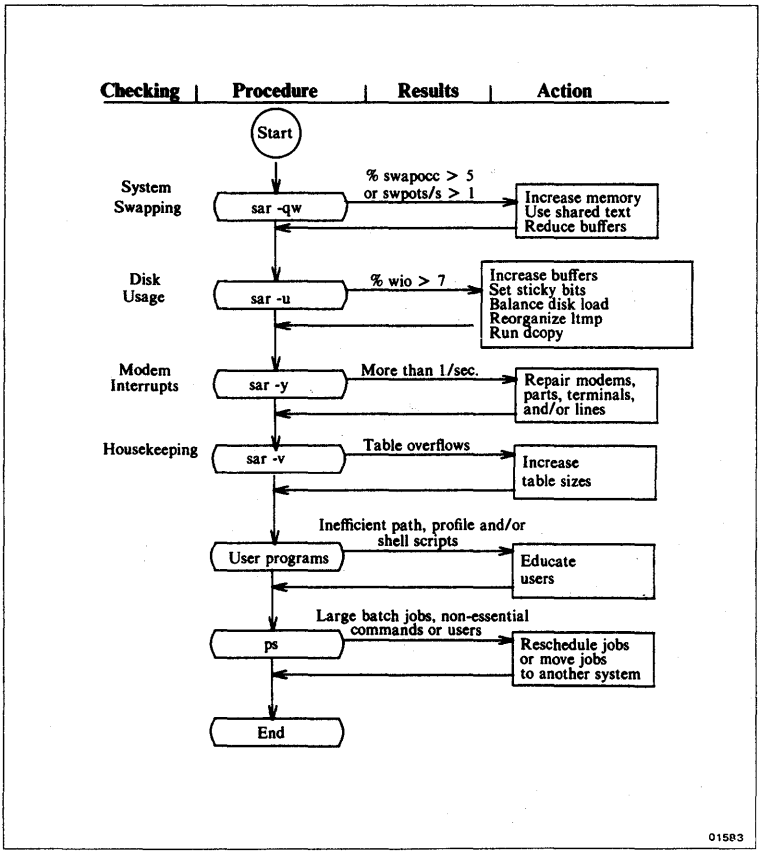


Figure 9-1. Outline of Typical Troubleshooting Procedure

PERFORMANCE TOOLS

Internal activity is measured by a number of counters contained in the operating system kernel. Each time an operation is performed, an associated counter is incremented. Sar(C) allows you to monitor the values of these counters.

To start up the data collection process, see sar(M).

The functions monitored by **sar** are discussed in the following sections.

Sar allows you to monitor the values of these counters by sampling cumulative activity counters and providing reports on various system-wide activities.

The following examples are from a system with 4 megabytes of main memory and about 80MB of hard disk. Command outputs are typical values for user workloads on the system. Values you receive may be quite different from values in the examples, depending on your application or benchmark. When tuning your system, it's best to use a benchmark or have the system under normal load for your application so you can tune for your specific application.

sar

Throughout this section, **sar**(C) options are described with an analysis of sample outputs of the options. **Sar** can be used either to gather system activity data or to extract what has been collected in data files created by **sa1** and **sa2**. **Sa1** and **sa2** are initiated by entries put in the **crontab.sys** file.

sar -a

The **-a** option reports the use of file access operations. The operating system routines reported include:

iget/s Number of files located by inode entry per second.

namei/s Number of file system path searches per second. **Namei** calls **iget**, so **iget/s** is always larger than **namei/s.**

dirbk/s Number of directory block reads issued per second.

Sample **sar -a** output, with a 30-second sampling interval, follows:

```
unix pubs 5.3.1 d 386/2000 11/5/87
12:41:40 iget/s namei/s dirbk/s
12:42:10       4       1       3
12:42:40       2       1       3
12:43:10       5       2       3
Average        4       1       3
```

The larger the values reported, the more time the kernel is spending to access user files. This indicates how heavily programs and applications are using the file system(s). The **-a** option is helpful for understanding how disk-dependent the application system is; it is not used for any specific tuning step.

sar -b

The **-b** option reports the following buffer activity:

bread/s Average number of physical blocks read into the system buffers from the disk (or other block devices) per second.

lread/s Average number of logical blocks read from system buffers per second.

%rcache Fraction of logical reads found in buffer cache (100% minus the ratio of **breads** to **lreads**).

bwrit/s Average number of physical blocks written from the system buffers to disk (or other block devices) per second.

lwrit/s Average number of logical blocks written to system buffers per second.

%wcache Fraction of logical writes found in buffer cache (100% minus the ratio of **bwrit/s** to **lwrit/s**).

pread/s Average number of physical read requests per second.

pwrit/s Average number of physical write requests per second.

The entries that you should be most interested in are the cache hit ratios **%rcache** and **%wcache**, which measure the effectiveness of system buffering. If **%rcache** falls below 90, or **%wcache** falls below 65, it may be possible to improve performance by increasing the number of buffers.

An example of **sar -b** output follows:

```
(to be reduced and inserted by Art)
                    " "
                    " "
                    " "
                    " "
                    " "
```

This example shows that the buffers are not causing any bottlenecks, because all data is within acceptable limits.

sar -c

The **-c** option reports system calls in the following categories:

scall/s All types of system calls per second, generally about 30/second on a busy 4-to-6 user system.

sread/s Read system calls per second.

swrit/s Write system calls per second.

fork/s Fork system calls per second, about 0.5/second on a 4-to-6 user system. This number will increase if shell scripts are running.

exec/s Exec system calls per second. (If **(exec/s)/(fork/s)** is greater than 3, look for inefficient $PATHs.)

rchar/s Characters (bytes) transferred by read system calls per second.

wchar/s Characters (bytes) transferred by write system calls per second.

Typically, reads plus writes account for about half of the total system calls, although this varies greatly with the activities that are being performed by the system.

An example of **sar -c** output follows:

```
unix pubs 5.3.1 d 386/2000 11/5/87
18:33:04 scall/s sread/s swrit/s fork/s exec/s rchar/s wchar/s
18:33:35      38      16       6   0.03   0.03    6089    1638
18:34:05      39      16       4   0.07   0.07    6123    1602
18:34:35      38      17       5   0.17   0.17    6042    1704
Average       38      16       5   0.09   0.09    6085    1648
```

A parallel option is available for systems with Remote File Sharing= installed. See the description of **sar -Dc** in the RFS manual.

sar -d

The **-d** option reports the activity of block devices.

device Name of the block device(s) that **sar** is monitoring. An "hd" label indicates a hard disk, and "fd" means a floppy disk. SCSI devices are identified with a "SCSI" label.

%busy Percent of time the device was servicing a transfer request.

avque The average number of requests outstanding during the period of time (measured only when the queue is occupied).

r+w/s Number of read and write transfers to the device per second.

blks/s Number of 512-byte blocks transferred to the device per second.

avwait Average time in milliseconds that transfer requests wait idly in the queue (measured only when the queue is occupied).

avserv Average time in milliseconds for a transfer request to be completed by the device (for

disks this includes seek, rotational latency, and data transfer times).

An example of **sar -d** follows:

```
unix pubs 5.3.1 d 386/2000 11/5/87
13:46:28  device %busy avque r+w/s blks/s  avwait  avserv
13:46:58  hd-0       6   1.6     3      5   13.8    23.7
          fd-0      93   2.1     2      4  467.8   444.0
13:47:28  hd-0      13   1.3     4      8   10.8    32.3
          fd-0     100   3.1     2      5  857.4   404.1
13:47:58  hd-0      17    .7     2     41     .6    48.1
          fd-0     100   4.4     2      6 1451.9   406.5
Average   hd-0      12   1.2     3     18    8.4    34.7
          fd-0      98   3.2     2      5  925.7   418.2
```

The above example was taken while transferring data from hard disk (hd-0) to floppy disk (fd-0).

Queue lengths and wait times are measured while the queue had something on it. If **busy** is small, large queues and service times probably represent the periodic **sync** efforts by the system to ensure that altered blocks are written to the disk in a timely way.

sar -m

The **-m** option reports on inter-process communication activities. Message and semaphore calls are reported as follows:

msg/s Number of message operations (sends and receives) per second.

sema/s Number of semaphore operations per second.

An example of **sar -m** output follows:

```
unix pubs 5.3.1 d 386/2000 11/5/87
15:16:58   msg/s  sema/s
15:17:32    0.00    0.00
15:18:02    0.00    0.00
15:18:32    0.00    0.00
Average     0.00    0.00
```

These figures will usually be zero (0.00) unless you are running applications that use the message or semaphore features.

sar -q

The **-q** option reports the average queue length while the queue is occupied and percent of time occupied.

runq-sz Run queue of processes in memory; typically, this should be less than 2. Consistently higher values mean you are CPU-bound.

%runocc The percentage of time the run queue is occupied; the larger this value is the better.

swpq-sz Swap queue of processes to be swapped out; the smaller this number is the better.

%swpocc The percentage of time the swap queue is occupied; the smaller this value is the better.

An example of **sar -q** follows:

```
unix pubs 5.3.1 d 386/2000 11/5/87
11:00:56 runq-sz %runocc swpq-sz %swpocc
11:01:07   1.7      98      1.5      36
11:01:17   1.0      63      1.0      31
11:01:27   1.0      58      1.0      49
Average    1.3      74      1.2      39
```

In this example, the processor utilization (**%runocc**) varies between 58% and 98%, while the fraction of time the swap queue is not empty (**%swpocc**) is 31% to 49%. This means that memory is not causing a major bottleneck in the system throughput, but more memory would help reduce the swapping/ paging activity.

If **%runocc** is greater than 90 and **runq-sz** is greater than 2, the CPU is heavily loaded and response is degraded. In this case, additional CPU capacity may be required to obtain acceptable system response. If **%swpocc** is greater than 20, more memory or fewer buffers would help reduce swapping/paging activity.

sar -u

The CPU utilization is listed by **sar -u** (default). At any given moment the processor will be either busy or idle. When busy, the processor will be in either user or system mode. When idle, the processor will either be waiting for input/output completion or idle. The **-u** option of **sar** lists the percent of time that the processor is in system mode (%sys), user mode (%user), waiting for input/output completion (%wio), and idle time (%idle).

In typical timesharing use, %sys and %usr are about the same value. In special applications, either of these may be larger than the other without anything being abnormal. A high %wio generally means a disk bottleneck. A high %idle, with degraded response time, may mean memory constraints; time spent waiting for memory is attributed to %idle.

An example of **sar -u** follows:

```
unix pubs 5.3.1 d 386/2000 11/5/87
09:20:05          %usr %sys    %wio %idle
09:40:12             6    7       2   86
10:00:03             7    9       3   80
10:20:07            14   16      10   61
Average              9   11       5   76
```

A parallel option is available for systems with Remote File Sharing installed. See the description of **sar -Du** in the RFS manual.

sar -v

The -v option reports the status of process, inode, file, shared memory record, and shared memory file tables. From this report you know when the system tables need to be modified.

proc-sz Number of process table entries presently being used/allocated in the kernel.

inod-sz Number of inode table entries presently being used/allocated in the kernel.

file-sz Number of file table entries presently being used/allocated in the kernel.

ov Number of times an overflow occurred. (One column for each of the above three items.)

lock-sz The number of shared memory record table entries presently being used/allocated in the kernel.

fhdr-sz No longer applicable.

The values are given as *level/table size.* An example of **sar -v** follows:

```
unix pubs 5.3.1 d 386/2000 11/5/87
17:36:05 proc-sz ov inod-sz ov file-sz ov lock-sz fhdr-sz
17:36:35 17/ 40  0   39/ 80  0   29/ 80  0 0/ 50   0/  0
17:37:05 19/ 40  0   46/ 80  0   35/ 80  0 0/ 50   0/  0
17:37:35 18/ 40  0   43/ 80  0   34/ 80  0 0/ 50   0/  0
```

This example shows that all tables are large enough to have no overflows. Sizes could be reduced to save main memory space if these are the highest values ever recorded.

sar -w

The -w option reports swapping and switching activity. The following are some target values and observations.

swpin/s Number of transfers into memory per second.

bswin/s Number of 512-byte-block units (blocks) transferred for swap-ins (including initial loading of some programs) per second.

swpot/s Number of transfers from memory to the disk swap area per second. If greater than 1, memory may need to be increased or buffers decreased.

bswot/s Number of blocks transferred for swap-outs per second.

pswch/s Process switches per second. This should be 30 to 50 on a busy 4-to-6 user system.

An example of **sar -w** output follows:

```
unix pubs 5.3.1 d 386/2000 11/5/87
19:53:44 swpin/s bswin/s swpot/s bswot/s pswch/s
19:53:58   0.0      0.0     0.0      0.0      37
19:54:14   0.0      0.0     0.0      0.0      39
19:54:24   0.0      0.0     0.0      0.0      39
Average    0.0      0.0     0.0      0.0      38
```

This example shows that there is sufficient memory for the currently active users, since no swapping is occurring.

sar -p

The **-p** option reports paging activity. The following page rates are recorded.

vflt/s Number of address translation page faults per second (valid page not present in memory).

pflt/s Number of page faults from protection errors per second (illegal access to page) or "copy-on-writes." **Pflt/s** generally consists entirely of "copy-on-writes."

pgfil/s Number of vflt/s per second satisfied by a page-in from the file system. (Each **pgfil** causes two **lreads**; see **sar -b**.)

rclm/s Number of valid pages per second that the system has reclaimed (added to list of free pages).

An example of **sar -p** output follows:

```
unix pubs 5.3.1 d 386/2000 11/5/87
12:01:51  vflt/s  pflt/s  pgfil/s  rclm/s
12:56:52  13.91    2.80     5.63    11.21
```

sar -r

The **-r** option records the number of memory pages and swap file disk blocks that are currently unused. The following are recorded:

freemem Average number of 4K-pages of memory available to user processes over the intervals sampled by the command.

freeswap Number of 512-byte disk blocks available for process swapping.

An example of **sar -r** output follows:

```
unix pubs 5.3.1 d 386/2000 11/5/87
12:01:51 freemem freeswap
12:56:52  208    5848
```

sar -y

The **-y** option monitors terminal device activities. If you have a lot of terminal I/O, you can use this report to determine if there are any bad lines. Activities recorded are defined as follows:

rawch/s Input characters (raw queue) per second.

canch/s Input characters processed by canon (canonical queue) per second.

outch/s Output characters (output queue) per second.

rcvin/s Receiver hardware interrupts per second.

xmtin/s Transmitter hardware interrupts per second.

mdmin/s Modem interrupts per second.

The number of modem interrupts per second (**mdmin/s**) shoul be close to 0, and the receive and transmit interrupts per second (**xmtin/s** and **rcvin/s**) should be less than or equal to the number of incoming or outgoing characters, respectively. If this is not the case, check for bad lines.

An example of **sar -y** output follows:

```
unix pubs 5.3.2 d 386/2000 11/5/87
16:50:11 rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
16:50:41    112      15     653     103     102       0
16:51:11    107       7     654     104     105       0
16:51:41     99       5     641      99     105       0
Average     106       9     649     102     104       0
```

sar -A

The **sar -A** option is equivalent to **sar -udDqbwcayvmprS.** The report includes RFS operations (the **-D** and **-S** options) described in the RFS manual. The **-A** option provides a view of overall system performance. Use it to get a more global perspective. If data from more than one time slice is shown, the report includes averages.

An example of **sar -A** follows:

(To be reduced and inserted by Art)

" "
" "
" "
" "
" "
" "
" "
" "
" "

RECONFIGURING THE OPERATING SYSTEM

This section explains how to rebuild the operating system after you modify either the hardware or operating system software (e.g., added a software driver) and tune it. Also, you'll learn how to recover if you create an unbootable operating system after attempting a system reconfiguration.

CAUTION

Do not change nodename during reconfiguration without coordinating it with interfacing systems if you are using any type of networking, including uucp.

System reconfiguration is necessary whenever the physical configuration of the computer or the software configuration of the operating system itself changes. This happens when you upgrade your system with certain hardware, add software drivers, or edit tunable parameters to improve performance. The only way the system can understand such changes is by rebuilding it from its (partly modified) source files. This procedure allows you to modify and reconfigure (rebuild) the system residing in **/unix**.

Preliminary Steps

When reconfiguring the operating system, do not arbitrarily change the node name (NODE) of the computer. Once basic networking has been established, a change in node name must be coordinated with all interfacing systems. If not properly coordinated, a calling system will fail sequence checks because the returned name does not match the name stored in **/usr/lib/uucp/Systems**. To just set or change a node name without reconfiguring the system, see the -N option of **uname**(C).

There are two major steps in reconfiguring the operating system:

1. Edit the **/usr/sys/master.d** files to modify the tunable parameters.

2. Rebuild the operating system.

Modifying the Tunable Parameters

The major steps in incorporating changes to the tunable parameters are as follows.

1. Log in as **root**.

2. Change the present working directory to **/usr/sys/master.d**:

```
# cd /usr/sys/master.d
```

3. Edit the applicable files in the **/usr/sys/master.d** directory to modify (increasing or decreasing the value of) the tunable parameters. Refer to the section titled "Tunable Parameters" for a listing of recommended values for the tunable parameters available with your system.

Modifying the Tunable Configuration Parameters

To set the system tunable parameters, edit the parameter entries in the **/usr/sys/master.d/kernel** file, or other files in **/usr/sys/master.d**, such as **shm, msg, sem,** or **disp.** For full definitions of the individual tunable parameters, and suggestions about setting them, refer to the section "Tunable Parameters" in this chapter. Use the **/etc/sysdef** command to see what the current values of the tunable parameters are in the present configuration of your system.

Generally, the default parameters for your configuration will result in acceptable performance. If, however, you are running an application that has special performance needs, you can use the tools described in the following section "Performance Tools" to measure system load and determine which parameters might be changed to improve performance. Two key parameters, NBUF (system buffers) and NHBUF (hash buffers), have a strong impact on the efficiency of disk utilization. They are discussed in the section "Improving Disk Utilization."

See the previous section for examples of editing the tunables and reconfiguring the system. Also, refer to the example of a typical reconfiguration that follows.

Sample System Reconfiguration

The following example shows how to change the buffer cach size from the default calculated value (dependent on memory size) to 3 MB. (See Figure 9-2 titled "Suggested Parameter Values" for parameter values recommended for your system.) Most of them are in the file **/usr/sys/master.d/kernel.**

The command line entries and system responses in the following illustration show the reconfiguration and rebooting of the operating system to support these new parameters.

The words that are **bold** are the input typed by the user. Comments are in parentheses and are not indented.

```
# cd /usr/sys
# cd master.d
# cp kernel kernel.old

(Save the current version in case of errors)
# vi kernel
/NBUF=
(Find the buffer paramter)
(Edit the line to read: NBUF=1500)
/NHBUF=
(Find the hash bucket entry)
(Edit the line to read: NHBUF=512)
/MAXBUF=
(Find the MAXBUF paramter)
(Edit the line to read: MAXBUF=1500)

:w
(Write file to kernel)
:q
(Quit vi)

# cd ..

# make
(Make the new unix in the /usr/sys directory)
```

Rebuilding the Operating System

After modifying the tunable parameters, rebuild the operating system as follows (note you must be logged in as **root**):

1. Change the present working directory to **/usr/sys**.

   ```
   # cd /usr/sys
   ```

2. Execute the **make** command to create a bootable object file for each of the files modified in **/usr/sys/master.d**. For example, if some of the tunable parameters in the **/usr/sys/master.d/kernel** were modified, you would see:

```
mkboot -m master.d  -d boot.d  -k kernel
```

If the tunables that you changed were in another file (for example, **/usr/sys/master.d/sem**), you would see:

```
/etc/mkboot -m master.d  -d boot.d boot.d/SEM
```

3. Next, you would see:

```
ldunix -b boot.d -s system
```

which loads the kernel into the temporary configuration files kimage and ksymbols.

4. Finally, you would see:

```
mv /unix /unix.old
mkunix -i KERNEL -o unix
rm kimage ksymbols
cp unix /unix
```

which creates a new bootable unix kernel.

5. Reboot the operating system so the new kernel configuration takes effect.

Recovering an Unbootable Operating System

The following procedure explains how to recover from an unbootable **/unix**, and describes how to get a viable version of the system running after an unsuccessful attempt at reconfiguring the system.

1. If the system does not boot successfully, reboot it.

2. If the system fails to come up, retry. When you see the words "Press any key to interrupt boot," press any key.

3. Next, the menu enabling you to boot from the hard or floppy disk is displayed. Press 1 to boot from the hard disk.

4. Press **Del** as soon as you see the message, "Loading kernel..."

5. Press Esc, and you should see a prompt like the following:

```
: hd(0,2)unix
```

6. Type **.old**, so that you now see:

```
: hd(0,2)unix.old
```

7. Press Retn to boot your old unix. The system boots the **/unix.old** that you made before reconfiguring the system.

8. After the system has rebooted and you have logged in again, move **/unix.old** back to **/unix:**

```
mv /unix.old /unix
```

This action protects you from having to repeat this procedure again in case of a sudden system crash.

NOTE

If you did not make **/unix.old**, reload the operating system: boot from the Root floppy diskette, mount the hard disk, and copy **/unix** from the floppy diskette to **/hd/unix** (see the Installation manual for your operating system).

9. Finally, try to determine what went wrong. Consider first that you may have placed an incorrect value in the parameter(s) you were working with.

10. Make the necessary corrections and rebuild the system using the previous steps. Try restoring your backed-up version of the file you modified. (See the previous section, "Modifying the Tunable Parameters," Step 3 for directions.) If you cannot determine what

went wrong, document what happened as thoroughly as you can, and then contact your service representative.

TUNABLE PARAMETERS

Use tunable system parameters to set various table sizes and system thresholds to handle the expected system load. Use caution when changing these variables since such changes can directly affect system performance. For the most part, the initial tunable parameter values for a new Altos computer are acceptable for most configurations and applications. If your application has special performance needs, you may have to experiment with different combinations of parameter values to find an optimal set.

Figure 9-2 that follows shows the recommended tunable parameter values for a system equipped with different sizes of Random Access Memory (RAM). Values shown as CALCULATED are calculated by the system at boot-time, an are represented in the actual table by a zero (0). These values are based on system configuration: RAM size, number of ports, and so forth. The parameters shown in the figure are defined in the following files:

- **/usr/sys/master.d/kernel**
- **/usr/sys/master.d/disp**
- **/usr/sys/master.d/msg**
- **/usr/sys/master.d/sem**
- **/usr/sys/master.d/shm**

The default parameter settings defined in the **/usr/sys/master.d/kernel** and **/usr/sys/master.d/disp** files are delivered with the Run-time system.

The following notes apply to Figure 9-2:

- The value of a few parameters are calculated each time a new kernel (**/unix**) is generated, unless the

value is manually overridden (see the following note).

- All the other parameters are set to specific values, as defined in the appropriate **/usr/sys/master.d** file. The default value and the size in bytes for each entry are shown in the following figure. These values are valid for most Altos Systems. Your specific system may have different default values, which will be documented in the Release Notes accompanying your operating system.

- A dash (-) is used in the size information to indicate parameters that set flags in the kernel. These parameters do not affect the size of the kernel when their values are changed; only the values of the specific flags are changed.

NOTE

A calculated parameter value is overridden by adding the value to the appropriate **/usr/sys/master.d** file. Overriding the calculated value causes the parameter to be set to the new value each time a new kernel (**/unix**) is generated. When calculated parameter values are overridden, however, any subsequent changes in the hardware configuration (adding new memory for example) require a change of the value that was manually set. This will allow you to optimize the performance of the new configuration.

Parameter	Equipped RAM					Default	Size per Entry
	2 Mb	4 Mb	8 Mb	8 Mb	16 Mb	Value	in Bytes
SYSTEM	500/ 1000	ALL	500/ 1000	2000	2000		
NBUF	120	250	500	500	1000	CALCULATED	2112
MAXBUF	1024	1024	1024	1024	1024	1024	-
NCALL	60	60	60	60	60	60	16
NINODE	200	400	500	500	1000	CALCULATED	68
NS5INODE	200	400	500	500	1000	CALCULATED	64
NFILE	250	500	800	1000	2000	CALCULATED	12
NMOUNT	25	25	25	25	25	25	36
NPROC	100	180	200	200	400	CALCULATED	140
NREGION	400	720	800	800	1600	CALCULATED	48
NCLIST						CALCULATED	128
MAXUP	30	30	30	30	30	30	-
NOFILES	50	50	50	50	50	50	-
NHBUF	64	128	128	128	256	128	12
NPBUF	8	8	8	8	8	8	64
BDFLUSHR	1	1	1	1	1	1	-
MAXPMEM	0	0	0	0	0	CALCULATED	-
SHLBMAX	2	2	2	2	2	2	-
NFLOCKS	200	500	800	1000	2000	CALCULATED	14
PUTBUFSZ	2000	2000	2000	2000	2000	2000	1
NSRMOUNT	10	10	10	10	10	10	28
MAXSLICE	50	50	50	50	50	50	

Figure 9-2. Suggested Parameter Values (1 of 3)

Parameter	Equipped RAM 2 Mb	4 Mb	8 Mb	16 Mb	Default Value		*Size
VHNDFRAC	16	16	16	16	16		-
VHANDR	1	1	1	1	1		-
GPGSLO	25	25	25	50	25		-
GPGSHI	40	40	40	80	40		-
MAXUMEM	2560	2560	2560	4096	2560		-
GPGSMSK	0x00000420	0x00000420	0x00000420	0x00000420	0x00000420		-
MAXFC	1	1	1	1	1		-
MAXSC	1	1	1	1	1		-
MINARMEM	25	40	40	40	25		-
MINASMEM	25	40	40	40	25		-
NQUEUE	256	384	512	512	20		36
NSTREAM	32	100	100	100	100		52
NMUXLINK	32	48	96	96	87		12
NSTREVENT	256	256	256	256	256		12
MAXSEPGCNT	1	2	4	4	1		2048
					w/o RFS	w/RFS	
NBLK4096	0	0	0	0	1	4	4142
NBLK2048	20	40	40	40	1	32	2094
NBLK1024	12	32	32	32	2	96	1070
NBLK512	8	18	18	18	2	16	558
NBLK256	16	48	48	48	0	16	302
NBLK128	64	128	128	128	0	96	174
NBLK64	40	256	256	256	0	40	110
NBLK16	40	256	256	256	0	40	61
NBLK4	40	128	128	128	0	40	50
NSTRPUSH	9	9	9	9	9		-
STRLOFRAC	80	80	80	80	80		-
STRMEDFRAC	90	90	90	90	90		-
STRMSGSZ	4096	4096	4096	4096	4096		-
STRCTLSZ	1024	1024	1024	1024	1024		-
NLOG	3	3	3	3	3		12
BSIZE	5	5	5	5	5		-

* per entry in bytes

Figure 9-2. Suggested Parameter Values (2 of 3)

Parameter	Equipped RAM 2 Mb	4 Mb	8 Mb	16 Mb	Default Value	Size per Entry in Bytes
MSGMAP	100	100	100	100	100	8
MSGMAX	2048	2048	2048	2048	2048	-
MSGMNB	4096	4096	4096	4096	4096	-
MSGMNI	50	50	100	100	50	48
MSGSSZ	8	8	8	8	8	1024
MSGTQL	40	40	40	40	40	12
MSGSEG	1024	1024	1024	1024	1024	8
SEMMAP	10	10	10	10	10	8
SEMMNI	10	10	10	10	10	32
SEMMNS	60	60	8	8	60	8
SEMMNU	30	30	30	30	30	8*(SEMUME+2)
SEMMSL	25	25	25	25	25	-
SEMOPM	10	10	8	8	10	8
SEMUME	10	10	10	10	10	8*SEMMNU
SEMVMX	32767	32767	32767	32767	32767	-
SEMAEM	16384	16384	16384	16384	16384	-
SHMMAX	524288	524288	524288	524288	524288	-
SHMMIN	1	1	1	1	1	-
SHMMNI	100	100	100	100	100	52
SHMSEG	6	6	6	6	6	12*NPROC
SHMALL	512	512	512	512	512	-

Figure 9-2. Suggested Parameter Values (3 of 3)

Kernel Parameters

The following parameters are defined in the file **/usr/sys/master.d/kernel.**

NBUF Specifies how many system buffers to allocate. The system buffers form a data cache: a memory array containing disk file information. Improvement in the hit rate of this cache increases with the number of buffers. Cache hits reduce the number of disk accesses and thus improve overall performance.

The entries are normally in the range of 100 to 600. NBUF=100 for systems less than 2 Mb. Otherwise, NBUF=MINBUF + 1/8 of memory over 512K but less than MAXBUF (default setting for MAXBUF is 1024). Each buffer contains 64 bytes of header information plus a data area equal to 1 disk block. Disk blocks are 2K. Hash buffers (NHBUF) should be increased along with system buffers (NBUF) for optimal performance. The value for NBUF is calculated automatically by the system, unless specifically set to a non-zero in the **/usr/sys/master.d/kernel** file.

MAXBUF See above.

NCALL Specifies how many call-out table entries to allocate. Each entry represents a function to be invoked at a later time by the clock handler portion of the kernel. This value must be greater than 2 and is normally in the range of 10 to 70. The default value is 60. Each entry contains 16 bytes.

Software drivers may use call-out entries to check hardware device status. When the call-out table overflows, the system outputs the following message on the system console:

```
WARNING:  Timeout table overflow
```

NINODE Specifies how many inode table entries to allocate. Each table entry represents an in-core inode that is an active file. For example, an active file might be a current directory, an open file, or a mount point. The file control structure is modified when changing this variable. The number of entries used depends on the number of opened files. This value is calculated by the system unless specifically set to a non-zero value. The entries are normally in the range of 100 to 1000. NINODE must always be less than or equal to NS5INODE. When the inode table overflows, the following warning message is output on the system console:

```
WARNING: inode table overflow
```

NS5INODE
: NS5INODE must always be equal to or greater than NINODE.

NFILE
: Specifies how many open file table entries to allocate. Each entry represents an open file. The entry is normally in the range of 200 to 2000. This value is calculated by the system unless specifically set to a non-zero value. Each entry contains 12 bytes. The NFILE control structure operates in the same manner as the NINODE structure. When the file table overflows, the following warning message is output on the system console.

```
NOTICE: file table overflow
```

As a reminder, this parameter does not affect the number of open files per process (see the NOFILES parameter).

NMOUNT
: Specifies how many mount table entries to allocate. Each entry represents a mounted file system. The root (/) file system is always the first entry. When full, the **mount**(S) system call returns the error EBUSY. Since the mount table is searched linearly, this value should be as low as possible.

NPROC
: Specifies how many process table entries to allocate. Each table entry represents an active process. The scheduler is always the first entry and **/etc/init** is always the second entry. The number of entries depends on the number of terminal lines available and the number of processes spawned by each user. The average number of processes per user is in the range of 2 to 5 (also see MAXUP). When full, the **fork**(S) system call returns the error EAGAIN. The NPROC entry is in the range of 100 to 400. Th value for NPROC is calculated automatically by the system unless specifically set to a non-zero value in the **/usr/sys/master.d/kernel** file. (See NREGION, which follows.)

NREGION Specifies how many region table entries to allocate. Each NREGION entry contains 36 bytes. Most processes have 3 regions: text, data, and stack. Additional regions are needed for each shared memory segment and shared library (text and data) attached. However, the region table entry for the text of a "shared text" program will be shared by all processes executing that program. Each shared memory segment attached to one or more processes uses another region table entry. A good starting value for this parameter is about 3.5 times NPROC. The default is to allocate 4*NPROC unless specifically set to a nonzero value. Shared libraries may need more. If the system runs out of region table entries, the following message is output on the system console.

```
Region table overflow
```

NCLIST Specifies how many character list buffers to allocate. Each buffer contains up to 122 bytes. The buffers are dynamically linked to form input and output queues for the terminal lines and other slow speed devices. The average number of buffers needed per terminal is in the range of 5 to 10. Each entry (buffer space plus header) contains 128 bytes. When full, input and output characters dealing with terminals are lost, although echoing continues. The default is to allocate 9 clists per port unless specifically set to a non-zero value.

MAXUP Specifies how many concurrent processes a non-superuser is allowed to run. The entry is normally 30. This value should not exceed the value of NPROC (NPROC should be at least 10% more than MAXUP). This value is per user identification number, not per terminal. For example, if 12 people are logged in on the same user identification, the default limit would be reached very quickly.

NOFILES Specifies the maximum number of open files per process. Default is 50. Processes using standard I/O subroutines are limited to 50, independent of the value of NOFILES. Unless an application package recommends that NOFILES be changed, the default setting should be left as is.

/bin/sh uses 3 file table entries: standard input, standard output, and standard error (0, 1, and 2 are normally reserved for stdin, stdout, and stderr, respectively). This leaves the value of NOFILES minus 3 as the number of other open files available per process. If a process requires up to three more than this number, then the standard files must be closed.

CAUTION

This practice is NOT recommended, and must be used with caution, if at all.

If the configured value of NOFILES is greater than the maximum (63) or less than the minimum (20), the configured value is set to the minimum (20), and a NOTICE message is sent to the console.

NHBUF Specifies how many "hash buckets" to allocate. These are used to search for a buffer given a device number and block number rather than a linear search through the entire list of buffers. **This value must be a power of 2.** Each entry contains about 12 bytes. The NHBUF value should be chosen so that the value NBUF divided by NHBUF is approximately equal to 4. While the value of NBUF is normally calculated by the system, that is not true for NHBUF. NHBUF must be specified in **/usr/sys/master.d/kernel.**

NPBUF Specifies how many physical input/output buffers to allocate. One input/output buffer is needed for each physical read or write active. Each entry contains 64 bytes.

BDFLUSHR
: Specifies the rate in seconds for checking the need to write the file system buffers to disk. The default is 1 second.

MAXPMEM
: Specifies the maximum amount of physical memory to use in pages. The default value of 0 specifies that all available physical memory be used.

SHLBMAX
: Specifies the maximum number of shared libraries that can be attached to a process at one time.

NFLOCKS
: Specifies the number of records that can be locked by the system. The default value is 100. Each entry contains 14 bytes. This value is calculated by the system unless specifically set to a non-zero value.

PUTBUFSZ
: Specifies the size of a circular buffer, **putbuf**, that is used to contain a copy of the last PUTBUFSZ characters written to the console by the operating system. This is the maximum length of a single kernel message that can be logged by the error logger.

WARNING

REL, SYS, and VER should not be changed, or some software packages may not install correctly.

REL
: Specifies the operating system release.

NODE
: Specifies the node name of the system. The default node name is empty. Node name can also be set with the **uname** -S command.

SYS
: Specifies the system name (default is **unix**).

VER
: Specifies the version.

MAXSLICE
: Specifies in clock ticks the maximum time slice for user processes. After a process executes for its allocated time slice, that process is suspended. The operating system then dispatches the highest priority process and allocates to it MAXSLICE clock ticks. MAXSLICE, defined in **/usr/sys/master.d/disp**, is normally 50 (one second).

Paging Parameters

There exists in the system a paging daemon, **vhand**, whose sole responsibility is to free up memory as the need arises. It uses a "least recently used" algorithm to approximate process working sets, and it writes those pages out to disk that have not been touched during some period of time. The page size is 4096 bytes. When memory is exceptionally tight, the working sets of entire processes may be swapped out.

The following tunable parameters determine how often **vhand** runs and under what conditions. The default values in **/usr/sys/master.d/kernel** should be adequate for most applications.

VHNDFRC
: Determines the initial value for the system variable VHANDL. VHANDL is set to the maximum useravailable memory divided by VHNDFRC or the value of GPGSHI, whichever is larger. The value of VHANDL determines when the paging daemon **vhand** runs. The amount of available free memory is compared with the value of VHANDL every VHANDR seconds. If free memory is less than VHANDL, then the paging daemon **vhand** is awakened.

  Decrease the value to make the daemon more active; increase the value to make the daemon less active (must be > 0 and < 25 percent of available memory).

VHANDR Specifies in seconds the maximum rate at which **vhand** can run. **vhand** will only run at this rate if free memory is less than VHANDL, as explained above for VHNDFRC. Increase the value to make the daemon less active (must be an integer > 0 and <= 300). If you have set the value higher, decreasing it makes the daemon more active.

GPGSLO Specifies the low water mark of free memory in pages for **vhand** to start stealing pages from processes. Increase the value to make the daemon more active; decrease the value to make the daemon less active (must be an integer >= 0 and < GPGSHI).

GPGSHI Specifies the high water mark of free memory in pages for **vhand** to stop stealing pages from processes. Increase the value to make the daemon more active; decrease the value to make the daemon less active. (The value must be an integer > 0, > GPGSLO and < 25 percent of the number of pages of available memory.)

GPGSMSK
Mask used by the paging daemon to determine the required number of aging passes before stealing a page. This value has been optimized by Altos and should not be changed.

MAXSC Specifies the maximum number of pages which will be swapped out in a single operation. The default value is 1.

MAXFC Specifies the maximum number of pages that will be added to the freelist in a single operation.

MAXUMEM
Specifies the maximum amount of memory a user process can occupy in pages. This value cannot be greater than 16MB worth of pages.

MINARMEM
Specifies the minimum number of memory pages reserved for the text and data segments of user processes.

MINASMEM
: Threshold value that specifies the number of memory and swap pages reserved for system purposes (unavailable for text and data segments of user processes).

Streams Parameters

The following tunable parameters are associated with Streams processing, and are defined in the file **/usr/sys/master.d/kernel.**

NQUEUE
: The number of STREAMS queues to be configured. Queues are always allocated in pairs, so this number should be even. A minimal Stream contains four queues (two for the Stream head, two for the driver). Each module pushed on a Stream requires an additional two queues. A typical configuration value is 8*NSTREAM.

NSTREAM
: The number of "Stream-head" (stdata) structures to be configured. One is needed for each Stream opened, including both Streams currently open from user processes and Streams linked under multiplexers. The recommended configuration value is highly application dependent, but a value of 32-40 usually suffices for running a single transport provider with moderate traffic.

NSTRPUSH
: The maximum number of modules that may be pushed onto a Stream. This is used to prevent an errant user process from consuming all of the available queues on a single Stream. By default this value is 9, but in practice, existing applications have pushed at most four modules on a Stream.

NSTREVENT

The initial number of Stream event cells to be configured. Stream event cells are used for recording process-specific information in the **poll**(S) system call. They are also used in the implementation of the STREAMS I_SETSIG **ioctl** and in the kernel **bufcall()** mechanism. A rough minimum value to configure would be the expected number of processes to be simultaneously using **poll**(S) times the expected number of Streams being polled per process, plus the expected number of processes expected to be using STREAMS concurrently. Note that this number is not necessarily a hard upper limit on the number of event cells that will be available on the system (see MAXSEPGCNT).

MAXSEPGCNT

The number of additional pages of memory that can be dynamically allocated for event cells. If this value is 0, only the allocation defined by NSTREVENT is available for use. If the value is not 0 and if the kernel runs out of event cells, it will under some circumstances attempt to allocate an extra page of memory from which new event cells can be created. MAXSEPGCNT places a limit on the number of pages that can be allocated for this purpose. Each event cell contains 12 bytes. Once a page has been allocated for event cells, however, it cannot be recovered later for use elsewhere. It is recommended that the NSTREVENT value be set to accommodate most load conditions, and that MAXSEPGCNT be set to 1 to handle exceptional load cases should they arise.

NMUXLINK

The maximum number of multiplexer links to be configured. One link structure is required for each active multiplexor link (STREAMS I_LINK **ioctl**). This number is application dependent; the default allocation equal to the number of Streams (NSTREAM) guarantees availability of links.

STRMSGSZ

The maximum allowable size of the data portion of any STREAMS message. This should usually be set just large enough to accommodate the maximum packet size restrictions of the configured STREAMS modules. If it is larger than necessary, a single **write**(S) or **putmsg**(S) can consume an inordinate number of message blocks. The recommended value of 4096 is sufficient for existing applications.

STRCTLSZ

The maximum allowable size of the control portion of any STREAMS message. The control portion of a **putmsg**(S) message is not subject to the constraints of the min/max packet size, so the value entered here is the only way of providing a limit for the control part of a message. The recommended value of 1024 is more than sufficient for existing applications.

NBLK*n*
RBLK*n*

The number of STREAMS data blocks and buffers to be allocated for each size class. Message block headers are also allocated based on these numbers: the number of message blocks is 1.25 times the total of all data block allocations. This provides a message block for each data block, plus some extras for duplicating messages (kernel functions **dupb**(), **dupmsg**()). The optimal configuration depends on both the amount of primary memory available and the intended application. The default values provided are intended to support a moderately loaded configuration using RFS and UUCP/CU over the network. The NBLK*n* values are used if RFS is not installed; the RBLK*n* values are used if RFS is installed.

STRLOFRAC

The percentage of data blocks of a given class at which low-priority block allocation requests are automatically failed. For example, if STRLOFRAC is 80 and there are 48 256-byte blocks, a low-priority allocation request will fail when more than 38 256-byte blocks are already allocated. The parameter is used to help

prevent deadlock situations by starving out low-priority activity. The recommended value of 80 works well for current applications. STRLOFRAC must always be in the range 0 <= STRLOFRAC <= STRMEDFRAC.

STRMEDFRAC
The percentage cutoff at which medium priority block allocations are failed (see STRLOFRAC discussion above). The recommended value of 90 works well for current applications. STRMEDFRAC must always be in the range STRLOFRAC <= STRMEDFRAC <= 100.

NOTE

There is no cutoff fraction for high-priority allocation requests; it is effectively 100.

Log Driver Parameters

The configurable parameters for the log driver are found in the file **/usr/sys/master.d/log.** They include the following:

NLOG The number of minor devices to be configured for the log driver; the active minor devices will be 0 through (NLOG-1). The recommended value of 3 services an error logger (**strerr**(M)) and a trace command (**strace**(M)), with one left over for miscellaneous usage. If only an error logger and a tracer are to be supported, this number can be set to 2. If there are several daemons for an application that may be submitting log messages, this number can be increased to accommodate the extra users.

BSIZE This number controls the number of log messages that will be kept in the log driver after being sent upstream to the error or trace logging process. Duplicates of only the last BSIZE messages are kept at the log driver; older messages are freed as new ones are submitted. The intent of this feature is to hold on to the last few log

messages in case a system crash prevents their being written to the log file. Use a system dump analysis tool to look at these messages to determine if they point to the cause of the crash. The recommended number of 5 is sufficient for most burst-traffic situations.

Message Parameters

The following tunable parameters are associated with interprocess communication messages, and are defined in the file **/usr/sys/master.d/msg**. They are described in the order in which they are defined in the output of the **/etc/sysdef** command.

MSGMAP Specifies the size of the control map used to manage message segments. Each entry contains 8 bytes.

MSGMAX Specifies the maximum size of a message. The maximum size is 64 kilobytes -1.

MSGMNB Specifies the maximum length of a message queue.

MSGMNI Specifies the maximum number of message queues system-wide (id structure).

MSGSSZ Specifies the size, in bytes, of a message segment. Messages consist of a contiguous set of message segments large enough to fit the text. The value of MSGSSZ times the value of MSGSEG must be less than or equal to 131,072 bytes (128 kilobytes).

MSGTQL Specifies the number of message headers in the system and, thus, the number of outstanding messages. Each entry contains 12 bytes.

MSGSEG Specifies the number of message segments in the system. The value of MSGSSZ times the value of MSGSEG must be less than or equal to 131,072 bytes (128 kilobytes).

Semaphore Parameters

The following tunable parameters are associated with interprocess communication semaphores. These parameters are defined in the **/usr/sys/master.d/sem** file. They are described in the order in which they are defined in the output of the **/etc/sysdef** command.

SEMMAP Specifies the size of the control map used to manage semaphore sets. Each entry contains 8 bytes.

SEMMNI Specifies the number of semaphore identifiers in the kernel. This is the number of unique semaphore sets that can be active at any given time. Each entry contains 32 bytes.

SEMMNS Specifies the number of semaphores in the system. Each entry contains 8 bytes.

SEMMNU Specifies the number of undo structures in the system. The size is equal to 8 x (SEMUME + 2) bytes.

SEMMSL Specifies the maximum number of semaphores per semaphore identifier.

SEMOPM Specifies the maximum number of semaphore operations that can be executed per **semop**(S) system call. Each entry contains 8 bytes.

SEMUME Specifies the maximum number of undo entries per undo structure. The size is equal to 8*(SEMMNU) bytes.

SEMVMX Specifies the maximum value a semaphore can have. The default value is the maximum value for this parameter.

SEMAEM Specifies the adjustment on exit for maximum value, alias **semadj**. This value is used when a semaphore value becomes greater than or equal to the absolute value of **semop**(S), unless the program has set its own value. The default value is the maximum value for this parameter.

Shared Memory Parameters

The following tunable parameters are associated with interprocess communication shared memory. These parameters are defined in the **/usr/sys/master.d/shm** file. They are described in the order in which they are defined in the output of the **/etc/sysdef** command.

SHMMAX Specifies the maximum shared memory segment size.

SHMMIN Specifies the minimum shared memory segment size.

SHMMNI Specifies the maximum number of shared memory identifiers system wide. Each entry contains 52 bytes.

SHMSEG Specifies the number of attached shared memory segments per process. The maximum value is 15.

SHMALL Specifies the maximum number of in-use shared memory text segments.

Remote File Sharing Parameters

RFS parameters are discussed in the RFS manual. NSRMOUNT is included in Figure 9-2 because it is located in the **/usr/sys/master.d/kernel** file. The values associated with NSRMOUNT are 0 unless the RFS package is installed. The rest of the RFS parameters are located in **/usr/sys/master.d/du.**

NSRMOUNT
Specifies the maximum number of resources from this system that can be mounted by other systems (two systems mounting the same resource use two entries).

SYSTEM PARAMETERS

Recommended parameters for memory, disk storage, and swap space follow. Your system may work better with different settings. Application programs that you install may have additional recommendations (see the application Release Notes for recommended settings).

Memory 100+ KB per process plus space for the kernel. It is not recommended to have less than 40 KB per process.

Disk storage
15 MB per each user who has an account on the system.

Swap space
256 KB per active user. This is determined by the data space used by each process. A practical minimum is 5 MB for small configurations and 15 - 20 MB for heavily loaded large configurations. Certain applications, most notable word processors, spreadsheets, and other data-intensive programs, may require even larger swap spaces.

(BLANK)

Chapter 10
Solving System Problems

(BLANK)

INTRODUCTION

This chapter explains how to solve problems that affect the operation of the system. The problems range in complexity from how to fix a non-echoing terminal, to how to restore lost system files.

RESTORING A NON-ECHOING TERMINAL

A non-echoing terminal is any terminal that does not display characters typed at the keyboard. This abnormal operation can occur whenever a program stops prematurely as a result of an error, or when the user presses the Break/Del key, the No Scroll key, Ctrl-s, or Ctrl-q.

To restore the terminal to normal operation, follow these steps:

1. Type Ctrl-j. The system may display an error message. If it does, ignore the message.

2. Type:

   stty sane

   and type Ctrl-j. The terminal does not display what you type, so type carefully.

After pressing the Ctrl-j keys, the terminal should be restored and you may continue your work.

FREEING A JAMMED LINEPRINTER

Lineprinter errors, such as running out of paper, can cause the **lpd** program to "lock up" the printing queue, preventing the current file and any other files in the queue from being printed. The **lpd** program is the "lineprinter daemon," the program that does the actual printing for the system print command **lpr**.

To free a jammed lineprinter, follow these steps:

1. Turn the printer off.

2. Log in as **root.**

3. Type:

   ps -a

   and press Retn to find the process identification number (PID) of the **lpd** program.

   (The PID is in the first column of the display.) The command display should look like this:

```
PID    TTY    TIME    COMMAND
34     01     0:08    sh
135    01     0:25    lpd
```

4. Type:

   kill -9 *PID*

   and press Retn. The *PID* is the process identification number of the program (135 in the example above).

5. Locate and fix the error that caused the lineprinter to become jammed.

6. Type:

   cd /usr/spool/lpd

   and press Retn to change to the lineprinter spool directory. This directory temporarily holds the files to be printed.

7. Type:

   rm -f lock

and press Retn to remove the lineprinter spool's lock file. This frees the queue and allows printing to continue.

After freeing the lineprinter, turn the printer on and type:

/usr/bin/lpd

STOPPING A LOCKED PROCESS

A locked process is a program that cannot be stopped from the terminal at which it was invoked. This occurs whenever an error in the program "locks up" the terminal, that is, prevents anything you type from reaching the system.

To stop a locked process, follow these steps:

1. Go to a terminal that is not locked up.

2. Log in as **root**.

3. Type:

   ps -a

   and press Retn. The system displays all current processes and their process identification numbers (PIDs). Find the PID of the locked process.

4. Type:

   kill -9 *PID*

   and press Retn. The *PID* is the process identification number of the locked program. The process should stop in a few seconds. If the process does not stop, repeat this step.

The last step is sure to stop the process, but may leave temporary files or a non-echoing terminal. To restore the terminal to normal operation, follow the instructions in the section "Restoring a Non-echoing Terminal" in this chapter.

STOPPING A RUNAWAY LOGIN

A runaway login occurs when you try to bring up your system in multiuser mode after enabling a port that should not have been enabled. This can happen as a result of the following actions:

- A port was enabled with **pconfig,** but there is no device physically attached to the port.
- The printer port was enabled for login.

During a runaway login the **getty** program forks the **login** program. **Login** dies because it tries to output to a device that does not exist and the process is repeated.

To stop a runaway login:

1. Everyone should log off, except the system administrator.
2. Login as **root.**
3. Type **ps -ef** several times. Each time you type the command, note which process ID numbers are increasing rapidly and the tty number associated with the ID.
4. Run **pconfig** to disable the offending port.

REPLACING A FORGOTTEN PASSWORD

The operating system does not provide a way to decipher an existing password. If a user forgets his password, the system administrator must change the password to a new one. To change an ordinary user password, follow the instructions in the section "Changing a User's Password" in Chapter 3.

REMOVING HIDDEN FILES

A hidden file is any file whose name begins with a dot (.), such as .login. You can list the hidden files in a directory by typing:

```
ls -a
```

and pressing Retn.

You can remove most hidden files from a directory by typing:

```
rm .[a-z]*
```

and pressing Retn. Remaining files can be removed individually.

RESTORING AN INOPERABLE SYSTEM

On very rare occasions, one or more of the critical system files may be accidentally modified or removed, preventing the system from operating. In such a case, you must reinstall the system, and restore user program and data files from backup disks. To reinstall the system, follow the instructions in the *Installation Manual.* To restore files from backup disks, follow the instructions in Chapter 6, "Restoring a Backup File."

RECOVERING FROM A SYSTEM CRASH

A system crash is a sudden and dramatic disruption of system operation that stops all work on the computer. System crashes occur very rarely. They are usually the result of hardware errors or damage to the root file system which the operating system cannot correct by itself. When a system crash occurs, the system usually displays a message

explaining the cause of the error, then stops. This gives the system administrator the chance to recover from the crash by correcting the error (if possible) and restarting the system.

A system crash has occurred if 1) the system has displayed at the system console a message beginning with "panic:", or 2) the system refuses to process all input (including Break/Del and QUIT keys) from the system console and all other terminals.

To recover from a system crash, follow these steps:

1. Use the error message(s) displayed on the system console to determine the error that caused the crash. If there is no message, skip to step 3.

2. Correct the error, if possible. A list of error messages and their descriptions appear in Appendix E. Even if the problem cannot be located or corrected, it is generally worthwhile to try to restart the system at least once by completing the remaining steps in this procedure.)

3. Turn off the computer and follow the steps described in Chapter 2, "Starting the System," to restart the system.

4. If the system will not restart, or crashes each time it is started, the operating system is inoperable and must be reinstalled. If you recently reconfigured the operating system, see the chapter titled "Performance Management." Follow the procedures described in the *Installation Manual* to reinstall the system and in Chapter 7, "Backing Up File Systems," to restore users' files. This procedure will write over all existing data on the hard disk.

5. If the system cannot be booted from the "Boot" diskette in the distribution set for installation, the computer has a serious hardware malfunction. Contact a hardware service representative for help.

CHANGING SYSTEM INITIALIZATION

One common problem is how to adapt the system initialization to suit your system environment. This problem occurs whenever you have added new devices such as terminals or disk drives to the system, and wish these devices to be automatically enabled or mounted whenever you start normal system operation. You can adapt system initialization by modifying the system initialization files.

The initialization files contain commands and/or data which the system reads at system startup or whenever a user logs in. The files typically mount file systems, start programs, and set home directories and terminal types. The initialization files are named .cshrc, .profile, /etc/brc, and files in the directory /etc/rc2.d.

The system administrator may modify these files to create any desired initial environment. The files are ordinary text files and may be modified using a text editor such as **vi** (see the *User's Guide)*, but not a word processor such as Uniplex.

NOTE

The files under /etc/brc, /etc/rc2.d, and .profile files contain system commands and comments, and have the command file format described in Chapter 7, "The Shell," in the *User's Guide.*

Changing the System Start Up Files

The contents of the /etc/brc file and /etc/rc2.d directory contain operating system initialization commands. The system executes the commands at startup time.

The /etc/rc1.d directory contains files executed by /etc/rc1 for transitions to system run-level 1 (single user). Files in this directory are linked from files in the /etc/init.d directory. The files begin with a **K**, where K indicates processes that are killed or stopped when entering single-user mode.

The /etc/rc2.d directory contains files executed by /etc/rc2 for transitions to system run-level 2 (multi-user). Files are linked from files in the /etc/init.d directory and begin with S, where S indicates processes that are started when entering multi-user mode.

For example, /etc/rc1.d directory contains the following files:

```
K20sys.stop  K75cron
```

The /etc/rc2.d directory contains the following files:

```
S01MOUNTFSYS  S02.printers  S20sys.start  S75cron
```

S or K	Defines whether the process should be started (S) or stopped (K).
nn	The two digit number indicates the order in which the files will be started or stopped.
name	The /etc/init.d filename this file is linked to.

In the examples, the init.d file *cron* is linked to both the rc1.d and rc2.d files. (Init.d is also linked to rc5.d; see Chapter 2 for details.) When you enter init level 2 (multi-user), this file is executed with the start option (S75cron). When you enter init level 1 (single user), the cron file is executed with the stop option (K75cron).

The files in /etc/rc1.d and /etc/rc2.d are shell scripts and can be modified. Follow these guidelines:

- Place the file in /etc/init.d
- Include the start and/or stop options.
- Specify the order in which the files will be started or stopped.
- Link the files to files in /etc/init.d.

CHANGING THE /etc/motd FILE

The message of the day file, /etc/motd, contains the greeting displayed whenever a user logs in. Initially, this file contains the name and version number of the operating system. It can be modified to include such messages as a reminder to clean up directories, a notice of the next periodic backup, and so on.

The /etc/motd file is an ordinary text file, so you can change the message by editing the file with a text editor. One common change is to include a reminder to delete unused files in order to preserve disk space. In general, you should limit the size of the file to include no more than a screenful of information.

(BLANK)

Appendix A
System Directories

(BLANK)

INTRODUCTION

This appendix lists the most frequently used files and directories in the system. Many of these files and directories are required for proper operation and must not be removed or modified. The following sections briefly describe each directory.

THE ROOT DIRECTORY

The root directory (/) contains the following system directories:

/bin	System command directory
/dev	Special (device) file directory
/etc	Additional program and data file directory
/lib	C program library directory
/usr	User home directories and some system programs
/tmp	Temporary directory (reserved for temporary file created by programs)

All of the above directories are required for system operation.

The root directory also contains a few ordinary files. Of these files, the most notable is the /unix file which contains the operating system kernel image.

THEN /bin DIRECTORY

The /bin directory contains the most common operating system commands, that is, the commands likely to be used by anyone on the system. The following is a list of a few of the commands.

basename	login	sleep	test
cp	mv	stty	
date	passwd	su	
echo	rm	sync	
expr	sh	tar	

These commands and all others in the /bin directory are required.

THE /dev DIRECTORY

The /dev directory contains special device files which control access to peripheral devices. All files in this directory are REQUIRED and MUST NOT BE REMOVED. The following is a partial list of the files.

/dev/console	System console
/dev/lp	Lineprinter
/dev/mem	Physical memory
/dev/null	Null device (used to redirect unwanted output)
dev/r*xx*	Unbuffered interface to corresponding device name
/dev/root	Root file system
/dev/swap	Swap area
/dev/syscon	Linked to console
/dev/systty	Linked to console
/dev/tty*xx*	Terminals
/dev/tty	The terminal you are using

THE /etc DIRECTORY

The /etc directory contains miscellaneous system program and data files. All files are required, but many may be modified.

The following program and data files should not be removed. /etc/init and /etc/inir must not be removed (the system will not boot without them).

/etc/mnttab	Mounted device table
/etc/mount	For mounting a file system
/etc/mkfs	For creating a file system
/etc/inir	Very first process started by the system
/etc/init	First process after boot
/etc/inittab	Init reads inittab
/etc/singleuser	Single user mode script

The following data files may be modified, if desired.

/etc/passwd	Password file
/etc/termcap	Terminal capability map
/etc/otd	Message of the day
/etc/profile	Startup file for /bin/sh users
/etc/cshrc	Startup file for /bincsh users

THE /lib DIRECTORY

The /lib directory contains Run-time library files for C and other language programs. This directory is required.

THE /tmp DIRECTORY

The /tmp directory contains temporary files created by operating system programs. The files are normally present when the corresponding program is running, but may also be left in the directory if the program is prematurely stopped. You may remove any temporary file that does not belong to a running program.

THE /usr DIRECTORY

The /usr directory contains the home directories of all users on the system. It also contains several other directories which provide additional operating system commands and data files.

The /usr/bin directory contains operating system commands. These commands are less frequently used or considered non-essential to system operation.

The/usr/include directory contains header files for compiling C programs.

The /usr/lib directory contains more libraries and data files used by various operating system commands.

The/usr/spool directory contains various directories for storing files to be printed, or passed through networks.

The/usr/tmp directory contains more temporary files.

The/usr/adm directory contains data files associated with system administration and accounting. The error file contains a log of system problems. See **errprint**(M) in the *Reference (M)*.

LOG FILES

A variety of directories contain log files that grow in size during the normal course of system operation. Many of these files must be periodically cleared to prevent them from taking up valuable disk space (see the section, "Clearing Log Files," in Chapter 6). The following table lists the files (by full pathname) and their contents.

Table A-1. Log Files

Filename	Description
/usr/adm/pacct	Records accounting information; grows rapidly when process accounting is on.
/usr/adm/stream/*	Records error messages generated by the system when started (see **strerr**(M)).
/usr/adm/wtmp	Records user logins and logouts.
/usr/adm/sulog	Records each use of the **su** command; grows only if option is set in the /etc/default/su file.
/usr/lib/cron/log	Records cron information.
/usr/spool/at/past	Records each use of the **at** command.

(BLANK)

Appendix B
Building a Communication System

INTRODUCTION

This appendix explains how to build a communication system for your computer using a normal telephone line and a Hayes Smartmodem 1200 or Hayes compatible modem.

A communication system provides a way to:

- Log in to the computer from a remote terminal or computer.
- Use the **cu** command to call and log in to other computers.
- Use the **uucp** command to copy files to and from remote computers.
- Use the **uux** command to execute a remote mail program (**rmail**) on a remote computer.

In other words, the communication system is intended to give access to terminals and computers that cannot be connected to your computer through a direct serial line.

All communication tasks are supported by a variety of files and directories. In addition, the tasks invoked by the **uucp** and **uux** commands are actually performed by a system of underlying programs, called the **uucp** system. The files and underlying programs are described in full later in this appendix.

The following sections explain how to prepare the programs you need to build a communication system. They also explain how to install and maintain a **uucp** system.

WHAT YOU NEED

To install a communication system on your computer, you will need:

- A modem (for example, a Hayes Smartmodem 1200)
- A standard telephone jack for access to the telephone system

- An RS-232 serial line (or serial port) on your computer
- An RS-232 cable to connect the serial line to the modem

The *Owner's Guide* that came with your system describes how to install the modem.

Test the Modem

Once you have verified that the modem is working, you can begin to use the communication system.

To test the modem, follow these steps:

1. Start the computer and log in as the **super-user.**
2. Create the call unit files as described in a following subsection. Configure the serial line as type "Modem" using **pconfig.**
3. Connect the modem to the serial line using the cable described in the *Owner's Guide*, and turn on the modem's power.
4. Make sure the volume control on the modem is at an appropriate level. You must be able to hear the modem to carry out this test successfully. See your modem manual for the location of this control.
5. Invoke the **cu** program using a command line of the form:

   cu -s *speed* **-l /dev/tty**nn *number*

   where **/dev/tty**nn is the filename of your serial line, and *number* is your telephone number (the number of the telephone jack your modem is connected to). For example, if your serial line is /dev/tty04 and number is "5551234," type:

   cu -s 1200 -l /dev/tty04 5551234

6. Listen carefully to the modem. You should hear each digit as the number is dialed, then hear the busy signal when the telephone system tries to make connection with your modem.

7. If the busy signal is present, wait a few moments and listen carefully for the modem to hang up. The modem automatically discontinues any call for which it cannot make a connection.

8. If the busy signal is not present, make sure you have connected the modem to the telephone jack. Make sure the jack is connected to the phone system. Make sure you gave the correct number when invoking **cu**.

9. If you did not hear the modem dial, make sure the volume is turned up. Make sure the modem is connected to the correct serial line and that the cable connection is tight. Make sure you gave the correct device name when invoking **cu**. Make sure the modem's power is on.

CREATING A DIAL-IN LINE

You can create a dial-in line for use by remote terminals or computers by enabling the modem's serial line with the **pconfig** command. Once the line is enabled, any user at a remote terminal or computer can log in to your computer by calling your modem and following the ordinary login procedure. To create a dial-in line, follow these steps:

1. Log in as the **super-user**.

2. Run **pconfig** (see Chapter 8, "Using Peripheral Devices") to set up the dial-in port and configure the line as type "Modem."

Now your computer can receive calls from remote terminals or computers and prompt for a login name.

CREATING A DIAL-OUT LINE

You can create a dial-out line by editing the /usr/lib/uucp/Devices file and creating the device files. A dial-out line lets you call and log in to other computers by using the **cu** command. The **cu** command uses the /usr/lib/uucp/Devices file to locate the modem's serial line and set the proper line speed when these values are not explicitly given in the **cu** command line.

Configure the Device Files

Follow these steps:

1. Log in as the **super-user.**

2. Use the **chmod** command to change the access mode of the device file to read and write for owner and group. For example, the command:

   chmod 660 /dev/tty14

   sets the appropriate permissions for the **/dev/tty14.**

3. Use the **chown** command to change the owner to uucp

   chown uucp /dev/tty14

4. Use the **chgrp** command to change the group to uucp:

   chgrp uucp /dev/tty14

Create the Devices File

The /usr/lib/uucp/Devices file defines the devices you intend to use to implement the dial-out line. The file is also used by programs in the **uucp** system (as described later).

Each entry in the Devices file has the following format:

Type Line Line2 Class Dialer-Token-Pairs

Each of these fields is defined below.

Type This field may contain one of the following keywords:

Direct This keyword indicates a Direct Link to another computer or a switch (for **cu** connections only).

ACU This keyword indicates that the link to a remote computer is made through an automatic call unit (Automatic Dial Modem). This modem may be connected either directly to your computer or indirectly through a Local Area Network (LAN) switch.

Network This keyword indicates that the link is established through a LAN switch where *Network* is replaced with either **micom** or **develcon.** These switches are the only ones that contain caller scripts in the Dialers file. Other switches may be issued if caller scripts are constructed and placed in the Dialers file.

System-Name
This keyword indicates a direct link to a particular computer where *System-Name* is replaced by the name of the particular computer. This naming scheme is used to convey the fact that the line associated with this Devices entry is for a particular computer.

The keyword used in the *Type* field is matched against the third field of /usr/lib/uucp/Systems file entries as shown below:

```
Devices: ACU tty14,M - 1200 hayes

Systems: eagle Any ACU 1200 3-2-5-1 ogin:  nuucp \
         ssword:  Oakgrass
```

Line This field contains the device name of the line (port) associated with the **Devices** entry. For instance, if the Automatic Dial Modem for a particular entry was attached to the /dev/tty14 line, the name entered in this field would be tty14. If the device is a modem, append ",M" to the device name.

Line2 If the **ACU** keyword is used in the *Type* field and the ACU has a separate dialer and modem, this field would contain the device name of the dialer. Therefore, a separate modem is required and would be connected to a different line, (defined in the *Line* field). This means that one line would be allocated to the modem and another to the dialer. Since the computer will not normally use this type of configuration, this field will be ignored; but it must still contain a pseudo entry as a placeholder (use a hyphen (-)).

Class If an **ACU** keyword is used, it may be just the spee of the device. It may contain a letter and a speec (for example, C1200, D1200) to differentiate between classes of dialers (Centrex or Dimension PBX This is necessary because many larger offices may have more than one type of telephone network. Or network may be dedicated to serving only internal office communications and another to the external communications. Therefore, it becomes necessary t distinguish which line(s) should be used for internal communications and which should be used for external communications. The same distinction mus be made in the Systems file because a match is ma against the fourth field of Systems file entries as shown below:

```
Devices: ACU contty - D1200 hayes

Systems: eagle Any ACU D1200 3-2-5-1 ogin: nuucp \
         ssword: Oakgrass
```

Some devices can be used at any speed, so the keyword **Any** may be used in the *Class* field. If **Any** is used, the line will match any speed requested in a Systems file entry. If this field is **Any** and the Systems file *Class* field is **Any**, the speed defaults to 1200 bps.

Dialer-Token-Pairs

This field contains pairs of dialers and tokens. The *dialer* portion may be the name of an automatic dial modem, a LAN switch, or it may be **direct** for a Direct Link device. The *token* portion may be supplied immediately following the *dialer* portion or if not present, it will be taken from a related entry in the Systems file.

This field has the format:

dialer-token dialer-token

where the last pair may or may not be present, depending on the associated device (dialer). In most cases, the last pair contains only a *dialer* portion and the *token* portion is retrieved from the *Phone* field of the Systems file entry.

The *Dialer-Token-Pairs* (*DTP*) field may be structured four different ways, depending on the device associated with the entry:

1. If a direct link is established to a particular computer, the *DTP* field of the associated entry would contain the keyword **direct**. This is true for both types of direct link entries, Direct and System-Name (refer to the discussion on the Type field).

2. If a direct link is established to a modem, the *DTP* field of the associated Devices entry will have only one pair. This pair would normally be the name of the modem. This name is used to match the particular Devices entry with an entry in the Dialers file. Therefore, this "dialer" must match the first field of a Dialers file entry as shown below:

```
Devices:  ACU contty - 1200 ventel
```

```
Devices:  ACU contty - 1200 ventel

Dialers:  ventel =&-% " \r\p\r\c $ <K\T%%\r>\c ONLINE!
```

Notice that only the *dialer* portion (**ventel**) is present in the *DTP* field of the **Devices** file entry. This means that the *token* to be passed on to the dialer (in this case the phone number) is taken from the *Phone* field of a **Systems** file entry.

3. If an automatic dialing modem is connected to a LAN, your computer must first access the switch, and the switch will make the connection to the automatic dialing modem. This type of entry requires two *dialer-token-pairs.* The *dialer* portion of each pair (fifth and seventh fields of entry) will be used to match entries in the Dialers file as shown below:

```
Devices:  ACU tty14 - 1200 develcon vent ventel

Dialers:  develcon " " \pr\ps\c est:\007 \E\D\e \007

Dialers:  ventel =&-% " \r\p\c $ <K\T%%\r>\c ONLINE!
```

In the first pair, **develcon** is the dialer and **vent** is the token that is passed to the Develcon switch to tell it which device (ventel modem) to connect to your computer. This token would be unique for each LAN switch since each switch may be set up differently. Once the ventel modem has been cc nected, the second pair is accessed, where ventel is the dialer and the token is retrieved from the Systems file.

4. If a computer with which you wish to communicate is on the same local network switch as your computer, your computer must first access the switch and the switch can make the connection to the other computer. In thi type of entry, there is only one pair. The *dialer* portion is used to match the following Dialers file entry.

```
Devices:  develcon tty13 - 1200 develcon \D

Dialers:  develcon ""\pr\ps\c est:\007 \E\D\e \007
```

As shown, the *token* portion is left blank, which indicates that it is retrieved from the Systems file. The Systems file entry for this particular computer will contain the token in the *Phone* field, which is normally reserved for the phone number of the computer (refer to Systems file, *Phone* field). This type of *DTP* contains an escape character (\D), which ensures that the contents of the *Phone* field will not be interpreted as a valid entry in the Dialcodes file.

There are two escape characters that may appear in a *DTP* field:

\T Indicates that the *Phone* (*token*) field should be translated using the Dialcodes file. This escape character is normally placed in the Dialers file for each caller script associated with an automatic dial modem (penril, ventel, etc.). Therefore, the translation will not take place until the caller script is accessed.

\D Indicates that the *Phone* (*token*) field should *not* be translated using the Dialcodes file. If no escape character is specified at the end of a Devices entry, the \D is assumed (default). A \D is also used in the Dialers file with entries associated with network switches (develcon and micom).

Configure the Serial Line

Configuring the serial line is the last step in creating a dialout line. To configure the serial line, follow these steps:

1. Make sure your modem has been installed and tested.

2. Make sure you are logged in as the super-user.

3. Use **pconfig** to configure the serial line as type "Modem" by adding device tty*nn* and choosing type "Modem."

4. If the line is to be used for both dialing in and out, select "Active" when the Action menu appears. If the line is to be used only for dialing out, select "Inactive."

You will now be able to call other computers that have dial-in lines by using the **cu** command. For a complete description of the command, see **cu**(C) in the *Reference (C)*.

INSTALLING A UUCP SYSTEM

A uucp system is a set of files and programs that let you use the **uucp** and **uux** commands to transfer files and commands between computers connected by your communication system. Before you can use the **uucp** and **uux** commands, you must install the uucp system by creating or modifying a number of uucp system files.

The uucp system actually provides two different methods of interaction with other computers. One method requires a dial-in line through which remote computers can log in and transfer files and commands. With this method, your computer is called a "dial-in site." The other method requires a dial-out line through which your computer can call other computers. With this method, your computer is called a "dial-out site." Each method requires its own set of uucp system files.

The following sections explain how to create files for both methods of interaction. They also explain how to create a transmission schedule and develop a **cron** script to implement the schedule.

Choose a UUCP Site Name

In a uucp system, every computer belongs to a given "site." A site is any computer that can communicate with the uucp system through a modem. To distinguish one site from another, every site must have a unique *site name*. A site name is any combination of letters and digits that begins with a letter and is no more than seven characters long. The site name may then by used in **uucp** and **uux** commands to direct transmissions to the appropriate computer.

The site name should suggest some characteristic of the site, such as its location or affiliation. For example, a site in Chicago can be named "chicago," or a site in the shipping department can be named "shipping." The *site name* must be unique. That is, no other computer that calls your computer or is called by your computer can have the same site name.

Once you have chosen a site name, you will need to set it with the **uname**(C) command. Type:

uname -S *sitename*

where *sitename* is the name you have chosen.

Create a Dial-In Site

You can create a dial-in site by installing the **uucp** login information required by other computers that wish to log in and transfer files and commands. This information consists of the following:

- One or more /etc/passwd file entries
- User access information in the /usr/lib/uucp/Permissions file

You can create this information by using a text editor and modifying or creating the appropriate files. The following sections explain the required format of the information.

Once the information is installed, you can enable the system for logins by creating a dial-in line. See the section "Creating a Dial-In Line" given earlier in this appendix.

Create UUCP Login Entries

A dial-in site must provide a login entry for the sites that call it. These entries must be placed in the /etc/passwd file. A **uucp** login entry has the same form as an ordinary user login entry (see Chapter 3 in this guide), but gives a special login directory and login program instead of the normal user directory and shell. To create a uucp login entry, follow these steps:

1. Choose a new login name and a user ID for the uucp login. The name may be any combination of letters and digits that is no more than eight characters long. The user ID must be an integer number in the range 1 to 60,000. Make sure the name and ID are unique; a uucp login entry must not have the same name or ID as any other login entry.

2. Invoke a text editor giving /etc/passwd as the file to edit.

3. Move to the end of the file and insert the login entry using the form:

   login-name::*user-ID*:*group-ID*::/usr/spool/uucppublic:
   /usr/lib/uucp/uucico

   where:

   login-name is the login name you have chosen

   user-ID is the user ID you have chosen

   group-ID is the ID of the uucp group

   For example, if you have chosen "uuchcg" for the login name and "12" for the user ID, add the following entry to the end of the file:

   uuchcg::12:5::/usr/spool/uucppublic:/usr/lib/uucp/uucico

4. Save the new file and exit the editor.

5. Create a new password for the login with the **passwd** command. Type:

   passwd *login-name*

   where *login-name* is the login name you have chosen. The command will ask you to type the new password twice. It will then add the encrypted password to the new login entry.

   Note that you can create new login entries for each site that calls your site, or use one entry for all sites.

Edit the Permissions File

The /usr/lib/uucp/Permissions file defines which directories a given site (or a given user) may access using the **uucp** and **uux** commands. You should create one Permissions entry for each site or user with a login entry in the /etc/passwd file.

How Entries are Structured

Each entry is a logical line with physical lines terminated by a backslash (\) to indicate continuation. Each option is a name/value pair (delimited by white space) in the following format:

name=value

Note that no white space is allowed within an option assignment. Comment lines begin with a "#," and they occupy the entire line up to a newline character. Blank lines are ignored (even within multi-line entries).

There are two types of Permissions file entries:

LOGNAME Specifies the permissions that take effect when a remote computer logs in on (calls) your computer.

MACHINE Specifies permissions that take effect when your computer logs in on (calls) a remote com puter.

LOGNAME entries will contain a LOGNAME option and MACHINE entries will contain a MACHINE option.

Considerations

The following items should be considered when using the Permissions file to restrict the level of access granted to remote computers:

- All login IDs used by remote computers to login for UUCP communications must appear in one and only one LOGNAME entry.
- Any site that is called whose name does not appear in a MACHINE entry, will have the following default permissions/restrictions:
  - Local send and receive requests will be executed.
  - The remote computer can send files to your computer's **/usr/spool/uucppublic** directory.
  - The commands sent by the remote computer for execution on your computer must be one of the default commands, usually **rmail.**

Options

This section describes each option, specifies how it is used, and lists default values.

REQUEST When a remote computer calls your computer and requests to receive a file, this request ca be granted or denied. The REQUEST option specifies whether the remote computer can request to set up file transfers from your computer. The string:

```
REQUEST=yes
```

specifies that the remote computer can request to transfer files from your computer. The string:

```
REQUEST=no
```

specifies that the remote computer cannot request to receive files from your computer. This is the default value. It will be used if the REQUEST option is not specified. The REQUEST option can appear in either a LOGNAME (remote calls you) entry or a MACHINE (you call remote) entry. A note on security: when a remote machine calls you, unless you have a unique login and password for that machine, you don't know if the machine is who it says it is.

SENDFILES

When a remote computer calls your computer and completes its work, it may attempt to take work your computer has queued for it. The SENDFILES option specifies whether your computer can send the work queued for the remote computer.

The string:

```
SENDFILES=yes
```

specifies that your computer may send the work that is queued for the remote computer as long as it logged in as one of the names in the LOGNAME option. This string is mandatory if your computer is in a "passive mode" with respect to the remote computer.

The string:

```
SENDFILES=call
```

specifies that files queued in your computer will be sent only when your computer calls the remote computer. The call value is the deremote computer. The call value is the default for the SENDFILE option. This option is only significant in LOGNAME

entries since MACHINE entries apply when calls are made out to remote computers. If the option is used with a MACHINE entry, it will be ignored.

READ and WRITE

These options specify the various parts of the file system that **uucico** can read from or write to. The READ and WRITE options can be usec with either MACHINE or LOG-NAME entries.

The default for both the READ and WRITE options is the **uucppublic** directory as shown in the following strings:

```
READ=/usr/spool/uucppublic  \
WRITE=/usr/spool/uucppublic
```

The strings:

```
READ=/ WRITE=/
```

specify permission to access any file that can be accessed by a local user with "other" permissions. The value of these entries is a colon separated list of path names. The REAL option is for requesting files, and the WRITE option for depositing files. One of the values must be the prefix of any full path name of a file coming in or going out. To grant permission to deposit files in /usr/news as well as the public directory, use the following values with the WRITE option:

```
WRITE=/usr/spool/uucppublic:/usr/news
```

It should be pointed out that if the READ and WRITE options are used, all path names must l specified because the path names are not adde to the default list. For instance, if the /usr/news path name was the only one specifi in a WRITE option, permission to deposit files in the public directory would be denied.

You should be careful which directories you make accessible for reading and writing by remote systems. For example, you probably

wouldn't want remote computers to be able to write over your /etc/passwd file so /etc shouldn't be open to writes.

NOREAD and NOWRITE

The NOREAD and NOWRITE options specify exceptions to the READ and WRITE options or defaults. The strings

```
READ=/ NOREAD=/etc  \
WRITE=/usr/spool/uucppublic
```

would permit reading any file except those in the /etc directory (and its subdirectories--remember, these are prefixes) and writing only to the default directory, /usr/spool/uucppublic. NOWRITE works in the same manner as the NOREAD option. The NOREAD and NOWRITE options can be used in both LOGNAME and MACHINE entries.

CALLBACK

The CALLBACK option is used in LOGNAME entries to specify that no transaction will take place until the calling system is called back. There are two examples of when you would use CALLBACK. From a security standpoint, if you call back a machine you can be sure it is the machine it says it is. If you are doing long data transmissions, you can choose the machine that will be billed for the longer call.

The string:

```
CALLBACK=yes
```

specifies that your computer must call the remote computer back before any file transfers will take place.

The default is:

```
CALLBACK=no
```

The CALLBACK option is very rarely used. Note that if two sites have this option set for each other, a conversation will never get started.

COMMANDS

The COMMANDS option can be hazardous to the security of your system. Use it with extreme care.

The **uux** program will generate remote execution requests and queue them to be transferred to the remote computer. Files and a command are sent to the target computer for remote execution. The COMMANDS option ca be used in MACHINE entries to specify the commands that a remote computer can execute on your computer. Note that COMMANDS is not used in a LOGNAME entry; COMMANDS i MACHINE entries define command permissions whether we call the remote system or it calls us.

The string:

```
COMMANDS=rmail
```

indicates the default commands that a remote computer can execute on your computer. If a command string is used in a MACHINE entry, the default commands are overridden. For instance, the entry:

```
MACHINE=owl:raven:hawk:dove \
COMMANDS=rmail:rnews:lp
```

overrides the COMMAND default so that the computers owl, raven, hawk, and dove can nov execute **rmail**, **rnews**, and **lp** on your computer.

In addition to the names as specified above, there can be full path names of commands. For example,

```
COMMANDS=rmail:/usr/bin/rnews:/usr/local/lp
```

specifies that command **rmail** uses the default path. The default paths for your computer ar /bin and /usr/bin. When the remote computer specifies **rnews** or **/usr/bin/rnews** for the command to be executed, **/usr/bin/rnews** will be

executed regardless of the default path. Likewise, **/usr/local/lp** is the **lp** command that will be executed.

Including the ALL value in the list means that any command from the remote computer(s) specified in the entry can be executed. If you use this value, you give the remote computer full access to your computer. BE CAREFUL. This allows far more access than normal users have.

The string:

```
COMMANDS=/usr/bin/rnews:ALL:/usr/local/lp
```

illustrates two points: The ALL value can appear anywhere in the string, and the path names specified for **rnews** and **lp** will be used (instead of the default) if the requested command does not contain the full path names for **rnews** or **lp.**

The VALIDATE option should be used with the COMMANDS option whenever potentially dangerous commands like **cat** and **uucp** are specified with the COMMANDS option. Any command that reads or writes files is potentially dangerous to local security when executed by the UUCP remote execution daemon (**uuxqt**).

VALIDATE The VALIDATE option is used in conjunction with the COMMANDS option when specifying commands that are potentially dangerous to your computer's security. It is used to provide a certain degree of verification of the caller's identity. The use of the VALIDATE option requires that privileged computers have a unique login/password for UUCP transactions. An important aspect of this validation is that the login/password associated with this entry be protected. If an outsider gets that information, that particular VALIDATE option can no longer be considered secure. (VALIDATE is merely an added level of security on top of the COMMANDS option, though it is a more secure way to open command access than ALL.)

Careful consideration should be given to providing a remote computer with a privileged login and password for UUCP transactions. Giving a remote computer a special login and password with file access and remote execution capability is like giving anyone on that computer a normal login and password on your computer. Therefore, if you cannot trust someone on the remote computer, do not provide that computer with a privileged login and password.

LOGNAME

The LOGNAME entry:

```
LOGNAME=uucpfriend  \
VALIDATE=eagle:owl:hawk
```

specifies that if one of the remote computers that claims to be eagle, owl, or hawk logs in on your computer, it must have used the login **uucpfriend.** As can be seen, if an outsider gets the **uucpfriend** login/password, masquerading is trivial.

But what does this have to do with the COMMANDS option, which only appears in MACHINE entries? It links the MACHINE entry (and COMMANDS option) with a LOGNAME entry associated with a privileged login. This link is needed because the execution daemon is not running while the remote computer is logged in. In fact, it is an asynchronous process with no knowledge of which computer sent the execution request. Therefore, the real question is how does your computer know where the execution files came from?

Each remote computer has its own "spool" directory on your computer. These spool directories have write permission given only to the UUCP programs. The execution files from the remote computer are put in its spool directory after being transferred to your computer.

When **uuxqt** daemon runs, it can use the spool directory name to find the MACHINE entry in the permissions file and get the COMMANDS list, or if the computer name does not appear in the permissions file, the default list is used.

The following example shows the relationship between the MACHINE and LOGNAME entries:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
COMMANDS=rmail:/usr/bin/rnews:ALL \
READ=/ WRITE=/

LOGNAME=uucpz VALIDATE=eagle:owl:hawk \
REQUEST=yes SENDFILES=yes \
READ=/ WRITE=/
```

These entries provide unlimited read, write, and command execution for the remote machines eagle, owl, and hawk. The ALL value in the COMMANDS option means that

any commnd can be executed by either of these machines. Using the ALL value gives the remote machine unlimited access to your computer. In fact, files that are only readable or writable by user "uucp" (like Systems or Devices) can be accessed using commands like **vi**. This means that a user on one of the privileged machines can write in the Systems file as well as read it!

In the first entry, you must make the assumption that when you want to call one of the computers listed, you are really calling either eagle, owl, or hawk. Therefore, any files put into one of the eagle, owl, or hawk spool directories is put there by one of those computers. If a remote computer logs in and says that it is one of these three computers, its execution files will also be put in the privileged spool directory. You therefore have to validate that the computer has the privileged login **uucpz**.

Machine Entry for Other Systems

You may want to specify different option values for the computers your computer calls that are not mentioned in specific MACHINE entries. This may occur when there are many computers calling in, and the command set changes from time to time. The name "OTHER" for the computer name is used for this entry as shown below:

```
MACHINE=OTHER \
COMMANDS=rmail:rnews:/usr/bin/Photo:/usr/bin/xp
```

All other options available for the MACHINE entry may also be set for the computers that are not mentioned in other MACHINE entries.

Combining MACHINE and LOGNAME Entries

It is possible to combine MACHINE and LOGNAME entries into a single entry where the common options are the same. For example, the entries

```
MACHINE=eagle:owl:hawk REQUEST=yes \
READ=/ WRITE=/

LOGNAME=uucpz REQUEST=yes SENDFILES=yes \
READ=/ WRITE=/
```

share the same REQUEST, READ, and WRITE options. These two entries can be merged as shown below:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
LOGNAME=uucpz SENDFILES=yes \
READ=/ WRITE=/
```

Polling

In addition, you may wish to poll certain machines at regular intervals.

The Poll file (/usr/lib/uucp/Poll) contains information for polling remote computers. Each entry in the Poll file contains the name of a remote computer to call, followed by a tab character (a space won't work), and finally the hours the computer should be called.

The format of entries in the Poll file are:

sys-name hour...

For example the entry:

```
eagle  0 4 8 12 16 20
```

will provide polling of computer eagle every four hours.

The uudemon.poll script does not actually perform the poll. It merely sets up a polling work file (always named C.*file*), in the spool directory that will be seen by **uusched**.

Create a Dial-out Site

You can create a dial-out site by installing the dialing information needed by your system to call and log in to other computers. This information consists of the following:

- Dialing abbreviations for remote computers in the /usr/lib/uucp/Dialcodes file.
- Information about logins on remote computers in the /usr/lib/uucp/Systems file.
- A transmission schedule in the form of a shell script to be called periodically by the **cron** program.

You can create this information by using a text editor and modifying or creating the appropriate files. The following sections explain the required format of the information.

Once the information is installed, you can enable the system for calling other computers by creating a dial-out line. See the section "Creating a Dial-Out Line" given earlier in this appendix.

Create the Dialcodes File

The /usr/lib/uucp/Dialcodes file defines abbreviations for often used telephone prefixes and area codes. You may use these abbreviations in the Systems file for the phone numbers of remote sites.

The Dialcodes file may contain one or more entries of the form:

abbreviation dial-sequence

where *abbreviation* is any combination of letters and digits that begins with a letter, and *dial-sequence* is any combination of digits that represents a telephone prefix, area code or any other part of a telephone number. For example, the entry:

ms 555

defines the abbreviation "ms" to be the telephone prefix "555."

Create the Systems File

The /usr/lib/uucp/Systems file defines the names, telephone numbers, and login information of all sites in the system. The file contains one or more entries of the form:

System-Name Time Type Class Phone Login

Each of these fields is defined in the following section.

System-name
: This field contains the node name of the remote computer.

Time This field is a string that indicates the day-of-week and time-of-day when the remote computer can be called. The day portion may be a list containing some of the following:

Su Mo Tu We Th Fr Sa for individual days

Wk for any week-day (Mo Tu We Th Fr)

Any for any day

Never for a passive arrangement with the remote computer. In this case, the computer will never initiate a call to the remote machine. The call must be initiated by the remote machine.

For example:

```
Wk 1700-0800SaSu
```

allows calls from 5:00 p.m. to 8:00 a.m., Monday through Friday, and calls any time Saturday and Sunday. The example would be an effective way to call only when phone rates are low, if immediate transfer is not critical.

The *time* portion should be a range of times such as 0800-1230. If no *time* portion is specified, any time of day is assumed to be allowed for the call. A time range that spans 0000 is permitted. For example, **0800-0600** means all times are allowed other than times between 6 a.m. and 8 a.m. An optional subfield is available to specify the minimum time (in minutes) before a retry, following a failed attempt. The default wait is 60 minutes. The subfield separator is a semicolon (;). For example, **Any;9** is interpreted as call any time, but wait at least 9 minutes before retrying after a failure occurs.

Type This field contains the device type that should be used to establish the communication link to the remote computer. The Devices file is searched for the device type listed, and the device found is used to establish the connection (if available). The following keywords may appear in this field:

ACU This keyword indicates that the link to a remote computer is made through an automatic call unit (automatic dial modem). This modem may be connected either directly to the computer or indirectly through a LAN switch.

Network
This keyword indicates that the link is established through a LAN switch where *Network* is replaced with either **micom** or **develcon.** These two switches are the only ones that contain caller scripts in the Dialers file (discussed in this appendix). Other switches may be used if caller scripts are constructed and placed in the Dialers file.

System-Name
This keyword indicates a direct link to a particular machine where *System-Name* is replaced by the name of the particular computer (should be the same as the first field).

The keyword used in this field is matched against the first field of Devices file entries as shown below:

```
Systems: eagle Any ACU D1200 3-2-5-1 ogin: nuucp \
         ssword: Oakgrass

Devices:  ACU contty - D1200 hayes
```

Class This field is used to indicate the transfer speed of the device used in establishing the communication link. It may contain a letter and speed (for example, C1200, D1200) to differentiate between classes of dialers (refer to the discussion on the **Devices** file, *Class* field). Some devices can be used at any speed, so the keywor **Any** may be used. This field must match the *Class* field in the associated **Devices** file entry.

```
Systems: eagle Any ACU D1200 3-2-5-1 ogin: nuucp \
         ssword: Oakgrass

Devices:  ACU contty - D1200 hayes
```

Phone This field is used to provide the phone number (token) of the remote computer for automatic dialers (LAN switches). The phone number is made up of an optional alphabetic abbreviation and a numeric part. The abbreviation must be one that is listed in the Dialcodes file. In this string, an equal sign (=) tells the ACU to wait for a secondary dial tone before dialing the remaining digits. A dash in the string (-) instructs the ACU to pause 4 seconds before dialing the next digit.

If your computer is connected to a LAN switch, you may access other computers that are connected to that switch. The Systems file entries for these computers will not have a phone number in the *Phone* field. Instead, it will contain the token that must be passed on to the switch so it will know which computer your computer wishes to communicate with. The associated Devices file entry should have a **\D** at the end of the entry to ensure that this field is not translated using the Dialcodes file.

Login: This field contains login information given as a series of fields and subfields of the format:

expect send

where *expect* is the string that is received and *send* is the string that is sent when the *expect* string is received.

The *expect* field may be made up of subfields of the form:

expect[*-send-expect*]...

where the *send* is sent if the prior *expect* is not successfully read and the *expect* following the *send* is the next expected string. For example, with **login--login, uucp** will expect

login. If **uucp** gets **login,** it will go on to the next field. If it does not get **login,** it will send nothing followed by a new line, then look for **login** again. If no characters are initially expected from the remote computer, the characters "" (null string) should be used in the first *expect* field. Note that all *send* fields will be sent followed by a new-line unless the *send* string is terminated with a **\c.**

There are several escape characters that cause specific actions when they are a part of a string sent during the login sequence. The following escape characters are useful in **uucp** communications:

\N Send a null character.

\b Send a backspace character.

\c If at the end of a string, suppress the new-line that is normally sent. Ignored otherwise.

\d Delay two seconds before sending or reading more characters.

\p Pause for approximately 1/4 to 1/2 second.

\n Send a new-line character.

\r Send a carriage-return.

\s Send a space character.

\t Send a tab character.

\\ Send a \ character.

EOT Send EOT character (a new-line sent twice).

BREAK Send a break character (ASCII NUL).

ddd Collapse the octal digits *(ddd)* into a single character.

Create a Transmission Schedule

In the uucp system, the **uucico** program carries out all transmissions between your site and other sites, sending and receiving files and commands as long as there is work for it to do. On a dial-in site, **uucico** is always started whenever a calling site logs in, but on a dial-out site, **uucico** is only started when an explicit invocation of the program is given. This means you must periodically invoke the program on a dial-out site to ensure that all transmissions requested by the **uucp** and **uux** programs are completed. You can do this in one of two ways: invoke the program manually whenever you need it, or create a shell script and let the **cron** program invoke **uucico** automatically according to a schedule of transmissions.

The most convenient method is to let **cron** invoke **uucico** for you. This is usually done by creating a crontab entry to invoke the uudemon.hour shell script once an hour:

minutes hour day month day-of-week command-line

where *minutes*, *hour*, *day*, *month*, and *day-of-week* give the exact day of the year and time of day to execute the given command-line. Each item, except the command-line, must be an integer number within an acceptable range, e.g., 0 to 59 for minutes. A sequence of values for one item may be given by separating the values with commas. Also, an asterisk (*) may be given to represent all acceptable values. The command-line is the name of the shell script, /usr/lib/uucp/uudemon.hour.

You can add an entry to the crontab file by using a text editor. For more information about the file, see **cron**(C) and **crontab**(C) in the *Reference (C)*. For example, the entry:

```
15 * * * * /usr/lib/uucp/uudemon.hour >/dev/null
```

invokes the shell script at 15 minutes past the hour.

The uudemon.hour shell script then calls **uusched** to execute **uucico** for any pending jobs.

MAINTAINING THE SYSTEM

This section explains how to maintain the uucp system. In particular, it explains how to display the contents of uucp log files, how to remove old requests and files from the spool directories, and how to solve some common problems.

Displaying and Merging Log Files

You can display a record of the transmissions requested and completed to a given site by using the **uulog** command. The log files contain information about queued requests, calls to remote sites, execution of **uux** commands, and file copy results. The command has the form:

uulog -s*sitename*

where **-s***sitename* names the site whose log files are to be displayed. If you do not give a sitename, log files for all sites are displayed.

Cleaning the UUCP Spool Directory

You can remove unwanted uucp system files from the uucp spool directory by using the **uucleanup** command. The command removes temporary data, log, system status, and lock files from the spool directory if they are more than a given number of hours old. The command has the form:

/usr/lib/uucp/uucleanup -C*time* **-D***time* **-X***time* **-o***time* **-ss**

where -C, -D, and -X remove C., D., and X. files that are more than *time* days old. The **-o** option specifies the time for all other files in the spool directories, and **-s***system* limits the action to the spool directory for a particular system.

The **uucleanup** program should be run once a day, once a week, or whatever the system requires. You can invoke it automatically by using a system daemon such as **cron**. Typically, **uucleanup** is invoked by uudemon.cleanup, which is run a regular intervals by **cron**.

The command:

uucleanup -D3

removes all temporary data files that are at least three days old.

The **uucleanup** command may also be run as needed to remove unwanted files after a system crash or an aborted uucp program.

Reclaiming Data Files After a Crash

You can check the status of files transmitted from a remote site and possibly reclaim some or all of the data lost during an aborted transmission by examining system data files. The data files contain the contents of files copied from remote sites. These files are temporarily kept in subdirectories of the /usr/spool/uucp directory; their names have the form:

X.*id*
C.*id*
D.*id*

where *id* is composed of the machine name, job priority, and a sequence number.

The temporary data files are normally moved to the requested destination immediately after the transmission has finished. However, if a transmission has failed or the system has crashed, the file remains in the spool directory. You can examine the contents of this file with the **cat** command. If desired, you can reclaim the file by moving it to a new location with the **mv** command. Leftover data files that cannot be reclaimed should be removed using the **uuclean** command.

Checking the Transmission Status

You can check the status of transmissions between sites in the uucp system by the **uustat** command. **Uustat** displays information about login, dialup, or sequence check failure, as well as the talking status when two machines are conversing.

Uustat has the following options:

-a Shows all jobs currently queued.

-k *job* Removes a job from the queue.

-m Reports the accessiblity of all machines.

-p Executes a **ps -flp** for all process IDs that are listed in lock files.

-q Lists in detail the jobs queued for each machine.

-r *job* Rejuvenates an old job.

-s *system*
Reports the status of **uucp** requests for the specified *system*.

-u *user* Reports the status of **uucp** requests for the specified *user*.

Checking for Locked Sites or Devices

You can make sure the uucp system is not intentionally preventing transmissions to a given site or through a given device by examining the system lock files. The uucp system creates a lock file for each site being called and for each device being used to call a site. Lock files prevent the uucp system from attempting to duplicate conversations with a given site, or from placing multiple calls on the same device. The lock files are kept in the /usr/spool/locks directory and their names have the form:

LCK..*str*

where *str* is either a site name or the name of the calling device.

Since lock files prevent all calls to a given site or through a given device, it is wise to make sure no unnecessary lock files are left in the directory. If a transmission has been aborted or the system has crashed, the lock files will prevent subsequent transmissions for about 24 hours. If you wish to place a call before this time, you must remove the file using the **uuclean** command.

DETAILS OF OPERATION

This section describes the details of uucp system program operation. It explains the processes used to create system communication and defines the files used to support the system.

UUCP Programs

The uucp system consists of four primary and two secondary programs. The primary programs are:

uucp This program creates work and gathers data files in the spool directory for the transmission of files.

uux This program creates work and execute files, and gathers data files for the remote execution of commands.

uucico This program executes the work files for data transmission.

uuxqt This program executes commands found in execution files.

The secondary programs are:

uulog This program reports on the status of uucp requests.

uucleanup
This program removes old files from the spool directory.

UUCP Directories and Files

During execution of the **uucp** programs, the uucp system uses files from the following three directories:

/usr/lib/uucp	This is the directory used for uucp syssystem files and all executable programs programs other than **uucp** and **uux.**
/usr/spool/uucp	This is the spool directory used during uucp execution.
/usr/spool/uucp/.Xqtdir	This directory is used during execution of execute files.

Files are created in a spool directory for processing by the uucp daemons. There are three types of files used for the execution of work:

- Data files — Contain data for transfer to remote sites
- Work files — Contain directions for file transfers between sites
- Execution files — Contain directions for command executions which involve the resources of one or more sites

UUCP Site to Site File Copy

The **uucp** program is the user's primary interface with the system. The **uucp** program was designed to look like the cp command. The syntax is:

uucp [*option ...*] *source ... destination*

where *source* and *destination* may contain the prefix *sitename*! which indicates the site on which the file or files reside, or where they will be copied.

The options interpreted by **uucp** are:

-d Make directories when necessary for copying the file (default).

-c Don't copy local file to the spool directory, but use the specified source for transfer.

-C Force the copy of local files to the spool directory for transfer.

-f Don't make intermediate directories for the file copy.

-g*grade* *Grade* is a single letter/number; lower ASCII sequence characters will cause the job to be transmitted earlier during a particular conversation.

-j Output the job identification ASCII string on the standard output. This job identification can be used by **uustat** to obtain the status or terminate a job.

-m Send mail to the requester on completion of the work.

-n*user* Notify *user* on the remote system that a file was sent.

-r Do not start the file transfer, just queue the job.

The following options are used primarily for debugging:

-s*file* Report status of the transfer to *file*. Note that *file* must be a full pathname.

-x*debug_level*
Produce debugging output on standard output. The *debug_level* is a number between 0 and 9; higher numbers give more detailed information.

The destination may be a directory name, in which case the file name is taken from the last part of the source's name. The source name may contain special shell characters such as "?[]." If a source argument has a *sitename*!

prefix for a remote site, the file name expansion will be done on the remote site.

The command:

uucp *filename* **chicago!/usr/dan**

will set up the transfer of the *filename* you specify to the /usr/dan directory on the chicago machine.

NOTE

If you execute **uucp** commands in the C-shell, escape the exclamation point (!).

The source and/or destination names may also contain a user prefix. This translates to the login directory on the specified site. The **uucp** program will only accept full pathnames. File names with "../" are not permitted.

The command:

uucp chicago!~dan/*.h ~dan

will set up the transfer of files whose names end with .h in dan's login directory on site chicago to dan's local login directory. For each source file, the program will check the source and destination filenames and the site-part of each to classify the work into one of five types:

1. Copy source to destination on local site.

2. Receive files from other sites.

3. Send files to remote sites.

4. Send files from remote sites to another remote site.

5. Receive files from remote sites when the source contains special shell characters as mentioned above.

After the work has been set up in the spool directory, the **uucico** program must be started to try to contact the other machine to execute the work.

Copying Files to a Local Destination

A **cp** command is used to do type 1 work. The **-d** and the **-m** options are not honored in this case.

Receiving Files from Other Sites

For type 2 work, a one-line work file is created for each file requested, and is put in the spool directory with the following fields, each separated by a blank:

[1] R

[2] The full pathname of the source or a user/pathname. The user part will be expanded on the remote site.

[3] The full pathname of the destination file. If the *~user* notation is used, it will be immediately expanded to be the login directory for the user.

[4] The user's login name.

[5] A "-" followed by an option list. (Only the **-m** and **-d** options will appear in this list.)

Sending Files to Remote Sites

For type 3 work, a work file is created for each source file and the source file is copied into a data file in the spool directory. (A **-c** option on the **uucp** program will prevent the data file from being made. In this case, the file will be transmitted from the indicated source.) Pathnames are checked using the Permissions file to verify access to the requested directory. The fields of each entry are given below.

[1] S

[2] The full pathname of the source file.

[3] The full pathname of the destination or user/filename.

[4] The user's login name.

[5] A "-" followed by an option list.

[6] The name of the data file in the spool directory.

[7] The file mode bits of the source file in octal print format (e.g., 0666).

Copying Files Between Sites

For type 4 and 5 work, **uucp** generates a **uucico** command line and sends it to the remote machine; the remote **uucico** executes the command line.

UUX - Site To Site Execution

The **uux** command is used to set up the execution of a command where the execution machine and/or some of the files are remote. The syntax of the **uux** command is:

uux [-] [*options*] ... *command-string*

where *command-string* is made up of one or more arguments All special shell characters such as "<>|^" must be quoted either by quoting the entire command string, or by quoting the character as a separate argument. Within the command string, the command and file names may contain a *sitename*! prefix.

The - option is used to indicate that the standard input for the given command should be inherited from the standard input of the **uux** command. The only option that is essential for debugging is -x*num*. X*num* directs the command to use *num* as the level of debugging output.

The command:

pr abc | uux - chicago!rmail joe

will set up the output of **pr abc** as standard input to a **mail** command to be executed on site chicago.

Uux generates an execute file which contains the names of the files required for execution (including standard input), the user's login name, the destination of the standard output, and the command to be executed. This file is

either put in the spool directory for local execution or sent to the remote site using a generated send command (type 3 above).

For required files which are not on the execution machine, **uux** will generate receive command files (type 2 above). These command-files will be put on the execution machine and executed by the **uucico** program. (This will work only if the local site has permission to put files in the remote spool directory as controlled by the remote Permissions file.)

The execute file will be processed by the **uuxqt** program on the execution machine. It is made up of several lines, each of which contains an identification character and one or more arguments. The order of the lines in the file is not relevant and some of the lines may not be present. Each line is described below.

User Line

U *user site*

where the *user* and *site* are the requestor's login name and site.

Required File Line

F *filename real-name*

where the *filename* is the generated name of a file for the execute machine and *real-name* is the last part of the actual file name (contains no path information). Zero or more of these lines may be present in the execute file. The **uuxqt** program will check for the existence of all required files before the command is executed.

Standard Input Line

I *filename*

The standard input is either specified by a "<" in the commandstring or inherited from the standard input of the **uux** command if the - option is used. If a standard input is not specified, /dev/null is used.

Standard Output Line

O *filename sitename*

The standard output is specified by a ">" within the command-string. If a standard output is not specified, /dev/null is used. (Note that the use of ">>" is not implemented.)

Command Line

C *command* [*arguments*] ...

The arguments are those specified in the command string. The standard input and standard output will not appear on this line. All required files are moved to the execution directory (a subdirectory of the spool directory) and the command is executed using the shell. In addition, a shell PATH statement is prepended to the command line as specified in the **uuxqt** program.

After execution, the standard output is copied or set up to be sent to the proper place.

UUCICO - Copy In, Copy Out

The **uucico** program will perform the following major functions:

- Scan the spool directory for work.
- Place a call to a remote site.
- Negotiate a line protocol to be used.

- Execute all requests from both sites.
- Log work requests and work completions.

Uucico may be started by a system daemon, by the user (this is usually for testing), or by a remote site. (The **uucico** program should be specified as the shell field in the /etc/passwd file for the uucp logins.)

When started with the **-rl** option, the program is considered to be in MASTER mode. In this mode, a connection will be made to a remote site. If started by a remote site, the program is considered to be in SLAVE mode. When using **uucico** in MASTER mode, you must also specify a site name with the **-s** option.

The **uucico** program must generally be started directly by the user or by another program, such as a shell script invoked by **cron**. There are several options used for execution:

-rl Start the program in MASTER mode. This is used when **uucico** is started by a program or **cron** shell. You must also use the -s option when using **-rl**.

-s*sitename*
Do work only for site *sitename*. If -s is specified, a call to the specified site will be made even if there is no work for site *sitename* in the spool directory, but will only call when times in the Systems file permit it. This is useful for polling sites which do not have the hardware to initiate a connection.

The following options are used primarily for debugging:

-d*dir* Use directory *dir* for the spool directory.

-x*num* Use *num* as the level of debugging output.

The next part of this section describes the major steps within the **uucico** program.

Scanning for Work

The names of the work related files in the spool directory have the format:

type . sitename grade number

where *type* may be "C" for copy command file, "D" for data file, "X" for execute file, *sitename* is the remote site, *grade* is a character, and *number* is a four digit, padded sequence number.

The file:

C.res45n0031

is a work file for a file transfer between the local machine and the "res45" machine.

The scan for work is done by looking through the spool directory for work files (files with prefix "C"). **Uucico** calls the site specified by the **-s** option and processes the corresponding work files.

Calling a Remote Site

The call is made using information from several files which reside in the **uucp** program directory. At the start of the call process, a lock is set to forbid multiple conversations between the same two sites. The lock filename has the form:

LCK..*str*

where *str* is the device name. The file is in the /usr/spool/uucp directory.

The site name is found in the Systems file. The information contained for each site is:

[1] Site name

[2] Times to call the site (days-of-week and time-of-day)

[3] Device or device type to be used for call

[4] Line speed

[5] Phone number if field [3] is "ACU" or the device name (same as field [3]) if not

[6] Login information (multiple fields)

The time field is checked against the present time to see if the call should be made.

The phone number may contain abbreviations (e.g., mh, py, boston) which get translated into dial sequences using the Dialcodes file.

The Devices file is scanned using device type and line speed fields from the Systems file to find an available device for the call. The program will try all devices that satisfy these fields until the call is made, or until all devices have been tried. If a device is successfully opened, a lock file is created so that another copy of **uucico** will not try to use it. If the call is complete, the login information in the last field of Systems is used to log in.

The conversation between the two **uucico** programs begins with a handshake started by the SLAVE site. The SLAVE sends a message to let the MASTER know it is ready to receive the site identification and conversation sequence number. The response from the MASTER is verified by the SLAVE and if acceptable, protocol selection begins. The SLAVE can also reply with a call-back required message in which case, the current conversation is terminated.

Selecting Line Protocol

The remote site sends a message:

```
Pproto-list
```

where *proto-list* is a string of characters, each representing a line protocol.

The calling program checks the protocol list for a letter corresponding to an available line protocol and returns a use protocol message. The message has the form:

```
Ucode
```

where *code* is either a one character protocol letter or "N" which means there is no common protocol.

Processing Work

The initial role of MASTER or SLAVE for the work processing is the mode in which each program starts. (The MASTER has been specified by the **-rl** option.) The MASTER program does a work search similar to the one used in the section "Scanning For Work" above. There are five messages used during the work processing, each specified by the first character of the message. They are:

S Send a file

R Receive a file

C Copy complete

X Execute a uucp command

H Hangup

The MASTER will send "R," "S," or "X" messages until all work from the spool directory is complete, at which point an "H" message is sent. The SLAVE will reply with the first letter of the request and either the letter "Y" or "N" for yes or no. For example, the message "SY" indicates that it is okay to send a file.

The send and receive replies are based on permission to access the requested file/directory using the Permissions file and read/write permissions of the file/directory. After each file is copied into the spool directory of the receiving site, a copy-complete message is sent by the receiver of the file. The message "CY" is sent if the file has successfully been moved from the temporary spool file to the actual destination. Otherwise, a "CN" message is sent. (In the case of "CN," the transferred file will be in the spool directory with a name beginning with "TM.") The requests and results are logged on both sites.

The hangup response is determined by the SLAVE program by a work scan of the spool directory. If work for the remote site exists in the SLAVE's spool directory, an "HN" message is sent and the programs switch roles. If no work exists, an "HY" response is sent.

Terminating a Conversation

When an "HY" message is received by the MASTER, it is echoed back to the SLAVE and the protocols are turned off. Each program sends a final "OO" message to the other. The original SLAVE program will clean up and terminate.

UUXQT - UUCP Command Execution

The **uuxqt** program is used to process execute files generated by **uux**. The **uuxqt** program is started by the **uucico** program. The program scans the spool directory for execute files (prefix X.). Each one is checked to see if all the required files are available and if so, the command line or send line is executed.

The execute file is described in the section "UUX - Site to Site Execution" above.

The execution is accomplished by executing the shell command:

sh -c *command*

with the command line, after appropriate standard input and standard output have been opened. If a standard output is specified, the program creates a send command or copies the output file, as appropriate.

Security

In an unrestricted uucp system, once a user logs in to another site through the uucp system, the user can execute any commands and copy any files normally accessible to the uucp login. It is up to the individual sites to be aware of this and apply the protections that they feel are necessary to prevent unauthorized use of files and commands.

The uucp system does provide a certain level of security. For example, a calling site does not get a standard shell when it logs in. Instead, the **uucico** program is started and all work is done through this program. The **uucico** program checks the pathnames of files to be sent or received, to prevent access to restricted directories. The Permissions file supplies the information for these checks. To prevent execution of possibly damaging commands, the **uuxqt** program can only execute the **rmail** program on a remote site. This special program is one of many underlying **mail** programs that help deliver mail. Finally, the Systems file is owned by **uucp** and has mode 0400 to protect the phone numbers and login information for remote sites.

CONFIGURING THE DIALERS FILE

This section explains how to configure your system for different autodial modems using the Dialers file.

Dialers File

The Dialers file (/usr/lib/uucp/Dialers) specifies the initial conversation that must take place on a line before it can be made available for transferring data. This initial handshaking is usually a sequence of ASCII strings that is transmitted and expected, and it is often used to dial a phone number using an ASCII dialer. The fifth field in a Devices file entry is an index into the Dialers file or a special dialer type (Hayes, TLI, or TLIS). Here an attempt is made to match the Devices field with the first field of each Dialers file entry. In addition, each odd numbered Devices field starting with the seventh position is used as an index into the Dialers file.

If the match succeeds, the Dialers entry is interpreted to perform the dialer negotiations. The first field matches the fifth and additional odd numbered fields in the Devices file. The second field is used as a translate string: the first of each pair of characters is mapped to the second character in the pair. This is usually used to translate = and - into whatever the dialer requires for "wait for dialtone" and "pause."

The remaining *expect-send* fields are character strings. Below are some character strings distributed with the Basic Networking Utilities in the Dialers file.

```
penril =W-P "" \d > Q\c : \d- > s\p9\c )-W\p\r\ds\p9\c-)
 y\c : \E\TP > 9\c OK
ventel =&-% "" \r\p\c $ <K\T%%\r>\c ONLINE!
hayes =,-, "" \dAT\r\c OK\r \EATDT\T\r\c CONNECT
rixon =&-% "" \d\r\r\c $ s9\c )-W\r\ds9\c-) s\c : \T\r\c $ 9\c LINE
vadiac =K-K "" \005\p  *-\005\p-*\005\p-* D\p BER? \E\T\e \r\c\LINE
develcon "" "" \pr\ps\c est:\007 \E\D\e \007
micom ""  "" \s\c NAME? \D\r\c GO
direct
```

The meaning of some of the escape characters (those beginning with "\") used in the Dialers file are listed below:

\p Pause (approximately 1/4 to 1/2 second)

\d Delay (approximately 2 seconds)

\D Phone number or token without Dialcodes translation

\T Phone number or token with Dialcodes translation

\K Insert a BREAK

\E Enable echo checking (for slow devices)

\e Disable echo checking

\r Carriage return

\c No new-line or carriage return

\n Send new-line

nnn Send octal number

\M Set CLOCAL bit

\m Clear CLOCAL bit

Additional escape characters that may be used are listed in the section discussing the Systems file.

The penril entry in the Dialers file is executed as follows. First, the phone number argument is translated, replacing any = with a W (wait for dialtone) and replacing any - with a P (pause). The handshake given by the remainder of the line works as follows:

"" Wait for nothing

\d Delay for 2 seconds

> Wait for a >

s\p9\c
Send an s, pause for 1/2 second, send a 9, send no terminating newline

)-W\p\r\ds\p9\c-)
Wait for a). If it is not received, process the string between the - characters as follows. Send a W, pause, send a carriage-return, delay, send an s, send a 9, without a newline, and then wait for the).

y\c Send a y

:	Wait for a:
\E\TP	Enable echo checking. (From this point on, whenever a character is transmitted, it will wait for the character to be received before doing anything else.) Then, send the phone number followed by a pause character (P). The \T means take the phone number passed as an argument and apply the Dialcodes translation and the modem function translation specified by field 2 of this entry.
>	Wait for a >
9\c	Send a 9 without a newline
OK	Waiting for the string OK

(BLANK)

Appendix C
The Print Spooler

INTRODUCTION

This appendix describes the **lp** and **lpr** print spoolers. Printer spooler error messages are in Appendix F. **Lp** is the standard print spooler and **lpr** is the Altos print spooler. Also see the *Reference (C)* for information about these spoolers.

This section tells you about:

- How the LP Spooling system works
- Sending a print request
- Using lp on a network
- Checking the status of a print request
- Canceling a print request
- The commands used to administer the system
- Printer interface programs
- LP Spooler files and directories

THE LP PRINT SPOOLER

LP spooling means that you can send a file to be printed (LP originally stood for Line Printer, but has come to include many other types of printing devices), while you continue with other work. The LP Spooling system is software that:

- Handles the task of receiving files users want printed
- Schedules the work of one or more printers
- Starts programs that interface with the printer(s)
- Keeps track of the status of jobs

- Issues error messages when problems arise

LP Spooler user commands are shown in Table C-1.

Table C-1. User Commands for the LP Spooling System

Command	Description
lpenable(C)	Activate the named printer(s).
cancel(C)	Cancel a request for a file to be printed.
lpdisable(C)	Deactivate the named printer(s).
lp(C)	Send a file or files to a printer.
lpstat(C)	Print the status of the LP system.

In addition to being able to send requests to the LP Spooling system, check the status of requests, and cancel requests, you can disable and enable a printer. The idea is that if you find a printer is malfunctioning in some way, it should not be necessary to call the administrator to turn the printer off.

The **/usr/lib/lpinit** shell script prompts you through the configuration/setup process of a line printer (see **lpinit**(M)).

Sending a Print Request

Print requests are sent to the line printer with the **lp** command. The **lp** program creates a unique id for each print request sent. That id is then used to identify the file when you are checking the status of queued print jobs or canceling a print request.

The syntax for using the **lp** command is:

$ **lp** [**-c**] [**-d***dest*] [**-m**] [**-n***number*] [**-o***option*] [**-s**] [**-t***title*] [**-w**] *files* Retn

You won't use all of these options every time you use the **lp** command, but you will find that the various combinations of options available make **lp** a versatile printing tool.

These options are:

-c Makes a copy of the file(s) to be printed. If you do not use this option, you cannot remove any file that is queued for printing until the print job is complete.

d*dest* Specifies the printer or class of printers where the file(s) will be printed. This option lets you specify a printer other than your default printer. Destination names vary between systems (if the value for *dest* is set, it resides in the environment variable LPDEST).

-m Sends you mail after the file(s) have been printed.

-n*number* Specifies the *number* of copies you want printed (if you don't use this option, only one copy will print).

-o*option* Specifies printer-dependent options, such as baud rate, parity, etc.

-s Suppresses messages coming to your terminal from the **lp** program, such as the message telling you the request id of the print job.

Typically, you won't use this option because you will want to know the print request id and make a note of it for later use with the **lpstat** or **cancel** commands.

-t*title* Prints the *title* you specify on the banner page of the printed file.

-w Writes a message to your terminal telling you the file(s) have been printed. If you are not logged in, mail will be sent instead.

For example, to print two copies of a file named holiday on the printer named spot, and have **lp** send you a message when the printing is finished, type:

lp -dspot -n2 -w holiday Retn

Using lp on a Network

To send a print request to a printer connected to another computer in your network, use the **-d** option with the name of the appropriate printer. If you don't know the name of the printer configured for this purpose, check with your system administrator.

The **lp** program will then dispatch the print request to the appropriate printer device.

Checking the Status of a Print Request

To check the status of a print request, type:

lpstat Retn

The system responds by displaying the following information about queued print requests:

```
canon1-300  eaa 80928   Oct 17  16:50
canon1-301  wmf  9025   Oct 17  16:59
```

In the above sample display, canon1 is the printer name; 300 and 301 are print request id numbers; eaa and wmf are the logged-in user names of the people printing the files; the numbers in the third column tell the size of the queued file in bytes; and the last two columns tell the date and time the print jobs were queued for printing.

There are several options you can use with the **lpstat** command to control what information is displayed. They are:

-a[*list*] Prints the acceptance status of the printer destination. The argument *list* lets you specify printer names and/or printer classes you want to know about; **lpstat** will then list only those acceptance statuses.

-c[*list*] Prints printer class names and the printers under them. The argument *list* lets you specify printer class names you want to know about; **lpstat** will then list only those names.

-d Prints the system default printer destination for **lp**.

-o[*list*] Prints the status of output requests. The argument *list* lets you specify the printer names, printer class names, and request ids you want to know about; **lpstat** will then list only those names.

-r Prints the status of the **lp** request scheduler.

-s Print a status summary with information about the status status of the line printer scheduler, the system default destination, a list of printer class names and printers in that class, and a list of printers and their associated devices.

-t Prints all status information.

-u[*list*] Prints the status of output requests for the user names specified in *list*.

-v[*list*] Prints the names of available printers and the path names of the devices associated with them. The *list* argument lets you specify the user names you want to know about.

Canceling a Print Request

If you decide you do not want a print request to print after all, type:

cancel *request id/printer name* Retn

You can cancel either a request id or a printer name. If you specify a request id, that print request will be cancelled even if it is currently printing. If you specify a printer name, the request that is currently printing on that printer will be cancelled; the next print request in the queue will then be printed.

Configuring a Printer

The system administrator uses the **lpadmin** command to define the names of printers used on your system and to associate each of those printer names with a model file.

The **lp** program comes with model files that define several types of printers, as well as providing for printing over a network and printing through **lpr.** If your system is using both the **lp** and **lpr** print spoolers to print on the same printer, **lpr** calls the **lp** program (see the *Release Notes*).

When you send a print request, you need only specify the name of the printer. The **lp** program then dispatches the print request to the appropriate printer device according to the information in the model file associated with that printer name.

ADMINISTRATIVE COMMANDS

A separate set of commands available for the LP administrator is shown in Table C-2. These commands are found in the /usr/lib directory. If you expect to use them frequently, you might find it convenient to include that directory in your PATH variable. To use the administrative commands, you must be logged in as **root.**

Table C-2. Administrative Commands for the LP Spooling System

Command	Description
/usr/lib/**accept**(C)	Permit job requests to be queued for a specified destination.
/usr/lib/**reject**(C)	Prevent jobs from being queued for a specified destination. Described on the same manual page as **accept**(C).
/usr/lib/**lpadmin**(M)	Set up or change LP configuration.
/usr/lib/**lpmove**(M)	Move output requests from one destination to another. Described on the same manual page as **lpsched**(M).
/usr/lib/**lpsched**(M)	Start the LP scheduler.
/usr/lib/**lpshut**(M)	Stop the LP scheduler. Described on the same manual page as **lpsched**(M).

In the section that follows, we describe the commands in the order in which they are usually used (also see the *Reference (C)* and *Reference (M)*).

Command Descriptions and Examples

/usr/lib/lpadmin

The **lpadmin**(M) command is used to add a new printer to the system, assign classes of printers, name or remove a default destination, and specify interface programs to be used. **Lpadmin** may not be used when the LP scheduler, **lpsched**(M), is running, except when the **-d** option is specified.

One (and only one) of the following three options must always be included on the command line when you execute **lpadmin:**

-d*[dest]*

-x*dest*

-p*printer*

No other options are allowed with **-d** and **-x.** However, many arguments are allowed with the **-p***printer* option, and at least one argument must always be present. The **-p** option names a printer to which the other argument(s) applies. If *printer* does not exist, it is created. The arguments that can be used with **-p** are as follows:

-c*class* — This argument assigns the printer specified in the **-p** option to the specified *class.*

-e*printer* — This argument allows you to use an existing interface program for a new printer that you are adding to the LP system. When you select this argument, the interface program for the printer specified in this argument is copied for the printer specified in the **-p** option.

-h — When adding a new printer, this argument means that the printer is hard-wired.

-i*interface* — Use this argument to specify a new interface program for the printer specified in the **-p** option. *Interface* is the path name of the new program.

-l — When adding a new printer, this argument means that the device associated with the printer is a login terminal.

-m*model* — Several *model* interface programs are supplied with the LP Spooling Utilities. Use this argument to select the model interface program that you want to use with the printer you are adding to the LP system.

-r*class* Use this argument to remove a printer from a class.

-v*device* This argument must be used when you add a new printer to the LP system. It associates the printer with the system file specified by *device.* The compute path name must be given for the file.

The **-d***[dest]* option is used to define an existing system destination as the new system default destination. If *dest* is not specified, there is no default destination. The LP scheduler may be running when you use this option. The default destination is used to determine where a file named in a user's **lp** command is sent (assuming the user does not specify a destination on the command line and the environment variable LPDEST is null). The destination *(dest)* must already exist.

To remove a destination *(dest),* the **-x***dest* option is used with **lpadmin.** This option cannot be invoked when the scheduler is running. If it is running, you must issue the **lpshut** command before **lpadmin.**

NOTE

In examples 2 through 7, it is assumed that the LP scheduler has already been stopped. This is done through the **lpshut**(M) command. Example 1 does not require the scheduler to be stopped since only the **-d** option to **lpadmin** is used. For all examples you must be logged in as root.

Example 1

Make printer dqp10_1 the system default destination.

```
lpadmin -ddqp10_1
```

Example 2

Add a new printer called dqp10_2 and associate it with device **/dev/tty11**. Use the dqp10 model interface program.

```
lpadmin -pdqp10__2 -v/dev/tty11 -mdqp10
```

When you add a new printer, it is left in a disabled state and does not accept requests.

Example 3

Create a hard-wired printer called lp1 on device **/dev/tty13**. Add lp1 to a new class called cl1, and use the same interface program that is used with printer lqp40_1.

```
lpadmin -plp1 -v/dev/tty13 -elqp40__1 -ccl1
```

Example 4

Change the interface program for printer lp1 to model interface program dqp10.

```
lpadmin -plp1 -mdqp10
```

Example 5

Add printer dqp10_2 to class cl1:

```
lpadmin -pdqp10__2 -ccl1 <CR> >s
```

Printers that are added to a class are ordered according to the sequence in which they are added. For example, assume that class cl1, in the example above, already had printers lp1 and lp2 as members. After adding printer dqp10_2, the order of the printers would be lp1, lp2, and dqp10_2. If all three printers are available, and a request is routed to class cl1, the request will be serviced by lp1. If all three printers are busy, the request will be serviced by the first available printer.

Example 6

Remove printers lp1 and lp2 from class cl1:

```
lpadmin -plp1 -rcl1
lpadmin -plp2 -rcl1
```

Example 7

No destination (class or printer) may be removed if it has pending requests. The pending requests must either be cancelled using the **cancel**(C) command or moved to other destinations using the **lpmove**(M) command before the destination can be removed.

Removing the last remaining member of a class causes the class to be deleted. If the destination removed is the system default destination, the system will no longer have a default destination.

However, the removal of a class does not imply the removal of printers that were assigned to that class.

```
lpadmin -xlp3
```

/usr/lib/lpsched

The **lpsched**(M) command starts the LP scheduler. The LP scheduler takes the top job request off the queue and "hands" it to the appropriate interface program to be printed on a printer. The LP scheduler keeps track of the job progress, and as soon as the job is completed, it takes the next job request off the queue and repeats the process. As long as the LP scheduler is running, jobs requested by **lp** will be printed. If the scheduler is not running, jobs will not be printed.

The LP scheduler is started automatically each time the system is turned on. This is done through an executable file called **lp** in the **/etc/rc.d** directory. The **lp** file is created when the LP Spooling system is installed.

Every time the scheduler is started, **lpsched** creates a file called SCHEDLOCK in the /usr/spool/lp directory. As long as the SCHEDLOCK file is present, the system will not allow another scheduler to run. When the scheduler is

stopped under normal conditions, either with **lpshut**(M) or as part of the normal shutdown procedure, the SCHEDLOCK file is removed. However, if the system comes down abnormally, there is a possibility that the SCHEDLOCK file may not get removed. To ensure that the SCHEDLOCK file does not exist, a file in /etc/rc2.d contains a command line to remove SCHEDLOCK first before it attempts to start the scheduler.

The command is entered without arguments.

lpsched

Notice in the example that the command shows no response to let you know that the scheduler is running. To verify that the scheduler is running, use the **lpstat**(M) command with the **-r** option.

lpstat -r

The screen displays:

```
scheduler is running
```

NOTE

If many job requests are queued, there may be a delay before the **lpstat** command reports that the scheduler is running.

/usr/lib/lpshut

Two of the three **lpadmin** command options (**-x** and **-p**) cannot be executed unless the LP scheduler is stopped. The **lpshut** command stops the LP scheduler and terminates all printing activity. All requests that were in the middle of printing will be reprinted in their entirety when the scheduler is restarted. The command is entered without arguments.

lpshut

The screen displays:

```
scheduler stopped
```

/usr/lib/lpmove

Occasionally, you may find it necessary to move output requests from one destination to another. For example, if you have a printer that was removed for repairs, you will want to move all the pending job requests to a destination with a working printer. This is done using the **lpmove** command. Be aware that job requests routed to a destination without a printer are automatically rejected.

Another use of the **lpmove** command is to move specific requests from one destination to another. When this is done, **lp** will no longer accept requests for the original destination (this is the same effect as a reject command). **Lpmove** refuses, however, to move requests while the LP scheduler is running. The general format of the **lpmove** command is as follows:

lpmove *requests dest*

Requests are the request identification numbers (request IDs) of jobs waiting to be printed, and *dest* is the destination to which the requests are to be moved. The destination can be a printer or a class of printers. You must be logged in as root to use the examples that follow.

Example 1

Move all the requests for printer lp1 to printer lp2. Moving the requests renames the request IDs from lp1-nnn to lp2-nnn. After the requests are moved, **lp** will no longer accept requests for lp1 (this is the same effect as a **reject lp1** command issued after the **lpmove**).

lpmove lp1 lp2

Example 2

Move requests lp1-54 and lp2-55 to printer dqp10_1:

lpmove lp1-54 lp2-55 dqp10__1

The screen displays:

```
total of 2 requests moved to dqp10_1
```

NOTE

The two requests are now renamed dqp10_1-54 and dqp10_1-55.

/usr/lib/accept

The **accept**(C) command allows job requests to be placed in a queue at the named destination(s), destination being the name of a printer or class of printers. The general format of the accept command is as follows:

accept *destination(s)*

Example

The sample command line allows printer dqp10_1 to start receiving requests.

/usr/lib/accept dqp10__1

The screen displays:

```
destination "dqp10_1 " now accepting requests
```

/usr/lib/reject

Sometimes it is necessary to stop **lp** from routing requests to a destination. For example, if a printer has been removed for repairs, or if too many requests are building at a destination, you may want to prevent new jobs from being queued at this destination. The **reject**(C) command performs this function.

Requests in the queue when the reject command is invoked will be printed as long as the printer is enabled. After the condition that led to denying requests has been corrected, use the **accept** command to allow requests to be received again. The general format of the **reject** command is as follows:

reject [-r[*reason*]] *destinations*

The **-r** option enables you to let users know why requests are being rejected by the specified destination. *Reason* is a brief explanation of the purpose for rejecting requests. If the reason consists of more than one word, enclose it in double quotes ("). The *destinations* are the printers that are not to accept requests any longer. You must be logged in as root to use the examples that follow.

Example 1

The example given here is for a printer, lqp40_1, that is being repaired. While lqp40_1 is out of service you want to prevent **lp** from routing requests to it.

reject -r "printer lqp40__1 under repair" lqp40__1

The screen displays:

```
destination "lqp40_1 " is not longer accepting requests
```

Users who try to route a job to lqp40_1 will receive the following message:

```
lp: can't accept requests for destination "lqp40_1" -
printer lqp40_1 under repair
```

PRINTER INTERFACE PROGRAMS

Printers that are used as LP Spooling printers must have a printer interface program. Every print request made with the **lp** command is routed through the appropriate printer interface program before the request is printed on a line printer. The printer interface program to use is specified by the **lpadmin**(M) command.

Model Interface Programs

Each type of printer requires its own interface program. Several, referred to as "model" interface programs, are furnished with the LP Spooling Utilities. The model interface programs are written as shell procedures, but they can be written as C programs or any other executable program. They are located in the /usr/spool/lp/model directory.

Writing Interface Programs

If you have a printer that is not supported by one of the model programs, you will have to furnish an interface program for it. The shell script for a dumb printer interface program (a model program) is shown in Figure C-1, later in this chapter. This program may be used as a guide if you have to provide one of your own.

When the LP scheduler routes an output request to a printer, the interface program for the printer is invoked in the directory /usr/spool/lp. The format is:

interface/*P id user title copies options file...*

Arguments for the interface program are:

P	printer name
id	request id returned by lp
user	logname of user who made the request
title	optional title specified by the user
copies	number of copies requested by user
options	blank-separated list of class or printer-dependent options specified by user
file	full path name of a file to be printed

When the interface program is invoked, its standard input comes from /dev/null and both the standard output and standard error output are directed to the printing device. Interface programs format their output based on the command line arguments. You want to make sure that the interface program has the proper stty modes (terminal characteristics such as baud rate, output options). You can do this by adding stty(C) command lines of the form:

stty *mode options <&1*

This command line takes the standard input for the **stty** command from the device. An example of an **stty** command line that sets the baud rate at 1200 and sets some of the option modes is shown below.

<stty -parenb -parodd 1200 cs8 cread clocal ixon 0<&1>s

Because different printers have different numbers of columns, make sure the header and trailer for your interface program correspond to your printer. When printing is complete, your interface program should exit with a code that tells the status of the print job. Exit codes are interpreted by **lpsched** as follows:

Code	Meaning to lpsched
0	The print job has completed successfully.
1 to 127	A problem was encountered in printing this particular request (for example, too many nonprintable characters). This problem will not affect future print jobs. The **lpsched** command notifies users by **mail**(C) that there was an error in printing the request.
greater than 127	These codes are reserved for internal use by **lpsched.** Interface programs must not exit with codes in this range.

When problems occur that may affect future print jobs--for example, a device filter program is missing--it is wise to have your interface program disable printers so that print requests are not lost. When an active printer is disabled, the interface program can be halted with signal 15 (see **kill**(C) and **signal**(S)).

```
# lp interface for dumb line printer
#
x="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
echo "$x\n$x\n$x\n$x\n"
banner "$2"
echo "\n"
user='grep "^$2:" /etc/passwd | line | cut -d: -f5`
if [ -n "$user" ]
then
        echo "User: $user\n"
else
        echo "\n"
```

Figure C-1. Dumb Line Printer Interface Program

```
fi
echo "Request id: $1    Printer: `basename $0`\n"
date
echo "\n"
if [ -n "$3" ]
then
        banner $3
fi
copies=$4
echo "\014\c"
shift; shift; shift; shift; shift
files="$x"
i=1
while [ $i -le $copies ]
do
        for file in $files
        do
                cat "$file" 2>&1
                echo "\014\c"
        done
        i=`expr $i + 1`
done
echo "$x\n$x\n$x\n$x\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n"
echo "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n$x\n$x\n$x\n$x\n$x"
echo "$x\n$x\n$x\n$x\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n"
echo "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n"
echo "$x\n$x\n$x\n$x\n$x\n$x\n$x\n$x\n$x\n"
exit 0
```

Figure C-1. Dumb Line Printer Interface Program (Cont.)

Files and Directories

This section describes the files and directories in the LP Spooling structure.

/usr/spool/lp/FIFO

FIFO is a special file that all the commands use to send messages to **lpsched.** Any of the LP commands may write to FIFO, but only **lpsched** may read it.

/usr/spool/lp/default

This file contains the name of the system default destination. If this file does not exist or if it is empty, the LP system has no default destination.

/usr/spool/lp/log

The purpose of the **log** file is to keep a record of all the printing activity that has taken place since the LP scheduler was last started. This file contains the **logname** of the user who made the request, the request id, the name o the printer that the request was printed on, and the date and time that printing started. Any **lpsched** error messages that occur are also recorded. The first line of the log file shows the time that the LP scheduler was started.

/usr/spool/lp/oldlog

The **oldlog** file contains a record of what was in the **log** file. When the scheduler is stopped, the **log** file is closed. When the scheduler is restarted, all the information that had accumulated in the **log** file is copied to the **oldlog** file, and a new **log** file is started. Any information that had been in the **oldlog** file is overwritten. The first line of the file tells the time that the scheduler was turned on, and the last line tells the time the scheduler was turned off.

/usr/spool/lp/outputq

When an output request is made by the **lp** command, an entry is made in this binary file. The LP scheduler takes the job request and hands it to the appropriate interface program to be printed. After the job is completed, the job request is removed, and the scheduler takes the next job request from this file and has it printed. Only those requests made since the last time the LP scheduler was started are contained in this file (that is, starting the LP scheduler clears this file).

Entries in outputq may be modified by the **lpmove, lpdisable,** and **lpsched** command. The **cancel, lpdisable,** and **lpsched** commands can mark entries in this file "deleted." If a job request is deleted before the job is completed, the entry will remain in the file.

/usr/spool/lp/pstatus

The binary file pstatus contains status information for each printer. Entries are added and removed from this file by the **lpadmin** command, and they are modified by the **cancel, lpenable, lpdisable,** and **lpsched** commands. When the **lpstat** command is invoked with the **-p** option, printer status information is obtained from this file.

/usr/spool/lp/qstatus

This binary file keeps track of whether a destination is accepting or rejecting requests. Entries are added or removed from this file by the **lpadmin** command and modified by the **accept** and **reject** commands. When the **lpstat** command is invoked with the **-o** option, the request status is obtained from this file.

/usr/spool/lp/seqfile

The seqfile file contains the sequence number of the last request id that was assigned by the **lp** command. The sequence number is incremented by **lp** for each request. When the number 9999 is reached, the sequence number is reset to 1.

/usr/spool/lp/class

This is a directory that contains one file for each LP class that has been identified. (The name of the file is the same as the name of the class.) The file identifies each member, in this case an LP printer, that is assigned to the class. Class files are created, modified, and deleted by the **lpadmin** command. Every class file must always have at least one member.

/usr/spool/lp/interface

The interface directory contains one executable interface program for each printer that is in the LP system. The filename of the interface program is the same as the printer name. The interface program is invoked with its standard error and standard output directed to the printer. Interface programs may be shell procedures or compiled C programs.

/usr/spool/lp/member

The member directory contains one file for each LP printer. The filename is the same as the printer name. Each file contains the pathname of the device to which the member is connected.

/usr/spool/lp/model

This is a directory that contains the printer interface programs that are distributed with the LP Spooling Utilities.

/usr/spool/lp/request

This directory contains a subdirectory for each destination in the LP system. The name of the subdirectory is the same as the name of the destination. When an **lp** request is made, a request file (or "r" file) and, in most cases, a data file (or "d" file) are created in the subdirectory of the destination to which the request is going. The data file stores the file to be printed until the scheduler is ready to print it. A data file is only created if the **-c** option is given to the **lp**(C) command.

The name of the request file is derived from the request identification number and is of the form *r-seqno*. The name of the data file is of the form *dn-seqno*, where *n* is a non-negative integer.

The request and data files are deleted by the **cancel** and **lpsched** commands. They may be moved from one subdirectory to another by the **lpmove** command.

Lock File

A lock file, SCHEDLOCK, is present while the LP scheduler is running to ensure that only one invocation of **lpsched** is active.

Cleaning Out Log Files

As described previously, when the scheduler is stopped, the log file is closed. When the scheduler is restarted, the log file is copied to /usr/spool/lp/oldlog, and a new log file is started.

If the scheduler is not stopped for long periods of time and if you have a large number of LP requests, the log file can grow to be a large file. You can manually remove the contents of this file, or you can let the system do it for you on a scheduled basis. To have the system clean out the log file, put an entry in a file in the /usr/spool/cron/crontabs directory. One way to do this is to log in as **root** and use the **crontab**(C) command. The other way is to edit a crontabs directory file.

The example below shows some typical **crontab** command lines. **Crontab** adds these command lines to the **root** file in the /usr/spool/cron/crontabs directory. Every Friday at 11:00 PM **cron**(C) executes the commands. First, the contents of the log file are copied to the oldlog file, and then the log file is cleaned out. You must be logged in as root to add these lines to crontab.

```
crontab -l 0 23 * * 5 /bin/su lp -c "cp /usr/spool/lp/log
/usr/spool/lp/oldlog" 1 23 * * 5 /bin/su lp -c
<2"> /usr/spool/lp/log" #>2
```

THE lpr PRINT SPOOLER

Like **lp**, the **lpr** print spooler lets you print in background on any printer in the system. When you use the **lpr** command, **lpr** places a copy of the file to be printed in the corresponding spool directory. The files from the spool directory are then sent to the requested printer in the order in which they were received.

NOTE

Beginning with Altos SystemV, release 5.3d, the **lpr** program is a filter for (calls) the **lp** program. See the beginning of this chapter for information on **lp** and how to set up a printer interface program (if needed).

Each printer connected to the system has a corresponding file name (device name) and a spool directory that receives a copy of each file to be printed and holds it until it can be printed. Printer device names follow a similar pattern to tty device names. The default printer on a system is /dev/lp (for line printer) and has a spool directory called /usr/spool/lpd. Additional printers are called /dev/lp*n*, where *n* is a single alphanumeric character. The additional printer /dev/lp2 has a spool directory called /usr/spool/lpd2.

Each printer also has at least one line in a printer configuration file where the spooler can find tty settings (baud rate, tabs, new line delays) for that printer. Printer configuration files are discussed later in this section.

When the **lpr** command is invoked, **lpr** looks in the printer configuration file for information (what printer, on what machine, with what tty settings) and sends this information with a copy of the file to be printed and the user name of the requestor to the spool directory. It also sends any information given in command line options. The tty settings include baud rate, carriage-return and newline delays, new line mapping, and tab expansion. For a complete list of tty modes, see **stty** in the *Reference (C)*.

Then, **lpr** runs a background process (/usr/lib/lpd), called a daemon, to do the actual printing. The daemon consults the control files in the spool directory for that job and sets the specified printer appropriately. The text file is then sent to the printer. When the daemon starts printing, a lock file is established in the spool directory to warn other daemons that printing is in progress. This keeps another daemon from trying to print simultaneously.

Files are printed in the order in which they were copied into the spool directory. The text file and its information files are then deleted from the spool directory. Periodically, the daemon reads the spool directory to see if new files have come in while it was busy printing. When no more files remain to be printed, the daemon terminates and removes its lock file so that a later daemon will be able to print.

Sending a Print Request

To send a file named text (in your current directory) to print on the default printer, type:

lpr text Retn

A copy of text is sent to the spool directory /usr/spool/lpd and then to the printer /dev/lp, where it is printed according to the default settings listed in the printer configuration file.

To print the file named text on printer 2 type:

lpr2 text Retn

A copy of text is sent to the spool directory /usr/spool/lpd2 and then sent to the printer /dev/lp2.

If your system is not set up for more than one printer, see the section "Adding a Printer to your System" to do so. When **lpr** is not followed by the name of a file, as here:

lpr Retn

the standard input (the keyboard) is assumed and the file is printed on the default printer.

The syntax for using the **lpr** command is:

lpr [*options*] Retn

These options are defined as follows (square brackets ([]) surround optional arguments):

-s*conf*	Selects a printer configuration line from the printer configuration file. *Conf* is an alphanumeric string that is used to pick out a single configuation line from the configuration file.
-b*x*	Adds a banner page at the beginning of a print job. The *x* argument (4 characters maximum) supplies the text of the header. Use of this option overrides the BANNER environment variable.
-p*k*	Prints on *k*, where *k* is a single-digit number that represents the printer device. For example, **lpr -p2** selects printer 2 (and is equivalent to lpr2).
-m[*login*]	Sends mail to the specified login name when printing is completed. If you don't specify a login name, you will be notified.
-N	Suppresses the formfeed after each file.
-n[*netname*]	Prints on a printer attached to another machine on the network (see "Using **lpr** with WorkNet").
-@[*netname*]	Same as -n[*netname*].
-S*modes*	Prints using printer settings (*modes*) supplied. Modes must be in the **stty** format, each one separated with a space. For example, **-S1200** selects 1200 as the baud rat for this print request.
-r	Removes a file from the spool queue.

For more detailed information on command line options for **lpr**, see the *Reference (C)*.

Using lpr on a Network

To use **lpr** on a network of systems, use the **-s** option to specify a line in the printer configuration file that lists a printer on a remote system, or specify a *machine-name* on the command line. The print spooler will look for the spool directory and the printer device on the remote machine. The text file and command files will be placed in the spool directory of the named machine and the daemon will run there locally.

Checking the Status of a Print Request

To check the status of a print request, type:

ps -e Retn

The system responds by listing every process currently running on your system in the following format:

```
PID     TTY     TIME    COMMAND
10084   co      0:02    sh
10087   co      2:55    uniplex
10318   co      0:01    lpd
```

When you see lpd in the command column, that means that your print job is queued for printing. On some printers, lpd will remain in the display until the last page of the print job has printed; on other printers, such as laser printers, the lpd will disappear from the screen as soon as the print job starts printing.

Canceling a Print Request

To cancel a print request, do the following:

1. Type:

   who; ps -e Retn

   The **who** command tells you the login name and terminal ID (tty) number of everyone logged in, and the **ps** command displays all the processes running on your system. It will look something like this:

```
admin    console    Feb 13    09:34
betsy    tty02      Feb 13    09:38
wendy    tty04      Feb 13    09:51

PID      TTY        TIME      COMMAND
1140     04         0:00      s
1141     04         0:02      lpd
1474     co         0:10      tar
```

2. Find the row that contains your terminal ID number (in the column headed TTY) and the process name lpd (in the column headed COMMAND). Note the number displayed in the column headed PID (process ID number) in this row; the PID number in the example if 1141. Enter the command:

   kill -9 *nnn* Retn

   where *nnn* is the PID number.

3. Next, enter the command:

   rm -f /usr/spool/lpd/lock Retn

   This command removes the lock file for your stopped print job, so you can start fresh with a new print request.

4. Realign the paper in the printer and send the print request again.

Configuring a Printer

Each printer must have at least one line in the printer configuration file /etc/printers. This is where **lpr** looks to find out what printer to print on, and with what tty settings. Each line in the printer configuration file gets a unique name. This lets you select a printer and a group of tty settings all at once by using the -s option on the command line. Each printer (each physical device) can be set with different tty modes for different printing needs. Each group of settings appears on a separate line.

Each line consists of several fields (arguments) separated by colons. Do not use spaces before and after the colons. A line in a printer configuration file has the following fields (square brackets surround optional fields; do not enter square brackets). The format is:

lpn:*line-name*:*printer-type*:[*machine-name*]:[*tty modes*]

The values for each part of a printer configuration file line are:

lp*n*	Printer device name.
line-name	User-selected printer name.
printer-type	Make of your printer (**lpr** does not use this field, but if the field is null, errors will occur.)
machine-name	Name of the machine on the network to which the printer is connected.
tty modes	The settings for the printer, separated by spaces, in **stty** format.

Here is a typical printer configuration file:

```
lp:agnes:x::
lp:bartleby:x::9600 nll -tabs
lpl:chauncy:x::1200 raw
lpl:desmond:x::1200 cll
lp:evelyn:x:machine2:9600 raw
lpl:fabian:x:machine2:1200 nll -tabs
```

This printer configuration file lists two printers on the home system (/dev/lp and /dev/lpl) and two printers on a remote system (machine2). Each of the two printers on th home system has two different tty settings. The default printer on the home system runs at 9600 baud (the default baud rate) and the other printer runs at 1200 baud. The first line listed for a printer is taken as the default setting for that printer.

The first line of the file (linename "agnes") lists no tty modes. This tells **lpr** to print using the system default tty settings. To see a list of these settings, enter the command:

stty < /dev/lp*n* Retn

where *n* is the printer digit. When **lpr** is invoked without arguments, and with a printer digit appended, as in:

lpr2 text Retn

The **lpr** program searches the printer configuration file for the first line that has lp2 in the first field, and uses the settings in that line.

Adding a Printer to Your System

To add a printer to your system, use the port configuration program, **pconfig**. When you run **pconfig** to configure a printer port, the program does the following:

- Makes a spool directory for the printer.
- Links the printer device to a tty device (port).
- Links the new printer to the **lpr** command.
- Enters the appropriate lines in the printer configuration file.

A printer /dev/lp may be attached to any port. On the 386 series machines, port 01 is set up to receive a printer. To move /dev/lp to another port use the **pconfig** command.

Using Printers on a Network

After your system has been successfully networked to another remote system (or systems), decide which systems have printers that you want to access.

From each of these systems, get a listing of their printer configuration file. To list the printer configuration file for a remote system named machine2, use this command:

cat @machine2/etc/printers Retn

Then enter new lines to your printer configuration file, consulting your listing, and following these guidelines:

Field 1: printer device name

You must use the same device name that the printer has on its home machine. This is because the **lpr** program runs there locally and needs to access certain information keyed to this name.

Field 2: *line-name*

Select a new line name appropriate for your system. Remember that it must be different from every other line name in your configuration file.

Field 3: *printer-name*

Leave this field as it is at the remote system.

Field 4: *machine-name*

Enter the network name of the remote machine.

Field 5: *tty modes*

Enter the correct baud rate for the printer. Then enter any other *tty modes* you want. The baud rate must be the same as that listed on the remote system Other *tty modes* do not have to be the same. You may print remotely on the printer using different settings from those that the local users use. If your printer configuration file is correctly set up and you still cannot access the remote printer, contact the system administrator of the remote system and verify permission settings.

Appendix D
File Transfer Program

(BLANK)

INTRODUCTION

The File Transfer Program (FTP) transfers ASCII text files or binary data files from UNIX to UNIX/XENIX, MP/M to UNIX, and UNIX to MP/M on Altos computer systems. You should be familiar with your operating system and MP/M before you use these programs. For some systems, Concurrent CP/M is used instead of MP/M.

The programs only transfer files; they do NOT convert MP/M programs to system compatible programs or system programs to MP/M programs.

Use the FTP program between Altos computer systems. For transferring files remotely between Altos computer systems, use the **cu** or **uucp** utility. These utilities are described in the *Reference (C)*.

NOTE

FTP (**ftp**(C)) has been recently renamed on Altos UNIX releases to **aftp**(C). The **aftp** command is in all other respects identical to **ftp**.

However, the FTP programs on other Altos operating systems have not changed. Therefore, the FTP command for CP/M and MP/M systems is still **ftp86**; and for XENIX systems it is still **ftp**.

Setup Procedures

Before you transfer files, do the following:

1. Connect the physical port on each machine via a null-modem cable. Refer to Table D-1 to determine the appropriate port. Refer to the *Altos Cable Guide* manual for your machine for how to make a null-modem cable to directly connect two Altos computers. You must be logged in as root to transfer files.

Table D-1. Default Ports

System	Sending/Receiving Operating System	Default Port
ALTOS 486	XENIX CP/M, MP/M	6
ALTOS 586 586T	XENIX CP/M, MP/M	6
ALTOS 986 986T	XENIX CP/M, MP/M	6 10
ALTOS 1086	XENIX	6
ALTOS 2086 3086	UNIX/XENIX	6
ALTOS 3068	UNIX CP/M, MP/M	any
ALTOS 386	UNIX System V Altos System V	6

You may need to disconnect the printer cable before installing the null-modem cable, or install a selector switch.

2. You must first disable the port you are going to use. To disable the sending/receiving port(s), become the system administrator, then type:

   # disable *device* Retn

   where:

   device the port to which the null-modem cable is attached. For example, /dev/tty06.

NOTE

If your Altos system has WorkNet, do not use the WorkNet port for the file transfer program.

If the cable gets disconnected during transmission, wait for the file transfer procedure to stop (takes about one minute) before restarting on the same port. Otherwise, the first file transfer procedure interferes with the second.

3. Select the same baud rates for both machines. For MP/M 16-bit machines, enter **MPMSETUP** Retn then alter the port configuration to set the correct baud rate.

   Altos systems can run at 9600 baud on send and receive. Use the **pconfig** command to set the correct baud rate.

4. Make sure file names are compatible between systems (later you can copy the file and rename it). Files sent from MP/M or CP/M systems to your operating system contain 11 characters; some characters may be trailing spaces. If you enclose the entire file name in quotes, the system recognizes it as the intended file name.

5. Use one of the following procedures, depending on the systems you are transferring between.

CP/M OR MP/M TO UNIX

FTP resides on both MP/M master distribution diskettes and transfers files to a UNIX system from any 8- or 16-bit Altos Computer System. FTP provides full error checking. Correction is accomplished by re-transmission of data blocks.

You must first do the setup procedures on the previous pages of this appendix.

It does not matter which side (sending or receiving) is started first, as long as both sides are started within one minute of each other.

Start the sending side by entering one of the following commands:

0C> **ftp86** *filename* Retn

or

0C> **ftp86** *a: filename* Retn

where:

filename the name of the file you are transferring.

a: the drive letter of the destination disk. If no drive letter is specified, the logged disk is the destination disk.

Next, a message on the screen asks you to choose the port number you are using for the FTP procedure.

The sending side displays an "s" every few seconds until communication is established with the other side.

Start the receiving side of the transfer by using the command format:

$ **aftp -f** *device* **-s** *speed name* Retn

where:

device the special file device that transfers files between machines. For example, the default device might be /dev/tty02 (port 2). If you don't specify the device, omit the **-f** also. Then, FTP uses the default device.

speed transmission speed, such as 1200, 2400, 4800, or 9600 bits per second. The default is 9600 baud. If you don't specify the speed, omit the -s also. Then, FTP uses the default speed.

name directory, if other than home directory. For example, if you want to transfer the file "update" to your directory "newdir," enter "newdir" as the name.

The receiving FTP periodically displays a "w" while waiting for the sender to become active.

For example, to transfer the file named "update" to the "newdir" directory on the UNIX system, enter:

Sending Side: 0C> **ftp86 update** Retn

Receiving Side: # **ftp -f /dev/tty02 -s 4800 newdir** Retn

If you do not start procedures within a minute of each other, the system times out and the $ prompt reappears. To redisplay the MP/M prompt, press Ctrl-c; then restart the procedures.

After the transfer, file names that don't have 11 characters contain trailing spaces. For example, the file named update has six letters and five trailing spaces.

Rename a transferred file so the operating system easily recognizes the file. Be sure to type the name and number of trailing spaces (up to 11 characters). Enclose the 11 characters in single quotation marks. For example,

$ mv 'UPDATE ' update Retn

UNIX TO UNIX/XENIX

FTP can transfer files between two Altos Computer Systems running the UNIX or XENIX operating system.

You must first do the setup procedures on the previous pages of this appendix.

It does not matter which side (sending or receiving) is started first, as long as both sides are started within one minute of each other.

Start the FTP utility by using the following command format on the sending computer (using **aftp** if it is used instead of **ftp** on your system):

ftp -f *device* **-s** *speed name*

where:

device the port to which the null-modem cable is attached. For example, the default device might be port 02. The sending/receiving port numbers don't have to be the same. If you don't specify the device, omit the -f also. Then, FTP will use the default device.

speed transmission speed, such as 1200, 2400, 4800, or 9600 bits per second. The default is 9600 baud. If you don't specify the speed, omit the -s also. Then FTP will use the default speed.

name the name of the file you are sending.

The sending side displays an "s" every few seconds until communication is established with the other side.

Enter the FTP utility on the receiving computer using the format below (and using **aftp** instead of **ftp** as explained above):

ftp -f *device* **-s** *speed name*

where device, speed, and name are the same as described previously for the sending computer.

The receiving device can differ from the sending device; however, the speed of the two systems must be the same. Enter the name only if you want to specify a directory for the transferred file other than your home directory.

The receiving side displays a "w" every few seconds. During the file transfer, the FTP outputs an "*" after each successful transfer of 128-byte block increments. A "?" is displayed each time a block is retransmitted to overcome a transmission error. If you receive many "?"s, decrease the baud rate.

For example, to transfer the file named "newfile" on the sending system to the directory "/tmp" on the receiving system, enter:

Sending Side: **# ftp -f /dev/tty02 -s 4800 newfile** Retn

Receiving Side: **# ftp -f /dev/tty05 -s 4800 /tmp** Retn

UNIX TO MP/M

The file transfer program, FTP, can transfer files from a UNIX system to an MP/M system.

FTP runs on the UNIX system, and FTP runs on the MP/M system during file transfer between your system and MP/M.

You must first do the setup procedures on the previous pages in this appendix.

It does not matter which side (sending or receiving) is started first, as long as both sides are started within one minute of each other.

Start the FTP utility by using the following command format on the sending system:

aftp -f *device* **-s** *speed name*

where:

device port to which the null-modem cable is attached. For example, /dev/tty02. If you don't specify the device, omit the **-f** also. Then FTP will use the default device.

speed transmission speed, such as 1200, 2400, 4800, or 9600 bits per second. The default is 9600 baud. If you don't specify the speed, omit the -s also. Then FTP will use the default speed.

name the name of the file you are sending.

For example, to transfer a file named SAMPLE.TXT to the MP/M system, enter:

```
# aftp -f /dev/tty2 -s 4800 SAMPLE.TXT Retn
```

The sending side displays an "s" every few seconds until communication is established with the other side.

Start the receiving side by entering one of the following commands:

```
0C>FTP86
```

or

```
0C>FTP86 u:
```

where:

u: the drive letter of the destination disk. If no drive letter is specified, the logged disk is the destination disk.

Next, a message on the screen asks you to choose the port you are using for the FTP procedure.

The receiving side periodically displays a "w" while waiting for the sender to become active. If the UNIX prompt reappears, the receiving side normally does not exit by itself; type Ctrl-c to get back to the MP/M prompt.

During the file transfer, the FTP utility outputs an "*" after each successful transfer of 128-byte block increments. A "?" is displayed each time a block is retransmitted to overcome a transmission error. If you receive many "?"s, decrease the baud rate.

Appendix E
Operating System Error Messages

(BLANK)

INTRODUCTION

This appendix contains information about operating system error messages. (For **lp** and **lpr** error messages, see the following appendix, "Print Spooler Error Messages.") System calls and library routines error messages are in the *Programmer's Guide.* Compiler error messages appear in the *C Compiler Library and User's Guide.*) Topics include:

- Sources and types of error messages
- Troubleshooting errors
- PANIC, WARNING, AND NOTICE error messages
- Other system messages
- Multidrop error messages

By using the information in this appendix, you should be able to:

- Understand the four different types of error messages
- Distinguish software error messages from hardware error messages
- Establish a troubleshooting strategy for possible recovery

WHERE DO ERROR MESSAGES COME FROM?

Error messages come from a variety of different software or hardware conditions. They are reported to the console from four different sources:

1) Application program error messages - an error message appears on the user's screen only.

2) Operating system error messages - a complete error message appears on the system console, and a partial error message may appear on the user's screen.

3) Firmware boot failure messages - a failure message appears when the operating system cannot be booted into main memory.

4) System diagnostic (SDX) diskette error messages - pass or fail messages appear as the diagnostics test the hardware.

You will usually see error messages from areas 1) and 2) only. The error messages from application programs are messages that notify the user of improper use of the application program. Most application programs provide an error message that appears only on the user screen when a user attempts an illegal action.

Application program error messages are usually easy to understand, and should be explained in the documentation provided with the application program. The operating system error messages are much more obscure; the rest of this appendix will be spent on describing operating system errors.

A message preceded by "PANIC," "WARNING," or "NOTICE" is an example of an operating system message. If your computer displays a "PANIC" error on the system console, the system shuts down, and you must reboot it before computing can continue. After the computer is rebooted, allow the file structure to be cleaned. You will only see error messages from 3) and 4) if your computer receives ar operating system error severe enough to prevent the operating system from being rebooted.

However, if a PANIC error message indicates an error condition that is permanent and non-recoverable (for example a hard disk controller failure), you won't be able to reboot. If this occurs, you will receive error messages from area 3). The hard drive controller IC will send error bytes to the firmware and a message similar to "File system not supported" will appear on the system console. At this point, you should boot the SDX diskette, which should isolate the failing hardware area. Refer to your *Owner's Guide* for information on how to use the SDX diskette.

The "Troubleshooting Errors" section that follows will give you some added guidelines for dealing with error messages.

TYPES OF OPERATING SYSTEM ERRORS

The operating system handles error conditions according to their degree of severity, and divides the error messages into three classes: NOTICE, WARNING, and PANIC.

NOTICE — This error message provides information on the system status. NOTICE error messages can help you anticipate potential problems before they occur.

WARNING — A WARNING error message indicates that the operating system may stop functioning if corrective action is not taken.

PANIC — PANIC error messages indicate a problem severe enough that the operating system stops, and you must reboot it system to resume operation. The cause is usually a hardware problem or a minor problem in the kernel software. Some file systems may be corrupted, but the operating system checks for this when you restart (reboot) the system. As with most sophisticated computer systems, PANICs will occasionally occur; they make the results of future computations uncertain. For example, if the operating system detects that system memory has internally contradictory data, the operating system must shut processes down to preserve the integrity of the existing data. If a particular PANIC message occurs repeatedly (or predictably), you should contact your service representative.

The messages in this appendix are listed in alphabetical order according to their class. Note that messages starting with ! (often come from the communications board) appear *only* in the system error log file, not on the system console. (See **errprint**(M) for information about this file.) Messages starting with ^ appear only on the console. All other messages appear both on the console and in the system error log.

If you get a device error, you may want to know the name of the device associated with the given major and minor numbers. To find out, use the command pipeline:

ls -l /dev|grep *nn*|**grep** *mm*

The variables (shown in *italic*) that occur in the text of this appendix are replaced by a real value when you see an error message. Those variables can be interpreted as follows:

d	signed decimal
o	unsigned octal
u	decimal
x or *X*	hexadecimal
s	string
c	character argument

In addition, abbreviations can be expanded as follows:

DU	Remote File Sharing (RFS)
S5	UNIX System V
WN	WorkNet

TROUBLESHOOTING ERRORS

As a system administrator, you need to monitor any error messages that occur on the console, and then assess the situation. It's a good idea to keep a written log of error messages. And, check the system error log by typing:

errprint

to display one screenful of the log at a time, or:

errprint > *filename*

which will put the errors in a file named by you. Part of your job as system administrator is to back up all files on a daily basis. That way if you have a problem, the files are backed up. Turn to "Backing Up File Systems" in this manual for backup procedures.

Be sure to check the alphabetical list of error messages that follows in this appendix. Then try to determine:

1. What type of error message is it: operating system, hardware, application program, etc.?

2. What were the different users on the system doing before the error message occurred?

3. If the system has stopped running, can you reboot it and continue? (Always allow the root file structure to be cleaned.) If the computer won't respond at all, do you have a terminal on the console port? Or, are you getting firmware failure messages because the operating system can't boot?

Your answers to the questions above should help you decide if you have a problem, and if so, the kind of problem it is.

If you receive a PANIC error message, your system won't reboot, or you are getting firmware boot failure messages, it is most likely that you have a hardware problem. At this point, you can run the SDX diagnostics to confirm the failure. (Then you will need to call your service representative to arrange for repairs.) If all the SDX tests run without failure, you might want to reinstall the operating system, and continue if possible.

There are many possible sources of PANIC error messages. For example, the source could be an untested path through the kernel, a botched hardware installation, or a transient hardware failure. It is also possible for some files to be corrupted, but the file system checks for this when the system is restarted.

NOTE

The best way to handle an isolated PANIC error message is to reboot the system, allow the file system check, and continue operation. As a general rule, an isolated PANIC failure shouldn't occur any more than once a month under continuous heavy use. Unless the PANIC error message is part of repetitious pattern, you should be able to clean the file system and resume computing.

Often an operating system error is caused by everyday events. For example, a problem is encountered while backing up to a floppy disk or streaming tape. In the following case, a user attempted to copy files to the system from a defective floppy using the **tar**(C) command. The following error message appeared on the user's terminal:

```
WARNING tape read error
```

A more detailed device error message is displayed on the system console:

```
WARNING error on dev fd (I/O), block = 9, cmd = 0004,
        status = 0106
```

The detailed device error message identifies the device, the location on the floppy where the error occurred, the command, and the status. This message is thoroughly explained in the WARNING section that follows.

Be aware that some software error messages resemble hard ware operating system errors. For example, the following error message could lead you to the false conclusion that your system memory failed during use:

```
memory fault - core dumped
```

In this case, the system memory didn't fail; something is wrong with the program you are using. The program attempted some impossible action, such as dividing by zero or accessing a nonexistent device. The operating system reacts to software errors of this kind by printing the error above, and dumping all the contents of memory into a file called "core." Later the software error can be removed or analyzed by an experienced programmer.

PANIC ERROR MESSAGES

Error Message		Description
PANIC	Allocated entry in s5inode free list	An inode on the free list was marked as already in use. This is a software problem.
PANIC	bumprcnt - region count list overflow	Ran out of region count entries when paging unused pages to swap disk.
PANIC	Coprocessor error with no numeric processor	An interrupt was received indicating that there was a coprocessor error, but no numeric processor chip is installed (or it was not detected during system boot). Preceded by a dump of the state of the system.
PANIC	Coprocessor overrun with no numeric processor	An interrupt was received by the numeric processor indicating that there was a coprocessor error, but no numeric processor chip is installed (or it was not detected during system boot). Preceded by a dump of the state of the system. Most likely a hardware problem.

Error Message		Description
PANIC	Couldn't allocate space for WorkNet buffers	System has too little memory, so the operating system couldn't allocate space for WorkNet buffers.
PANIC	Duinit: duinit not found in fstypsw	This error message is the same for UNIX System V and WorkNet. The initialization routine for the file system type has been called without being in the fstypsw table. This is an operating system problem.
PANIC	Finddbd: can't find page table entry	The page table for one of the processes' regions could not be found in the list of the page tables. This is an operating system problem.
PANIC	Floating point in kernel with no hardward	A floating point operation was attempted by the operating system. Usually an operating system problem.
PANIC	Get pages - pbremove	When a page in memory ha been modified, and has already been swapped to disk at least once, the previously used swap page must be released. This error occurs when the swa page can not be found on the hash chain. Reboot the system.

Error Message		Description
PANIC	Iget - mounted on inode, not in mount table	The inode of the directory on a device is mounted but could not be found in the operating system's internal list. Reboot the system.
PANIC	Init process exited with status %x	The initialization process has been terminated. This is extremely rare. If this does occur, the process table fills up. The error message usually stems from a problem in the /etc/inir or /etc/init programs.
PANIC	invalid root filesystem type	The type of root file system was not a supported file system type.
PANIC	I/O error on restart partition; error: %d	A disk error occurred while saving the system state after a power failure (UPS is installed). Reboot the system.
PANIC	Iupdat - fifo iaddress> 24	Block number in inode is too big.
PANIC	Iupdat - iaddress > 24	Block number in inode is too big.

Error Message		Description
PANIC	Kernel mode trap. Type 0x%x	The 386 processor unexpectedly registered an error. The actual trap type is provided by the 386 error handling routines. See the "Interrupt Vector Assignments" in the "Interrupts" section of the Intel 80386 data book for dee scriptions of trap types. Run the SDX tests, and check for any CPU failures.
PANIC	Kseg - ptemail failed	Not enough physical memory available for kernel or driver when needed.
PANIC	Main - copyout of icode failed	The kernel was not able to copy the assembly code that is used to start up the system.
PANIC	Main - swapadd failed	The kernel was not able to attach to the first swap area. Check to see if there is a swap area available.
PANIC	memory parity error	See System or User mode memory parity error.
PANIC	NULL fs pointer in s5input	An attempt was made to write an inode to disk with an invalid data structure. System inconsistency, fatal.

Error Message		Description
PANIC	Newproc - fork failed	The kernel was not able to create one of the kernel processes while booting. Check parameters, particularly NPROC in master.d directory.
PANIC	Newproc - no procs	The kernel ran out of process table slots while creating a required system process.
PANIC	No floating point emulator	No numeric processor is installed, the file /etc/emulator could not be loaded during system boot, and a floating point instruction was executed. Fatal.
PANIC	Pinsert - pinset dup	The kernel was attempting to add a page to the page cache but the page was found to be in the cache already. Probable operating system problem.
PANIC	S5init: s5init not found fstypsw	The inialization routine for the file system type has been called without being found in the fstypsw table. This is an operating system problem.
PANIC	Slave %d returned from swtchs	One of the system processes has returned from its processing loop. This is an operating system problem.

Error Message		Description
PANIC	Small model shared data copy failure	When shared data was copied to a user process user process with previously validated addresses, the copy failed. Inconsistency in shared data tables.
PANIC	Spinwait timed out on address %x	A multi-processor lock has been locked too long (about 5 seconds).
PANIC	Srmount - cannot mount root	An error occurred while trying to read the superblock of the root file system. Either the hard disk is damaged or the layout is wrong. This error should never happen, since the kernel boot would fail first.
PANIC	Srmount - not a valid root	The superblock read from the file system does not appear to be valid. System inconsistency, fatal.
PANIC	Svirtophys - not present	Could not convert a virtual address to a physical address. This is an operating system problem, because of an invalid address.
PANIC	Swapseg - i/o error in swap	An I/O error occurred during a transfer to or from the swap area.
PANIC	System process %s exited with status 0x%x	A system process such as the swapper terminated. This is an operating system problem.

Error Message		Description
PANIC	System or User mode bus timeout error	This is a hardware error; if this error appears to be part of a failure pattern, and not an isolated incident (see the prior "Troubleshooting Errors" section), back up all your files, run SDX, and call customer support. When a system bus cycle exceeds 14 microseconds, the operating system sends this error message. This error message is always triggered by a hardware error.
PANIC	System or User mode memory array parity error	This is a hardware error; if this appears to be part of a failure pattern, and not an isolated incident (see the prior "Troubleshooting Errors" section), back up all your files, run SDX, and call customer support. This error message indicates that the parity detection circuitry in system memory has detected internally contradictory memory contents. Run the following SDX tests and check for failure messages. 1. System Memory Address Ripple Test. 2. System Memory Parity Test

Error Message		Description
		3. System Memory Content Test 4. System Memory Refresh Test
PANIC	System or User mode unknown NMI %x	This is a hardware error; if this error appears to be part of a failure pattern, and not an isolated incident (see the prior "Troubleshooting Errors" section), back up all your files, run SDX, and call customer support. This error message indicates that an interrupt occurred that does not correspond to any configured device.
PANIC	Trap ax=% bx=% cx=% ...	These are the values of the 386 registers at the time of the fault.
PANIC	u-cdir not set in name for local file	When looking for a file in the filesystem that did not start with "/", the internal variable u.u-cdir for the current directory wasn't set. This is an operating system problem.
PANIC	User mode double fault	A second fault occurred while trying to save the state from a fault. This is an operating system problem.

Error Message		Description
PANIC	User mode invalid tss fault	The tss tables have become corrupted. This means a jump through a tss failed due to an invalid tss. This is an operating system problem.
PANIC	User mode segment not present fault	The kernel segment tables have been corrupted. This is an operating system problem.
PANIC	Userxmemflt. impossible condition %x	A page fault occured that was not caused by copy on write. A page not present. This is an operating system problem.
PANIC	Vfault - bad dbd type	The page being faulted is not a a recognized type (one of demand fill, demand zero, in filesystem or on swap). This is an operating system problem.
PANIC	WNinit: WNinit not found in fstypsw	The initialization routine for the file system type has been called without being in the fstypsw table. This is an operating system problem.
PANIC	WNinit: can't get inode for '@' directory	Unable to set up the internal entry for the specified directory during the boot process. This is an operating system problem.

WARNING ERROR MESSAGES

Error Message		Description
WARNING	Bufcall: could not allocate stream event	Too many stream events are pending. The event is lost. If the problem continues, increase the value of NSTREVENT in the master.d directory, and rebuild the operating system.
WARNING	Cannot allocate memory for emulator	The system has too little physical memory. Increase the size of physical memory on the system or decrease the sizes of system tables. No floating point operations will work.
WARNING	Cannot open %s partition; %s errno %d	The restart partition can not be opened. Reboot the system.
WARNING	Could not open %s partition; error %d	Could not open restart partition to save the system state after a power failure with UPS. Reboot the system.
WARNING	Device 200 not ready	Cannot open the tape drive, usually because there is no tape in the drive, it wasn't initialized at boot time, or it was disconnected from the bus.

WARNING error on dev *name* (*nn/mm*), sector *X*, cmd *X*, status *X*

This message can indicate routine error conditions (problems encountered while moving files onto a worn floppy, for example), or indicate impending serious problems (constant, recurrent hard disk errors, for example). The message indicates the problem that the device driver had when it attempted to execute a command. Use **ls -l /dev/*** to see a listing of device names.

The message is composed of the name of the failing device, the physical sector on which the error occurred, the command that was being attempted when the error occurred, and the status of the attempted command. These are defined as follows:

name A two letter mnemonic (code) for the device that caused the error.

nn The major number of the device.

mm The minor number of the device.

These are translated as follows:

Device Name *name*	**Device Number Major/Minor** *nn/mm*	**Description**
hd	00/XX	hard disk
fd	01/XX	floppy disk
tp	02/XX	tape
pp	03/XX	parallel printer
sc	04/XX	SCSI

where *XX* is:

01	swap area
02	root file system
18	second hard drive
34	third hard drive

sector X — The number of the disk block or sector where the error occurred.

cmd X — The request command (shown as a code) that was attempted by the system. See Tables E-1 and E-2.

status X — The status of the command (shown as a code) that was attempting to execute. See Table E-3, E-4, and E-5.

Tables E-1 and E-2 show the request command codes for the SCSI hard disk, and the SCSI tape, respectively.

In the following tables, an **0x** prefix indicates hexadecimal format.

Table E-1. Request Commands For SCSI Hard Disk (In Hexadecimal)

Command	Description
0x00	test unit ready
0x01	rezero unit
0x03	request sense
0x04	format unit
0x08	read
0x0A	write
0x12	inquiry
0x15	mode select
0x1A	mode sense
0x1B	start/stop
0x25	read capacity

Table E-2. Request Commands For SCSI Tape (In Hexadecimal)

Command	Description
0x00	test unit ready
0x01	rewind
0x03	request sense
0x08	read
0x0A	write
0x10	write file marks
0x11	space over block, file mark
0x12	inquiry
0x15	mode select
0x19	erase
0x1A	mode sense
0x1B	load unload

Tables E-3 and E-4 show the status of the command that was attempting to execute.

The SCSI device (hard drive or tape) returns a status byte to the SCSI controller if various error conditions occur. There are two error conditions: the check condition (status messages beginning with an 02) and the terminated condition (status messages beginning with an 80). In the former error condition, the SCSI device returned a status, in the latter the device terminated the transfer. Check conditions are less serious than terminated conditions.

After the check condition error message (02), futher information is provided to determine the nature of the condition.

Table E-3. SCSI Check Condition Error Codes

Status Reported by SCSI Controller (Hexadecimal)	
0x0200	No sense
0x0201	Recovered error
0x0202	Not ready
0x0203	Medium error
0x0204	Hardware error
0x0205	Illegal request
0x0206	Unit Attention
0x0207	Data protect
0x0208	Blank check
0x0209	Reserved
0x020A	Copy aborted
0x020B	Aborted command
0x020C	Reserved
0x020D	Volume overflow
0x020E	Miscompare
0x020F	Reserved

Table E-4. SCSI Terminated Condition Error Codes

Status Reported by SCSI Controller (Hexadecimal)	
0x8040	An invalid command was issued
0x8041	An unexpected disconnect caused a command to terminate
0x8042	A time-out occurred during a select or reselect command
0x8043	A parity error caused a command to terminate
0x8044	A parity error caused a command to terminate

Table E-4. SCSI Terminated Condition Error Codes (Cont.)

Status Reported by SCSI Controller (Hexadecimal)	
0x8045	Logical address exceeded the the disk boundaries
0x8046	The wrong target device reselected the SCSI controller
0x8047	Incorrect message, status or command byte was received
0x8048	An unexpected information phase was requested

Table E-5. Floppy Disk Error Codes

Status Reported by the Floppy Controller (Hexadecimal)	
0x0001	Missing Address Mark. The Floppy Disk Controller (FDC) cannot detect the ID Address Mark.
0x0002	Not Writable. The FDC detects a Write Protect condition from the floppy disk.
0x0004	No Data. The FDC cannot find the appropriate sector, or read an ID field.
0x0008	Not used.
0x0010	Overrun.

Table E-5. Floppy Disk Error Codes (Cont.)

Status Reported by the Floppy Controller (Hexadecimal)	
0x0020	Data Error. When the FDC detects a CRC error in the ID field or the data field, this flag is set.
0x0040	Not used.
0x0080	End of cylinder. This flag is set if the FDC tries to access beyond the final sector.

After you determine the nature of the error or the device, you may want to run one or more the following diagnostic tests:

1. IOC Packer Function Function Test
2. IOC Floppy Controller Chip Test
3. IOC Floppy Disk Random Seek Test
4. IOC SCSI Disk Verify Test
5. SCSI Disk Verify Test

Error Message		Description
WARNING	Iget - inode table overflow	Out of free slots in the inode table. Too many files open at once. Reduce the number of simultaneous processes, or increase of inode table entries.
WARNING	Illegal *major/ minor* for *%s* partition: *%s*	An invalid device was specified for the restart partition with an UPS. Reboot the system.
WARNING	I/O error on restart partition while saving memory; error: *%d*	A disk error occurred while saving the system occurred while saving the system state (saving memory) after a power failure (UPS is installed). Reboot the system.
WARNING	I/O error on restart partition while writing header: error *%d*	A disk error occurred while saving the system state (writing the header) after a power failure (UPS is installed). Reboot the system.
WARNING	I/O error on *%s* partition: error *%d*	A disk error occurred with the system state after a power failure (UPS is installed). Reboot the system.
WARNING	Mfree map overflow %x lost %d items at %d	If recurring problem, reconfigure the system with a larger value of SPTMAP. The first variable (%d) is the address of the map that overflowed, the second variable (at %d) is the address that has to be placed in the free list.

Error Message		Description
WARNING	NULL mp in s5getinode	An attempt was made to allocate an inode for a before the pipe device was set up. This is an operating system problem.
WARNING	No floating point emulator found in /etc/emulator	No floating point emulator was found. This is not a problem unless a floating point instruction is executed when no 387 is present.
WARNING	No swap for u-area	Run fewer simultaneous processes or repartition the disk with increased swap space. Use the swap command to add more swap on another disk partition.
WARNING	Not enough 2K stream buffer for RFS, RFS failed	Increase the value of NBLK2048, and rebuild the kernel.
WARNING	Null mount in iget mp:	The inode of a mounted file system was not set in the mount table. File system is not accessible. This is an operating system problem.
WARNING	POWER FAILURE CONDITION	UPS detected a power failure. Reboot the system when power returns.
WARNING	POWER FAILURE DETECTED DURING RESTART - RESTART ABORTED	UPS detected a power failure during its restart operation, and aborted. Reboot the system.

Error Message		Description
WARNING	Rdev==NODEV in ialloc	The device field of an inode for a device was not valid. Usually indicates file system damage. Run a file system check (fsck).
WARNING	Region table overflow	Each text, data stack and memory process segment requires one entry in the region table. The system call that tried to allocate another region failed. Reduce the number of processes, or increase the number of region table entries.
WARNING	Sum of NLOCAL and NREMOTE exceeds NBUF	The partitioning of disk buffers between local disks and RFS is incorrect. The partitioning is set so that each gets at least 1/3 of v.v-buf. Should be corrected in the master.d file.
WARNING	Swap space is running out. Needed %d pages	This message is recorded in the errorlog. There is insufficient space on the swap disk to hold a task. The system refuses to create task when it feels there is insufficient disk space, but it is possible to create situations to fool this mechanism. Abnormal condition, increase the swap area to remedy.

Error Message		Description
WARNING	System must be reset	This message appears after any failure to restart, such as an I/O error on restart partition or power failure of UPS. Reboot the system.
WARNING	Tried to demand load a page that was part the end of the file	A page fault occurred, and the page requested was past the end of the file. The process is terminated. This is an operating system problem.
WARNING	unknown interrupt 0x%x	This message is recorded in the errorlog. The CPU received an interrupt via a supposedly unused vector. Typically, this event comes about when a hardware failure miscomputes the vector of a valid interrupt.

NOTICE ERROR MESSAGES

Error Message		Description
NOTICE	All processes signalled	The SIGPWR signal was sent to all processes to inform them that the system is shutting down. The processes then take the appropriate action(s) to prepare for shutdown.

Error Message		Description
NOTICE	AC back on.... continuing	AC power came back on. The system is continuing with UPS powerup.
NOTICE	Can't allocate message buffer	The system is out of message buffers. Either try again later, or send fewer inter-process communication messages.
NOTICE	Devices restarted	This is a progress message during restart after a UPS power failure.
NOTICE	Devices shutdown	This is a progress message during shutdown after a UPS power failure.
NOTICE	getcpages - waiting for %d contiguous pages	A request was made for contiguous memory pages that could not be satisfied without waiting until other pages were released.
NOTICE	IGNORED because power returned	UPS detected a power fail-failure, but no actual failure occurred and the system continued operating.
NOTICE	INITIATING SHUTDOWN PROCEDURES (SHUTKILL)	Beginning the shut-down procedure after a power failure, if UPS is ininstalled.
NOTICE	INITIATING SHUTDOWN PROCEDURES (SHUTSAVE)	Saving the system state and beginning the shutdown procedure after a power failure (UPS is installed).

Error Message		Description
NOTICE	POWER FAILURE RESTART SUCCESSFULLY COMPLETED	UPS successfully restarted the system.
NOTICE	POWER FAILURE SHUTDOWN (SHUTSAVE) COMPLETED SUCCESSFULLY	UPS successfully shut down using **shutsave.**
NOTICE	POWER FAILURE SHUTDOWN (SHUTKILL) COMPLETED	UPS successfully shut down using **shutkill.**
NOTICE	ROUTINE_NAME - insufficient memory to [allocate|lock] page(s)	A request was made for some pages that could not be satisfied. Typically, the process that needed the pages is terminated.
NOTICE	semaphore table overflow	During a creatsem or opensem system call, no more semaphores were available. If problem persists, increase XSEMMAX in the master.d directory and rebuild the kernel.
NOTICE	shared data table overflow	During an sdget system call, no more shared data structures were available. If problem persists, increase NSDSEGS and rebuild kernel.
NOTICE	swapdel - too few free pages	An attempt to delete a swap partition has failed because it would result in too little remaining space. Usually a result of bad arguments to the "swap" program.

Error Message		Description
NOTICE	s5ialloc: inode was already allocated	While getting an unused inode from the disk, the chosen inode had already been allocated. No action begun while waiting for disk read.
NOTICE	tune.t_maxfc reduced duced to *%d*	The tunable parameter **maxfc** was found to be greater than the system-imposed limit. It has been automatically reduced to that limit. When convenient, correct the kernel master file, reconfigure, and reboot.
NOTICE	tune.t_maxsc reduced to *%d*	The tunable parameter **maxsc** was found to be greater than the system-imposed limit. It has been automatically reduced to that limit. When convenient, correct the kernel master file, reconfigure, and reboot.
NOTICE	useracc - couldn't lock page	Insufficient main memory available to lock a user user data page in memory to service a read or write system call to a raw device. Reduce system load, reduce size of raw I/O request, or add more memory.

Error Message	Description
NOTICE %s - swpuse count overflow	An attempt was made to use the same swap page for more than 255 processes. No action needed; more pages are used on the swap device.

OTHER SYSTEM MESSAGES

Error Message	Description
ABNORMAL System Shut-down	This message appears when errors occur during normal system shutdown. It is usually accompanied by other system messages. No action necessary.
bad block on dev *maj/min*	A nonexistent disk block was found on, or is being inserted in, the file system's free list. Run **fsck.**
Bad free count on dev *maj/min*	A structural inconsistency in the superblock of a file system. The system attempts a repair, but this message will probably be followed by more complaints about this file system.

Error Message	Description
cannot allocate stream data blocks	There was not enough memory to allocate the system buffers during boot up. No streams services are available. Probably caused by configuring a system table with too large a value. Reduce the size of the table.
could not perform unlink ioctl, closing anyway	When a stream was being closed, one of the modules pushed on the stream could not be unlinked (operating system error). May not be able to use module after this. Contact your service representative.
DANGER: Out of swapspace.	The system is running out of swap space. If condition continues, add swap space on another disk with the **swap** command, or reinstall the system with a larger swap area.
DANGER: Out of swap space. Waiting for *%d* pages.	The system is running out of swap space. If condition continues, add swap space on another disk with the **swap** command, or re-install the system with a larger swap area.
lp: cannot get a page	When the parallel printer port was opened, no memory could be obtained. Usually causes print jobs to fail; however spooler will try to resend.

Error Message	Description
lp: not ready or out of paper	When the parallel printer port was in use, the printer went off line or ran out of paper.
maxserve changed to *%d* - not enough send descriptors	When RFS started, the value of MAXSERVE was greater than NSNDD in the master.d directory. Discrepancy ignored; the smaller of the two values was used.
minserv changed to *%d* (maxserv)	When RFS started, the value of MINSERVE was greater than MAXSERVE in the master.d directory. Discrepency ignored; the smaller of the two values was used.
namei: Invalid fs dependent *namei* return	A file system specific function returned an invalid value; the system call failed (operating system error).
No file	There are too many open files; the system has run out of entries in its "open file" table. When this table overflows, the specific request is refused. Although not fatal, this event may damage the operation of various spoolers, daemons, the mailer, and other important utilities. Anomalous results and missing data files are a common result. Close some of the open files to prevent overflow.

Error Message	Description
no idler process	The kernel has one idler process for each CPU board. This message indicates that the idler process for the CPU couldn't be found. Not fatal by itself, but PANICs usually follow.
no space on dev *maj/min*	This message means that the specified file system has run out of free blocks. Although not normally as serious, the warnings discussed for "inode table overflow" apply: often programs are written casually and ignore the error code returned when they tried to write to the disk; this results in missing data and "holes" in data files. The system administrator should keep close watch on the amount of free disk space and take steps to avoid this situation.
** Normal System Shutdown **	This message appears when the system has been shutdown properly. It indicates that the machine may now be rebooted or powered down.
not enough memory for more stream events	Too many streams events are pending and no more memory could be allocated. Some streams events will be lost. The protocol using streams will try to recover.

Error Message	Description
Out of inodes on dev *maj/min*	The indicated file system has run out of free inodes. The number of inodes available on a file system is determined when the **mkfs**(C) command is run. The default number is quite generous; this message should be very rare. The only recourse is to remove some worthless files from that file system, or dump the entire system to a backup device, rerun **mkfs** with more inodes specified, and restore the files from backup.
out of stream queues	No more stream queues could be allocated. The open on the stream will fail. If problem persists, increase NQUEUE and rebuild the kernel.
out of streams	No more stream structures could be allocated. The open on the stream failed. If the problem persists, increase the value of NSTREAM and rebuild the kernel.
page read error on dev *mm/nnn*	An error occurred while trying to read a page into memory during a pagefault. The program that filed will be terminated. Usually a disk error.

Error Message	Description
SCSI interrupt, but SCSI driver not installed	An interrupt from a SCSI device was received, but no SCSI driver was installed to handle it. The interrupt is ignored.
System V inode table overflow	There are too many open files; the system has run out of entries in its "open file" table. When this table overflows, the specific request is refused. Although not fatal to the system, this event may damage the operation of various spoolers, daemons, the mailer, and other important utilities. Anomalous results and missing data files are a common result. Close some of the open files to prevent overflow.
value configured for v.v_nqueue was not an even number	The value of NQUEUE in the configuration file in the master.d directory must be even. One queue (the last one) will be ignored. The value should be reconfigured when convenient.

MULTIDROP ERROR MESSAGES

The following messages come from the Multidrop board and are displayed in the error log file, which can be viewed by typing **errprint.** Most of the time, the system recovers (by resending data), and recovery is transparent to the user. If an error persists, suspect faulty wiring (for example, wire either too long, missing a terminator, or crimped).

Error Message	Description
bad packet length received on multidrop board %d, channel %d, (%d times)	The packet count was wrong. The packet is resent.
bad packet length received on multidrop board %d, channel %d, (%d times)	The length of the packet didn't match the packet header. The packet is resent.
bad receiver packet on multidrop board %d, channel %d, (%d times)	There was a bad control byte (contains the packet type and sequence type). Ignored, but the remote device will probably resend on its own.
bad secondary packet on multidrop board %d, channel %d, (%d times)	Packet from a bad secondary (TCU) address.
carrier present too long on multidrop board %d, channel %d, (%d times)	A device on the Multidrop cable asserted carrier for too long. No data can be received or transmitted until corrected. Physically disconnect each remote device one at a time to isolate the problem (others should then start to work). Then service the bad device.

Error Message	Description
%d carrier present too long on net channel errors on multidrop board %d	Some device on the WorkNet cable asserted carrier for too long. No data can be received or transmitted until corrected. Physically disconnect each remote computer one at a time to isolate the problem (others should then start to work). Then service the bad computer.
CRC error received	A CRC (Cyclic Redundancy Check) error was detected, because something in the packet is bad. Packet resent.
framing error on multidrop board %d, channel %d, (%d times)	A framing error occurred, possibly due to two devices sending at the same time. Packet resent.
input char(s) lost on multidrop board %d, channel %d, (%d times)	Input characters were discarded because the system buffers were full.
iopmem.c: Unknown mdc interrupt; vec=%d, board=%d, chan=%d	An unknown interrupt occurred when running foreign code on the Multidrop board.
%d lost packets (host buffers full) on multidrop board %d	Net packets were discarded because all the buffers were in use.
Multidrop %d found system bus or RAM error on multidrop board %d, channel %d, (%d times)	A fatal error occurred in the download code on the Multidrop board. Hardware failure; service the board.

Error Message	Description
Multidrop %d got unknown interrupt on multidrop board %d, channel %d, (%d times)	Software problem in the download code.
Multidrop %d had local RAM error on multidrop board %d, channel %d, (%d times)	Hardware problem detected by the download code; if this error appears to be part of a failure pattern, and not an isolated incident (see the prior "Troubleshooting Errors" section), back up all your files, run SDX, and call customer support.
%d Net channel receiver overrun errors on multidrop board %d	Too many receive packets were sent at about the same time. Resent again.
%d Net receive CRC errors on multidrop board %d	Something in the network CRC was bad; resent again.
no response from secondary on multidrop board %d, channel %d, (%d times)	A secondary (TCU) that ha open ports has failed to respond. Check to see that it has power.
packet from bad address on multidrop board %d, channel %d, (%d times)	A packet arrived with an invalid address. A new computer was connected that wasn't assigned an address in the /etc/net/hosts file. Assign a unique address, restart the network, and resend the packet.

Error Message	Description
parity error on multidrop board %d, channel %d, (%d times)	The download code has detected a parity error, which usually stops that board. Hardware error; if it appears to be part of a failure pattern, and not an isolated inciddent (see the prior "Troubleshooting Errors" section), back up all your files, run SDX, and call customer support.
receiver overrun on multidrop board %d, channel %d, (%d times)	This overrun is usually due to more than one packet arriving at about the same time. Packets resent.
transmit never overrun (chip error) on multidrop board %d	This is a hardware error; if this error appears to be part of a failure pattern, and not an isolated incident (see the prior "Troubleshooting Errors" section), back up all your files, run SDX, and call customer support.
transmit overrun early (sent short packet) errors on multidrop board %d	A packet was sent and there was an overrun, so a short packet was sent. Packet is resent. If this happens repeatedly, suspect a hardware problem.

Error Message	Description
Unknown multidrop error type *%d* on multidrop board *%d*, channel *%d*, (*%d* times)	There was an unknown Multidrop error caused by a software error in the download code.
unknown multidrop net error *%d* on multidrop board *%d*	There was an unknown Multidrop net error on the named Multidrop board caused by a software error in the download code.

Appendix F
Print Spoller Error Messages

(BLANK)

This appendix describes **lp** and **lpr** print spooler error messages.

INTRODUCTION

This section provides a description of the error messages that are associated with **lp** and **lpr** commands. The following variables are used in the error messages:

file(s)	Indicates the file or files that are to be printed.
dest	Indicates the name of the destination printer.
printer-id	Indicates the request identification number of the printout. For example, local-02 is the printer name followed by the request identification number.
printer-name	Indicates the name of the printer.
program-name	Indicates the program name that was executed.
user	Indicates the user who requested the printout.

LP ERROR MESSAGES

In this appendix, each message has an explanation of the probable cause of the error and the corrective action to take. If you are not able to correct all the error conditions you encounter, call your service representative for assistance.

Some lengthy error messages that appear as one line on the screen may be shown as two or more lines in this documentation.

Error Message	Description/Action
dest is an illegal destination name	The *dest* you used is not a valid destination name. Use the **lpstat -p** command to list valid destination names.
file is a directory	The file name you typed is a directory and cannot be printed.
xx is not a request id or a printer	The argument you used with the **cancel** command is not a valid request identification number or a printer name. Use the **lpstat -t** command to give you all the printers and requests waiting to get printed.
xx is not a request id	The request identification number you used with the **lpmove** command is not a valid request identification number. To find out what requests are valid, use the **lpstat -u** command.
xx not a request id or a destination	You used an invalid request identification number or destination with the **lpstat** command. To find out what is valid, use the **lpstat -t** command.
dest not accepting requests since *date*	Requests to the printer which you are trying to use have been stopped by the **reject** command.
Can't access FIFO	The named pipe file **/usr/spool/lp/FIFO** is incorrect. The mode should be 600.
Lp Administrator not in password file	You must have an entry in the **/etc/passwd** file for "lp," and you must belong to the group "bin."

Error Message	Description/Action
acceptance status of destination *printer-name* unknown	Use the **accept** command to enable the printer so that it will accept requests.
can't access file *xx*	The mode could be wrong on your directory or the file that you are trying to access.
can't create class *xx* - it is an existing printer name	The class name you are trying to use has already been given to a printer. You will have to use another name or remove the printer to use the class name.
can't create new acceptance status file	The mode may be wrong on the **/usr/spool/lp** directory. It should be 755 with the owner "lp" and the group "bin."
can't create new class file	The mode may be wrong on the **/usr/spool/lp** directory. It should be 755 with the owner "lp" and the group "bin."
can't create new interface program	The mode may be wrong on the **/usr/spool/lp/interface** directory. It should be 755 with the owner "lp" and the group "bin."
can't create new member file	The mode may be wrong on the **/usr/spool/lp/member** directory. It should be 755 with the owner "lp" and the group "bin."
can't create new printer status file	The mode may be wrong on the **/usr/spool/lp/pstatus.** It should be 644 with the owner "lp" and the group "bin."
can't create new request directory	The mode may be wrong on the **/usr/spool/lp/request** directory. It should be 755 with the owner "lp" and the group "bin."

Error Message	Description/Action
can't create printer *printer-name* -- it is an existing class name	The printer-name you are trying to use has already been used as a class name. You will have to assign another name for the printer.
can't create new output queue	The mode on the file **/usr/spool/lp/seqfile** is incorrect. It should be 644, and the mode on the directory should be 755. The owner should be "lp," and the group should be "bin." This may be corrected by typing the command at a later time.
can't create new sequence number file	The mode on the file **/usr/spool/lp/seqfile** is incorrect. The mode of the file should be 644, and the mode of the directory should be 755. The owner should be "lp," and the group should be "bin." This may be corrected by typing the command at a later time.
can't create request file *xx*	The mode on the file **/usr/spool/lp/request/***printer-nam* is incorrect. *Printer-name* is the name of the printer such as dqp10, and *r-id* is the request identification number. The mode of the file should be 444, and the mode of the directory should be 755. The owner should be "lp," and the group should be "bin." This may be corrected by typing the command at a later time.

Error Message	Description/Action
can't fork	You either have several processes running and are not allowed to run any more, or the system has all the processes running that it can handle. You will have to re-run this command later.
can't lock acceptance status	This is a temporary file in **/usr/spool/lp** that prevents more than one "lp" request from being taken at any given instant. You will have to rerun this command later.
can't lock output queue	The file **/usr/spool/lp/QSTATLOCK** prevents more than one "lp" request from being printed on a printer at a time. You will have to rerun this command later.
can't lock printer status	The temporary file **/usr/spool/lp/PSTATLOCK** prevents more than one "lp" request from being printed on a printer at a time. You will have to rerun this command later.
can't lock sequence number file	The file **/usr/spool/lp/SEQLOCK** prevents more than one "lp" request from being printed on a printer at at time. You will have to rerun this command later.
can't move request *printer-id*	*Printer-id* is the request identification number that cannot be moved. You will probably have to change the modes on the files and directories in **/usr/spool/lp/request.** Also, you will have to manually move the request from the disabled printer directory to the new destination after you shut down the LP scheduler.

Error Message	Description/Action
can't open class file	The **lp** program is trying to access the list of classes for printers. One reason it may not be able to open the class file is that the system could have the maximum number of files open that are allowed at any time. This can be corrected by typing the command at a later time.
can't open member file	The **lp** program is trying to access the list of members in the directory **/usr/spool/lp/member.** The system could have the maximu number of files open that are allowed at any time. This can be corrected by typing the command at a later time.
can't open *xx* file in member directory	There are a couple of reasons why file *xx* in the **/usr/spool/lp/member** directory cannot be opened. The mode on the file could be incorrect. It should be 644. Another possibility is that the system could have the maximum number of files open that are allowed at any time. This can be corrected by typing the command at a later
can't open *xx* file in class directory	One possibility why file *xx* cannot be opened is that the mode or the file or directory is incorrect. The file mode should be 644, and the directory mode should be 755. Another possibility is that the system has the maximum number of files open tha are allowed at any time. The latter problem can be corrected by typing the command at a later time.

Error Message	Description/Action
can't open *xx*	You cannot print on printer *xx* because the mode is incorrect on **/dev/tty**. The mode should be 622.
can't open FIFO	The mode on the named pipe file **/usr/spool/lp/FIFO** may be incorrect. It should be 600. Or, the system could have the maximum number of files open that are allowed at any time. The latter problem can be corrected by typing the command at a later time.
can't open member directory	The mode on the directory **/usr/spool/lp/member** could be incorrect. It should be 755. Another possibility is that the system could have the maximum number of files open that are allowed at any time. If the maximum number of files are open, try typing the command at a later time.
can't open acceptance status file	The mode on the file **/usr/spool/lp/qstatus** may not be correct. It should be 644. Another possibility is that the system could have the maximum number of files open that are allowed at any time. The latter problem can be corrected by typing the command at a later time.
can't open system default destination file	Check the mode on the file **usr/spool//lp/default.** The mode should be 644. If the mode is okay, it could be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command at a later time.

Error Message	Description/Action
can't open file *filename*	The *filename* was incorrectly typed or you don't have the correct modes set. The mode should be at least 400 if you are the owner.
can't open output queue file	Check the mode on the file **/usr/spool/lp/outputq.** It should be 644. This error message could also be generated if the system has the maximum number of files open that are allowed at any one time. Try entering the command at a later time.
can't open printer status file	The mode on the file **/usr/spool/lp/pstatus** is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command at a later time.
can't open request directory *directory-name*	The mode on the directory name **/usr/spool/lp/request** is incorrect. The mode should be 655. It could also be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command at a later time.
can't open request file *xx*	The mode on the file **/usr/spool/lp/member/request/***xx* is incorrect. The mode should be 644. It could also be that the system has the maximum number o files open that are allowed at any one time. This can be corrected by trying the **lpmove** command at a later time.

Error Message	Description/Action
can't open system default destination file	The mode on the file **/usr/spool/lp/default** is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command again at a later time.
can't open temporary output queue	The mode on the file **/usr/spool/lp/outputq** is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command at a later time.
can't proceed -- scheduler running	Many of the **lpadmin** command options cannot be executed while the scheduler is running. Stop the scheduler using the **lpshut** command and then try invoking the command again.
can't read current directory	The **lp** and **lpadmin** commands cannot read the directory containing the file to be printed. The directory name may be incorrect or you do not have read permission on that directory.
can't remove class file	The mode may be wrong on the **/usr/spool/lp/class.** It should be 755. The owner should be "lp," and the group should be "bin." Another possibility is the file in that directory may have the wrong mode. It should be 644.

Error Message	Description/Action
can't remove printer	The mode may be wrong on the **/usr/spool/lp/member** directory. It should be 755, and the files in that directory should be 644. Both the directory and the files should be owned by "lp," and the group should be "bin."
can't remove request directory	The mode may be wrong on the **/usr/spool/lp/request** directory. It should be 755 and should be owned by "lp," and the group should be "bin." The directory may still have pending requests to be printed which will have to be removed before the directory can be removed.
can't set user id to Lp Administrator's user id	The **lpsched** and **lpadmin** commands can only be used when you are logged in as "lp" or or "root."
can't unlink old output queue	The **lpsched** program cannot remov the old output queue. You will have to remove it manually by using the command **rm/usr/spool/lp/outputq.**
can't write to *xx*	The **lpadmin** command cannot write to device *xx*. The mode is probably wrong on the /dev/tty*xx* file. It should be 622 and owned by "lp."
cannot create temp file *filename*	The system may be out of free space on the **/usr** file system. Use the command **df /usr** to deter mine the number of free blocks. Several hundred blocks are required to insure that the system will perform correctly.

Error Message	Description/Action
class *xx* has disappeared!	Class *xx* was probably removed since the scheduler was started. The system may be out of free space on the /usr file system. Use the command **df /usr** to find out. Use the **lpshut** command to stop the scheduler and restore the class from a backup.
class *xx* non-existent	The class *xx* may have been removed because the system is out of free space on the **/usr** file system. Use the command **df /usr** to fine out how much free space is available. The class will probably have to be restored from a backup.
class directory has disappeared!	The **/usr/spool/lp/class** directory has been removed. The system may be out of free space on **/usr**; use the **df /usr** command to find out. The class directory contains all the data for each printer class. To restore this directory, get these files and directory from a backup.
corrupted member file	The **/usr/spool/lp/member** directory has a corrupted file in it. You should restore the directory from backup.
default destination *dest* nonexistent	Either the default destination is not assigned or the printer *dest* has been removed. Use the **lpadmin** to set up a default destination or set LPDEST to the value of the destination.
destination *dest* has disappeared!	A destination printer, *dest* has been removed since **lpsched** was started. Use the **lpadmin** command to remove the printer.

Error Message	Description/Action
can't accept requests for destination *printer-name*	The printer has been disabled using the **reject** command. The printer can be reenabled using the **accept** command.
destination *dest* non-existent	The destination printer you specified as an argument to the **accept** or **lpadmin** command is not a valid destination name, or it has been removed since the scheduler was started.
destination *printer-name* was already accepting requests	The destination printer was previously "enabled." Once a printer is accepting requests, issuing any more **accept** commands to it are ignored.
destination *printer-name* was already not accepting requests	A **reject** command was already sent to the printer. Use the **accept** command to allow the printer to start accepting requests
destination *printer-name* is not accepting requests move in progress ...	The printer has been disabled by the **reject** command, and requests are being moved from the disabled printer to another printer. The printer can be enabled again by the **accept** command.
destinations are identical	When using the **lpmove** command, you need to specify a printer to move the print requests from and a different printer to move the requests to.
disabled by scheduler: login terminal	The login terminal has been disabled by the LP scheduler. The printer can be reenabled by using the **enable** command.

Error Message	Description/Action
error in printer request *printer-id*	*Printer-id* is the actual request identification number. The error was most likely due to an error in the printer. Check the printer, and reset it if needed.
illegal keyletter *xx*	An invalid option, *xx*, was used. See the manual page for the correct options.
keyletters -*xx* and -*yy* are contradictory	This combination of options to the **lpadmin** program cannot be used together.
keyletter *xx* requires a value	The option *xx* requires an argument. For example, in the command line **lpadmin -m** *model*, the argument to the **-m** option, is the name of a model interface program.
keyletters -e, -i, and -m are mutually exclusive	These options to the **lpadmin** command cannot be used together. Refer to the manual page for more information on usage.
lp: *xx*	The variable *xx* could be one of several arguments. Typically, it is a message telling you the default destination is not assigned.
member directory has disappeared!	The **/usr/spool/lp/member** directory has been removed. The system is probably out of free disk space in the **/usr** file system. You need to clean up the **/usr** file system, and then install the LP commands or retrieve them from a backup.

Error Message	Description/Action
model *xx* non-existent	The name that you are using for a model interface program is not a valid one. A list of valid models is in the **/usr/spool/lp/rmodel** directory.
new printers require -v and either -e, -i, or -m	A printer must have an interface program, and this is specified by **-e, -i,** or **-m** options. The -v option specifies the device file for the printer. For more information on these options, refer to the **lpadmin** manual page.
no destinations specified	There are no destination printers specified. Use the **lpadmin** command to set one up.
no printers specified	There are no printers specified. Use the **lpadmin** command to set one up.
non-existent printer *xx* in PSTATUS	A printer with the name *xx* is in the **/usr/spool/lp/pstatus** file but no longer exists. The printer should be removed using the **lpadmin** command.
non-existent printer *printer-name* in class *xx*	The printer that you are trying to address in class *xx* has been removed from that class.
out of memory	The message implies that there is not enough memory to contain the text to be printed. Obtain more memory by archiving/removing files no longer needed.
printer *printer-name* already in class *xx*	The printer you are trying to move to class *xx* is already in that class. You cannot move a printer to a class that it is already in.

Error Message	Description/Action
printer *printer-name* has disappeared!	The printer has been removed, and the **enable** command cannot find it. The printer was most likely removed since the machine was rebooted or since the scheduler was started.
printer *printer-name* nonexistent	Printer-name is the name of a printer that has been removed since the scheduler has been started. You must use the **lpadmin** *-xprinter-name.*
printer status entry for *printername* has disappeared	The **/usr/spool/lp/pstatus** file must have been corrupted. You will have to resubmit the printer request.
printer *printer-name* was not busy	The printer is not printing a request at this time. Either the request you wanted to cancel is finished printing or you have specified the wrong printer.
request *printer-id* non-existent	You are attempting to cancel a request that does not exist. You may have given the wrong printer name or wrong request id number or the request may have finished printing.
request not accepted	The request was not accepted by **lp.** The scheduler may not be running. Use the **lpstat -t** command to find out more information.
requests still queued for *printer-name* -- use lpmove	*Printer-name* is the printer that still has requests waiting to get printed. You need to use the **lpmove** command to get those requests moved to another printer.

Error Message	Description/Action
scheduler is still running -- can't proceed	You cannot perform this command while the scheduler is running. You will have to use the **lpshut** command first.
spool directory non-existent	The directory **/usr/spool** has been removed. You will have to use the **mkdir** command to restore the directory. This has probably removed some of the necessary LP files. You may have to reinstall the LP commands.
standard input is empty	You specified an invalid file name either by incorrectly typing a name or by specifying a non-existent file. Nothing will be printed on the printers from this request.
this command for use only by LP Administrators	This command is restricted to someone logged in as root or lp.
too many options for interface program	The **lp** command called the appropriate interface program with too many arguments. For more information on the options and arguments that can be used with the **lp** command, refer to the **lp** manual page.
unknown keyletter *xx*	An invalid option was supplied to the **lp** or **lpadmin** commands (see **lp**(C) or **lpadmin**(M) in the *Reference (C)* and *Reference (M).*
unknown option *xx*	This message is displayed in response to an invalid option supplied to the **disable**(C), **lpstat**(C), or **reject**(C) commands. Refer to the *Reference (C)* for correct usage.

Error Message	Description/Action
usage: disable [-c] [-r[reason]] printer	The syntax for the **disable** command is not correct. The valid options are: **-c** to cancel the currently printing request, and **-r** followed by the reason that you are disabling the printer.
usage: reject [-r[reason]] dest...	The syntax for the **reject** command is not correct. The proper format is to specify the reason why the printer is not taking any more print requests and to identify the destination printer.
usage: accept dest	The syntax for the **accept** command is to specify a destination printer. You are setting up a printer to accept requests, and you did not specify what printer should accept requests.
usage: enable printer	The syntax for the **enable** program is to specify a destination printer.
usage: cancel id.... printer...	The syntax for the **cancel** command is not correct. The proper format is to specify the request identification number or the printer name.
usages: lpadmin -pprinter [-vdevice] [-cclass] [-rclass] [-eprinter | -iinterface | -mmodel] [-h | -l] -or- lpadmin -d[destination] -or- lpadmin -xdestination	The correct syntax for the **lpadmin** command is to specify at least one of the options mentioned.

Error Message	Description/Action
your printer request *printer-id* was cancelled by *user*	The printer request did not finish printing because another *user* cancelled it. Typically, you will get this message in your mail. One reason a person may cancel a request other than their own is because the request is not printing correctly.

LPR ERROR MESSAGES

When errors occur while you are using **lpr**, one or more of the following messages appear in the spool directory LOGFILE:

Error Message	Description/Action
Cannot store pid	A daemon could not write its process id number into a lock file.
Lock has disappeared	The lock file made by the running daemon has mysteriously vanished. This is because it has been removed, but the daemon is still running.
Cannot open lock	A lock file has been established in the spool directory, but an incoming daemon can't open it. Permissions on lock files should be 644.
Cannot open directory	The spool directory has not been created, or has been created with the wrong permissions. Permissions should be 777.

Error Message	Description/Action
Can't read *filename*	A file was created in the spool directory with incorrect permissions. All files in the spool directory should be 644.
Can't send *filename*	The printer device cannot be opened for writing.
Can't mail: *person*	The **mail** option was selected from the command line, but *person* either does not exist, or cannot be mailed to for some other reason.
Daemon killed, Now asleep, Now awake	These messages are the result of various signals sent to a daemon. The signals involved are: kill daemon and remove lock - SIGTERM (15) sleep daemon - SIGPIPE (13) wake daemon - SIGFPE (8)
Unable to fork	The daemon program could not fork to put itself in background. The process limit may have been reached.
printer device write error	In the middle of printing a file, the daemon got a write error on the tty.
Cannot restore tty modes, Cannot save tty modes	The daemon saves the current modes on a tty before changing them, and restores them after each file is printed. If these things can't happen, tty modes will probably not get set correctly on the port.

Error Message	Description/Action
Cannot fork to set modes for *filename*	The daemon couldn't fork an **stty** to set tty modes on a printer device. The process limit may have been reached.
Cannot reopen stdin as /dev/ttyn	Part of the process of running the **stty** command from the daemon requires that the tty be reopened as stdin. If this cannot happen, tty modes will not be set properly on the port.
Cannot exec stty for *filename*	The daemon couldn't find the stty program, which it expects to be in the /bin directory.
Daemon stty child interrupted	The **stty** command run by the daemon was prematurely terminated.

Appendix G
Porting Guide

INTRODUCTION

This appendix is written for customers who have XENIX System III or XENIX System V software and want to transfer that software to UNIX/XENIX System V/386. The Porting Guide describes software transitions in four major configurations:

From	To
386 Series XENIX System V.2/386	386 Series UNIX System V.3/386
286 Systems XENIX System III	386 Series UNIX System V.3/386
286 Systems XENIX System III	386 Series XENIX System V.2/386
8086 Systems XENIX System III	386 Series UNIX System V.3/386

Although much of the information required to make any of the above transitions is similar, separate sections in this appendix describe each configuration. Thus, you can read only the section that meets your particular needs.

NEW FILE FORMAT FOR UNIX SYSTEM V/386

Altos (UNIX) System V/386 introduces a new file format, Common Object File Format (COFF). COFF is the AT&T UNIX standard. The Altos strategy is to support COFF objects and binaries as the default format for UNIX System V/386. While our software developers are in transition from X.OUT to COFF, the UNIX System V/386 operating system and C compiler development environment will support both COFF and X.OUT formats. Software developers should move to COFF, so that Altos may better support your development efforts.

For more information regarding COFF, please consult the *C Compiler Library and User's Guide.* Please review the chapter titled "The Common Object File Format (COFF)." This manual is provided with the Altos (UNIX) System V/386 Development System.

XENIX SYSTEM V.2/386 TO UNIX SYSTEM V.3/386

Compatibility

Altos UNIX System V/386 protects your investment in application software with built-in compatibility with popular operating systems in the UNIX market today. The following matrix illustrates compatibility points:

Operating System	Compatibility with Altos UNIX V/386 Source	Object	Binary
Altos XENIX System III/286	Yes **	No	Yes
Altos XENIX System V/386	Yes	Yes *	Yes
SCO XENIX System V/286	Yes **	No	Yes
SCO XENIX System V/386	Yes	No	Yes
Intel/AT&T UNIX V.3	Yes ***	Yes	Yes
Microport UNIX V.3	Yes ***	Yes	Yes

* Object compatibility is possible with use of **-omf** option to the C compiler.

** If 80286 C source code does not depend on 16-bit integers, it will compile and run on Altos UNIX System V/386 without modification. The 80286 **far** and **near** keywords are ignored by the Altos UNIX V/386 C compiler.

*** The C compiler delivered with these products is based on the AT&T portable C compiler (pcc) while Altos XENIX uses the Microsoft C compiler. Depending on how C programs were written using pcc, some modifications may be necessary to run under UNIX V/386.

Binary Compatibility

All executables generated on XENIX System V/386 will run on UNIX System V/386 without modification.

Object File Compatibility

XENIX System V/386 object file compatibility is supplied on UNIX System V/386, within certain constraints. The Cmerge C Compiler supplied with the UNIX System V/386 Development System produces COFF object files by default. The option **-omf** has been supplied; when invoked this option will cause the C compiler to produce XENIX-style X.OUT object files.

Three utilities have been supplied in both COFF and X.OUT formats. The COFF versions are: **ld**(CP), **ar**(CP), and **nm**(CP). The X.OUT versions are: **xld**(CP), **xar**(CP), and **xnm**(CP). The C compiler will automatically invoke the **xld**(CP) linker if the **-omf** option has been specified. Both COFF and X.OUT libraries have been included; all X.OUT libraries begin with the letter 'S'. Also, both COFF and X.OUT include files have been provided; the COFF include files are in /usr/include, and the X.OUT include files are contained in /usr/include.xenix. The C compiler will use the X.OUT libraries and include files automatically if the **-omf** option has been specified.

NOTE

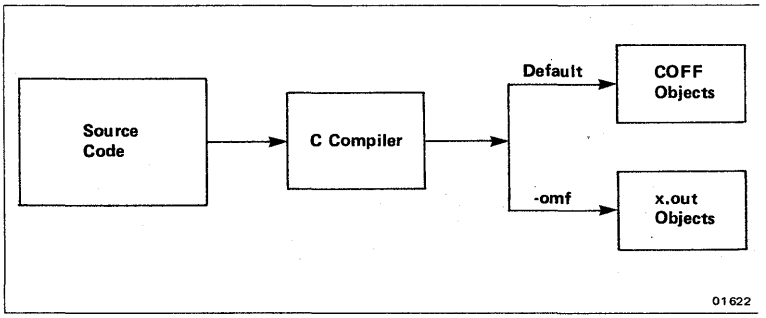
Under no circumstances can COFF object files be linked with X.OUT object files.

Data File Compatibility

The alignment of long data items within structures has changed from XENIX System V/386 to UNIX System V/386. Under XENIX, long items are aligned on a two-byte boundary. Under UNIX, long items are aligned on a four-byte boundary. This should be transparent unless the user is attempting to read a data file that was created by a XENIX program from a UNIX program. Even then, this issue should not be noticeable unless structures were written to the data file in binary format.

Source File Compatibility

UNIX System V/386 is fully source-compatible with XENIX System V/386. The use of the **near** and **far** keywords is discouraged. If your source code contains these keywords, it will still compile correctly with the **-Me** option, but these keywords will be ignored. If you do not use the **-Me** option, the compiler will abort with a syntax error on the lines where these keywords are used.



Conversion Utility

Object File Conversion Utility

A conversion utility, called **fixobj**(CP), is supplied with the release. This utility will convert X.OUT object files (.o files) to COFF object files.

CAUTION

Special care should be taken when using the fixobj(CP) utility, as it does nothing to resolve the data alignment issues mentioned previously.

System Limits

UNIX System V/386 system limits for the Altos 386 Series 1000:

	Memory Size		
	2MB	4MB	8MB (Or Greater)
Open Files Per User	50	50	50
Open Files Per System	250	500	800
Inodes	200	400	500
Record/File Locks	250	500	800
Processes Per User	30	30	30
Processes Per System	160	180	200
Buffers	116	244	500*

* For each additional 1MB of RAM, add 64 buffers.

UNIX System V/386 system limits for the Altos 386 Series 2000:

	Memory Size		
	2MB	8MB	16MB
Open Files Per User	50	50	50
Open Files Per System	500	1000	2000
Inodes	400	500	1000
Record/File Locks	500	1000	2000
Processes Per User	30	30	30
Processes Per System	180	200	400
Buffers	244	500	1024

XENIX SYSTEM III/286 TO UNIX SYSTEM V.3/386

There are two ways you can move your software from the 80286-based systems running XENIX System III to the Alto 386 Series running UNIX System V/386:

- Run 80286-based binaries in "compatibility mode" on the Altos 386 System.

  Choosing the compatibility mode option means that yo must focus on compatibility issues between XENIX System III and UNIX System V. This means less development time and a rapid entrance into the 80386 market. This is a good short-term option.

- Recompile your code on the Altos 386 System.

  Recompiling your source code is the recommended method for moving to UNIX System V. You will see substantial performance gains because the Altos 386 C Compiler takes full advantage of the enhanced instruction set of the 80386. This is the best long-term strategy; it will help you prepare for the evolving 80386-based competition.

Compatibility

Altos UNIX System V/386 protects your investment in appl cation software with built-in compatibility with popular operating systems in the UNIX market today. The followi matrix illustrates compatibility points:

Operating System	Compatibility with Altos UNIX V/386 Source	Object	Binary
Altos XENIX System III/286	Yes **	No	Yes
Altos XENIX System V/386	Yes	Yes *	Yes
SCO XENIX System V/286	Yes **	No	Yes
SCO XENIX System V/386	Yes	No	Yes
Intel/AT&T UNIX V.3	Yes ***	Yes	Yes
Microport UNIX V.3	Yes ***	Yes	Yes

* Object compatibility is possible with use of **-omf** option to the C compiler.

** If 80286 C source code does not depend on 16-bit integers it will compile and run on Altos UNIX System V/386 without modification. The 80286 **far** and **near** keywords are ignored by the Altos UNIX V/386 C compiler.

*** The C compiler delivered with these products is based on the AT&T portable C compiler (pcc) while Altos uses the Microsoft C compiler. Depending on how C programs were written using pcc, some modification may be necessary to run under UNIX V/386.

Special Files

There are some special files that have changed from XENIX System III/286 to UNIX System V386.

Crontab

The /usr/lib/crontab format has changed in UNIX System V. Any application that adds entries to /usr/lib/crontab under XENIX System III (at installation or at any other time during execution of the program) must use the new format. Please refer to the *Reference (C)*, **crontab**(C) for further information.

/etc/rc

Applications that update the /etc/rc script at installation must now update the /etc/rc2.d directory. For more information, refer to the *Reference (M)*, **rc0**(M) and **rc2**(M).

Inittab

The /etc/inittab file replaces /etc/ttys and /etc/ttytpe in UNIX System V. These files are still supplied on the Altos 386 for applications that use them, but the operating system itself no longer uses these files for tty information. The /etc/ttys and /etc/ttytype files are also included in UNIX System V and are updated by **pconfig**(C), but changing these files will not affect the system. Please refer to the *Reference (M)*, **inittab**(M) for more information on **inittab.**

Utmp (wtmp)

The /etc/utmp file format has changed in UNIX System V. Any programs that access the utmp file directly must be recompiled using the new utmp structure defined in /usr/include/utmp.h. The **getlogin**(S) and **cuserid**(S) library routines, which access the utmp file, have been updated in the 286 libraries included with the 286 C Compiler to read either XENIX System III or UNIX System V utmp format and respond accordingly. Programs that call **getlogin**(S) and **cuserid**(S) and were linked on the Altos 80286-based systems with a version earlier than 3.4b of the 286 C Compiler will not run in compatibility mode on the 386. These programs must be re-linked with the 3.4b version of the C Compiler on the 286, or recompiled on the Altos 386.

Libraries and System Calls

Brk System Call

The return value from the **brk**(S) system call has been changed in UNIX System V. In XENIX System III, the **brk**(S) routine returned a pointer to the beginning of newly allocated space. In UNIX System V, **brk**(S) returns 0 on no error and -1 if an error is detected. Upon successful completion, the 80386 kernel will return a pointer to the beginning of newly allocated space for those applications executing in compatibility mode (80286 binary), and return a value of 0 to the 80386 program. Applications using **malloc**(S) do not require modification prior to recompilation.

Lock System Call

The **lock**(S) system call takes different arguments in UNIX System V. In XENIX System III, the **lock**(S) system call with a zero argument will unlock the process, while the non-zero argument will lock the process in core. In UNIX System V, the lock system call with a zero argument will unlock the process, while an argument of one will lock the process in core.

Object Modules

The format of relocatable object modules (i.e., ".o" files) has changed for the 80386. Any .o files or libraries created on the Altos 80286-based systems cannot be linked with .o files or libraries you create on the Altos 386.

Record and File Locking

There are subtle differences between XENIX System III and UNIX System V related to the **locking**(S) system call. Most applications will not be affected by these differences, but for those that will be affected, the following information is included:

1. UNIX System V disallows LK_RLCK and LK_NBRLCK requests on a file that is not opened for reading (i.e., "read locks" are not allowed on "write-only" files). Likewise, UNIX System V disallows LK_LOCK and LK_NBLCK requests on a file that is not opened for writing (i.e., "write locks" are not allowed on "read-only" files). XENIX System III allows both of the above conditions.

2. On XENIX System III, if a negative number is specified for the number of bytes to lock, the entire file is locked. On UNIX System V, a negative numbe will lock that number of bytes before the current file position. Note that the correct way to lock the entire file is to specify zero (0) as the number of bytes to lock.

3. On XENIX System III, a "read lock" cannot overlap with any other locked region of the file. UNIX System V allows a "read lock" to overlap with anothe "read lock."

4. In XENIX System III, a LK_NBLCK request will retur the error EACCES if any part of the region is alread locked. On UNIX System V, the error EAGAIN is returned in such cases.

Terminfo

In UNIX System V, the **curses**(S) library uses /usr/lib/terminfo directory instead of the /etc/termcap file to derive terminal information. If your program is running in compatibility mode, you will not have a problem because /etc/termcap is included on the UNIX System V r lease. If you want to recompile your program on the 386 and continue using termcap instead of terminfo, you must link with the **ocurses** library instead of the **curses** library. Refer to the *Reference (M)*, **terminfo**(M) for more informa tion.

liscellaneous

ile System

If your application is shipped on a diskette with a mountable file system, you will not be able to mount that diskette on the Altos Series 386. Altos recommends that you distribute software on tar format diskettes to avoid this problem.

UNIX System V uses the standard UNIX file system format (XENIX System III used the Berkeley File System on the Altos 1086, 2086 and 3086 Systems). The block size is 2K in UNIX System V (XENIX System III was 1K). Programs dependent on the superblock or inode structures must be modified to use the new UNIX System V structure formats.

hared Data Segments

Both the Altos 80286-based systems and the Series 386 provide a maximum of 120 shared data segments per system. The Series 386 will allow the following number of shared data segments per process:

- 386 process - 11
- 286 process (large model) - 10
- 286 process (small model) - 11
- 286 process (small model - non-separate I & D) - 12

Data-type Compatibility

Some 80386 C data types have been modified in size from the 80286-based systems:

	286	386
ints	2 bytes	4 bytes
ptrs (small/medium models)	2 bytes	4 bytes
ptrs (large model)	4 bytes	4 bytes
far ptr	4 bytes	N/A

Floats and doubles are compatible between the Altos 80286 based and Altos Series 386. Shorts on both the Altos 80286-based and Altos 386 are two bytes long. To read data files created on the Altos 80286-based system in whic integers are written to files, you must redefine integers (ints) as shorts. It should not be necessary to use far pointers on the Altos Series 386.

Program Models

All programs compiled on the Altos 386 have one code seg ment and one data segment (i.e., small model). Their com bined size cannot be greater than the sum of physical men ory and disk swap space on your particular machine config uration. This sum cannot exceed three gigabytes.

Data File Compatibility

The alignment of long data items within structures has changed from XENIX System III to UNIX System V/386. Under XENIX III, long items are aligned on a two-byte boundary. Under UNIX V, long items are aligned on a fou byte boundary. This should be transparent, user is at- tempting to read a data file that was created by a XENIX program from a UNIX program. Even then, this issue shou not be noticeable unless structures were written to the data file in binary format.

System Limits

UNIX System V/386 system limits for the Altos 386 Series 1000:

	Memory Size		
	2MB	4MB	8MB (Or Greater)
Open Files Per User	50	50	50
Open Files Per System	250	500	800
Inodes	200	400	500
Record/File Locks	250	500	800
Processes Per User	30	30	30
Processes Per System	160	180	200
Buffers	116	244	500*

* For each additional 1MB of RAM, add 64 buffers.

UNIX System V/386 system limits for the Altos 386 Series 2000:

	Memory Size		
	2MB	8MB	16MB
Open Files Per User	50	50	50
Open Files Per System	500	1000	2000
Inodes	400	500	1000
Record/File Locks	500	1000	2000
Processes Per User	30	30	30
Processes Per System	180	200	400
Buffers	244	500	1024

XENIX SYSTEM III/286 TO XENIX SYSTEM V.2/386

There are two ways you can move your software from the 80286-based systems running XENIX System III to the Altos 386 Series running XENIX System V.2/386:

- Run 80286-based binaries in "compatibility mode" on the Altos 386 System.

  Choosing the compatibility mode option means that you must focus on compatibility issues between XENIX System III and XENIX System V. This means less development time and a rapid entrance into the 80386 market. This is a good short-term option.

- Recompile your code on the Altos 386 System.

  Recompilation is the most recommended method of conversion. Recompilation also means you will see substantial performance gains because the Altos 386 C Compiler takes full advantage of the enhanced instruction set of the 80386. This is the best long-term strategy; it will help you prepare for the evolving 80386-based competition.

Memory Recommendations

For each process that the Altos 386 system forks, there is an overhead of 24K that is not swapped or paged. This means the number of processes that can run on your system is directly related to the size of available physical memory. The following table shows the recommended number of users/RAM:

Number of Users	Recommended RAM
16 Users	4 MB
32 Users	8 MB
64 Users	16 MB

Therefore, it may be advantageous to have one large program rather than several small ones. In order to reduce the number of processes extra shells should be minimized. In addition, unused ports should not be enabled.

Special Files

There are some special files that have changed from XENIX System III/286 to XENIX System V/386.

Crontab

The /usr/lib/crontab format has changed in XENIX System V. Any application that adds entries to /usr/lib/crontab under XENIX System III (at installation or at any other time during execution of the program) must use the new format. Please refer to the *Reference (C)*, **crontab**(C) for further information.

/etc/rc

Applications that update the /etc/rc script at installation must now update the /etc/rc2.d directory. For more information, refer to the *Reference (M)*, **rc0**(M) and **rc2**(M).

Inittab

The /etc/inittab file replaces /etc/ttys and /etc/ttytpe in XENIX System V. These files are still supplied on the Altos 386 for applications that use them, but the operating system itself no longer uses these files for tty information. The /etc/ttys and /etc/ttytype files are also included in XENIX System V and are updated by **pconfig**(C), but changing these files will not affect the system. Please refer to the *Reference (M)*, **inittab**(M) for more information on **inittab**.

Utmp (wtmp)

The /etc/utmp file format has changed in XENIX System V. Any programs that access the utmp file directly must be recompiled using the new utmp structure defined in /usr/include/utmp.h. The **getlogin**(S) and **cuserid**(S) library routines, which access the utmp file, have been updated in the 286 libraries included with the 286 C Compiler to read either XENIX System III or XENIX System V utmp format and respond accordingly. Programs that call **getlogin**(S) and **cuserid**(S) and were linked on the Altos 80286-based systems with a version earlier than 3.4b of the 286 C Compiler will not run in compatibility mode on the 386. These programs must be re-linked with the 3.4b version of the C Compiler on the 286, or recompiled on the Altos 386.

Libraries and System Calls

Brk System Call

The return value from the **brk**(S) system call has been changed in XENIX System V. In XENIX System III, the **brk**(S) routine returned a pointer to the beginning of newly allocated space. In XENIX System V, **brk**(S) returns 0 on no error and -1 if an error is detected. Upon successful completion, the 80386 kernel will return a pointer to the beginning of newly allocated space for those applications executing in compatibility mode (80286 binary), and return a value of 0 to the 80386 program. Applications using **malloc**(S) do not require modification prior to recompilation.

Lock System Call

The **lock**(S) system call takes different arguments in XENIX System V. In XENIX System III, the **lock**(S) system call with a zero argument will unlock the process, while the non-zero argument will lock the process in core. In XENIX System V, the lock system call with a zero argument will unlock the process, while an argument of one will lock the process in core.

Object Modules

The format of relocatable object modules (i.e., ".o" files) has changed for the 80386. Any .o files or libraries created on the Altos 80286-based systems cannot be linked with .o files or libraries you create on the Altos 386.

Record and File Locking

There are subtle differences between XENIX System III and XENIX System V related to the **locking**(S) system call. Most applications will not be affected by these differences, but for those that will be affected, the following information is included:

1. XENIX System V disallows LK_RLCK and LK_NBRLCK requests on a file that is not opened for reading (i.e., "read locks" are not allowed on "write-only" files). Likewise, XENIX System V disallows LK_LOCK and LK_NBLCK requests on a file that is not opened for writing (i.e., "write locks" are not allowed on "read-only" files). XENIX System III allows both of the above conditions.

2. On XENIX System III, if a negative number is specified for the number of bytes to lock, the entire file is locked. On XENIX System V, a negative number will lock that number of bytes before the current file position. Note that the correct way to lock the entire file is to specify zero (0) as the number of bytes to lock.

3. On XENIX System III, a "read lock" cannot overlap with any other locked region of the file. XENIX System V allows a "read lock" to overlap with another "read lock."

4. In XENIX System III, a LK_NBLCK request will return the error EACCES if any part of the region is already locked. On XENIX System V, the error EAGAIN is returned in such cases.

Terminfo

In XENIX System V, the **curses**(S) library uses /usr/lib/terminfo directory instead of the /etc/termcap file to derive terminal information. If your program is running in compatibility mode, you will not have a problem because /etc/termcap is included on the XENIX System V release. If you want to recompile your program on the 386 and continue using termcap instead of terminfo, you must link with the **ocurses** library instead of the **curses** library. Refer to the *Reference (M)*, **terminfo**(M) for more information.

Miscellaneous

File System

If your application is shipped on a diskette with a mountable file system, you will not be able to mount that diskette on the Altos Series 386. Altos recommends that you distribute software on tar format diskettes to avoid this problem.

XENIX System V uses the standard UNIX file system format (XENIX System III used the Berkeley File System on the Altos 1086, 2086 and 3086 Systems). The block size is 2K in XENIX System V (XENIX System III was 1K). Programs dependent on the superblock or inode structures must be modified to use the new XENIX System V structure formats

Shared Data Segments

Both the Altos 80286-based systems and the Series 386 provide a maximum of 120 shared data segments per system. The Series 386 will allow the following number of shared data segments per process:

- 386 process - 11
- 286 process (large model) - 10
- 286 process (small model) - 11
- 286 process (small model - not separate I & D) - 12

Data-type Compatibility

Some 80386 C data types have been modified in size from the 80286-based systems:

	286	386
ints	2 bytes	4 bytes
ptrs (small/medium models)	2 bytes	4 bytes
ptrs (large model)	4 bytes	4 bytes
far ptr	4 bytes	N/A

Floats and doubles are compatible between the Altos 80286-based and Altos Series 386. Shorts on both the Altos 80286-based and Altos 386 are two bytes long. To read data files created on the Altos 80286-based system in which integers are written to files, you must redefine integers (ints) as shorts. It should not be necessary to use far pointers on the Altos Series 386.

Program Models

All programs compiled on the Altos 386 have one code segment and one data segment (i.e., small model). Their combined size cannot be greater than the sum of physical memory and disk swap space on your particular machine configuration. This sum cannot exceed three gigabytes.

XENIX SYSTEM III FOR 8086-BASED SYSTEMS TO UNIX SYSTEM V.3/386

It's best to recompile when moving your software from the Altos 8086-based systems to the Altos 386 Series. Substantial performance gains will be realized when you complete this process, because the Altos C Compiler takes full advantage of the enhanced instruction set of the 80386.

Binary Compatibility

UNIX System V.3/386 is binary compatible with XENIX System III on the 8086-based systems.

Object Compatibility

There is no object compatibility between UNIX System V.3/386 and XENIX System III.

Data File Compatibility

Data file compatibility must be evaluated on a case-by-case basis. It may be necessary to develop a data file conversion utility to run data files created on 8086-based systems on the 386 Series.

System Limits

UNIX System V/386 system limits for the Altos 386 Series 1000:

	Memory Size		
	2MB	**4MB**	**8MB (Or Greater)**
Open Files Per User	50	50	50
Open Files Per System	250	500	800
Inodes	200	400	500
Record/File Locks	250	500	800
Processes Per User	30	30	30
Processes Per System	160	180	200
Buffers	116	244	500*

* For each additional 1MB of RAM, add 64 buffers.

UNIX System V/386 system limits for the Altos 386 Series 2000:

	Memory Size		
	2MB	**8MB**	**16MB**
Open Files Per User	50	50	50
Open Files Per System	500	1000	2000
Inodes	400	500	1000
Record/File Locks	500	1000	2000
Processes Per User	30	30	30
Processes Per System	180	200	400
Buffers	244	500	1024

TERMS

COFF

This is the AT&T UNIX standard Common Object File Format.

Compatibility

Binary — A binary file is an executable file; it is produced by the linker. An operating system is "binary compatible" with another operating system if it can execute programs that also execute on the other operating system.

Object — An object file is a file produced by the compiler (with UNIX, object files typically have a ".o" suffix). Several object files are linked together (by the program **ld**) to create an executable file. An object file is compatible between two operating systems if it can be linked on either system with other object files.

Source — A source file is a file that contains the program written in a language - such as "C". An operating system is "source compatible" with another system if it can compile and execute source programs that can be compiled and executed on the other operating system.

Demand Paging

Both Altos 80286-based and the Series 386 programs are demand paged. The only difference between the two is in implementation. The 80286-based programs begin with all code and data in memory and are demand paged to and from the swap area. When 386 programs begin, a small working set is loaded into memory and then the program is demand paged from the file system.

Implementation of demand paging on the Altos Series 386 allows much larger programs than are possible on the Altos 80286-based systems. Due to the demand paging capabilities, Altos recommends that frequently used routines be grouped together to improve performance.

Virtual Memory
Virtual memory allows you to run programs that are larger than physical memory, that is, an address space is built that only requires a small amount of physical memory (where the program is executing) to be "physically" resident in the machine's RAM.

x.out
This is the XENIX standard object file format.

8086-based systems
References to 8086-based systems include the Altos 486, 586, 586T, 986, and 986T.

80286-based systems
References to 80286-based systems include the Altos 686, 886, 1086, 2086 and 3086.

80386-based systems
References to 80386-based systems include Altos 386 Series 1000 and 386 Series 2000.

(BLANK)

Glossary

A

address
A number, label, or name that indicates the location of information in the computer's memory.

advertise
To make resources available from a local host to other hosts in a Remote file Sharing environment.

a.out
The default name of a freshly compiled object file, pronounced 'A-dot-out'; historically a.out signified assembler output.

archive
1. A collection of data gathered from several files into one file. 2. Especially, such a collection gathered by **ar**(CP) for use as a library.

automatic calling unit
A hardware device used to dial stored telephone numbers; allows the system to contact another system over phone lines without manual intervention.

B

bad block
A section of a storage medium which cannot store data reliably.

block
The basic unit of buffering in the kernel, 2048 bytes; see indirect, logical, and physical blocks.

block device
A device upon which a **file system** [1] can be mounted typically a permanent storage device such as a tape or disk drive, so called because data transfers to the device occur by character device.

boot
To start the operating system, so called because the kernel must bootstrap itself from secondary storage into an empty machine. No **login** [3] or process persists across a boot.

boot program
A program that loads the operating system into core.

buffer
1. Arbitrary-length transactions are collected into convenient units for system operations; the **file system** [3] uses buffers, as does stdio. 2. As a verb: to use buffers.

buffer pool
A region of storage available to the **file system** [3] for holding blocks; all but raw input-output for block devices goes through the buffer pool so read and write operations may be independent of device blocks.

C

cartridge tape
A storage medium that consists of a magnetic tape wound on spools housed in a plastic container.

character device
A device upon which a **file system** [1] cannot be mounted such as a terminal or the null device.

child process
See **fork.**

client
A host that has mounted an advertised resource from another host in a Remote File Sharing environment.

command
1. An instruction to the shell, usually to run a **program** [1] as a child process. 2. By extension, any executable file, especially a utility program.

command file
See **shell script.**

configuration
The arrangement of the software or hardware of a system, peripheral, or network as defined by the nature, number, and chief characteristics of its functional units.

controller
A device that directs the transmission of data over the data links of a network.

core file
For debugging; a core file is created under the name "core" in the current directory of the process.

core image
A copy of all the segments of a running or terminated program; the copy may exist in main storage, in the swap area, or in a core file.

crash
If a hardware or software error condition develops that the system can't handle, it takes itself out of service, or crashes. Such conditions occur when the system can't allocate resources, manage processes, respond to requests for system functions, or when the electrical power is unstable.

cron
A command that creates a daemon, which invokes commands at specified dates and times.

cylinder
The set of all tracks on a disk that are the same distance from the axis about which the disk rotates.

D

daemon
A background process, often perpetual, that performs a system-wide public function, such as **calendar**(C) and **cron**(C).

destination
The remote system that ultimately receives a file transferred over a network.

device
1. A **file** [2] that is not a plain file or a directory, such as a tape drive, or the null device; a special file. 2. A physical input-output unit.

diagnostic
A message printed at your terminal that identifies and isolates program errors.

directory
A file that comprises a catalog of **filenames** [2]; the organizing principle of the **file system** [2], a directory consists of entries which specify further files, and constitutes a node of the directory tree.

directory entry, entry
1. An association of a name with an inode number appearing as an element of a directory. 2. The nam part of such an association.

directory hierarchy
The tree of all directories, in which each is reachable from the root via a chain of subdirectories.

directory tree
See **directory hierarchy.**

disk
A platter coated with magnetic material on which dat can be stored.

diskette
A magnetic storage medium which is smaller and more flexible than a hard disk.

domain
A logical grouping of hosts in a Remote File Sharing environment. Each host in a domain relies on the same domain name server(s) for certain resource sharing and security services. Each domain has one primary and zero or more secondary domain name servers.

domain name server
A computer that creates and maintains the following information for hosts in a Remote File Sharing domain: advertised resources, host names and passwords, names and addresses for name servers of other domains (optional), host user and group information used for ID mapping (optional).

drive
The hardware device that holds magnetic disks, diskettes, and tapes while they are in use.

dump
A copy of the core image of the operating system.

E

environment
1. A set of strings, distinct from the arguments, made available to a process when it **executes** [2] a file; the environment is usually inherited across **exec**(S) operations. 2. A specific environment maintained by the shell.

error
An error occurs when a hardware or software condition prevents the successful execution of a system or a user process.

error message
A message sent from the system to the system console or a terminal when an error occurs.

exec
A system call that allows the user to request the execution of another program.

executable file
1. An object file that is ready to be copied into the address space of a process to run as the code of that process. 2. A file that has execute permission, either an executable file [1] or a shell script.

execute
1. Informally, to run a program. 2. To replace the text segment and data segments of a process with a given program.

F

FIFO
A named permanent pipe that allows two unrelated processes to exchange information using a pipe connection.

file
1. In general, a potential source of input or destination for output. 2. Most specifically, an inode and/or associated contents, i.e., a plain file, a special file, or a directory. 3. A directory entry; several directory entries may name the same file.

file descriptor
A conventional integer quantity that designates an open file.

filename
1. A pathname. 2. The last component name in a pathname.

file system
1. A collection of files that can be mounted on a block special file; each file of a file system appears exactly once in the i-list of the file system and is accessible via some path from the root directory of the file system. 2. The collection of all files on a computer. 3. The part of the kernel that deals with file systems.

filter
A **program** [1] that reads from the standard input and writes on the standard output, so called because it can be used as a data-transformer in a pipeline.

firmware
Programs contained in Read Only Memory (ROM), usually used to initially start the system on powerup.

flush
To empty a buffer, for example to throw away unwanted input-output upon interrupt or to release output from the clutches of stdio.

fork
To split one process into two, the parent process and child process, with separate, but initially identical, text, data, and stack segments.

formatting
The process of imposing an addressing scheme on a disk. This includes the establishment of the mapping of both sides of the disk into tracks and sectors.

free list
In a **file system** [1], the list of blocks that are not occupied by data.

G

getty
One of a series of processes that connect the user to the operating system. **Getty**(M) is invoked by **init**(M), and in turn invokes **login**(M).

group
1. A set of permissions alternative to owner permissions for access to a file. 2. A set of userIDs that may assume the privileges of a group. 3. The groupID of a file.

group id
An integer value, usually associated with one or more login names; as the user id of a process becomes the owner of files created by the process, thus the group id of a process becomes the **group** [3] of such files.

H

hole
A gap in a file caused by seeking while writing; **read**(S) takes data in holes to be zero; a block in a hole occupies no space in its file system.

host
A computer that is configured to share resources in a Remote File Sharing environment.

I

ID mapping
A means of setting the permissions that each remote user and group will have for a host's advertised resources in a Remote File Sharing environment.

i-list
The index to a **file system** [1] listing all the inodes of the file system; see **inode number.**

indirect blocks
Data blocks that are not directly referenced by an inode (because the file is larger than ten 2048-byte blocks); the inode has three addresses that indirectly reference (by a cascade of pointers) some 134,480,384 data blocks (an extremely large potential file size). The inode has one address that points to 512 more data blocks; a second address that points to 512 blocks that each point to 512 data blocks; and, finally, a third address that points to 512 blocks each of which point to another 512 blocks, each of which point to 512 data blocks. The actual maximum size of a file, however, is only 2 gigabytes.

init
A general process spawner that is invoked as the last step in the boot procedure; it regularly checks a table that defines what processes should run at what run level.

inode
An element of a **file system** [1]; an inode specifies all properties of a particular **file** [2] and locates the file's contents, if any.

inode number, i-number
The position of an inode in the i-list of a file system.

instruction
The data that actually tells the computer what to do.

integrity
In a file system, the quality of being without errors as a result of bad blocks.

interface programs
Shell scripts furnished with the LP spooling software which interface between the user and the printer.

InterProcess Communication
InterProcess Communication (IPC) describes software that enables independent processes running at the same time, to exchange information through messages, semaphores, or shared memory.

interrupt
1. A signal that normally terminates a process, caused by a break or an interrupt character. 2. A signal generated by a hardware condition or a peripheral device. 3. Any signal.

K

kernel
The operating system proper; resident code that implements the system calls.

kernel address space
A portion of memory used for data and code addressable only by the kernel.

L

line discipline
A module to handle protocol or data conversion for a stream. A line discipline, unlike a filter, is part of the kernel.

link

1. To add an entry for an existing file to a directory; converse of unlink. 2. By extension, a directory entry. 3. Any but one primary directory entry for a given inode; either linked or a symbolic link.

link count

The number of directory entries that pertain to an inode; a file ceases to exist when its link count becomes zero and it is not open.

log files

Contain records of transactions that occur on the system; software that spools, for example, generates various log files.

logical block

A unit of data as it is handled by the software; the operating system handles data in 2048-byte logical blocks.

login

1. The program that controls logging in. 2. The act of logging in. 3. By extension, the computing session that follows a login.

M

memory

1. Same as **memory image.** 2. Physical memory represents the available space in main memory; programs are either swapped or paged into physical memory for execution. 3. Virtual memory management techniques permit programs to treat disk storage as an extension of main memory.

memory image

See **core image.**

mode, file mode

The permissions of a file; usually referred to by a three-digit octal number, e.g., "a 755 file"; see **chmod**(C).

mount
To extend the directory hierarchy by associating the root of a **file system** [1] with a directory entry in an already mounted file system; the converse is unmount, spelled "umount."

N

namelist
See **symbol table.**

network
The hardware and software that constitute the interconnections between computer systems, permitting electronic communication between the systems and associated peripherals.

networking
For computer systems, means sending data from one system to another over some communications medium (coaxial cable, phone lines, etc.). Common networking services include file transfer, remote login, remote execution.

node name
An up-to-eight character name for the system; used as the official name of the machine in a network. The node name resides in the NODE parameter.

null device
A **device** [1] that always yields end of file on reading and discards all data on writing.

O

object file
A file of machine language code and data; object files are produced from source programs by compilers and assemblers, and from other object files.

operating system
The program for managing the resources of the computer. It takes care of such things as input/output procedures, process scheduling, and the file system.

open file
1. The destination for input or output obtained by opening a file or creating a pipe; a file descriptor; open files are shared across forks and persist across executes. 2. A file that has been opened, however an open file need not exist in a **file system** [1], and a file may be the destination of several open files simultaneously.

other
1. A set of permissions regulating access to a file by processes with userID different from the owner, and groupID different from the group of the file. 2. The customary name of the default **group** [2] assigned upon login.

owner
The userID of the process that created a file; the owner has distinctive permissions for a file.

P

page
A fixed length, 4096-byte block that has a virtual address, and that can be transferred between main and secondary storage.

paging
The process by which programs are truncated into pages and transferred between main and secondary storage by the virtual handler (or paging daemon).

parent process
See **fork.**

partitions
Units of storage space on disk.

path, pathname
A chain of names designating a file; a relative pathname leads from the current directory, for example, a path to directory A, thence to directory B, thence to file C is denoted A/B/C; a full pathname begins at the root, indicated by an initial backslash (/), as in /A/B/C.

permission
A right to access a file in a particular way; read, write, execute (or look up in, if a directory); permissions are granted separately to owner, group, and others. Each permission is encoded into one bit in an inode.

physical block
See **sector.**

physical
See **memory.**

pipe
A direct stream connection between processes, whereby data written on an open file in one process becomes available for reading in another.

pipeline
A sequence of programs connected by pipes.

polling
The interrogation of devices by the operating system to avoid contention, determine operation status, or ascertain readiness to send or receive data.

ports
The point of physical connection between a peripheral device (such as a terminal or a printer) and the device controller (ports board), which is part of the computer hardware.

primary name server
The computer on which administration for a Remote File Sharing domain is performed.

process
A connected sequence of computation; a process is characterized by a core image with instruction location counter, current directory, a set of open files, control terminal, userID, and groupID. The process id is an integer that identifies a process.

process number
See **process id.**

profile
1. An optional shell script, ".profile," conventionally used by the shell upon logging in to establish the environment and other working conditions customary to a particular user. 2. To collect a histogram of the values of a process's instruction location counter.

program
1. An executable file. 2. A process.

Q

queue
A line or list formed by items in a system waiting for service.

R

raw device
A block device, read and write operations to which are not buffered, and are synchronized to natural records of the physical device.

reboot
To restart the system. See **boot.**

region
A group of machine addresses that refer to a base address, such as the text or data regions of a **process.**

release
A distribution of fixes or new functions for an existing software product.

Remote File Sharing
A software utilities package that enables computers to share resources across a network.

resource
A directory that is advertised in a Remote File Sharing environment. When a resource is mounted on a client, the contents of the directory (files, devices, and named pipes) and any of its subdirectories are potentially available to users on the client.

retension
The process of re-winding the tape in a cartridge tape device to make sure it is at the correct tautness for accurate recording of data.

root
1. A distinguished directory that constitutes the origin of the directory hierarchy in a **file system** [1]. 2. Specifically, the origin for the **file system** [2], with the conventional pathname "/". 3. The origin of the directory hierarchy in a **file system** [1]. 4. The login name for the super-user.

rotational gap
The gap between the actual disk locations of blocks of data belonging to the same file. The rotational gap compensates for the continuous, high-speed rotation of the disk so that when the controller is ready to refer to the next physical block, the read-write head is positioned correctly at the beginning of that block.

run level
A software configuration of the system that allows a particular group of processes to exist.

S

schedule
To assign resources (main storage and CPU time) to processes.

scheduler
A permanent process, with process number 1 and associated kernel facilities, that does scheduling.

search path
In the shell, a list of pathnames of directories that determines the meaning of a command; the command name is prefixed with members of the search path in turn until a pathname of an **executable file** [2] results; the search path is given by the shell variable PATH.

secondary name server
A host that is configured to take over domain name server responsibilities temporarily in case the primary name server goes down.

section, sector
A 512-byte portion of a track which can be accessed by the magnetic disk heads in the course of a predetermined rotational displacement of the storage device.

segment
A contiguous range of the address space of a process with consistent store access capabilities; the four segments are (1) the text segment, occupied by executable code, (2) the data segment, occupied by static data that is specifically initialized, (3) the bss segment, occupied by static data that is initialed by default to zero values, and (4) the stack segment, occupied by automatic data, see **stack**; sometimes (2), (3), and (4) are collectively called data segments.

semaphore
An IPC facility which allows two or more processes to be synchronized.

server
A host that is actively sharing one of its advertised resources with another host in a Remote File Sharing environment.

set userid
A special permission for an **executable file** [1] that causes a process executing it to have the access rights of the owner of the file; the owner's user id becomes the effective user id of the process, distinguished from the real user id under which the process began.

set userid bit
The associated permission bit.

shared memory
An IPC facility which allows two or more processes to share the same data space.

shell
1. The program **sh**(C), which causes other programs to be executed on command; the shell is usually started on a user's behalf when the user logs in. 2. By analogy, any program started upon logging in.

shell script
An executable file of commands taken as input to the shell.

signal
An exceptional occurrence that causes a process to terminate or divert from the normal flow of control; see **interrupt, trap.**

single-user
A state of the operating system in which only one user is supported.

source file
1. The uncompiled version of a program. 2. Generally, the unprocessed version of a file.

special file
An inode that designates a device, further categorized as either a block special file describing a block device, or a character special file describing a character device.

spool
To collect and serialize output from multiple processes competing for a single output service.

spool area
A directory in which a spooler collects work.

spooler
A daemon that spools.

stack
A region of the address space into which automatic data and subroutine linkage information is allocated in last-in-first- out fashion; the stack typically occupies the largest data addresses and grows downward towards static data.

standard input, output, and error
Open files, customarily available when a process begins, with file descriptors 0, 1, 2 and stdio names 'stdin', 'stdout', 'stderr'; where possible, utilities by default read from the standard input, write on the standard output, and place error comments on the standard error file. Initially, all three of these files default to your terminal.

startup
See **boot.**

sticky bit
A permission flag that identifies a file as a sticky file.

sticky file
A special permission for a shared text file that causes a copy of the text segment to be retained in the swap area to improve system response.

super-block
The second block in a **file system** [1], which describes the allocation of space in the file system; see **boot block.**

super-user
User id 0 - can access any file regardless of permissions and can perform certain privileged system calls, for example, setting the clock. Root is typically the login name for the super-user.

swap
To move the core image of an executing program between main and secondary storage to make room for other processes. See **paging.**

swap area
The part of secondary store to which core images are swapped; the swap area is disjointed from the file system.

symbolic link
An inode that contains the pathname of another. References to the symbolic link become references to the named inode.

symbol table
Information in an object file about the names of data and functions in that file; the symbol table and address relocation information are used by the link editor to compile object files and also by debuggers.

system calls
1. The set of system primitive functions through which all system operations are allocated, initiated, monitored, manipulated, and terminated. 2. The system primitives invoked by user processes for system-dependent functions, such as I/O, process creation, etc.

system console
The directly connected terminal used for communication between the operator and the computer.

system name
An up-to-eight character name for the system; resides in the SYS parameter.

T

table
An array of data, each item of which may be uniquely identified by means of one or more arguments.

text file
A file whose bytes are understood to be in ASCII code.

track
An addressable ring of sections on a disk or diskette; each disk or diskette has a predefined number of concentric tracks, which allow the disk head to properly access sections of data.

trap
A method of detecting and interpreting certain hardware and software conditions via software; a trap is set to catch a signal (or interrupt), and determine what course of action to take.

tunable parameters
Variables that are used to set the sizes and thresholds of the various control structures of the operating system.

tuning
1. Modifying the tunable parameters so as to improve system performance. 2. The reconfiguration of the operating system to incorporate the modifications into an executable version of the system.

U

user id
An integer value, usually associated with a login name; the user id of a process becomes the owner of files created by the process and its descendent (forked) processes.

utility, utility program
A standard, generally useful, permanently available program.

V

version
A separate program product, based on an existing one, but containing significant new code or new functions.

virtual memory
See **memory.**

Index

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

W

Change Information

This is a summary of the changes that have been made to the previous version of this manual. The chapters, page numbers, and/or paragraphs mentioned in this summary reference the previous manual.

Title: Altos System V Series 386 Operations Guide

Revised Part Number: 690-21171-007

Previous Part Number: 690-21171-006

Date: June 1989

Changes:

Run level 0 is now the preferred system state for gracefully halting the system. Run level 5, the previous system halt run level, is still present for compatibility.

The exact file names for scripts contained in **/etc/rc?.d** directories may vary from release to release, but always conform to the general naming style given in examples.

Changed the following pages:

Page	Description
2-6 2-17	The **shutdown** command uses the **-g** option to specify the number of seconds to wait before shutting down the system.
2-13	The scripts in **/etc/rc2.d** no longer start **gettys** or hard disk daemons.
5-6	Removed reference to setting the PATH variable for the "adm" login, since this is a dummy login only.

9-6 The suggested buffer space allocated for a 4 MB system has changed from 600 KB to 500 KB. The suggested number of buffers changed from 125 to 250.

9-6
9-13
9-33 Removed references to machines with less than 2 MB of RAM. New examples now describe a system with 4 MB of RAM and an 80 MB hard disk (instead of 2 MB of RAM and 30 MB hard disk).

9-26 Changed the example used in "Sample System Reconfiguration."

9-27
9-28 Deleted the description of a group of commands (**mkboot, ldunix, mkunix,** and **ln**) shown after you execute **make,** since these commands are not meant to be executed by a user; they are automatically executed by **make.**

9-31
9-33 Values calculated at boot-time were previously shown in Figure 9.2 as "CALCULATED" but are actually represented in the table as "0" (zero).

9-31 Re-emphasized that the values in Figure 9.2 are valid for most Altos systems, but that your system may differ. Refer to your Release Notes.

9-33 Inserted values for the MAXBUF parameter.

9-50 Swap space is calculated according to each active user. Other new information regarding swap space was added.

A-7 The **/usr/lib/cron/log** file records **cron** information, not accounting information.

B-43 When using **uucico** in the MASTER mode, you must also specify a site name with the -s option.

B-49 Added appropriate character strings for Penril modems suitable for the Dialers file.

D-3 The file Transfer Program for Altos 386 machines has been renamed to **aftp.**

READER'S COMMENTS

Manual Title: Altos System V Series 386 Operations Guide

Part Number: 690-21171-007

Altos Computer Systems' Publications Department wants to provide documents that meet the needs of all our customers. Your comments help us produce better manuals.

Please Rate This Manual:	Excellent	Good	Average	Fair	Poor
Completeness of information	☐	☐	☐	☐	☐
Organization of manual	☐	☐	☐	☐	☐
Adequate illustrations	☐	☐	☐	☐	☐
Overall manual	☐	☐	☐	☐	☐

Do you find any of the chapters confusing or difficult to use? If so, which ones and why?

_____

_____

What could we do to improve the manual for you?

_____

_____

If you find errors or other problems when using this manual, please write them below. Do include page numbers or section titles.

_____

_____

Name: _____ Title: _____

Company: _____ Type of system: _____

Phone: (_____) _____ - _____ ext. _____

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

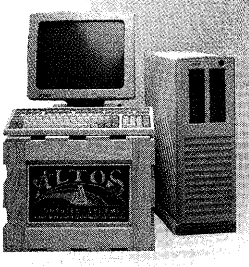
FIRST CLASS PERMIT NO. 7399 SAN JOSE, CA 95134

POSTAGE WILL BE PAID BY ADDRESSEE

Altos Computer Systems
ATTN: PUBLICATIONS DEPARTMENT
2641 Orchard Parkway
San Jose, CA 95134-9987
USA

Fold Here

P/N 690-21171-007
Printed in U.S.A.
9/89



Altos Computer Systems
2641 Orchard Parkway, San Jose, CA 95134
408/946-6700, FAX 408/433-9335