# HP 9000 Series 800

## System Administrator's Manual

HEWLETT
PACKARD

Edition 4 E1188

92453-90004
Printed in U.S.A. 11/88

## Notice

## Printing History

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition or a new update is published. No information is incorporated into a reprinting unless it appears as a prior update; the edition does not change when an update is incorporated.

The software code printed alongside the date indicates the version level of the software product at the time the manual edition or update was issued. Many product updates and fixes do not require manual changes, and conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual updates.

| | | |
|---|---|---|
| First Edition | November 1986 | |
| Update 1 | December 1986 | |
| Second Edition | July 1987 | |
| Update 1 | November 1987 | |
| Third Edition | February 1988 | A.B2.00 |
| Fourth Edition | November 1988 | A.B3.00 |

# Preface

This manual describes the responsibilities and duties of the System Administrator. To use it, you should be familiar with the basic commands and structure of HP-UX as it functions on HP 9000 Series 800 computers. However, the material is presented with step-by-step instructions for the training of a new System Administrator. The manual is organized as follows:

| | |
|---|---|
| **Chapter 1** | presents introductory information, such as the responsibilities of the System Administrator, conventions used in this manual, and a user survey form. |
| **Chapter 2** | tells you how to get started, where to go for installation and update information, describes tasks you need to perform after installation, and discusses the System Administration Manager (SAM). |
| **Chapter 3** | describes the system boot and the system shut down procedures. |
| **Chapter 4** | explains how to customize your system to fit your users' needs. |
| **Chapter 5** | presents some tasks you will perform as the System Administrator. These include: adding and removing users, adding and removing peripheral devices, and backing up the file system. |
| **Chapter 6** | describes the system accounting features of HP-UX. These features allow you to keep track of who is using the system, when they are using it, and how they are using it. You can also keep track of incorrect login tries and charge users for computer time and/or file space. |
| **Chapter 7** | describes system reconfiguration. It explains how to reconfigure the kernel using the **uxgen** command. It also explains how to restructure the file system; and creating, mounting, and unmounting file systems. |
| **Chapter 8** | presents some troubleshooting techniques. These include system panic, power fail recovery, and file recovery. |
| **Chapter 9** | presents systems management concepts, such as the superuser, processes, file system implementation, and memory management. |
| **Appendix A** | lists error messages, possible causes, and may recommend an action to resolve the error. |
| **Appendix B** | lists system accounting files. |
| **Appendix C** | explains the **fsck** command. |
| **Appendix D** | defines the system parameters. |

## Additional Documentation

The *Documentation Guide* (92453-90009) contains a list of HP 9000 Series 800 documentation. The manuals described below may also provide information that will help you perform your duties as System Administrator.

The *Installation and Configuration Guide* for your Series 800 computer tells you how to install computer hardware, interface cards, and peripherals. This manual supplies the hardware-specific information you will need to set up the HP-UX system.

*Installing and Updating HP-UX* (92453–90019) explains how to install, update, and modify the HP-UX operating system.

*System Administration Basics: Using SAM* (92453–90017) describes the System Administration Manager (SAM). SAM enables you to perform system administrative tasks using menus. Some of the tasks you can do using SAM are adding users, creating a new file system, review disk information, back up the system, add terminals, and set up UUCP.

The *HP-UX Reference* (09000–90009) contains syntactic and semantic details of all commands and application programs, system calls, subroutines, special files, file formats, miscellaneous facilities, and system maintenance procedures for the HP-UX operating system. Command descriptions, called "man pages" are also available online (use the **man** command). Section 1M of the *HP-UX Reference* contains descriptions of commands and programs used mainly for system maintenance and administration.

The *HP-UX User's Guide* (92453-90001) contains tutorials on user tasks and commands. A simple tutorial section is provided for first–time users. This section provides users with hands-on experience to gain familiarity with the system. A more detailed tutorial is also provided in this manual for those wishing to learn additional HP-UX commands.

The *HP-UX Real-Time Programming Manual* (92453-90003) explains the real-time programming features and capabilities of HP-UX. It contains programming examples to help users write realtime application programs.

The *HP-UX Concepts and Tutorials* set contains information on a broad range of HP-UX topics and tools.

| | | |
|---|---|---|
| *Concepts and Tutorials:* | *Text Editors and Processors* | 97089-90022 |
| *Concepts and Tutorials:* | *Text Formatters* | 97089-90032 |
| *Concepts and Tutorials:* | *Programming Environment* | 97089-90042 |
| *Concepts and Tutorials:* | *Device I/O and User Interfacing* | 97089-90052 |
| *Concepts and Tutorials:* | *UUCP* | 97089-90053 |
| *Concepts and Tutorials:* | *Shell and Miscellaneous Tools* | 97089-90062 |
| *Asynchronous Serial Communications Programming Manual* | | 92453-90002 |

Manuals provided with optional feature products, such as Network Services (NS), provide information specific to the product.

In addition to these manuals, your documentation package includes the appropriate manuals for the programming languages included with your system.

# Contents

# Contents (continued)

# Contents (continued)

# Contents (continued)

# Contents (continued)

# Contents (continued)

# Contents (continued)

# Contents (continued)

# Contents (continued)

# Contents (continued)

# Contents (continued)

# Contents (continued)

# Tables

# Figures

1

# Introduction

This manual is written for the HP-UX System Administrator. It is written to serve people with different levels of expertise. It contains procedures that help someone who is not an expert on HP-UX (or any other operating system) to install the HP-UX software and start up the system. For those who are familiar with computers but are new to HP-UX, system management concepts and descriptions are provided to help them manage the HP-UX system effectively. For those familiar with the UNIX* operating system, detailed information about HP-UX is provided to help you get the most out of system resources and to fine-tune the HP-UX system. This manual also contains information that will help you identify and solve system problems.

## The System Administrator

The System Administrator is responsible for installing the HP-UX operating system software, configuring the operating system for the proper applications, updating the software, tuning the system for optimum performance, maintaining the system, and repairing the system when something goes wrong. Additionally, the System Administrator should be the HP-UX expert who helps other HP-UX users.

To perform the system administration tasks effectively, the System Administrator must meet the following qualifications:

- Have a thorough user-level knowledge of HP-UX commands
- Be familiar with Series 800 computer hardware and other HP peripheral devices
- Have knowledge of the HP-UX file system
- Be familiar with HP-UX privilege groups and real-time features
- Be familiar with HP-UX documentation set

* UNIX is a registered trademark of AT&T in the U.S. and other countries.

## Responsibilities of the System Administrator

The major responsibilities of the HP-UX System Administrator are to:

- Install and test hardware
- Evaluate users' needs
- Install the HP-UX operating system
- Configure the HP-UX operating system
- Allow users access to the system
- Add peripheral devices
- Monitor file system use and growth
- Update the HP-UX operating system
- Back up and restore the file system
- Detect and correct file system errors
- Assist other users
- Provide a backup System Administrator

### Installing and Testing the Hardware

As System Administrator, you should make sure that your computer is installed and operating properly. The computer hardware must function correctly before you install HP-UX. Refer to the appropriate *Installation and Configuration Guide* for your computer for more information.

### Evaluating Users' Needs

You must know your potential users. Knowledge of the number of users, the characteristics of each user, the system resources and peripherals required by each user, and the information that must be shared by various user groups, will help you set up HP-UX for optimum performance.

To aid you in this analysis, a sample user survey form is provided at the end of this chapter. You may want to modify this survey to fit your particular needs. Most users think in terms of "I need to do this job" not "I need FORTRAN, Graphics, a plotter, and 500000 bytes of data storage." The survey should help you identify the needs of the system users and translate those needs into data relevant to system configuration.

## Installing the HP-UX Operating System

The HP-UX operating system is supplied on cartridge tape or 9-track magnetic tape. The operating system is installed onto a Command Set '80 (CS/80) or Subset '80 (SS/80) disk drive. As System Administrator, you are responsible for installing HP-UX. Instructions for accomplishing this task are provided in the manual *Installing and Updating HP-UX.*

## Configuring the HP-UX Operating System

You can change the values and parameters that control how the system uses resources. Configuring the system influences its efficiency and response time. Once familiar with the system, you can use the instructions in Chapters 4, 5, and 7 to alter the system configuration.

## Allowing Users Access to the System

Once HP-UX is installed, you are responsible for allowing access by other users. This involves providing each user with a login name, a password, and a home directory. Instructions for adding users and assigning passwords are contained in Chapter 5, "System Administration Tasks."

## Adding Peripheral Devices

You are responsible for adding or removing system peripherals, such as printers, terminals, or mass storage devices). A list of peripherals supported by the HP-UX system can be found in the *Installation and Configuration Guide* for your system. Directions for installing the peripherals can be found in Chapter 5 of this manual.

## Monitoring File System Use and Growth

As HP-UX is used, files are added to and deleted from the file system. If unused files are not removed, the amount of space required to store files eventually exceeds available space. One of your responsibilities is to monitor the amount of file system space used and identify unused files. Unused files should be archived (if someone will need them later) and then removed from the file system. Also, you should watch for files that continually increase in size. For user files, ask the file's owner if the file is needed, or if its size can be reduced. Generally, you are concerned with system files that grow steadily. You should monitor these files and set up procedures to truncate them periodically.

## Updating the HP-UX System

By purchasing HP support services, you will receive periodic software updates. These updates provide the latest version of the software which typically modifies existing capabilities and adds new capabilities to the system. You may also receive manual updates, if needed.

As System Administrator, you are responsible for installing each software update. You should create and maintain a log showing when each update is installed. Notify all system users of the changes caused by the update. Because each update depends on changes made by the previous update, you should install each update when it arrives. Instructions for accomplishing this task are provided in the manual *Installing and Updating HP-UX*.

## File System Backup and Restore

The HP-UX operating system, programming languages, and applications software represent a large investment of time and money. Files can be unintentionally removed or corrupted. One critical error can cause additional errors and corrupt the file system.

Loss of the system can also occur through carelessness (such as spilled coffee, smoke contamination, or dust) or by other accidents damaging a mass storage device and/or its media.

As System Administrator, you should make a backup copy of the HP-UX operating system, file system, and programming languages. If your system is destroyed, you can recover by using a copy of the latest system backup. If a user accidentally removes a needed file, the file (or a previous version of it) can be recovered by copying it back into the file system from the backup. Note that using files on a system backup is the **only** way to recover a deleted or destroyed file.

Depending on your system usage, consider backing up the system daily. Generally, base the frequency of your system backups on your answer to this question: "How much data can I afford to lose?". Instructions for backing up and restoring the file system are given in Chapter 5.

## Detecting and Correcting File System Errors

Every day the system is used, numerous files are created, modified, and removed; each action requires an update to the file system. It is possible that an update could fail (for example, because of abnormal system shutdown). When an update fails, the file system can become corrupt.

HP-UX provides the **fsck** command, a program that checks the integrity of a file system and can also repair the file system. Each time you boot your HP-UX system, HP-UX automatically checks to see if your system was improperly shutdown. If HP-UX detects an improper shutdown, it automatically checks, and if necessary, repairs common file system errors. If **fsck** finds abnormal conditions that it doesn't know how to repair, it stops the system and prompts you to make corrections. You should, in addition, check the file system whenever you observe unexpected system behavior. Continuing to use a corrupt file system only further corrupts the system. Instructions for verifying and repairing the file system are located in the "Using the **fsck** Command", Appendix C. Also, see **fsck(1M)** in the *HP-UX Reference*.

## Assisting Other Users

As System Administrator, you are the person users come to for help with the system. You should plan on allocating some of your time for consulting and problem solving.

If you have purchased support services, you have access to direct technical support from Hewlett-Packard. As the System Administrator, you are the only person authorized to use this service. If other system users have difficulty with the system, they should direct their questions to you. If you cannot solve the problem, then call your support person at HP.

### Providing a Backup System Administrator

At least one other person should be trained as a backup System Administrator to handle your responsibilities in your absence.

To ease your job as System Administrator and the job of the backup System Administrator, you should automate as many of your tasks as possible. By scheduling programs using system routines (for example, **cron**), you can automatically back up the system or initiate communications between your system and another HP-UX system (using the **uucp** utilities). Refer to the *HP-UX Reference* for details about these routines and utilities.

## Conventions Used in this Manual

The following conventions are used throughout this manual.

**Bold font** indicates files and HP-UX commands, system calls, or subroutines. A font similar to that shown on your terminal display is used for items either typed by the user or displayed by the system as discussed below.

The first time a file is mentioned, the complete pathname is given; subsequent references to that file may contain only the filename unless there is some chance of ambiguity. For example, **/mnt/pubs/yl/myfile** is used for the first time and subsequently, **myfile** may be used.

Bold font is also used when a word is first defined (for example, **inode**) and for general emphasis (for example, **do not touch**).

User inputs are highlighted to differentiate them from program prompts and outputs. For example:

```
54% date
Fri Mar 28 09:31:07 PST 1988
```

Optional parameters and arguments in a command line are indicated in the examples and command syntax with square brackets, "[" and "]".

Environment variables such as **PATH** or **MAIL** are represented in uppercase characters.

Unless otherwise stated, all references such as "see the **login(1)** entry for more details" refer to entries in the *HP-UX Reference*. Some of these entries are presented under an associated heading. For example, the **chgrp(1)** entry is under the **chown(1)** heading.

## User Survey Form

Name ____________________ Location ____________________

Phone ____________________

Location where you will be using the system ____________________

____________________________________________

User Category

| | | |
|---|---|---|
| ___ | Engineers and Managers | (run existing application programs, enter data, create models) |
| ___ | Technical Data Entry Operator | (run existing application programs, enter data or automatically read data from instrumentation) |
| ___ | Secretary - Word Processing Operator | (run existing application programs, enter data/text) |
| ___ | General Programmer | (develop application programs) |
| ___ | System Programmer Support Personnel | (develop programs for improving computer system performance or for use by other programmers) |

Describe your application ____________________

____________________________________________

____________________________________________

____________________________________________

____________________________________________

What programming language(s) will you use? ____________________

____________________________________________

____________________________________________

What applications software (such as graphics) will you use? ____________________

____________________________________________

____________________________________________

____________________________________________

What computer hardware or peripherals will you need to access?

|  |  |
|---|---|
| ___ Inkjet printer | ___ Plotter |
| ___ Impact printer | ___ Removable mass storage device |
| ___ Graphics terminal | ___ Other ____________________ |
| ___ Laser printer | ____________________ |
|  | ____________________ |

Are there other users with whom you want to share programs or data? __________

If so, list them. ______________________________

____________________________________________

Will you be generating or using large amounts of data? ____________________

If so, how much must be "online" (accessible at all times)?

____________________________________________

____________________________________________

What long term data storage does your application require? ____________________

____________________________________________

____________________________________________

How many programs/processes will you be running at one time? ______________

____________________________________________

Which programs are interactive, which will be run in a background mode? Can any programs be run overnight? ______________________________

____________________________________________

# 2 Getting Started

This chapter describes how to get started using an HP-UX system. It tells you where to go for installation and update information, lists tasks you need to perform before using the system, and summarizes the System Administration Manager.

## Installing HP-UX

The HP-UX operating system is supplied on magnetic tapes or cartridge tapes. To install HP-UX, the system presents a series of menus in which you need to supply information about your system configuration. The system provides online help information to help you install the operating system. Refer to the document *Installing and Updating HP–UX* for information on how to install your system.

## Updating HP-UX

If you are updating your system to a new release of the operating system, you must use the **update** utility program. You normally run update from the shell to install a new release of the software by typing:

```
update RETURN
```

When installing HP-UX, the installation program automatically calls the update program so you can install the operating system and optional products. Refer to the document *Installing and Updating HP–UX* for information on how to update your system.

## After Installing HP-UX

**NOTE**

**After your system is installed, you can modify and customize the system as described in the next section. If you purchased software support services from Hewlett-Packard, you can make only limited changes to certain files without voiding your agreement. Consult your support plan or your HP Customer Engineer regarding intended changes.**

After your HP-UX system is installed, you should take the following steps to protect and customize the system:

1. Set the system protection
2. Customize the system
3. Back up the system

Minimum system protection should include setting the password for **root**, the name of the superuser account. Customizing the system is discussed in Chapter 4 of this manual. Backing up the system is discussed in Chapter 5 (which describes the manual method of backup) and in the manual: *System Administration Basics* (which describes the menu-oriented approach).

---

**CAUTION**

**Do not execute any commands except those specified in this section while logged in as superuser until you are familiar with the system. Otherwise, you may inadvertently damage the operating system.**

---

## Setting the Password for Root

After installation, the system automatically reboots and displays a system login prompt. If there is no display, press RETURN to display the system prompt. Then login as the root user.

The system responds with:

```
WARNING: YOU ARE SUPERUSER!!

#
```

To protect your system, assign a password to the root user by typing:

```
passwd root RETURN
```

The system prompts for a password. Enter at least six characters (at least one should be a numeric or special character) and then press RETURN. Note that the password is not displayed on the console. The system then prompts you to re-enter the password to confirm it. If the two entries match, the program accepts the new password. If the two entries do not match, the system gives you another chance to type the password and verify it.

**DO NOT** forget the password you assign. You need it to log on as superuser and perform system administration duties. If you forget the password, reboot the system and bring it up in single-user mode. To reassign a password:

1. Sync the disks using the **sync** command
2. Reset the system
3. Override the autoboot to get the ISL prompt
4. Enter **hpux -is** (to get into single-user mode)
5. Execute **passwd** (to assign another password)
6. Return the system to its normal operation

### System Console

Your system requires a system console. If the console is a terminal, set the console configuration to "RecvPace = Xon/Xoff, XmitPace = Xon/Xoff, and ASCII 8 bits = YES".

Never leave the system console unattended. To ensure the security of your system, you should locate this terminal in a controlled environment. Otherwise, unauthorized users could access this terminal and become the root user during the boot phase of system operation. This defeats system security. Remember that any user logged on as the root user has superuser privileges (a user ID of 0) and can execute any HP-UX command. This situation can be hazardous to the integrity of your system.

## Customizing a System

After installing HP–UX, you may customize your system. Chapter 7 describes system reconfiguration, the process you need to follow to customize your system.

Edit **/etc/csh.login**, **/etc/rc**, **/etc/profile**, **/.profile**, and **/etc/powerfail** to set the correct time zone information. The fields XXX and YYY represent the 3–letter codes for the standard and daylight time zones in your area and H represents the difference between standard local time and Greenwich Mean Time, in hours. (YYY is only required if Daylight Savings Time is observed in your geographic area.)

Insert or modify the lines in **/etc/rc**, **/etc/profile**, **/.profile**, and **/etc/powerfail**:

```
TZ=XXXHYYY
export TZ
```

Insert or modify the line in **/etc/csh.login**:

```
setenv TZ XXXHYYY
```

For example:

In Eastern time zone, use `TZ=EST5EDT`

In Central time zone, use `TZ=CST6CDT`

In Arizona, where Daylight Savings Time is not observed, use `TZ=MST7`

Next, execute the **date** command to set the current time and date. For more information pertaining to setting the system clock, see the **date (section 1)** command in the *HP-UX Reference*.

After setting the time and date for the system, you should proceed to customize the system. Some common tasks are described in Chapter 4 of this manual. However, it is likely that you will also want to perform some of the tasks described in Chapters 5 and 6. A sample list of administrative tasks follows.

- Adding/changing passwords
- Adding new users
- Adding new peripherals
- Setting up an accounting system
- Mounting file systems
- Creating system run levels

After customizing the system, you should perform a full system backup as described in Chapter 5 of this manual or in Chapter 4 of *System Administration Basics*.

## Using the System Administration Manager

The System Administration Manager (SAM) is a tool designed to simplify some system administration tasks. For example, you can use SAM to add users, add a peripheral device, or to set up the LP spooler on the system. Instead of manually changing system files to reflect the changes you want to make to the system, you can use SAM's menus to supply necessary information.

This section briefly introduces SAM. In general, this document describes the manual methods of performing system administration and states whether or not you can use SAM menus to perform the task.

The manual, *System Administration Basics,* describes SAM, how the menus work, and how to accomplish certain administrative tasks using the menus. You can also learn about using SAM by invoking it, reading through the menus, and reading the online help information.

To use SAM, type

```
sam
```

and press the RETURN key. SAM's main menu is displayed on the screen. It lists several major topics that lead to other menus that show tasks you can perform using the various menus. Table 2-1 lists the topics and tasks listed on the SAM menus.

**Table 2-1. SAM Menu Topics and Tasks**

| Menu | Topics/Tasks |
|---|---|
| Main Menu | Working with Individual Accounts<br>File System Management<br>Backup and Recovery<br>Configuration of the System<br>Spool System Configuration<br>UUCP Management<br>How to Use SAM |
| Working with Individual Accounts | Add a New User to the System<br>Delete/Deactivate a User from the System<br>Reactivate a User<br>Show a User's Information<br>Modify a User's Information<br>Add a New Group to the System<br>Delete a Group Account<br>Show Users in a Group<br>Add or Delete Users in a Group |
| File System Management | Create a New File System<br>Add an Entry to File System List<br>Remove an Entry from File System List<br>Change an Entry in File System List<br>Mount a File System<br>Unmount a File System<br>List Swap Devices<br>Enable a Swap Device<br>Show Disk Information |
| Backup and Recovery | Perform a Full Backup Online<br>Perform an Incremental Backup Online<br>Set up a Schedule for Automated Backup<br>Recover an Index of Files on Tape<br>Recover Selected Files from a Tape<br>Display Index Files<br>Display Backup Logfile<br>Display and Modify List of Backup Filesets |
| Configuration of the System | Add Additional Terminals<br>Kernel Configuration<br>Convert File System to Allow Long Filenames |
| Spool System Configuration | Add a Printer<br>Startup/Shutdown the Spool System<br>Remove a Printer<br>Start/Stop a Printer<br>List Status<br>Set Default Printer |

Table 2-1. SAM Menu Topics and Tasks (continued)

| Menu | Topics/Tasks |
|---|---|
| UUCP Management | Show/Modify Device Configurations<br>Add a Device<br>Remove a Device<br>Show/Modify Systems Configurations<br>Add a System<br>Add/Modify Permissions for a System<br>Remove a System |

# System Startup and Shutdown

From the time you switch on power to the computer until you have successfully logged on, the system automatically performs many tasks. These tasks include testing the computer hardware, loading the Initial System Loader (ISL), and loading the HP-UX operating system. You can choose the mode of loading and initializing the operating system. To manage your HP-UX system effectively, you must understand which tasks are performed at which times.

This chapter describes the computer's activities from power-on through successful completion of the **login** program. A system shutdown procedure is also provided in this chapter.

## System Boot

When more than one operating system is present on the system's mass storage devices, both the location of the operating systems and the type of media on which they are stored determine which operating system is loaded. The primary boot path in stable storage determines the default boot path. Stable storage is the memory in Series 800 computers reserved for maintaining critical configuration parameters used during system boot. For example, the primary and alternate boot paths, console path, and autoboot settings are stored in stable storage.

### What the Boot ROM Does

The boot ROM (Read Only Memory) contains general-purpose software that was specifically developed to support the present and future Hewlett-Packard operating systems. This section describes how the ROM boots the HP-UX operating system.

When you turn the computer on, the boot ROM goes through the following sequence:

1. Performs System Processing Unit (SPU) self-test.
2. Reads the console path from stable storage, tests it, and assigns a display terminal for use as a system console.
3. Reads the boot device path from stable storage, searches for the paths on Model 825 and 835 computers (see the section called "Autosearch"), or lets you enter the path from the system console and tests the boot device path.
4. Loads the Initial System Loader (ISL) into memory.

The following shows a typical display of the boot ROM's operation on a Model 840 computer. Note that the display varies depending on the version of the boot ROM and the model of your computer.

```
Processor Dependent Code (PDC) Revision 2

Console path = 8.1.0.0
Primary boot path = 8.0.0.0
Alternate boot path = 8.2.3.0
```

Your HP-UX system is supplied with autoboot enabled. For more information on enabling and disabling autoboot, see **isl(1M)**. Autoboot is enabled by setting a flag in stable storage. You can change this flag using the ISL **autoboot** command. If autoboot is enabled, the following is displayed:

```
Autoboot from primary boot path enabled.
To override, press any key within 10 seconds.
```

If autoboot is disabled, the following is displayed:

```
Boot from primary boot path (Y or N)?>
```

If you type N, the loader prompts for alternate boot path.

```
Boot from alternate boot path (Y or N)?>
```

If you type N, the loader prompts for the boot path.

```
Enter boot path, command, or ?>
```

When you specify a valid boot path, the system displays messages similar to the following:

```
Interact with IPL (Y or N)?> Y

Booting.

Console IO Dependent Code (IODC) Revision 15
Boot    IO Dependent Code (IODC) Revision 15

 HARD Booted

ISL Revision 2803 - January, 1988

ISL>
```

You must provide some input to the ISL prompt. For example, the **hpux** command is entered with the appropriate parameters to specify the boot device, the device address, etc. See **hpuxboot(1M)** for more information on the ISL **hpux** command.

To boot to multiuser mode if partition 0 contains root:

```
ISL> hpux
```

To boot to single-user mode if partition 0 contains root:

```
ISL> hpux -is
```

The two commands shown above boot the kernel (**hp-ux**) on the root volume. To boot an alternate kernel (for example, one with the pathname ALTKERNELNAME) to single-user mode:

```
ISL> hpux -is ALTKERNELNAME
```

Again, the above command only works if partition 0 contains the root directory. If root is not in partition 0, you need to specify its path in the boot command:

hpux -is *disc#* (*path*; *section*) *kernel*

Where:

*disc#* is **disc0** or **disc2** depending on the type of driver required for the disk that contains the root directory.

*path* specifies the disk drive from which to boot and has the following format:

*channel_address.cio_slot.bus_address*

*section* is the number of the section containing the root partition.

*kernel* is the name of the kernel you want to boot.

For example, if root is on section 3 of an HP-IB disk, you would boot to single-user mode as follows:

```
hpux -is disc0(4.1.0;0x3)hp-ux
```

After you boot the system, a series of messages appear. Some of the information of interest is listed below:

```
real mem = amount of physical memory on your system
lockable mem = amount of memory that may be locked via plock(2) or
               shmctl(2)
avail mem = amount of real memory available to user processes
Console information
Processor information
Interface cards
Subsystem information
File system information
Buffer information
Root Device information
Copyright information
```

The media on which your bootable system resides has a boot area and a root partition. The boot area contains the bootstrap program and other files needed for bringing up the system. See **ISL(1M)** for more information on these files. The disk section where the boot area resides is **/dev/dsk/c0d0s6**. The root partition contains a file, called **hp-ux**, which is the operating system. The typical disk section for the root partition on a system using an HP-IB interface card is **/dev/dsk/c0d0s0**.

## The Boot ROM's Search Sequence

The boot ROM initializes the primary boot path, loads ISL, and allows you to select either the manual or autoboot mode. On Model 825 computers, an additional mode called **autosearch** is available when autoboot is enabled. In manual mode, you can select the boot device from all the available peripheral devices. In autoboot mode, the boot ROM automatically boots the operating system from the primary boot path defined in stable storage.

### Autoboot

You should use **autoboot** except for first-time installation and operating system reconfiguration. To enable autoboot, use the ISL **autoboot on** command. Otherwise, no reboot or automatic reboot on panic is possible. Panic is a condition when the system becomes inoperative due to an abnormal condition detected by the kernel. Before using autoboot, make sure the boot device is fully powered up and ready for operation before you turn on your computer. Autoboot is selected if you let the 10 second override period expire.

### Manual Boot

Manual boot can be entered by pressing any key during the 10-second override period in the beginning of the autoboot sequence. When manual mode is activated, the boot ROM prompts for the path to be used. The primary boot path is not altered or disabled.

If you do not want to boot automatically from the primary boot path during the boot process, disable the autoboot flag in stable storage. You can do this using the ISL **autoboot off** command. However, this is not normally done because the autoboot feature makes your system administration tasks more efficient. Disabling autoboot requires your intervention each time the system is rebooted. To re-enable autoboot, use the ISL **autoboot on** command.

### Autosearch

Model 825 computers have a feature called **autosearch.** If the system cannot locate the console using the console path from stable storage, it always automatically searches for a console device. Then, if the system cannot locate the boot device using the primary or alternate boot paths and the autosearch flag is set, the system continues to search for a boot device.

The autosearch flag is much like the autoboot flag. It is in stable storage and can be enabled or disabled on Model 825 computers. Use the ISL **autosearch on** command to enable autosearch and the ISL **autosearch off** command to disable the feature.

When autosearch is invoked (if the two paths specified in stable storage fail), the following messages appear on the console:

```
Autosearch for boot device enabled.
To override, press any key within 10 seconds.
```

If you press a key, the system responds:

```
Do you want to continue an interactive search? (Y or N)?>
```

If you answer "no," autosearch halts at that point and proceeds to manual boot. If you answer "yes," the system searches for a boot path, states it, and asks you if you want to boot from it. If you respond "yes," the system uses that path. Otherwise, it presents other logical paths until you respond positively or until it finds no other paths. It then proceeds to manual boot.

## Booting Problems

If you have problems booting HP-UX, the following information may help:

- If, for any reason, the boot ROM is unable to find and load an operating system or locate a system console, messages are displayed on the system console. (Refer to "Booting Diagnostic Error Messages" in Appendix A and the **pdc**, **isl**, and **hpuxboot** manual pages for interpretations of these messages and information on booting HP-UX.) Verify that the boot path and console path are correct. Check all cable connections and be sure that the boot device is where specified. Make sure that the boot device is functioning properly and that it is online. Ensure that the HP-UX operating system is on the disk.

- Remember that the disk drive containing the HP-UX operating system must be powered up and be in a "ready" state **before** you turn on the computer. If the disk drive has not completed its power-up sequence (which can take several minutes on some disk drives), the boot ROM cannot access the disk and load the system.

- The boot ROM follows a specific search for the system console. First, it tries the primary console path as defined in stable storage. If that fails, it goes through Mid-bus/CIO Module 8, MUX card in CIO slot 1 on Model 840 computers. On Model 825 computers, the system searches the Mid-bus/CIO Modules and all CIO slots until it finds a terminal or graphics monitor that can function as a system console. The MUX card for the console must exist for the boot to succeed. On Model 850 computers, the boot ROM refers only to stable storage for the primary console path.

- During the booting process, the system automatically runs a program called **dasetup**. This program scans the I/O tree in the kernel and searches for cards (such as MUX cards) that are configured, then downloads the firmware to the card. This can cause boot warning messages to appear on the console. Refer to Appendix A, "Boot Warning Messages" for information on the messages and what they mean.

- The **/etc/bcheckrc** script executes the **fsck -P** command at bootup which checks the root file system (typically, c0d0s0, c2000d0s0, or c1000d0s0 as specified in **/etc/checklist**), and checks other file systems if they were incorrectly shutdown. The file system consistency check program (**fsck**) is vital to the maintenance of your file system. Continuing to use a corrupted file system invites disaster. For this reason, **fsck** halts the system if it locates serious file system errors. In that case, you must boot the system to single-user mode and run **fsck** interactively on the corrupted file system to correct the errors. Refer to Appendix C, "Using the **fsck** Command," for details on checking file systems.

## After Booting

After your system is booted, it performs a series of checks and starts system processes.

The following sections discuss the root file system and system processes that take place after booting.

### The Root File System

Once the HP-UX operating system is located and loaded successfully, HP-UX searches for the root file system. The **root file system** is the portion of the file system that forms the base of the file system hierarchy. The root file system contains the files required by the HP-UX operating system. It can exist on any mass storage medium supported by Series 800 computers with at least 24 MB of storage.

### The init Process

After finding the root file system, HP-UX sets up its first process, **/etc/init**. The **init** process becomes process 1 and has no parent.

After booting, the **init** process reads the configuration file **/etc/inittab**. Each line in the **inittab** file describes some activity for the system to take under certain circumstances. The entries are of the form:

```
id:rstate:action:process
```

where:

| | |
|---|---|
| `id` | is a unique one- or two-character identification code. |
| `rstate` | specifies the run levels to which this entry applies. |
| `action` | tells **init** what to do with the entry. |
| `process` | is an HP-UX command to execute. |

A run level can be viewed as a software configuration of the system where each configuration allows only a selected group of processes to exist. The run levels are described in the **inittab(4)** manual page.

For example, if you want the system console to have a **getty** (be able to log in) for every state you define:

```
co:2345:respawn:/etc/getty console console
```

The **respawn** action tells **init** to recreate a **getty** process at the console (in run levels 2 through 5) every time the process it spawned terminates. Leaving the **rstate** field null (as shown below) will cause execution in run levels 0 through 6:

```
co::respawn:/etc/getty console console
```

This example is highly recommended for the system console.

---

**NOTE**

**The remaining sections in this chapter describe the operation of the system as shipped to you; however, by altering certain configuration or system files, any of the following procedures can change. If, for example, you write your own /etc/rc script, the following paragraphs may no longer apply. If you change any system files, save a copy of the original file, document the changes, and keep a copy of that documentation with this manual.**

---

After **init** begins, but before it makes the first transition into states 0 to 6, all entries marked **boot** or **bootwait** are executed. As shipped, **init** runs **/etc/bcheckrc** which runs **fsck** on file systems listed in **/etc/checklist** that are not clean. There is a flag in the superblock of each file system that is set if the system was shut down improperly. It indicates that it is not safe to use the file system until **fsck** fixes the errors. You cannot mount such a file system until the errors are fixed by **fsck**.

Once the booting processes have been run, **init** comes up in the "initdefault" run level as defined in **/etc/inittab**. Refer to the **init(1M)** and **inittab(4)** entries in the *HP-UX Reference* for more information. After installation, the system will come up in run level 2, which is the default multiuser run level. Refer to Chapter 5 for information on how to change the system run level.

## Single-User and Multiuser Modes

The two most common run levels on your HP-UX system are run level 1 and run level 2 (single-user and multiuser modes). Use single-user mode during system installation and maintenance.

Use multiuser mode during normal system operation. The system will boot up to multiuser mode at run level 2. The other multiuser modes (run levels 3 through 6) are undefined. Each can be defined for specific applications. For example, in run level 2 (the default multiuser mode), there is a **getty** process for each terminal configured in the system. Other multiuser run levels can be defined for users to include a subset of the terminals or to run only a selected group of processes.

HP-UX has a special single-user mode: run level s. If the **initdefault** entry in **/etc/inittab** is s, then you immediately get a Bourne shell at the system console, logged in as root. The **bootwait** and **boot** entries are not executed. In this run level, no actions are taken by the script **/etc/rc**. The following are not executed: **syncer, bcheckrc**, and **fsck**. Run level s is not recommended by Hewlett-Packard for routine use because in run level s, certain processes that monitor and check your system do not run. Use run level s for file system repair, for fixing system problems, and for configuring your system.

## The /etc/bcheckrc Script

The **/etc/bcheckrc** (Boot CHECK Run Command) script is run by **init** before your system makes the first transition to any of states 0-6. Script **bcheckrc** checks to see if the system was properly shut down.

Program **bcheckrc** prompts with:

```
    Do you want to check the file systems, (y or n)?
```

If you do not respond within 10 seconds, the system assumes "y" is the answer and "**fsck -P**" is executed. This command looks in the primary superblock of each file system. In the superblock is a **clean byte**. When a file system is created, this byte is set to FS_CLEAN. When the file system is mounted, the clean byte is set to FS_OK. If the file system is unmounted during system shut down (see "Shutting Down Your System"), the clean byte is reset to FS_CLEAN.

If **fsck -P** does not find FS_CLEAN in the clean byte, **fsck** will run using the **preen** mode, correcting most errors found (see the discussion on the preen mode in Appendix C, "Using FSCK").

Anytime **fsck** makes changes to the root (/) file system, the system must be rebooted to force the memory-resident disk information to incorporate the changes **fsck** made to the disk. This is because the root file system is always mounted. Do not check file systems (except the root volume) when they are mounted.

You can, on a system with multiple file systems:

1. Bring the system up
2. Run **fsck**
3. Reboot the system
4. Run **fsck** again

The first time you run **fsck**, it fixes the root volume; the second time fixes all the additional, as yet unmounted, disks. The second **fsck** does not cause the system to reboot because the disks it fixes are not mounted.

If the **fsck** command run by **bcheckrc** fails for any reason, **bcheckrc** halts the system. You will then need to boot the system to run-level s and then run **fsck** manually to ensure the integrity of your file system.

Some file system problems must be fixed manually because of the risk of data loss. When you have completed running the manual **fsck** on the root volume, you might be instructed to reboot the system. If you are instructed to reboot, you must reboot the system using **reboot -n** to ensure the integrity of your system. If **fsck** does not tell you to reboot, make the transition to the desired state by running "init run-level". When **bcheckrc** prompts for file system check, respond with "n".

## The /etc/rc File

Each time **init** changes run level, either at boot time or when invoked manually, **/etc/inittab** is read. After reading **/etc/inittab** and signalling processes as required, a line in **/etc/inittab** invokes **/etc/rc**. This script uses the following two arguments returned by doing a **who -r**: the current run level and the number of times this run level has been entered previously. At power-up, the values of these arguments are 2 and 0 respectively, if you have not modified your system. The following paragraph describes the actions that take place whenever **/etc/rc** is invoked. This is followed by a section describing the contents of **rc**.

Upon starting, the **/etc/rc** script sets the environment variable TZ (for time zone). The **/etc/rc** file then exports the TZ variable (using the **export** command). Exporting TZ causes **rc** (and any child process of **rc**) to override the default time zone (**MST7MDT**); for more information, see the **ctime(3C)** entry in the *HP-UX Reference*. Note, however, that setting TZ in **/etc/rc** does not set the time zone for the system or for users. You do this using startup shell scripts for users and in the **uxgen** input file for the system.

### /etc/rc in Run Level 2

You can customize **/etc/rc** to perform functions that should occur every time the system is booted, or whenever there is a change in run level which **init** does not handle. As shipped, the **/etc/rc** shell script will print the date.

**The following are done only during the first time the system enters run level 2, or when the system is rebooted.** These functions are not performed if run level 2 is entered without a system reboot. For example, after a shutdown for system maintenance, issuing an **init 2** command will not perform the tasks listed below.

1. Mounts the file systems configured in **/etc/checklist** (see **checklist(4)**).
2. Runs **savecore** to save a core image of a previously crashed system (refer to Chapter 8 for a complete description of **savecore**).
3. Enables swapping on additional disk sections as configured in **/etc/checklist**, see **checklist(4)**. As shipped, the HP-UX system has a default swap section in **/dev/dsk/c0d0s1**.
4. Runs **syncer**, which executes the **sync** command periodically to write all in-core superblocks to disk. This will limit data corruption in case of a system crash.
5. Executes the **/etc/cron** program, which executes commands at specific dates and times, according to the instructions submitted by the **crontab** command (see **crontab(1)**).
6. Starts diagnostic logging, see **delog(1)**.
7. Performs miscellaneous "housekeeping" chores, such as preserving editor files (if they exist) and starting up the **lp** daemon. (A daemon is a background process.)
8. Saves various logging files (by renaming them).

The following files are not run by the default **rc** shell script. You may want to add them.

- Networking startup files. Refer to the *Network Services* manual set for information.

- **rc.local** file that includes any system specific additions to the **rc** file. You need to create it with **root** as the owner and a group ID of **bin**. Its permission should be rwxr--r--. The file is used to localize changes made for a particular computer system. By using this file, all computer systems in a network can have the same **/etc/rc** with changes for the host system in **/etc/rc.local**. To execute **/etc/rc.local** at startup time, add the following to the end of **/etc/rc** (after the section which saves the old log files):

  ```
  if [ -x /etc/rc.local ]
  then /etc/rc.local
  fi
  ```

### Returning Control to the /etc/init Process

When **/etc/rc** finishes its run level 2 execution, control returns to **/etc/init** which executes the commands from the command field of all run level 2 entries in **/etc/inittab**. Typically, **/etc/inittab**'s run level 2 command field entries consist of one **/etc/getty** command for each terminal on which users are to log in. The **getty** command runs the login program and eventually runs the shell program when a user successfully logs in.

## Execution of /etc/getty

The first command (**/etc/getty**) executed for each login terminal specifies the location of the terminal and its default communication protocol. Additionally, it causes the first **login:** prompt to be displayed. Eventually, the getty process slot is occupied by your shell (see the following section, "Login").

When your shell is terminated (that is, when you log out), the **/etc/init** process is signalled and "wakes up". **init** then checks **/etc/inittab** to see if the process that signalled it is marked as "respawn" in the **inittab** entry. If the process is marked "respawn", **init** again invokes the command in the command field of the appropriate **inittab** entry as described above (the getty runs and a new **login:** prompt appears). If the process is not marked to respawn, it is not restarted.

**NOTE**

**Do not add** /etc/getty **entries to** /etc/inittab **for terminals that are not connected to the computer. If you do,** getty **sends an error message to the console, waits 20 seconds, and then exits. If such** /etc/getty **entries are flagged as "respawn," these error messages are logged every 20 seconds.**

## Login

This section describes what the operating system does when you log into the system.

1. The login process begins when you supply a user name in response to the **login:** prompt generated by **/etc/getty**. Once a user name is entered, **/etc/getty** executes **login** with the supplied user name; **/bin/login** checks the name against the list of valid user names in **/etc/passwd**.

2. If the user name is valid, **login** checks to see if there is a password associated with the user name (the encrypted form of the password is stored in **/etc/passwd**). If a password is associated with the user name, the system prompts you to enter the password. The supplied password is encrypted and compared to the encrypted password stored in **/etc/passwd**. If a valid user name is supplied and that name has no password associated with it, you are logged in without further prompting.

   For security reasons, if the user name supplied is invalid (it is not found in **/etc/passwd**), the system still prompts you to enter a password before denying access to the system. This makes it more difficult for an intruder to find and use a valid user name. Once access is denied, **login** displays its **login:** prompt and waits for another user name to be entered.

   Successful logins are recorded in the **/etc/wtmp** file; unsuccessful logins are recorded in the **/etc/btmp** file (these files are explained later in this chapter).

3. The **login** program next sets your numeric user and group IDs. The values are taken from the values supplied in the user ID and group ID fields of the **/etc/passwd** file.

4. Next, **login** changes the current working directory to that supplied in the home directory field in **/etc/passwd**.

5. **Login** then executes (using the **exec** system call) the command in the command field of your **/etc/passwd** entry. You can place any command in the command field of **/etc/passwd**, though the most common ones are **/bin/sh** and **/bin/csh**. If no command is present, **/bin/sh** is executed by default.

6. Next (assuming a shell was started in step 5), the shell executes the shell script **/etc/profile** for the Bourne shell (**sh**), and the Korn shell (**ksh**), or **/etc/csh.login** for the C shell (**csh**). As shipped to you, these scripts define and export the environment variables PATH, TZ, and TERM. You can modify **/etc/profile** and **/etc/csh.login** to change each user's default settings for the environment variables.

7. The files **/etc/profile** and **/etc/csh.login** also define the path for the MAIL environment variable and they perform the following tasks:

   - Display the message-of-the-day (contained in **/etc/motd**).
   - Use **mail -e** to detect if any mail is present; if there is any mail, the message "You have mail." is displayed.

8. Finally, the shell executes the script **.profile** (for the Bourne and Korn shells) or **.cshrc** (for the C shell) if it exists in the user's home (login) directory. Typically, a

**.profile** or **.cshrc** file is created by the System Administrator for each user and is customized by the user for his environment on the HP-UX system. For example, you might create or modify your **.profile** file to alter the primary and secondary prompts, change one or more environment variables, set up your terminal for a particular application, or invoke application programs automatically.

   In addition to this, the C shell also executes the file **.login** (if it exists) when you first log in. File **.login** is executed following the execution of **.cshrc**. When you log off, csh executes a file called **.logout**.

9. Now that you have successfully logged in, you can invoke an application program or execute any system command.

## Changes You Might Want to Make

You should now have an understanding of how the operating system is loaded and initialized and the procedures used to log onto the system. Now, consider how you can change the boot and login procedures to customize your HP-UX system.

This chapter has discussed system boot and login, and contains suggestions on the type, scope, method, and advisability of the changes that you can make to the system. Note that changes to the files **/etc/inittab, /etc/rc, and /etc/passwd** should not be major ones; the system will not boot if these files are incorrectly modified. Before modifying these files you should ensure that you have a backup of the HP-UX operating system. The procedures for making the suggested modifications are detailed in Chapters 4 and 5. The following is an example of a typical system boot, describing changes you might want to make in this process.

1. The computer is powered up and the operating system is loaded using the primary boot path.

   You can change boot parameters during installation, if necessary, such as changing the address of the system disk or assigning a new primary or alternate boot path using ALTPATH or PRIMPATH at the ISL prompt. Type ? at the ISL prompt or refer to **ISL(1M)** for additional information.

2. The **init** process is brought up and run.

   The **init** process calls **bcheckrc**, and if your system was improperly shut down, **fsck** is run.

3. The system automatically goes into run level 2 which executes certain commands from **/etc/rc**.

   **/etc/rc** mounts all the file systems listed in **/etc/checklist**.

   **/etc/rc** turns on swapping for alternative swap devices listed in **/etc/checklist**.

   As shipped, the **/etc/rc** file executes the **cron** command, which runs programs at a scheduled time.

You may want to add entries with the **crontab** command to perform certain procedures automatically and periodically. For example, you might add entries for **cron** to:

- Back up the system.
- Call other HP-UX systems for mail and other **uucp** transactions.
- Execute System Accounting commands (described in Chapter 6).

4. Commands are executed from the command field of run level 2 entries in **/etc/inittab.**

   Ensure that the value of "2" is in the **rstate** field of each terminal **/etc/getty** line. The terminal **/etc/getty** should have state numbers for each state you want the user of that terminal to be able to log in. You should have the console defined in all states. Add **getty** entries for terminals you wish to add.

5. A user attempts to log in to HP-UX.

   Before any user can log in, an entry must be made for that user in the **/etc/passwd** file. Details are in Chapter 5.

   To keep track of all bad login attempts, create a log file called **/etc/btmp**. The system uses this file to log unsuccessful login attempts, and you may use this file to help determine if unauthorized users are attempting to login. A summary of these attempts may be viewed using **lastb**. If this file is created, you should also provide some automated means to truncate the file periodically so that disk space is not wasted.

   To keep track of all successful login attempts, create a file called **/etc/wtmp** which the system uses to log successful login attempts. A variety of login and logout information may be accessed with the **last** command.

   You can specify which "tty" device (login communication port) the root user can login on by creating a file called **/etc/securetty**. This file should contain the device names of the tty files where the root user can log in. The entries only contain the name of the special (device) file for that tty but not the pathname (typically, the **/dev** directory). The file may contain multiple entries but only one entry per line. If you do not create this file, the user root may log in on any tty device. Note that this security feature does not restrict a normal user from using **su** to become the superuser on any device. You can restrict use of the **su** command by executing **chmod 500 /bin/su**. Here is a typical **/etc/securetty** file with two entries:

   ```
   console
   tty05
   ```

6. A user successfully logs in.

   After some initialization, login changes the user's current working directory to the user's login directory. To set the user's login directory, edit the file **/etc/passwd** (see **passwd(4)** in the *HP-UX Reference*). **mkdir** creates the login directory (also called the home directory). The **chmod, chown,** and **chgrp** commands set up the recommended ownership and permissions.

Login executes a command from the user's entry in **/etc/passwd**. Typically, the command invokes a shell for the user. However, you might wish to execute an application program for the user. This is advisable when the user has no knowledge of HP-UX and only wants to use the system to run a specific application. If no entry exists in the **/etc/passwd** command field, **/bin/sh** is executed by default.

For example, suppose that an inexperienced user wanted to access HP-UX only to run the program **testx** (a program written by your company that tests widgets) contained in his/her login directory. You might add an entry to **/etc/passwd** of the form:

```
john::135:12:run testx only:/users/john:/users/john/testx
```

**john** is the user's login name and the values **135** and **12** are his user ID and group ID respectively. The field **run testx only** is a comment in **/etc/passwd**. The field **/users/john** is the user's login directory. The last field is the command field; it specifies that when the user logs in, the program **/users/john/testx** is automatically run (instead of the shell). In this example, after the user **john** finishes executing the program **testx**, he will automatically be logged off the system.

The same thing happens when a user executes a shell: when the shell is terminated (with a **ctrl-D, logout**, or **exit** command), the user is logged off.

The command field of **/etc/passwd** enables users to access information without logging onto the system. For example, the system is shipped with a **passwd** entry containing the user name **who** with the command **/bin/who** in the entry's command field. Supplying **who** in response to the login prompt displays a list of all system users currently logged in (including the user **who**). The user is logged in only during the execution of the **who** program and is logged off when the program terminates. This login allows anyone to determine a valid login name; you might want to delete this entry to provide more security.

7. The shell scripts **/etc/profile** or **/etc/csh.login** are automatically executed.

   **/etc/profile** (for the Bourne and Korn shells) or **/etc/csh.login** (for the C shell) allows you to force execution of one or more commands for each user when the user logs in. This is ideal for forcing the execution of commands that each user should execute at login. For example, **/etc/profile** and **/etc/csh.login** (as shipped to you) assume that **/etc/motd** (message-of-the-day) contains one or more messages and display the contents of that file on each user's terminal at login. To change the message sent to each user, simply edit **/etc/motd**. Note from the previous discussion of **/etc/profile** that many of these commands are only executed if the user is executing **/bin/sh** or **/bin/rsh**. **/etc/csh.login** is executed if the user is executing **/bin/csh**.

   Edit **/etc/profile** and **/etc/csh.login** to add, remove, modify shell commands. New commands can be added to these scripts; less useful commands can be removed or modified. Changes to these scripts do not take effect until the script is executed (either at login or explicitly).

   The shell script **.profile** (for the Bourne and Korn shells) is automatically executed if it exists in the user's login directory. For the C shell, scripts **.cshrc** and **.login** are automatically executed if they exist in the user's login directory.

The scripts **.profile** or **.cshrc** execute commands that users want executed at login, as well as each time a new shell is spawned. For example, a user might want prompts from the shell that are different from the default prompts. To change these, the user can add PS1="new prompt" and PS2="new secondary prompt" to these scripts, in the Bourne and Korn shells, and set prompt = "new prompt" in the C shell. Remember, in the Bourne and Korn shells, variables must be exported as environment variables to have an effect upon subsequent processes. In the C shell you must use **setenv** to set environment variables.

Following the execution of **.cshrc** by the C shell, the file **.login** is executed at log in. Edit **$HOME/.profile** or **$HOME/.login** (where $HOME is the user's login directory pathname set at login) when you want to alter its function. You may add new commands to the script; old commands may be removed or modified. Typically, a user uses **.profile** to:

- Set and export (with the **export** command) environment variables such as shell prompts (PS1 and PS2) and the default search path (PATH).
- Execute commands at login (for example, the **who** command, to see who else is on the system and the **ls** command to list the names of files in the login directory).
- Set terminal options with the **stty** command.

Do not change the **stty** settings until you understand how they work. Many commands (such as the **vi** text editor) require settings similar to those shipped with your system to operate properly. See the **stty(1)** command in the *HP-UX Reference* and "The vi Editor" section in the *HP-UX Concepts and Tutorials: Text Editors and Processors.*

Changes made to the **.profile** script are not active until the script is executed. Depending on your current shell, specifying one of the following commands executes **.profile** or **.cshrc** in the current environment.

| | |
|---|---|
| `. .profile` | In the Bourne and Korn shells |
| `source .cshrc` | In the C shell |

## Changing the System Run Level

As shipped, your system will "come up" in run level 2 at bootup. This means your system executed the **inittab** commands associated with run level 2.

If you change to a different run level, the processes corresponding to entries in **inittab** that do not explicitly include the new run level automatically terminates when entering that run level. For example, if you have not added a respawn **getty** (see discussion below) entry for run level 3 for the console, entering run level 3 from run level 2 causes the console to die.

Each terminal or RS-232 port used as an incoming terminal device needs to have a process called a **getty** running on the line for users to log on to. **inittab** sets up the gettys.

For each incoming port, **inittab** must have one line that looks like:

```
03:236:respawn:/etc/getty tty0p3 9600
```

**03** is a two-character ID field (by convention the name of the associated tty); **236** are run levels 2, 3, and 6 (there should be at least one run-level entry for each incoming terminal device); **respawn** tells **init** to restart the **getty** after it exits; and the colons (:) are mandatory separators that should be used as shown. You may include comment lines in **inittab** as follows:

```
xx::off:#This is a comment
```

See the **inittab(4)** entry in the *HP-UX Reference* for further details.

To create new run levels, make entries in **/etc/inittab** defining how the system operates in its new run level. For example, identify the run level entry by a run level number (in the range 1 to 6), identify the command you want executed for each run level entry, and list any flags that are to be considered. Once **/etc/inittab** contains all of the entries you want for the new run level, be certain to warn all users to log off before you change run levels. The **wall** command is a useful tool for warning all users to log off. Changing run levels while users are logged on terminates their login processes during execution unless the **getty** for their terminal is defined as "respawn" for the new run level.

Once all the users that need to be off the system are off, change run levels by making an entry in **/etc/inittab** of the form:

```
init new_rstate
```

where `new_rstate` is the number of the new run level.

Depending on how the new run level is defined, even the foregoing might not be necessary. For instance, you can move freely between run levels as long as entering the new run level does not kill user or system processes that may have begun in the previous run level. Watch for side effects, however. Consider the case where a user logs off, you then change run levels (for example, from run level 2 to run level 4). The user will not be able to log in next time because an **/etc/getty** entry does not now exist in run level 4.

When the system enters the new run level, its actions are similar to the actions described in the system boot section of this chapter except that the commands executed are those identified by the new run level number. Some files, such as **/etc/rc**, have no entries for run levels other than run levels 1 and 2. The **/etc/rc** script may be customized (by adding **/etc/rc.local**, for example) to include more run levels.

## Summary of Environment Files

As you have seen, the system boot and login processes provide many opportunities to customize the environment in which your system operates. Customization is achieved primarily by altering the contents of one or more files known as **environment** files. The following list summarizes the files that you might want to alter, and identifies the types of changes you might want to make. Use these suggestions with those presented in the previous section to determine which files to modify. All of the files listed here have commented versions that were shipped with your system. Files discussed here that do not include a specific pathname are contained in the **/etc** or **/etc/newconfig** directories.

The **/etc/newconfig** directory is shipped with updates of the HP-UX operating system and generally contains new versions of the files shipped in the **/etc** directory. If you have never updated your system, no **/etc/newconfig** directory will exist.

---

**NOTE**

**The system may not boot if** /etc/inittab, /etc/rc, **and** /etc/passwd **are radically modified. The parts of these files that are shipped should not be changed, though additions can be made as necessary for terminals, commands, and users. You should check the** /etc/passwd **and** /etc/group **files with the** pwck **and** grpck **commands whenever you modify them.**

---

### /etc/inittab

This file contains entries for the different run levels (supplied or created) to which the system administrator may want to change the system.

### /etc/rc

This shell script defines miscellaneous actions for each system run level change. It will take different actions for different run levels. This script typically contains commands to perform many tasks. These tasks are described previously in the "**/etc/rc** in Run Level 2" section.

The following are not in the default **rc** file. You may want to add them.

- Networking startup files. Refer to the *Network Services* manual set for more information.
- **rc.local** file that includes any additions to the **rc** file.
- Removal of **/tmp** files.

## /etc/passwd

This text file identifies the user name, real user and group IDs, home (login) directory, and execution command for every valid user on the system. The execution command is the command executed when the user correctly logs in. **You must add an entry to this file for each additional user who wants to log in to your system.**

## /etc/group

This text file identifies the users that form a group. It contains a list of users and associates those users with a group name and a group ID.

## /etc/motd

This text file contains messages you want to send to each user (such as a message specifying a new system update). Type the message in **/etc/motd** using any text editor. As each user logs in, the system displays the message (assuming that the scripts **/etc/profile** and **/etc/csh.login** are not modified to remove the command that prints **/etc/motd**).

## /etc/profile or /etc/csh.login

These shell scripts are automatically executed for each user, upon successful login. The Bourne shell (**sh**) and Korn Shell (**ksh**) executes **/etc/profile**; the C shell (**csh**) executes **/etc/csh.login**. This is an ideal location to place commands that you require every user to execute. For example, you may want every user to read the message of the day file (**/etc/motd**) because it contains information that each user should see before beginning any task. Do this by entering the statement

```
        cat /etc/motd
```

in the **/etc/profile** or **/etc/csh.login** shell scripts. These scripts are also an ideal location to define and export default environment variables (such as PATH and TZ) in case the user does not set them in his local shell startup script.

## /etc/wtmp

**/etc/wtmp** is a text file where the system keeps a history of successful logins, logouts, and date changes. You need to create this file for system use. You can access the contents of this file with the **last** command. It must be truncated periodically.

## /etc/btmp

**/etc/btmp** is a text file that the system uses to log bad login attempts. You must explicitly create this file to use this feature. The contents of **btmp** are accessed with the **lastb** command. This file must be truncated periodically.

## /etc/securetty

**/etc/securetty** is a text file that specifies on which tty files the **root** user can login. You must explicitly create this file and specify the tty special filenames there to use this feature.

## $HOME/.profile, $HOME/.cshrc, or $HOME/.login

These shell scripts, when located in the user's home (login) directory, are automatically executed each time the user successfully logs in. It is a good location to place commands that customize users' environments and perform functions users want to execute each time they log in.

For example, the scripts might include a definition of the default shell prompt (the **PS1** and **PS2** environment variables in the Bourne shell, **prompt** in the C shell), the default search path (the PATH environment variable), and some editor information (the EXINIT environment variable). It also generally includes the execution of one or more commands such as the **export** command (in **sh**) to export environment variable definitions, the **who** command to identify who is logged in on the system and the **mail** command to automatically display mail that has been received.

## $HOME/.exrc

This file maps terminal characteristics and sets up new key definitions so that features such as arrow keys can be used with the **ex(1)** family of HP-UX editors (**vi, ex**, etc.). The editor searches for **$HOME/.exrc** and, if it exists, uses the definitions to create extra editor features. It will then implement the instructions in **$HOME/.exrc**, if it exists.

Note that the **.exrc** file is functional *only if the EXINIT environment variable is not defined*. EXINIT is generally defined and exported from either **/etc/profile**, **$HOME/.profile** or **.cshrc**. The **.exrc** file serves a function similar to EXINIT.

## /usr/lib/terminfo

This subsystem identifies terminal capabilities for programs such as the **vi** text editor. It defines terminal attributes for all Series 800 models and HP-supported terminals. It might contain terminal attributes for terminals **not** supported by Series 800 computers; these are provided for your convenience only. Hewlett-Packard does not support their use. See **terminfo(4), tic(1M)**, and **untic(1M)** for more information.

## /etc/checklist

The **/etc/checklist** text file lists all file systems that need to be mounted. It includes entries for how disks are laid out with regard to parts of the file systems. Default entries for all disks (including the installation device) are automatically entered into this file during HP-UX installation. The **/etc/checklist** text file and all the fields it contains are explained in more detail on the **checklist(4)** manual page in the *HP-UX Reference*. For information on NFS-related mount points, refer to the *LAN,NS,ARPA Services/9000 Series 800 Node Manager's Guide*.

For example, default entries for systems installed on 7933, 7935, or 7937 disks connected to HP-IB interface cards are:

```
/dev/dsk/c0d0s0       /         hfs  rw  0 1   # root disk
/dev/dsk/c0d0s3       /tmp      hfs  rw  0 2   # /tmp directory
/dev/dsk/c0d0s4       /usr      hfs  rw  0 3   # /usr directory
/dev/dsk/c0d0s5       /extra    hfs  rw  0 4   # extra space
/dev/dsk/c0d0s10      /mnt      hfs  rw  0 5   # /mnt directory
```

And the default entries for systems installed on 7937 disks connected to HP-FL interface cards are:

```
/dev/dsk/c2000d0s0    /         hfs  rw  0 1   # root disk
/dev/dsk/c2000d0s3    /tmp      hfs  rw  0 2   # /tmp directory
/dev/dsk/c2000d0s4    /usr      hfs  rw  0 3   # /usr directory
/dev/dsk/c2000d0s5    /extra    hfs  rw  0 4   # extra space
/dev/dsk/c2000d0s10   /mnt      hfs  rw  0 5   # /mnt directory
```

The **/etc/checklist** text file that is installed with your system contains entries for the configuration of your computer.

If you add a disk to the system, you need to add entries for each additional disk section containing a file system that you want automatically mounted, and for each disk section used as a swap device. (The file **/etc/disktab** provides information on the section sizes of all supported disks.) Refer to Chapter 7, "Creating a New File System," for how to change **/etc/checklist**.

Be sure that there are no blank lines in this file. Blank lines can cause **fsck** to skip some file systems or to fail during autoboot.

When you execute the **fsck** command and do not specify device files, **fsck** checks the file systems marked "hfs" in **/etc/checklist**. This file is also used by the system accounting **diskusg** command, and the **mount**, **umount**, **swapon**, and **fsclean** commands. The order of entries in **/etc/checklist** is important because the commands that use this file iterate through it sequentially.

## /etc/catman

Executing the **catman** command expands the **nroff** versions of manual pages into their "processed" form as well as creating the **/usr/lib/whatis** database, used by the **man** command. When the pages are accessed using **man**, the processed manual page is retrieved, significantly improving response time to the command. The price for the improved speed is disk space. This roughly doubles the amount of disk space required for manual pages because the original files remain intact. By default, running **catman** causes manual pages in all the **/usr/man/manX**, **/usr/contrib/man/manX**, and **/usr/local/man/manX** directories (where **X** is 0 through 8 or 1M) to be processed and stored in the corresponding **/usr/man/catX** directories.

Running **catman** with the **-w** option will create the **/usr/lib/whatis** database without processing the **man** pages. This is necessary for the **-k** option of the **man** command to work.

The **cat** directories are not shipped with your system. You must explicitly create them for **catman** to run successfully. Use the following script to create the appropriate **cat** directories:

```
for I in /usr/man /usr/contrib/man /usr/local/man
do
     for num in 0 1 1m 2 3 4 5 6 7 8
     do
     mkdir $I/cat$num
     done
done
```

Once the **cat** directories exist, you have three alternatives:

1. Create all the processed manual pages by executing **/etc/catman** with no parameters. This process can take as long as five or six hours to complete so you might want to run it at night.

2. Process sections of manual pages by executing **/etc/catman sections** where **sections** is the sections to process.

3. Do not execute **/etc/catman** at all. Because the **/usr/man/cat** directories now exist, the first time **man** is executed for any given manual entry, the entry is processed, added to the appropriate **cat** directory, and used in subsequent accesses.

   The third alternative is recommended if you can spare some disk space but do not want to use more than necessary. With this "build-as-you-go" alternative, the system only fills the **cat** directories as manual pages get accessed by **man**.

When the processed **man** pages exist, you can remove the **nroff** source files.

## Shutting Down the System

System shut down is performed to bring the system to the single-user mode for system maintenance or to halt the system for powering down.

Do not allow any user to shut down the system. It should only be done by the System Administrator or designated superuser. The **shutdown** command warns all users to log off the system, giving them a reasonable amount of time to finish their tasks before shutting down the system.

---

### WARNING

**Powering down the computer improperly can corrupt the file systems. If using network services, do not run SHUTDOWN when logged in remotely via** rlogin. **The shut down process logs you out prematurely and returns control to the system console.**

---

The **shutdown** command is normally used to shut down the system. It terminates, in an orderly and cautious manner, all processes currently running on the system. This allows you to power down the system hardware without adversely affecting the file system.

The **shutdown** command kills all unnecessary processes, forces the contents of the file system's I/O buffers to be written to the disk (with the **sync** command), and takes the system into the single-user state. It will also optionally halt or reboot the system.

To shut down the system from a normal operating mode, perform the following steps.

1. Login as the superuser, **root**. You must know the root password.
2. Move to the root directory of the file system by entering the command:

   ```
   cd /
   ```

3. Execute the **shutdown** command.

   The **shutdown** command allows you to specify a **grace_period**, which is the number of seconds you want **shutdown** to wait before terminating all processes. You can also use the **-r** option to automatically reboot the system after reaching run level "s", or the **-h** option to halt the system.

   The **shutdown** command looks like (see the examples below):

   ```
   /etc/shutdown [-r|-h] [grace_period]
   ```

   If there are other users on the system, **shutdown** prompts to see whether you wish to send the standard broadcast message or enter your own message. If you elect to send your own broadcast message, type the message on the terminal. When you are finished typing the message, press RETURN and then the CONTROL-D keys to signify the end of the message. If a value of 0 is entered, **shutdown** asks if you want to continue.

If **grace_period** is omitted, **shutdown** allows a grace period of 60 seconds before shutting down the system. When **shutdown** completes its task, it displays a message telling you to halt the system when you are ready.

4. If you have halted the system (**shutdown -h**) you can now power down the system. To bring the system back up, go to step 5.

   If you have not halted the system, to power down the system, you have two alternatives:

   a. If you ran **shutdown** without the **-h** option, you are now in run level s (single-user mode), and can halt the system by typing:

   ```
   reboot -h
   ```

   b. If you have not yet run the **shutdown** program, you are still in the normal operating run level and can halt the system by typing in:

   ```
   shutdown -h
   ```

5. If you do not want to power down the system and now need to resume normal operating run level, type:

   ```
   reboot
   ```

   For example: To activate a newly configured kernel, shut down the system with no grace period and automatically reboot:

   ```
   shutdown -r 0
   ```

   To install an interface card, halt the system by typing:

   ```
   shutdown -h 0
   ```

   Wait for the "halted" message, then turn the power off to the computer.

   To back up the system, change to run level s by typing:

   ```
   shutdown 0
   ```

   After running backup, bring the system up with all the daemons running by typing:

   ```
   reboot
   ```

---

**NOTE**

**Do not execute shutdown -r from run level s. You must reboot using the reboot command.**

---

To halt the system from run level s with no daemons or programs running, type:

```
reboot -h
```

4

# Customizing the HP-UX System

After your system has been installed and started, you may want to customize the system. The tasks described in this chapter include:

- Setting the System Clock
- Maintaining System Security
- Creating Groups
- Setting Up the LP Spooler
- Administering UUCP

## Setting the System Clock

Make sure that the system clock always has the correct time and date. Many commands use the system clock, so the system time and date must be correct if these commands are to accomplish their tasks successfully.

You may sometimes need to set or reset the system clock. Only the System Administrator can change the system clock.

### Time and Date

To set the current time zone and date:

1. Log in as the superuser or **root**.
2. Make sure that the time zone environment variable (TZ) is set properly. Check the variable declarations in **/etc/rc**, **/etc/profile**, **/etc/csh.login**, **/etc/powerfail**, and **/.profile**.

   Typically, the TZ value is set with a variable declaration (as shown below) in the files **/etc/profile** or **/etc/csh.login**. TZ can also be set from an application program using the **tzset** library routine.

   As shipped to you, the system is set up to run in the Mountain Standard Daylight Time Zone (MST7MDT). To change it to your time zone, modify these lines in **/etc/profile** and **/etc/csh.login**. The lines you must modify have the following format:

```
TZ=XXXHYYY
export TZ
```
(in /etc/profile for **sh** and **ksh**)

and

```
setenv TZ XXXHYYY
```
(in **/etc/csh.login** for **csh**)

In these examples, **XXX** and **YYY** are three-letter representations of the standard and daylight time zones for your area. The letter **H** represents the difference (in hours) between current local time and Greenwich Mean Time (GMT). The **export TZ** line will never change in **/etc/profile**. For example, in San Francisco, California, enter the following:

```
TZ=PST8PDT
export TZ
```
(for **sh** and **ksh**)

and

```
setenv TZ PST8PDT
```
(for **csh**)

where PST stands for Pacific Standard Time and PDT stands for Pacific Daylight Time. There is an 8-hour time difference between GMT and San Francisco local time. Here are some other examples:

For St. Clair Shores, Michigan: **TZ=EST5EDT** and **setenv TZ EST5EDT**

For Norman, Oklahoma: **TZ=CST6CDT** and **setenv TZ CST6CDT**

For Corvallis, Oregon: **TZ=PST8PDT** and **setenv TZ PST8PDT**

For Hawaii: **TZ=HST10** and **setenv TZ HST10** (Hawaii has no Daylight Savings Time)

3. If you are changing the time by more than 20 minutes, terminate the **cron** process if it is running. Do this so that any processes that are set to execute will not do so at an incorrect time. To terminate **cron**, you must first determine the **cron** process **id**. Enter:

   ```
   ps -ef | grep cron
   ```

   Next, terminate **cron** by typing:

   ```
   kill -9 <pid>
   ```

   where **pid** is the process **id** associated with **cron**.

4. Now that the time zone is set, you can set the correct time and date (using the **date** command) by typing an entry of the form:

   ```
   date MMddhhmm[yy]
   ```

   **MM** is a two-digit integer representing the month. For example, 03 represents March.

   **dd** is a two-digit integer representing the day of the month. For example, 02 represents the second day of the month.

**hh** is a two-digit integer specifying the current hour in terms of a twenty-four hour clock. For example, 03 specifies 3:00 am and 14 specifies 2:00 pm.

**mm** is a two-digit integer specifying the number of minutes past the stated hour. For example, 04 specifies four minutes past the hour.

[yy] is an **optional** two-digit integer specifying the last two digits of the current year; this parameter may be omitted if the year is already correct. For example, 85 specifies 1985 as the current year. Execute **date** and it echoes the time and date on your screen.

5. Restart **cron** if you terminated it in step 3.

   To restart **cron**, enter:

   ```
   /etc/cron
   ```

## Special Considerations - Backups, Make, and Cron

The **make** program (see the **make** entry in the *HP-UX Reference*) uses a file's time and date information and the current value of the system clock. Although setting the clock forward does not affect **make**, setting the clock backward may cause **make** to behave unpredictably. Avoid setting times earlier than the current system clock's value. If the change you are making could effect any system or user application programs, you should change the time and date while running in single-user mode.

The process of doing incremental backups depends on the correctness of the date. This is because incremental backups are often made in relation to a dated file. This is another reason to keep the date correct on your system. If your backup method uses a time stamped file to determine the files to be included in an incremental backup, changing the date may affect the accuracy of the backup.

**CAUTION**

**Altering the system clock may cause unexpected results for routines scheduled by** cron.

Altering the system clock may also cause some unexpected results for routines scheduled by **cron**; see the **cron(1M)** entry in the *HP-UX Reference*. When setting time back, **cron** does not run until the clock "catches up" to the point from which it is set back. For example, if you set the clock back from 8:00 to 7:30, **cron** will not begin executing until the clock again reads 8:00. If you are setting the clock ahead, **cron** attempts to "catch up" by immediately executing all routines scheduled to run between the old time and the new time. For example, if you set the clock ahead from 9:00 to 10:00, **cron** immediately executes all routines scheduled to run between 9:00 and 10:00.

## Maintaining System Security

Establish system security by setting your own rules, procedures, and strategies to discourage illicit activities and inadvertent actions. Security can be considered in three categories; physical, password, and file security.

**Physical security** should include items listed below:

- Maintain back-up tapes or discs
- Keep back-up media locked and protected
- Identify terminal lines to system users
- Protect system console (or any terminal you are logged onto as superuser)
- Protect the ability to reboot the system (via the "reset" button or **/etc/reboot**)

**Password security** is another way to protect your system. Your system has a password file, **/etc/passwd**, that can be read by the public but can only be changed by the superuser. The actual password for each user is encrypted. You should require all users to use a login password.

Another level of system security is **file security**. The HP-UX system has a file mode which allows users to set the access permission of their files. Refer to the *HP-UX User's Guide* for more information about file access permissions.

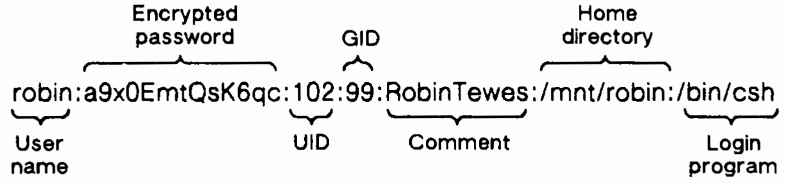
Your system files must have the correct file modes. Check the following items:

- The root directory should have the file mode 775.
- The file **/etc/passwd** should have the file mode 664.
- New files should be write protected to all but the owner. Use **umask** of 022 and include this entry in **/etc/profile** and **/etc/csh.login** to create a default file creation mode.
- System directories such as **/usr**, **/lib**, **/usr/lib**, **/bin**, **/usr/bin**, and **/etc** should have the file mode 775 to prevent users from adding or deleting files.
- Temporary directories (such as **/tmp** and **/usr/tmp**) should have the file mode 777.

Refer to the *HP-UX Reference* for more information about **umask(1)**.

### The Password File

The password file (**/etc/passwd**) contains descriptions of the accounts on your HP-UX system. The file is readable by all users, but only the superuser can modify it. Each entry in the file has seven fields, separated by colons. A typical password file entry looks like this:



```
robin:a9x0EmtQsK6qc:102:99:RobinTewes:/mnt/robin:/bin/csh
```

The fields contain the following information:

- The user name (or login name) consisting of up to eight characters.
- The encrypted password.
- The user ID number (UID), usually the next available sequential user number. It must be an integer less than 60000.
- The group ID number (GID), taken from the **/etc/group** file. This number is shared by accounts belonging to the same group, and must be an integer less than 60000.
- The comment field, usually used for identifying information, such as the user's full name.
- The home directory, which is the initial login directory for the user.
- The login program pathname, which is executed when the user logs in.

## Changing a Password

This section explains how either a user or the System Administrator may change passwords manually.

**NOTE**

You can also use SAM to change passwords using menus. Refer to *System Administration Basics* for information about using SAM.

Any user on the system may change his/her own (but not other user's) password by entering:

```
passwd
```

The **passwd** command prompts for the existing (old) password before allowing the user to continue. Once the correct old password is entered, the command prompts for a new password. Enter at least six characters including at least one numeric and/or special character then press RETURN. The password can have fewer characters if the superuser generates it. The password can include control characters such as those generated by pressing the backspace key (but this is not recommended). When you type it, the password is not echoed on the screen so others cannot read it. The command then prompts you to reenter the password to confirm it. Do so and, if the two entries match, the program accepts the new password. If the two entries do not match, you are prompted to enter it twice again.

Users will occasionally assign passwords to their accounts and then forget the password. Once a user has forgotten his/her password, he/she cannot log in to the system and will probably come to you for help. Because only the encrypted form of the password exists in **/etc/passwd**, you cannot recover the user's password. Therefore, you must assign the user a new password.

To change a user's password, log in as superuser and type:

```
passwd username
```

Where *username* is the user's login name. You will be prompted for the user's new password. Only the superuser may use this method for changing a password. You must tell the user the new password. Alternately, you may delete the encrypted password from that user's entry in **/etc/passwd**. This allows the user to log in without any password and then use the **passwd** command to set the password immediately.

You may also force users to set their password immediately upon login. Enter ",.." in the password field of the **/etc/passwd** entry. For example:

```
hp:,..:200:20::/mnt/mktg/hp:/bin/csh
```

This is the password aging feature of the system. It sets up a null password to allow the user to log in. However, immediately after login, a new password is requested from the user. A valid password must be entered before login can be completed.

There are other password aging commands that cause a password to expire after a specified time period. Then the user must assign a new password when they log in. Initially, you must enter the command using a series of aging codes. Details are given in the **passwd(4)** entry in the *HP-UX Reference*.

To protect the system, **/etc/passwd** should be owned by **root** (the superuser) and no one should have write permission to the file, not even the superuser. If you, as the superuser, want to modify **/etc/passwd**, temporarily change the permission using **chmod**, modify **/etc/passwd**, then change the permission of **/etc/passwd** back. You could also, as the superuser, use the enforced write command of your editor (":w!" in **vi**) to override the file protection. Actually, only those with superuser privilege should be allowed to write to **any** of the files in the **/etc** directory, or to the directory itself.

If the superuser password is lost or forgotten, no one can log in as superuser. If you ever forget your superuser password, the only way to gain access as superuser is to reboot your system, bring it up in single-user mode, and assign a new password.

## Creating Groups

Using groups is a way to organize users on the system. This section describes how to create groups manually.

---

**NOTE**

You can also use SAM to create groups. Refer to *System Administration Basics* for information about using SAM.

---

A group is defined by a single line in the **/etc/group** file. Each entry in the file consists of four fields, separated by colons. To create a group, edit the **/etc/group** file and make an entry for the group. A typical entry in **/etc/group** looks like this:

```
groupname:password:groupid:member1, member2, ..., memberN
```

The **groupname** field contains the name of the group.

The **password** field is used by **newgrp**. Placing an asterisk (*) in the **password** field prevents non-group members from switching to this group.

The **groupid** field is the unique integer ID shared by all group members.

The **member1, member2, ..., memberN** is a list of user names of each group member; user names are separated by commas.

## Changing Group Membership

To change a group's membership, simply modify the membership field for the group entry in the **/etc/group** file. When you are satisfied with the group definition, write the modified file to disc and terminate the editing session.

If your system has a file called **/etc/logingroup**, users can belong to multiple groups simultaneously. Since **/etc/logingroup** is usually linked to **/etc/group**, the fields are the same as for **/etc/group**, but only the group ID and the list of users is significant. At log in, each user belongs to all groups which list that user name as a member. Each user may be associated with a maximum of 20 groups in **/etc/logingroup**; there is no limit for the number of groups the user can belong to in **/etc/group**.

To change the effective group ID to that of another group if there is only the **/etc/group** file, use the **newgrp** command. At log in, each user is placed in the group specified by his group ID entry in **/etc/passwd**.

## Privileged Groups

Privileged groups control user access to various HP-UX features such as real-time priorities, file locking, and changing file ownership. Initially, your HP-UX system allows all users CHOWN (change ownership) privileges. Features and capabilities of privileged groups are described in the *HP-UX Real-Time Programming Manual* under the **setprivgrp** command. This section explains briefly the command formats and parameters.

The **setprivgrp** command lets you change group privileges. The command formats are summarized here:

> **setprivgrp -g**|**-n**| *groupname* [*privileges*]
> **setprivgrp -f** *filename*

where:

- **-g** = all groups
- **-n** = no groups
- *groupname* is the group whose privileges are being changed.
- *privileges* are as listed below.

| **Privileges** | **Description** |
|---|---|
| RTPRIO | Real-time priorities |
| MLOCK | Locking processes into memory |
| CHOWN | Change file ownership |
| LOCKRDONLY | Permit locking of read-only files using the **lockf(2)** system cell. |

These privileges are stored in memory and you must reset them when you reboot the system. Typically, privileges are set in **/etc/rc**. The **-f** option causes privileged groups to be set from the named file, which typically is **/etc/privgroup**. The system sets global CHOWN privileges as a default.

Only the superuser can execute this command. Therefore, the System Administrator or the designated superuser must maintain the file **/etc/privgroup**. The format of this file is the same as the arguments to **setprivgrp**.

**CAUTION**

**Taking away CHOWN privilege can produce undesired results. For example, mail runs as "set group id mail", so if the mail group does not have CHOWN privileges, local mail will not work.**

## Setting Up the LP Spooler

On a multiuser system, access to printers requires careful management and control. Many users need to print documents at the same time so there needs to be a way to make sure each file is printed separately and to determine which file will be printed first.

**NOTE**

You can also use SAM to set up the spooler using menus. Refer to *System Administration Basics* for information about SAM.

HP-UX provides a series of commands, collectively referred to as the Line Printer Spooler (or LP Spooler), which are used to set up and control line printer spooling. Line printer spooling allows each user to print files as desired yet have the system organize how it is to be done (see Figure 4-1). Spooling is temporarily storing files to be printed in a spool directory until they can be printed. You can customize the LP

Spooler to spool files to different printers on both local and remote systems. In addition, you can group printers into classes so that files are printed on the next available printer to increase system efficiency.

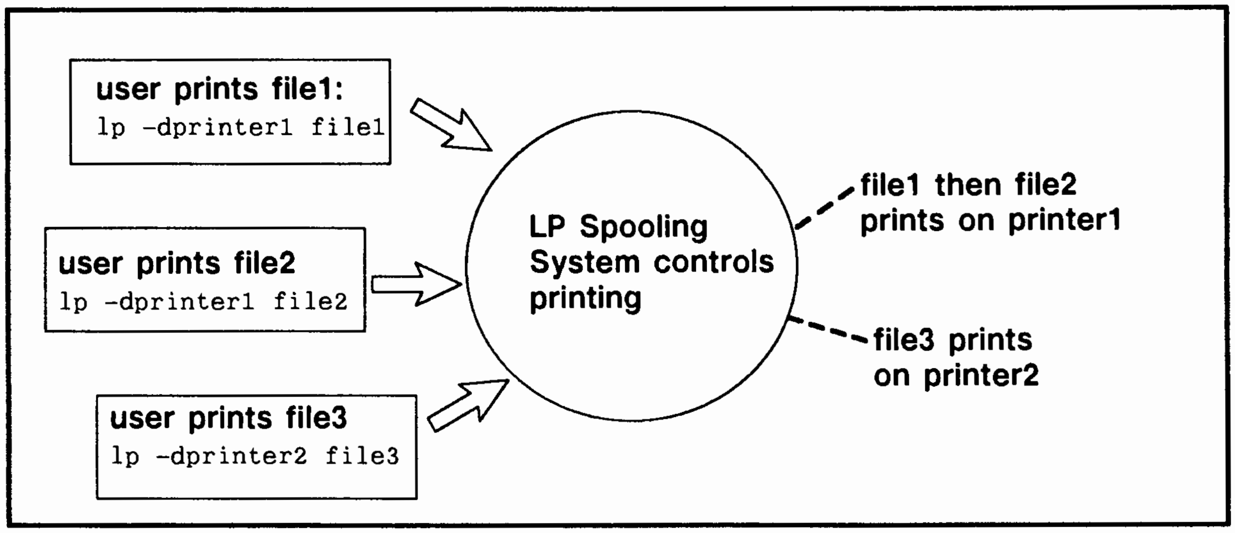


**Figure 4–1. Line Printer Spooling**

The LP Spooler system has many features that provide for smooth running of the printers and minimum administrator intervention. Users submit their print requests to the system, then continue working at their terminals. Later they can use a command to check whether or not the file has printed. Users can also stop files from printing after having executed the **lp** command.

On some systems, a person is designated to serve as the LP Spooler administrator. In this section, it is assumed that the LP administrator is the same person as the System Administrator.

This section presents an overview of how the spooler operates then briefly describes all LP spooling commands: those available to all users, those available only to the superuser (the user with ID=0) or LP Spooler administrator (the user **lp**), and those invoked for internal operation. It then describes how to install and configure the LP Spooler, and concludes with some tips on monitoring and maintaining the spooler.

## LP Spooler Terminology and Overview

A complete LP Spooler configuration for a system consists of:

- devices
- destinations (printer names and classes)
- interface programs
- the LP Spooler directories including the LP Spooler commands in **/usr/bin** and **/usr/lib**

When a user executes the **lp** command or prints a file, the system generates a spooling request. A **spooling request** is a combination of one or more files to be printed and all associated information such as its destination, number of copies, and other **lp** options. The request is stored temporarily in a spool directory until the file is printed. The **lp** command assigns an ID to the request then passes the request to the **LP scheduler.** (You need to start up the scheduler using the **lpsched** command after configuring the spooling system.) The LP scheduler routes the request to the proper **interface program** to do the actual printing on a device, handling the interface between **lpsched** and printing devices.

**Models** of interface programs are supplied with the LP Spooler and, in some cases, have options to use specific printer features, such as expanded or compressed print. The models can be used as is, modified for your specific needs, or used as examples for creating new interface programs. Models are shell scripts that run the commands necessary to allow the scheduler **lpsched** to communicate with printing devices. Several model scripts are shipped with your system and are located in the **/usr/spool/lp/model** directory. As shipped, this directory includes model scripts for a generic "dumb" printer, the HP 2225D, HP 256X, HP 2686A, HP 2932A, and HP 2934A. Refer to the **/etc/mklp** script for a description of the provided models.

## Default Printing

The **lp** command prints the output on a default printer unless a different destination is specified as an option when **lp** is executed. The System Administrator can set or change the default destination using the **lpadmin** command when configuring the LP Spooler. A **destination** is a printer or a class of printers. A **class of printers** must contain at least one printer although a printer may belong to more than one class. If the destination is a queue for a specific printer, the output can only go to that printer. If the destination is a queue for a class of printers, the output is sent to the first available printer in that class. You should inform all system users where their files will print by default; a login message is a good way to do this.

Although you can specify a default printer for all users, you can also assign a default printer queue or printer class queue for each user (that is, a default destination). You do this using an environment variable. Where you do this depends on which shell the user is set up to work from. If the user runs the Bourne shell (**sh**) or the Korn shell (**ksh**) add the following line to .profile in the user's home directory:

```
LPDEST=destination;export LPDEST
```

If the user runs the C shell (**csh**), add the following line to .login or .cshrc in the user's home directory:

```
setenv LPDEST destination
```

By setting up default printers for each user, you can allow users to print on printers that are located nearest to them. In some environments, this is not an important consideration because printers may be located in a central location. As an administrator, you need to determine the best way to set up spooling for your system.

The LP Spooler distinguishes between logical destinations and physical destinations. A logical destination is a name for one or more physical destinations. You connect logical destinations with physical destinations using the **lpadmin** command when configuring the spooler. Each device has an associated special file that stands for the physical destination. One physical destination can be associated with one or more logical destinations on the system. **Lp** requests are directed to a logical destination as long as that destination was set up to accept requests (see **accept(1M)**). Otherwise, the **lp** request is rejected. When a corresponding physical destination (a printer) is enabled (see **enable(1)**) and available for use, the request is transferred there.

## General-Purpose LP Spooler Commands

The following is a brief overview of the LP Spooler commands available to all users; for further details consult the *HP-UX Reference*.

| | |
|---|---|
| **cancel(1)** | Cancels requests to an LP Spooler made with the **lp** command. The user may address a specific printer or a specific request ID number (explained in the **lp(1)** entry in the *HP-UX Reference*). Users may cancel their own print requests. The superuser may cancel any request. |
| **disable(1)** | Disables one or more physical printers such that they will not print **lp** requests. Refer to the **enable(1)** entry in the *HP-UX Reference*. |
| **enable(1)** | Activates one or more physical printers to print **lp** requests. |
| **lp(1)** | Sends requests to an LP Spooler line printer. Requests are files and associated printing information (flags, etc.) sent to the spooler. The **lp** command returns (to standard output) a unique ID associated with a request. |
| **lpstat(1)** | Prints current LP Spooler status information such as requests, IDs, and scheduler information. |

## System Administrator LP Spooler Commands

The following commands are only for use by the System Administrator (the user **root**) or the LP Spooler administrator (the user **lp**). Further details are contained in this section and in the *HP-UX Reference*.

| | |
|---|---|
| **accept(1M)** | Allows lp requests to occur on one or more logical destinations where a "destination" is a printer or class of printers. |
| **lpadmin(1M)** | Configures the LP Spooler system by describing printers, classes, and devices. The LP scheduler must **not** be running when most **lpadmin** command options are used. |
| **lpmove(1M)** | Moves requests queued by the LP scheduler from one destination to another. The LP scheduler must **not** be running when **lpmove** is used. See the **lpsched(1M)** entry in the *HP-UX Reference*. |
| **lpsched(1M)** | Schedules requests taken by **lp** for spooling to line printers. |
| **lpshut(1M)** | Shuts down the LP scheduler. See the **lpsched(1M)** entry in the *HP-UX Reference*. |
| **reject(1M)** | Rejects **lp** requests on one or more logical destinations where a "destination" is a printer or class of printers. See the **accept(1M)** entry in the *HP-UX Reference*. |

## Remote Spooling Commands

Remote spooling commands allow the spooler to send requests back and forth among HP 9000 Series 300, 800, and BSD 4.2 or 4.3 systems. Refer to the section "Configuring the LP Spooler for Remote Operation" for details on how to set up remote spooling.

| | |
|---|---|
| **rcancel(1M)** | When a remote printing request is canceled, **cancel(1)** calls this command to actually cancel the printing request. |
| **rlpstat(1M)** | If status is requested using **lpstat(1)** and there are remote printers on the system, **lpstat** calls this command to report on remote spooling status information, such as each request, its ID, the username, the size of the request, and scheduling information. |
| **rlp(1M)** | If printing to a remote printer, the printer model file associated with that printer calls this command to connect to the remote system and sends the spooling request. |
| **rlpdaemon(1M)** | This daemon program or spool area handler runs on a remote system that receives requests to be printed. It handles remote system communications, such as receiving files for printing, returning status to the system that originated the remote spooling request, and canceling remote spooling requests. To set up remote spooling on your system, you can start this program from **inetd(1M)** or directly at boot time in the **/etc/rc** file. |

## LP Spooling Directories

The spooling system involves several directories that contain the commands, the spooler setup, and the temporary storage area where printing requests wait to be printed. The directories are described here and shown in Figure 4-2.

| | |
|---|---|
| **/usr/spool/lp** | LP Spooler system parent directory. All information about the setup and printing queues is located here. |
| **/usr/spool/lp/class** | Printer classes directory. This contains the files that define how printers are grouped. |
| **/usr/spool/lp/model** | System-supplied interface programs. This directory contains the model shell scripts designed for particular printer models. |
| **/usr/spool/lp/interface** | Interface programs in use on your system. This has shell scripts from **/usr/spool/lp/model** that may be modified for particular printers. If interfacing a printer for which there is no model file, you may need to create an interface program for it. |
| **/usr/spool/lp/request** | Destination queues. This is where all **lp** requests are queued. It usually contains a subdirectory for each printer configured on the system. |

| | |
|---|---|
| **/usr/bin** | Contains user-executable commands, such as the LP Spooler commands that general users can execute. |
| **/usr/lib** | Contains administrator-executable commands, such as the LP Spooler commands that only **root** or **lp** can execute. |
| **/usr/spool/lp/cmodel** | Contains system-supplied interface programs, in the form of model shell scripts for processing remote cancel requests. |
| **/usr/spool/lp/smodel** | Contains system-supplied interface programs, in the form of model shell scripts for processing remote status requests. |
| **/usr/spool/lp/cinterface** | Contains the shell scripts from **/usr/spool/lp/cmodel.** |
| **/usr/spool/lp/sinterface** | Contains the shell scripts from **/usr/spool/lp/smodel** for the installed printers. |

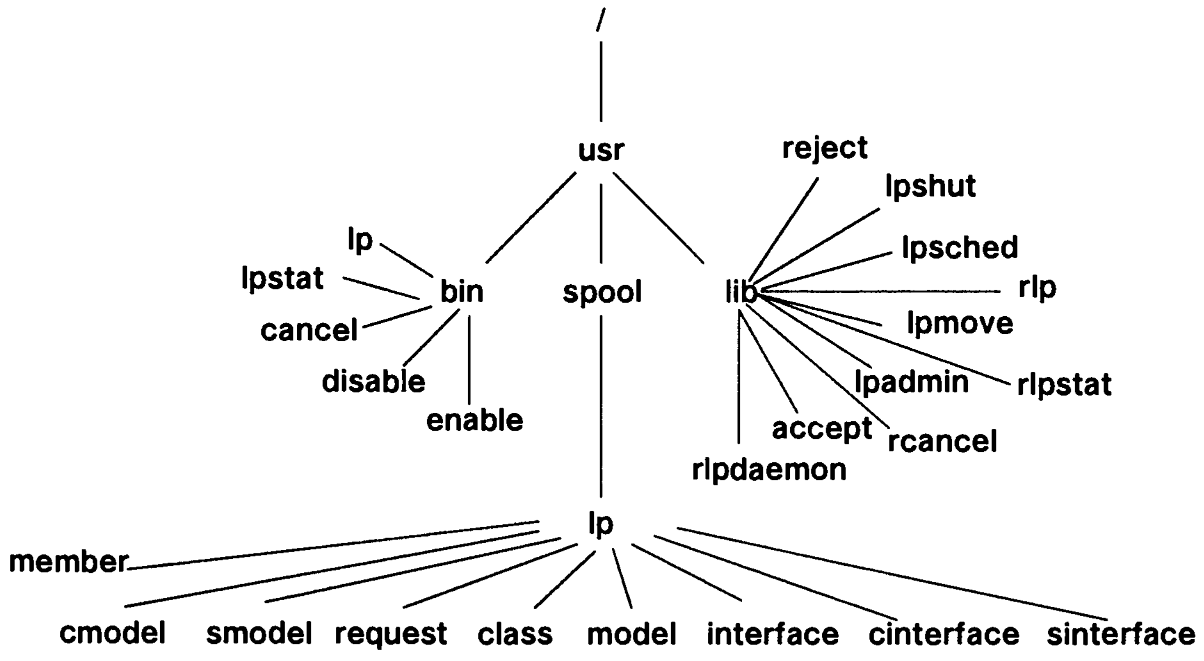


**Figure 4-2. LP Spooler Directories and Files**

## Installing the LP Spooler

**NOTE**

You can also use SAM to set up the spooler. Refer to *System Administration Basics* for information about SAM.

To install the LP Spooler, you have to log in as **root**.

You **must** check and possibly modify three configuration files for the LP Spooler to work properly:

- **/etc/passwd**
- **/etc/group**
- **/etc/rc**

These files vary depending on the version of your HP-UX system. If you have just **installed** a new system, the files should be in the **/etc** directory and contain the appropriate information. If you just **updated** an existing system, you need to update all three files using the information contained in the files in the **/etc/newconfig** directory. Note that the **/etc/newconfig** directory is only shipped with updates. Check the configuration files and update them, if necessary.

### Check /etc/passwd:

The **/etc/passwd** file should contain:

```
lp::9:2::/usr/spool/lp:/bin/sh
```

providing ownership of the LP Spooler to the user **lp**.

You may want to add password protection to the **lp** user. To do this from the **root** account, execute the **passwd lp** command. Or you may place an asterisk (*) in the passwd field of **/etc/passwd** for **lp**. See the **passwd(1)** entry in the *HP-UX Reference* for details.

### Check /etc/group:

The **/etc/group** file should contain:

```
bin::2:root,bin,lp
```

providing group ownership of the LP Spooler to the user **bin**. Other users may also be associated with the group **bin** in your system configuration.

### Check /etc/rc:

The **/etc/rc** file should contain:

```
# Start lp printer scheduler
     /usr/lib/lpshut>/dev/null 2>&1
     if [ -s /usr/spool/lp/pstatus ]
     then
         rm -f /usr/spool/lp/SCHEDLOCK
         /usr/lib/lpsched
         echo line printer spooler started
     fi
```

to start the LP scheduler every time the system is booted.

To enable remote spooling and allow the computer to receive remote requests, you need to start the remote spooling daemon in **/etc/rc** or through the use of **inetd(1M)** as follows:

If using **/etc/rc** alone:

```
/usr/lib/rlpdaemon -l
```

To specify the use of **inetd(1M)**:

```
/usr/lib/rlpdaemon -i -l
```

Refer to the *LAN, NS, ARPA Services/9000 Series 800 Node Manager's Guide* (50980-90010) for more information on how to use **inetd(1M)** to start **rlpdaemon**.

To set up a particular printer to be used with the LP Spooler, you can edit and use the **/etc/mklp** script, or type in the commands directly from the keyboard. The first step is to make a special (device) file for the printer using the **insf** or **mksf** command. Refer to the "Adding/Removing Peripheral Devices" section in Chapter 5 for details.

## Using the mklp Script to Configure the Spooler

The **/etc/mklp** file is a shell script that presents sample commands that you can use as a start to write a script to configure your LP Spooler system. The file consists of three sections:

- Section 1 sets the ownership, group, and access mode for each file used by the LP Spooler
- Section 2 lists each supported printer and lists the commands required to configure it.
- Section 3 explains how to set up a default printer.

If you plan to use the script, you will need to edit **/etc/mklp** so it fits your system needs, and then execute the script.

The **mklp** script needs to be edited in several places.

1. These lines appear in the introduction, just before Section 1. Comment them out:

   ```
   echo "mklp:  template version -- customize script before using it"
   exit 1
   ```

2. In Section 2, uncomment the lines:

   ```
   cd /dev
   name= ?? (name of the device)
   dev= ?? (the device file)
   ```

3. Edit the lines you just uncommented to contain the name of the device and to specify the device file. For example:

   ```
   cd /dev
   name=lp (now contains lp, the name of the device)
   dev= /dev/lp3 (now contains lp3, the name of the device file)
   ```

4. Also in Section 2, select the device model by uncommenting a line that contains the model name. For example:

   ```
   #    model=hp2225a          # for HP 2225
        model=hp2563a          # for HP 2563
   #    model=hp2686a          # for HP 2686 (serial) laser jet
   #    mode=hp2932a           # for HP 2932A
   ```

   In this example the line containing the HP 2563 printer was uncommented, so that was the printer model selected.

5. Finally, in Section 2, uncomment the lines:

   ```
        $lpshut
        $lpadmin -p$name -v$dev -m$model -h
        $accept $name
        $enable $name
   #    $lpadmin -d$name       #uncomment if this is the default
        $lpsched
   ```

   Note that you will uncomment the fifth line only if the device is a default device.

6. Uncomment the last line of the **mklp** script:

   ```
   exit 0
   ```

The **mklp** script contains commented information on how to modify it for your system. Before you modify the file, read the next section. It explains the commands you have to use to configure the system. Be sure to refer also to the section "Configuring the LP Spooler for Remote Operation" if you plan to connect remote printers to your system.

## Configuring the LP Spooler

To configure the LP Spooler system, you can follow the steps in this section or read these steps, then modify and run the **/etc/mklp** script.

1. Log in as superuser (**root**) and shut down the LP scheduler:

   ```
   /usr/lib/lpshut
   ```

2. Refer to **/usr/spool/lp/model** or the description of supported printers in **/etc/mklp** and select the appropriate model shell scripts for the printers you want to add to

the system. Make sure the model scripts have a permission mode of 644 and that they are owned by **lp** and group **bin**.

If you want to modify one of the models for your system needs, copy then edit the model file and then associate the new model with a printer using **lpadmin** with the **-i** (interface) option.

3. Execute the **lpadmin** command with the **-p** option to name a printer. Repeat the command for each printer you want to configure in to the system (including remote printers - see the next section for additional options).

   For example, if you have an HP 2934A that is accessible through the device file **/dev/lp**, you can use the following command line:

   ```
   /usr/lib/lpadmin -plp -v/dev/lp -mhp2934a -h
                      ↑      ↑         ↑
            printer name device file  model
   ```

   where:

   **-plp** names the printer **lp** (the logical destination).

   **-v/dev/lp** specifies the full path name of the printer's special file, the physical destination.

   **-mhp2934a** specifies the printer model hp2934a from the **/usr/spool/lp/model** directory.

   **-h** means the printer is "hard-wired".

4. For each of the printers defined with **lpadmin**, execute **accept** and **enable** to allow requests to reach the printer:

   ```
   /usr/lib/accept lp
   /usr/bin/enable lp
   ```

5. Select a printer as the system default. For example, to make the printer **lp** the default, use the command:

   ```
   /usr/lib/lpadmin -dlp
   ```

6. Restart the LP scheduler:

   ```
   /usr/lib/lpsched
   ```

   then check that the LP spooler's "scheduler" is running properly:

   `lpstat -t`  [prints out status information]

7. If the scheduler is not running, try removing the file SCHEDLOCK:

   ```
   rm -f /usr/spool/lp/SCHEDLOCK
   ```

   Then repeat step 6. SCHEDLOCK acts as a "semaphore" to prevent more than one scheduler from running at a time. The **lpshut** command automatically removes the SCHEDLOCK file when it terminates the LP scheduler. The **shutdown** command executes **lpshut**.

You may also need to remove the file FIFO before the scheduler will work properly. FIFO is a named pipe created by **lpsched** for LP scheduler communications. Do this by entering:

```
rm -f /usr/spool/lp/FIFO
```

## Configuring the LP Spooler for Remote Operation

The remote spooler schedules print jobs on remote systems. You add remote spooling capabilities to the system when configuring the rest of the system. The spool system supports system names up to eight characters in length. Note that you must use remote spooling options with each **lpadmin** command that you execute to configure a remote printer. That is, when you get to step 3, execute **lpadmin** as follows:

```
/usr/lib/lpadmin -pname -v/dev/device -mmodel# remote_sp_options
```

where:

| | |
|---|---|
| *name* | is the name users will use to print to that printer. |
| *device* | specifies the full pathname of the special (device) file of the printer. |
| *model#* | is the printer model (must be listed in the **/usr/spool/lp/model** directory). |

*remote_sp_options* are any of the options shown in Table 4-1.

**Table 4-1. Remote Spooling Options to lpadmin**

| Option | Description |
|---|---|
| **-ob3** | Use 3-digit request numbers associated with the printer directory. This is for consistency with BSD systems. The default is to use 4-digit request numbers. |
| **-oci***rcancel* or | Causes *cancel* to use *rcancel* to cancel requests to remote printers. Specify the full pathname to ensure that the correct command is used. |
| **-ocm***rcancel* | The model *rcancel* will be used to cancel requests to remote printers. |
| **-orm***machine* | The name of the remote machine is *machine*. |
| **-orp***printer* | The name of the printer to use on the remote machine is *printer*. |
| **-orc** | Restricts users to canceling only their own requests. The default is **not** to restrict the *cancel* command. |
| **-osi***rlpstat* or | Causes *lpstat* to use *rlpstat* to obtain the status of requests on remote printers. Specify the full pathname to ensure that the correct command is used. |
| **-osm***rlpstat* | The *rlpstat* model will be used to obtain the status of requests to remote printers. |

For example:

```
/usr/lib/lpadmin -plp3 -mrmodel -v/dev/null -ocm rcmodel
   -osm rsmodel -ormsystem2 -orplp3 -ob3
```

This example sets up remote spooling on a system. Although the command is shown on two lines, you need to type it continuously, with no Returns, at your terminal, or by typing a backslash (the Escape character) before any Returns.

The example names the local printer (lp3) and its model (rmodel). The device destination is specified as **/dev/null** because the Spooler requires a device name. In this case, networking software takes care of which device to send a request to.

The **-ocm** option specifies a model (**rcmodel**) that is used to cancel requests to the remote printer. The **-osm** option specifies the command (**rsmodel**) to use to obtain the status of remote printers. The **-orm** option specifies the system name to send remote spooling requests to: **system2.** The **-orp** option gives the remote printer name (lp3). You must include the **-ob3** option if connecting an HP-UX system to a BSD Unix system. This is because HP-UX systems use 4-digit spool requests and the BSD systems use 3-digit spool request numbers. The option makes the two spooling systems compatible.

On a remote system, the **rlpdaemon** command directs print jobs into the spooler.

To set up a remote spooler, execute the following commands:

```
/usr/lib/lpshut (To shut down the scheduler.)
/usr/lib/lpadmin -prem_lj -v/dev/null -mrmodel -ocmrcmodel -osmrsmodel
-ormhptml -orplj2
/usr/lib/accept rem_lj
/usr/bin enable rem_lj
/usr/lib/lpsched
```

On the remote system, add this line to the **/etc/rc** file:

```
/usr/lib/rlpdaemon -l
```

or, in the **/etc/inetd.conf** file, add this line:

```
printer stream tcp nowait root /usr/lib/rlpdaemon rlpdaemon -i -l
```

and then execute the command:

```
inetd -c
```

For these programs, set the user id to **root**:

- **/usr/lib/rcancel**
- **/usr/lib/rlp**
- **/usr/lib/rlpdaemon**
- **/usr/lib/rlpstat**

## Determining LP Spooler Status

As a System Administrator, you need to monitor the spooling system. Various options to the **lpstat** command allow you to print out the status of printers, list jobs yet to be printed, etc. Used without options, **lpstat** prints the status of all printing requests that the current user has made. An important option for you, the **-t** option, provides complete LP Spooler status information. For example, executing

```
lpstat -t
```

provides the following information:

```
scheduler is running
system default destination: lp
device for lp: /dev/lp
lp accepting requests since Jul 14, 15:37
printer lp now printing lp-165.  enabled since Jun 23 13:31
lp-165          williams        62489    Jul  9  12:53 on lp
lp-166          jones            1374    Jul  9  13:39
```

### lpstat Options

You can specify any combination of the following options when you execute **lpstat.** You can also include particular request IDs in the command to acquire status information about those requests.

| | |
|---|---|
| **-a***[list]* | Print the request acceptance status (with respect to **lp**) of logical destinations. *List* is a list of intermixed printer names and class names. If you do not specify *list*, the acceptance status of all logical destinations is printed. |
| **-c***[list]* | Print class names and their members. *List* is a list of class names. If you do not specify *list*, all classes and their members are printed. |
| **-d** | Print the system default destination for **lp**. |
| **-i** | Inhibit the reporting of remote status. |
| **-o***[list]* | Print the status of requests. *List* is a list of intermixed printer names, class names, and request IDs for which you want request status. If you do not specify *list*, **lpstat -o** has the same effect as **lpstat** (with no options). |
| **-p***[list]* | Print the status of printers. *List* is a list of logical printer names. If you do not specify *list*, the status of all printers is printed. |
| **-r** | Print the status of the LP scheduler. |
| **-S** | Print a status summary that includes the status of the LP scheduler, the name of the system default destination, a list of class names and their members, and a list of logical printers names and their associated special (device) filenames. |
| **-t** | Print all status information. |

| | |
|---|---|
| **-u***[list]* | Print the status of requests for particular users specified by the login names in *list*. If you do not specify *list*, the status of all user requests is printed. |
| **-v***[list]* | Print the path names of the physical devices associated with the logical printer names specified in *list*. If you do not specify *list*, the names of all of the logical printers and their associated physical devices are printed. |

## Controlling the LP Scheduler

The LP scheduler has to be running for files to be sent to a printer using the spooling system. The scheduler services all **lp** requests by routing them to an interface program associated with the specified printer or class of printers. Interface programs control the actual printing on the devices. As System Administrator, you will have to start up and shut down the LP scheduler, for example, when configuring a new printer on the system or when taking a printer out of service and moving requests from one printer to another.

To start the LP scheduler, enter:

```
/usr/lib/lpsched
```

Note that you **must** shut down the scheduler before using either **lpadmin** or **lpmove**. To shut down the scheduler, enter:

```
/usr/lib/lpshut
```

Remember to restart the scheduler after executing **lpadmin** or **lpmove**.

## Building Printer Classes

A **class** is a name assigned to represent one or more printers. Printers in the same class should all have the same ob3 option. When requests are sent to a class, they are serviced by the first available printer that is a member of that class.

The **-c** option of the **lpadmin** command inserts a printer into a particular class. If the class does not already exist, it is created. For example, you could associate the printer described above to a class with:

```
/usr/lib/lpadmin -plp -cclass1
```

This creates **class1** (unless it already exists) and places the printer **lp** in it.

To add another printer to **class1**, use:

```
/usr/lib/lpadmin -plp1 -cclass1
```

## Removing Destinations

As System Administrator, you are likely to have to remove a previously configured printing destination. LP Spooler destinations (printers or classes) are removed with the **lpadmin** command. To remove a printer from a specific class, use the **-r** option:

```
/usr/lib/lpadmin -plp -rclass1
```

Removing the last remaining member of a class causes the class itself to be deleted. .

To remove an entire class of printers, use the **-x** option:

```
/usr/lib/lpadmin -xclass1
```

To remove a printer that is not a member of a class, use **-x** option as follows:

```
/usr/lib/lpadmin -xlp
```

---

**NOTE**

**No printer or class of printers can be removed if it has any pending requests. Check the status of the printers using** lpstat. **If there are pending requests, use** lpmove **or** cancel **to move or delete the requests.**

---

## Moving Requests

As System Administrator, you may need to move requests from one destination to another, such as if one printer is down for repairs. You do this using the **lpmove** command. Before moving any print requests, shut down the LP Scheduler:

```
/usr/lib/lpshut
```

You can use **lpmove** in one of the following ways:

1. Move all requests for printer **lp1** to printer **lp2**:

```
/usr/lib/lpmove lp1 lp2
```

2. Move the request using its ID number **lp1-103** to printer **lp2**:

```
/usr/lib/lpmove lp1-103 lp2
```

**Lpmove** never checks the acceptance status of the new printer (that is, whether or not **accept** was executed on it) when it moves requests; therefore, you should execute:

```
lpstat -alp2
```

to see if the logical destination can accept requests before redirecting requests to it.

---

**NOTE**

**HP does not recommend using** lpmove **to move requests associated with remote printers, because the remote IDs may conflict and destroy a request.**

---

## LP Spooler Errors

In the event of errors, check these error message and perform the suggested corrective actions given below.

1. Corrupted member reported by **lpadmin**.

   Replace contents of **/usr/spool/lp/member** with **/dev/lp**, or the correct physical device name associated with the line printer.

2. Error message "lp has disappeared" occurs.

   Use **lpadmin -xlp** to remove the entry and use **lpadmin** to reinitialize the spooling system.

3. Error message "lp: destination 'lp' non-existent" occurs.

   Check mode, permission, group, ownership and existence of all **lp** directories and files. You may also need to remove **lp** from the member directory and make **/usr/spool/lp/default** a zero-length file.

4. Error message "lp: Can't open acceptance status file" occurs.

   Check ownership of **lp** directory.

5. Error message "lpadmin: Can't lock printer status" occurs.

   Check **lp** directory. It should belong to **lp** and should be in the group **bin**.

6. Error message "lp: Can't access 'file name'" occurs.

   The directory does not permit other to read.

7. Error message "lp: Can't open file" occurs.

   The file cannot be read by general users. Change the access permission mode of the file.

8. Error message "Cannot open /dev/lp" occurs.

   Printer off-line, cable disconnected, or printer power off. Correct problem and issue the enable **lp** command.

## Administering UUCP

This section briefly describes the UUCP network, outlines how to set up the network with direct and modem connections, and describes the files you will have to modify.

---

**NOTE**

**You can also use SAM to set up UUCP. Refer to *System Administration Basics* for information about SAM.**

---

## The UUCP Network

The UUCP network (UNIX to UNIX copy) is a collection of C language and shell programs that allow an HP-UX system to communicate with other systems. The network is connected to remote systems either through direct cables or through telephone lines.

Each system in the network is known as a node. Systems that can initiate conversations with other machines are called "active" nodes. These systems have an automatic call unit installed or a direct cable connection in place. Systems that cannot initiate conversations are called "passive" nodes. Passive nodes must wait until they are called by an active node before any communication can occur.

In addition to the hardware you install, the UUCP network comprises programs and support files. The UUCP system software is generally located in five directories:

- **/usr/lib/uucp** - executable programs and support files
- **/usr/bin** - more executable programs
- **/usr/spool/uucp** - spooling area and lock files for hardware devices
- **/usr/spool/uucppublic** - open directory accessible to remote systems
- **/usr/admin/menu** - administration subcommands

For more details, refer to the UUCP description contained in the *UUCP* volume of the Concepts and Tutorials set.

## HoneyDanBer UUCP Conversion

As of Release 2.0, HP-UX includes HoneyDanBer UUCP. The HoneyDanBer version of UUCP is easier to administer and provides more advanced communications devices and networks than Version 2. HoneyDanBer is also known as Basic Networking Utilities (BNU). If you have already updated your system to Release 2.0 or any later release, it is likely that this conversion has already been done.

Otherwise, you need to convert your Version 2 UUCP files to HoneyDanBer files. The files that need conversion are in the directory **/usr/lib/uucp**. Here are the Version 2 files and the corresponding HoneyDanBer files:

| Version 2 | HoneyDanBer |
|---|---|
| L-devices | Devices |
| L-dialcodes | Dialcodes |
| (no file) | Dialers |
| (no file) | Maxuuscheds |
| (no file) | Maxuuxqts |
| L.cmds and USERFILE | Permissions |
| (no file) | Poll |
| L.sys | Systems |

To perform the conversion:

```
mv /usr/lib/uucp/newconfig/Cvt /usr/lib/uucp

mv /usr/lib/uucp/newconfig/awk* /usr/lib/uucp

mv /usr/lib/uucp/newconfig/Make.cvt /usr/lib/uucp/Makefile

cd /usr/lib/uucp

make
```

Then you must also add a machine entry to the Permissions file and both **uux** and **uucp** commands to the COMMANDS line entry.

The following messages are displayed on your terminal:

```
mv Devices Devices.old
awk -f awkdevices L-devices > Devices
cp L-dialcodes Dialcodes
mv Permissions Permissions.old
awk -f awkperm L.cmds USERFILE > Permissions
mv Systems Systems.old
awk -f awksystems L.sys > Systems
```

Once you have used the **Make.cvt** script to convert your UUCP files to HoneyDanBer format, you must perform the following steps (refer to *UUCP: Concepts and Tutorials* before editing any of the HoneyDanBer UUCP files):

1. Move these files from **/usr/lib/uucp/newconfig** to **/usr/lib/uucp**:
   **Dialers**
   **Maxuuscheds**
   **Maxuuxqts**
   **Poll**

2. In **Systems**, be sure that the **time** and **retry** fields are separated by a semicolon ( ; ) rather than a comma ( , ), like this:

   ```
   sysname time[;retry] device[,protocol] baud_rate phone login_info
   ```

   Edit the **Systems** file to match this format.

3. In **Devices**, the HoneyDanBer format is:

   ```
   type device caller speed dialer
   ```

   For example, some lines in the **Devices** file might look like this after the conversion to the HoneyDanBer format:

   ```
   Direct tty0p4 0 9600 direct (Direct line to a remote system)
   Direct cua0p1 0 1200 direct (Direct line connected to a modem)
   ```

   You would edit the lines shown in this example to look like this:

   ```
   sys1 tty0p4 - 9600 direct (Direct line to a remote system)
   Direct cua0p1 - 1200 direct (Direct line connected to a modem)
   ```

   The corresponding Systems file entry would look like this:

   ```
   sys1 Any;5 sys1,g 9600 - login:uucp word:foo
   ```

When a device indicates a direct line to a remote system, the first field of the **Devices** file **must** be changed to match the third field in the **Systems** file (of an entry for the same device). Also, change **0** to a – for the caller entry, as shown in the example.

If you want to use the Version 2 **dialit** file instead of HoneyDanBer's dialing capability, make an entry like this in the **Devices** file:

```
ACU cul0pl cua0pl 1200 PROG/usr/lib/dialit ACUHAYESSMART /dev/cua0pl \T \S \P
```

In an entry like this, the `\T` is translated into a phone number at the appropriate speed (1200). `\S` is translated into the specified speed. `\P` is translated into the protocol specified in the **Systems** file.

---

**NOTE**

**In the Devices file, you must change all of the direct remote systems connections from Direct to <system_name>.**

---

4. If you are using Hayes/Hayes compatible modems, and the modem configurations are not set by the DIP switches (or stored in the nonvolatile RAM of the modem), then you will have to edit the following line in the **Dialers** file:

   ```
   hayes =,-,  "" /dATZ/r/c <rest of line>
   ```

   to look like this:

   ```
   hayes =,-,  "" /dAT/r/c <rest of line>
   ```

   The required configuration for DIP switches is: switches 3 and 8 are up; all other switches are down.

5. If the system name is not specified in **USERFILE**, add an entry to the **Permissions** file that looks like this:

   ```
   MACHINE = OTHER
   REQUEST = yes
   SENDFILES = yes
   READ = /usr/spool/uucppublic
   CALLBACK = no
   COMMANDS = rmail:uucp:uux:rnews
   ```

   After the conversion, execute the command:

   ```
   uucheck -v
   ```

   to check the contents of the Permissions file.

6. If you want to use **mh** for mail messages, edit this line in **/usr/lib/sendmail.cf** from:

   ```
   FU/usr/lib/uucp/L.sys %s
   ```

   to look like this:

   ```
   FU/usr/lib/uucp/Systems %s
   ```

Your UUCP conversion is now complete. Check the results of your conversion before you purge your old UUCP files.

After you have purged your Version 2 files, move all of the spooled files into their subdirectories under **/usr/spool/uucp**. To do this use the command:

```
sh Cvt
```

Some things to think about before you begin your conversion:

- If your files contain bad or duplicate information, they will still contain bad or duplicate information after you convert them. Clean up these files before converting them so you are working with valid data.
- Check the **Dialers** file for the type of modem you will use. If it is not in this file, add an entry to **Dialers** and also make changes in **Devices** to use that modem type.
- If you have modified **/usr/lib/dialit.c** and still want to use dialit for your connections, you will have to edit **Devices** to use the dialer **PROG/usr/lib/dialit**, with the appropriate parameters.
- The file **Permissions** will allow calls to come in and go out for all machines and logins. You should probably edit this file before you use the UUCP network.

## The Link

When you set up your UUCP link, you need to tell UUCP about the systems it will be talking to. Find out what other systems you want to be linked to and talk to their systems administrators to get the following information:

- System node name
- Times the system will accept calls
- Phone number of the modem
- Login name for **uucico**
- Login password

Provide the other System Administrators with the same information about your system. Enter this information into the file **Systems**, which is described later in this section.

## Direct Connect Installation

Direct connection between two Series 800 computers requires that the master-slave relationship be established. The connection is done through a MUX port on each computer.

1. Connect the two computers via a MUX port with the RS-232 cable.
2. Determine the device file name of the two ports.
3. Choose one system to be the master and the other the slave.

   The master system always initiates **uucp** transfers. The slave system has a **getty** process detecting any transfer initiated. The master system has no **getty** on the line.

4. Remove the **getty** entry from the **/etc/inittab** file on the master system.
5. Set up a **getty** entry on the slave system.
6. Set up a **uucp** log-in account in **/etc/passwd** and **/etc/group** on the slave system.
7. Customize **Devices**.
8. Customize **Systems**.
9. Customize **Permissions**.
10. Customize **Dialers**.
11. Make sure that the hostname is set in **/etc/rc**.

## Modem Connect Installation

This configuration requires access to a remote call-up system.

1. Install phone line.
2. Connect modem to a MUX port.
3. Determine the device file of that port; create the file with **insf** or **mksf** if necessary.
4. Set up a **getty** in **/etc/inittab**.
5. Set up a **uucp** log in entry in **/etc/passwd** and **/etc/group**.
6. Customize **Devices**.
7. Customize **Systems**.
8. Customize **Permissions**.
9. Customize **Dialers**.
10. Customize **/usr/lib/uucp/dialit.c** for modem type.
11. Compile **/usr/lib/uucp/dialit.c** to **/usr/lib/uucp/dialit**.
12. Copy **/usr/lib/uucp/dialit** to **/usr/lib/dialit**.
13. Make sure that the hostname is set in **/etc/rc**.

## File Descriptions

The following are descriptions of the supporting database files **Devices**, **Systems**, **Permissions**, **Dialers**, **Dialcodes**, **Maxuusched**, and **Maxuuxqts**, **uudemon.admin**, **uudemon.cleanu**, **uudemon.poll**, **Poll**, and **uudemon.hour**. You must modify these files to customize your system's use of the uucp network. Samples of these files are available in the directory **/usr/lib/uucp/newconfig**.

### Devices

The **Devices** file contains information on direct links and automatic call units.

Begin each entry in column 1 of the **Devices** file, otherwise you will get the message:

```
NO DEVICES AVAILABLE
```

when you try to use the device. An entry (for modems) in the **Devices** file typically looks like this:

```
 (type)    (device)    (caller)    (speed)    (dialer)
  ACU       cul0p2      cuaop2      1200       hayes
```

An example of an entry for a direct line to a remote system:

```
sys1       tty0p8       -          9600       direct
```

An example of an entry for a direct line connected to a modem:

```
Direct     cua0p1       -          2400       direct
```

- The **type** describes the type of link. The link can be:

| | |
|---|---|
| **ACU** | For links made through a modem. |
| **Direct** | For links made through a line to a computer or to a modem. |
| **sysname** | For direct links to a particular machine where **sysname** is the remote machine name. This indicates that the line associated with this entry in **Devices** is for a specific computer in the **Systems** file. Be sure that the entry in the **type** field matches the entry in the third field of the system file. |

- **Device** refers to the name of the special file in the **/dev** directory that corresponds to the serial port used for the UUCP link.
- **Caller** refers to the name of the special file in the **/dev** directory that specifies the automatic call unit.

- **Dialer** refers to the name of the dialer subroutine or program. These are some examples of the dialers supplied with the Dialers file:
  - direct (No dialer. The line is direct.)
  - 801 (Standard Bell 801 autodialer with 212 or 103 modem.)
  - hayes (Hayes modem)
  - ventel (Ventel modem)
  - vadic (Vadic modem)
  - HP2334Ag (with g protocol)
  - HP2334Af (with f protocol)
  - HP35141A
  - HP37212A
  - HP2334A (with interactive protocal)
- **Speed** contains the speed (baud rate) of the device.

You can also instruct UUCP to use another program (in this example, **dialit**) instead of **Dialers** to create a connection between systems. An example of such an entry (in your file, the entry will be all on one line):

```
(type)  (device)  (caller)  (speed)     (program name)      (devices used by dialit)
ACU     cul0p2    cua0p2    1200     PROG/usr/lib/dialit      /dev/cul0p2
ACUHAYESSMART               \T          \S          \P
  (ACU type)              (phone)     (speed)     (protocol)
```

## Systems

The **Systems** file contains information on modem connections. An entry in this file usually looks like this:

```
 (system) (schedule)

                (device) (speed)  (phone no.)     (chat script)
  losangeles Any ACU  1200  1-916-3618897  ogin:--ogin:\
  nuucp ssword:  external
```

- **System** (in this case **losangeles**) refers to the node name of the remote system. The first seven characters of the system name must be unique relative to other entries in the **Systems** file.
- The **schedule** field contains when the local system can call the remote system. It includes a day subfield and a time subfield, which are written with no spaces between the two subfields. In the example, the local can call the remote **any** day of the week. **Never** means that the local system should never call, but should wait to be called by the remote system. **Wk** means any weekday. You can also specify certain days of the week by using one or more of the keywords **Su**, **Mo**, **Tu**, **We**, **Th**, **Fr**, or **Sa**.

  The time subfield is specified using two 24-hour clock times separated by a dash. For example, 1800-2300 means that a system can only be called between 6 p.m. and 11 p.m. If the time subfield is omitted, the local system can be called any time of the day.

An example for a system that can be called on weekday evenings and all day on Saturday and Sunday:

```
chicago Wk1700-2200,SaSu
```

- The **device** field contains a pointer to an entry in the **Devices** file. This will be the device used for the call. For systems that are connected through a hardwired line (direct connection), this field should be the same as the first one.

  For example, sys in the **Systems** file

  ```
  sys Any sys 9600 - ogin:uucp ssword:none
  ```

  corresponds to sys in the device

  ```
  sys tty0p5 - 9600 direct
  ```

- **Speed** refers to the baud rate of the device.
- The **phone number** field contains the phone number that the modem uses to call the remote system. For direct lines, use a '-'. An equal sign in the phone number tells the system to wait for a dial tone (for example, 8 = means to dial 8 and then wait for a dial tone before dialing the rest of the phone number).
- The rest of the line is the **chat script**, a string that describes the initial conversation between the two systems. It describes the login password and special character sequences needed to complete the login procedure. The **chat script** contains sets of "expect-send" pairs separated by spaces, with optional "subexpect-subsend" pairs separated by hyphens:

  ```
        (subsend)     (send)          (send)

  ogin:-BREAK-ogin:  nuucp  ssword:  external
  (expect)  (subexpect)       (expect)
  ```

  In other words, expect **ogin**:, send **break** if timeout, then expect **ogin**:, send **nuucp**, expect **ssword**, and send **external**. This script uses the subsend-subexpect pair to give **uucico** more than one chance to log in.

## Permissions

The **Permissions** file contains two types of entries. These entries are **LOGNAME** entries that allow you to specify permissions for individual systems, and **MACHINE** entries that allow you to specify permissions for the local system when you call a remote system. A typical set of entries looks like this:

```
LOGNAME=uhpxsoe2 VALIDATE=hpxsoe2 REQUEST=yes SENDFILE=yes \
READ=/ WRITE=/ \

COMMANDS=/bin/rmail/:pr:col:lp:cat:date:ls:uucp:uux

MACHINE=hpxsoe2 REQUEST=yes \
READ=/ WRITE=/ \

COMMANDS=/usr/bin/uuname:hostname:rmail:pr:date:ls:uucp:uux:cat
```

These entries consist of option/value pairs. You can have as many of these option/value pairs as you want. Table 4-2 lists the options and their allowed values. The *Class* column uses the codes **M** or **L** to show whether an option can be used in a **LOGNAME** entry, a **MACHINE** entry, or both.

**Table 4-2. Option Values**

| Option | Class | Values | Description |
|---|---|---|---|
| LOGNAME | L | *login[:login:. . .]* | Specifies the login ids of remote systems that can log into the local system. |
| MACHINE | M | *sys[:sys2:. . .]* | Specifies machines that the local system can call under specified conditions. |
| REQUEST | M, L | *yes no* | Whether the remote system can request to set up file transfers from the local system. The default is "no"; it will be used if the REQUEST option is not given. |
| SENDFILES | L | *yes call* | Whether the remote site can execute locally queued requests during a session. "Yes" means that your system may send jobs queued for the remote system as long as it is logged in as one of the names in the LOGNAME option. The default is "call" (the queued files are sent only when the *local* system calls the remote machine). |
| READ | M, L | *path[:path2:. . .]* | Specifies directories that uucico can use for requesting files; the default is the uucppublic directory. If this option is used, all pathnames must be specified since these are not added to the default list. |
| WRITE | M, L | *path[:path2:. . .]* | Specifies the directories to which uucico can deposit files. The default is the *uucppublic* directory. If this option is used, all pathnames must be specified since these are *not* added to the default list. |
| NOREAD | M, L | *path[:path2:. . .]* | READ option or default exceptions. |
| NOWRITE | M, L | *path[:path2:. . .]* | WRITE option or default exceptions. |
| CALLBACK | L | *yes no* | Whether or not the local system will call back the calling system before transaction can occur. The default is "no". |
| COMMANDS | M | *cmd[:cmd2. . .]* | Commands that the remote system can execute locally. This option defaults to the command list in the *parms.h* header file, and overrides the default command list. |
| VALIDATE | L | *sys[sys2. . .]* | Verifies calling system's identity. Can be used with the COMMAND when specifying commands that may violate your system's security. |
| MYNAME | M | *name* | Links another system name to the local system. |
| PUBDIR | M, L | *dir* | Specifies the directory for local access. Default is the public directory. |

Here are some rules for modifying **Permissions**:

- Each option/value pair has this format:

  *option=value*

  Blanks are not allowed before or after the equal sign (=).
- Each line corresponds to one entry. A line may be continued by a backslash (\).
- Blanks are used to separate option-value pairs. If an option has more than one value, the values are separated by a colon (:).
- Comment line begins with a pound sign (#) and end with a new-line. Blank lines are ignored.

## Dialers

The **Dialers** file contains the instructions for dialing the various modems supported by uucp. Some dialers, such as the 801 dialer, are part of the the internal uucp code and do not have entries in this file. A typical entry looks like this:

```
rixon =&-%     "" \r\r\d $ s9\c )-W\r\ds9\c-) s\c :\T\r\c $ 9\c LINE
```

An entry in in this file has the following format:

*dialer subs expect-send [expect-send] . . .*

| **Field** | **Function** |
|---|---|
| *dialer* | The type of dialer. This field must match the fifth or any following odd-numbered field in **Devices**. In this field, “direct” means a direct cable connection, and no additional entries are needed. |
| *subs* | The translate string. The first pair of characters is mapped to the second character in the pair in the phone number dialed. This string is used to translate the characters = and - ( which UUCP uses to represent “wait for a dialtone” and “pause” ) into whatever equivalent characters the modem uses. |
| *expect-send* | This field is defined in the chat script field of the **Systems file**. The strings “\T” and “\D”are used to substitute the phone number string passed to the dialer. “\T” means to expand to the number in the **Dialcodes** file. “\D” means do not expand the string. |

As supplied, **Dialers** contains entries for the following modems:

- Hayes
- Ventel
- Vadic
- Rixon
- HP35141A
- HP37212A
- HP2334A

You can use various escape characters in the **Dialers**. These include:

| **Escape** | **Function** |
|---|---|
| \c | No new-line |
| \d | Delay for about two seconds |
| \D | Phone number without **Dialcodes** translation |
| \*ddd* | Send octal number *ddd* |
| \e | Disable echo checking |
| \E | Enable echo checking for slow devices |
| \K | Insert a break |
| \n | Send a new-line |
| \p | Pause for an instant |
| \r | Send a carriage return |
| \T | Send phone number with **Dialcodes** translation |

## Dialcodes

The **Dialcodes** file contains a location prefix, an area code, and any digits needed to get to an outside line. A typical entry in this file:

```
sunnyvale 9=1-405-555
```

In **Systems**, a corresponding entry could be:

```
calif Any ACU 1200 sunnyvale5557
```

After **uucp** reads this entry, it refers to **/usr/lib/uucp/Dialcodes** and sends the completed dialing sequence "9=1-405-5555557". The "=" tells the dialer to wait for a secondary dial tone before dialing the rest of the number.

## Maxuuscheds

**Maxuuscheds** specifies the number of **uusched** programs that can run simultaneously. The default is 2 programs.

## Maxuuxqts

**Maxuuqts** specifies the maximum number of **uuxqt** programs that can be run simultaneously. The default is 2 programs.

## uudemon.admin

The **uudemon.admin** file contains a shell script that sends status information to the UUCP login (**uucp**) using **uustat** commands with the -p and -q options. Execute **uudemon.admin** daily through an entry in the root **crontab** file. The default **uudemon.admin** entry in root **crontab** is:

```
48 8,12,16 * * * /bin/su uucp -c" /usr/lib/uucp/uudemon.admin" > /dev/null
```

## uudemon.cleanu

The **uudemon.cleanu** file contains a shell script that cleans up the HoneyDanBer log files and directories. This script updates archived log files so that no log information more than three days old is retained. The **uudemon.cleanu** script performs a log file cleanup by taking log files for individual machines from **/usr/spool/uucp/.Log**, merging the files, and then putting them in **/usr/spool/uucp/.Old** with other old log information. The script removes unneeded files and directories from the spool directories. After cleaning is complete, **uudemon.cleanu** sends **uucp** a summary of the information gathered during the day.

Execute **uudemon.cleanu** through an entry in the root **crontab** file. Run **uudemon.cleanu** daily, weekly, or as often as you want to, depending on the amount of UUCP activity that occurs on your system. For **uudemon.cleanu**, the default root **crontab** entry is:

```
45 23 * * * ulimit 5000; /bin/su uucp -c"/usr/lib/uucp/uudemon.cleanu" >
/dev/null 2>&1
```

You can increase the **ulimit** if your log files are large.

## uudemon.poll

The **uudemon.poll** file contains a shell script that polls the remote machines listed in the **Poll** file. This script creates work files for systems according to entries in **Poll**. This script should be executed by an entry in the root crontab file, and should be set to run once an hour (just before **uudemon.hour**) so that the work files will exist when **uudemon.hour** is run. The default root **crontab** entry:

```
1,30 * * * * " /usr/lib/uucp/uudon.poll > /dev/null
```

### Poll

This file contains information for polling certain systems. Each entry in **Poll** has the name of the remote system and the hours the system can be called. For example, this entry will allow polling of the "bearcat" system every three hours:

```
    bearcat 0 3 6 9 12 15
```

### uudemon.hour

The **uudemon.hour** file contains a shell script that calls UUCP programs each hour. The **uudemon.hour** script calls the **uusched** program to search the spool directory for unprocessed work files and to schedule these files to be sent to a remote system. The script then calls the **uuxqt** daemon to search for execute files in the spool directory that were received by the local system, but have not yet been processed. Execute **uudemon.hour** through an entry in the root **crontab** file. The default root crontab entry is:

```
    41,11 * * * * " /usr/lib/uucp/uudemon.hour > /dev/null
```

If you have a lot of UUCP traffic on your system, execute **uudemon.hour** once or twice an hour. If you do not have a large amount of UUCP activity, execute **uudemon.hour** about every four hours.

5

# System Administration Tasks

This chapter describes common system administration and maintenance tasks. To perform these tasks, you must log on as the root user with the root login password. If you do not remember it and have no means of obtaining the password, you must reboot the system, start up the system in single-user mode, and assign a new password for root, following the description given in Chapter 2 of this manual.

**CAUTION**

**After logging in as the superuser, do not leave the terminal unattended. Anyone can access the system as superuser and change or damage it inadvertently.**

## Communicating with System Users

The following are ways to communicate with your users via the HP-UX system:

- The **/usr/news** directory and the **news** command provide a way to send brief announcements to system users.
- You may put more pressing items (such as announcing an upcoming backup) in the message-of-the-day file, **/etc/motd**. Keep these messages short and current so users can read them quickly. Make sure that **/etc/csh.login** and **/etc/profile** contain the line "**cat /etc/motd**" so users receive the messages.
- Send longer messages or even major documents intended for specific users using the **mail** command.
- To write to users who are already logged on, use the **write** (for specific users) or **wall** (for all users) commands. The **write** command is intended for user-to-user dialog, while **wall** is intended for system-wide announcements. Note that if a user has executed the **mesg** command with the **-n** (no) option, write permission to that user's terminal is denied, and you cannot communicate with that terminal using the **write** command.
- When the superuser executes the **wall** (write all) command, all user protections are overridden; the command immediately sends its message to every user's terminal,

regardless of the tasks they are performing. Thus, if you are logged on as the superuser, avoid using **wall** unless it concerns a pressing matter such as an impending system shutdown (consider a user's irritation at receiving an unimportant message while editing a file).

As shipped, the files **/etc/profile and /etc/csh.login** contain entries to automatically notify each user at login of news, message of the day, and mail.

## Adding/Removing Users

This section covers the software or configuration aspects of adding or removing a user to or from the HP-UX system. If you are adding a terminal for the new user, you need to install and configure it before the user can use the system; see the "Adding/Removing Peripheral Devices" section of this chapter.

---

**NOTE**

**You can also use SAM to add or remove users using menus. Refer to** *System Administration Basics* **for more information.**

---

Each user's login environment is defined by an entry in **/etc/passwd**. Without this entry, the user cannot log in. To add a user to the system, you must add a line to this file and customize the user environment by specifying a home directory, selecting a shell or user interface program, etc. A complete description of the **/etc/passwd** file is in the **passwd(4)** entry in the *HP-UX Reference*.

Two approaches for adding users to the system are offered here. Both approaches require adding an entry in the **/etc/passwd** file and the procedure for creating that entry is explained next. Following that, a listing of a shell script (that partially automates the process of adding users to the system) is supplied. Finally, a step-by-step method for adding users is provided. If you expect to continually add users to the system, it saves time to create a shell script from the one included in the section "The Makeuser Script." If you have a single-user system or expect to add only one or two users to the system, the step-by-step process is probably your best choice. Both approaches accomplish the same task; the automation of adding users is the only functional difference between the two.

### Adding a New User

1. Add entries for the new user in **/etc/passwd** (and **/etc/group** if group membership is in effect).
2. Create a directory for that user on the appropriate file system, usually under **/mnt**.

3. Create and personalize **.login, .cshrc, .mailrc, .logout** and **.profile** in the user's home directory.

4. Set the user directory and file permissions, changing ownership of the files mentioned above to the new user.

5. Set an initial password, tell the user the password you set, and ask the user to change the password during the first login session. You may also implement password aging and require users to add a password during the first login.

## Removing a User

1. Copy the user account to the disk or tape.

2. Remove the user's directory (this is optional).

3. Disable the user account by placing an asterisk in the password field of the **/etc/passwd** entry. Or, you can delete the entry if you have removed the user's directory and the user is not the owner of other files on the system.

4. If the user is moved to another account, you can forward mail to the new account. Add the words "Forward to <forwarding **uucp** address>" as the first line in the **/usr/mail/$LOGNAME** file.

## Creating the /etc/passwd Entry

To create an entry in the **/etc/passwd** file for a new user, first log in to the system as the **root** user.

The format for each line (or each user entry) in the **/etc/passwd** file is:

- login name
- encrypted password
- numerical user ID (UID)
- numerical group ID (GID)
- comment field used for identification
- initial working directory
- command to execute

This file is an ASCII file. Each field within each user's entry is separated from the next by a colon. Each entry is separated from the next by a new-line character. Some sample **/etc/passwd** entries are shown below.

```
glenn:H.b7dJs6hlfPo:102:11:Glenn West x75147:/mnt/mktg/glenn:/bin/csh
lpe:lhKlPhw7fcG/o:104:11:Learning Products Engr:/mnt/mktg:/bin/csh
job:Z3jd;askj5087:106:11:Word Processing x78607:/mnt/mktg/prod:/bin/csh
```

The format for **/etc/group** is:

- group name
- encrypted password
- numerical group ID
- list of users allowed, comma used as a separator

This is an ASCII file. The fields are separated by colons. Each group is separated from the next by a new-line. For example:

```
mktg::11:glenn,job,lpe
```

If this is the first time you are following this procedure, it is a good idea to copy the original **/etc/passwd** file that was shipped with your system before continuing. This insures that you have a backup. To copy the file, type:

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

where **/etc/passwd.old** is the copy of the file.

Next, edit the file **/etc/passwd**. Add a line to the file describing the new user. The colon character (:) is used on this line to separate the various fields. The new line must have the form:

```
username::userid:groupid:comment:logindirectory:command
```

where:

| | |
|---|---|
| `username` | is the user's login name. It consists of 1 to 8 alphanumeric characters; the first character must be a lowercase alphabetic character. |
| `::` | represents an empty password field. Passwords and the **passwd** command are discussed later in this section. |
| `userid` | is the user ID, a **unique** integer value that the system uses to identify the user. If the real user ID is 0, that user has superuser capabilities. As the system was shipped to you, the real user ID 0 is associated with the **root** user. By convention, the values 0 through 99 are reserved for system use and the value 100 is used for a guest account. Therefore, pick any unused number greater than 99, but less than 60001, (except 200 and 201 which are used by the windex program) for this field. User IDs greater than 60001 are mapped onto 60001. The user ID is the third field in the **/etc/passwd** file entry: `yl:password:102:11:yitty x 2010:/mnt/mktg/eck:/bin/csh` The user ID (UID) in this example is 102. |

## NOTE

**You should assign unique user ID numbers to normal users.**

| | |
|---|---|
| `groupid` | is the group ID, an integer value shared by all members of the same group. This entry corresponds with the group entry in **/etc/group**; refer to the "Creating Groups/Changing Group Membership" section in Chapter 4 for details. The group ID is in the fourth field in the **/etc/passwd** entry. In the example shown previously, the group ID (GID) is 11. The group ID is in the third field of the group file. |

| | |
|---|---|
| `comment` | is a word or phrase that identifies the user or specifies the reason for the entry. Typically, this field contains the user's full name and other information such as the user's location or phone number. This information is displayed on the header of requests spooled to the printer via the **lp** command. |
| `logindirectory` | is the absolute pathname of the user login directory. This becomes the user's working directory upon login. The directory need not exist when the entry to **/etc/passwd** is made. However, the directory must exist before the user can log in. A user login directory is usually a subdirectory of the **/mnt** directory and may have the same name as the user login name. For example, a user whose last name is Young might have login name **young** and home directory **/mnt/young**. The directory should be owned by the user. |
| `command` | is the name of a single command to be executed for the user at login. This should be an absolute pathname. Typically, **/bin/sh** (or **/bin/csh**) is placed in this field to invoke the Bourne shell (or C shell) for the user. However, the name of any executable program or command may be placed in this field. The command can be either a compiled program or a shell script but no arguments to the command or script should be supplied. If the command field is left blank, **/bin/sh** is executed by default. When the user logs in, the command listed in this field is executed and control is passed to that program. When the program terminates, the user is logged out. |

## The "Makeuser" Script

This section describes the shell script for adding users to the system. This script assumes that certain files are located where they were when the system was shipped. If you have moved these files, edit the script to match their new locations. To use this script, you need to:

- Log in to the system as superuser or **root**.
- Use one of the text editors to create a file using a name such as **/etc/makeuser.** Call it what you want; the name **/etc/makeuser** is only a suggestion.
- Change the mode of the file by typing:

  ```
  chmod 744 /etc/makeuser
  ```

  This gives you read, write, and execute permission to the file but restricts the access of all other users to read permission.
- After creating the new user's **/etc/passwd** entry, execute the script by typing:

  /etc/makeuser *username*

  where *username* is the new username from the **/etc/passwd** entry.

Here is a partial "makeuser" Bourne shell script:

```
: #/etc/makeuser: create a new user

      USERS=/mnt/mktg
      if [ $# != 1 ]
      then
      echo "Usage:  makeuser name"
      exit 1
      fi
      if [ -d $USERS/$1 ]
      then
      echo "Home directory already exists."
      exit 1
      fi
      if grep \^$1\: /etc/passwd >/dev/null
      then
      :
      else
      echo "No password entry."
      exit 1
      fi
      mkdir $USERS/$1
      chown $1 $USERS/$1

      ls -ld $USERS/$1
      ls -la $USERS/$1
      echo "Remember to add the new user to a group."
```

## The Step-by-Step Method

If you decide not to use the "makeuser" script, here is a step-by-step procedure for accomplishing the same task. The following procedure assumes that certain files are located where they were when the system was shipped. It also assumes that an appropriate entry for the new user already exists in the **/etc/passwd**.

1. Create a login directory for the user with the **mkdir** command by typing

   ```
   mkdir /mnt/username
   ```

   where *username* is the new user's login name. The entire pathname (**/mnt/***username*) must match the **logindirectory** field of the user's **/etc/passwd** entry.

2. Create the login files for the user (the name and contents of the login file depends on the default shell, for example, **.profile** for **sh** and **.cshrc** and **.login** for **csh**). If the login file exists in a user's login directory, the shell attempts to execute that file at the end of the login process. This file typically contains shell commands and environment variable definitions that customize the user's environment and/or automatically run one or more programs.

3. Because you (**root** or superuser) created the new user's directory and copied the login file(s) into the user's directory, you own the user directory and all files in that directory. To change the ownership of the directory to the user, type:

   ```
   chown username /mnt/dirname
   ```

   where: *username* is the name of the user. **/mnt/***dirname* is the login directory name of the user.

To change the ownership of all the files in the user directory, type:

```
chown username /mnt/dirname/.[a-z]*
```

where: *username* is the name of the user. **/mnt/***dirname***/.[a-z]*** matches all the files in the directory that begin with a period and are followed by a lowercase letter and then anything else.

4. Check the ownership and permissions of the user's directory by typing:

   ```
   ls -ld /mnt/dirname
   ```

   and check the status of the user's files by typing:

   ```
   ls -la /mnt/dirname
   ```

   See the **ls(1)** entry in the *HP-UX Reference* for an explanation of the display.

5. Add the user to the **/etc/group** file. If you are using the group access features available on HP-UX, use the **chgrp** command to change the group ownership of all files created for the user. The **chgrp** command format is:

   ```
   chgrp group-name /mnt/dirname/.[a-z]*
   ```

   Refer to the *HP-UX Reference* for a detailed description of the **chgrp** command.

Whether you used the "makeuser" method or the step-by-step method, the new user is now installed on the system. Proceed to the following section for further considerations.

### Some Optional Items

**If you are using HP-UX's group access capability**, you may want to add a user to a group or change the group ID associated with the user's files. The user's group must exist in **/etc/group** and the user must be made a member of that group before you can use the **chgrp** command to change the group ID associated with the user's files. A user can be a member of multiple groups. For details on these operations, refer to the "Creating Groups/Changing Group Membership" section in Chapter 4 and the **chgrp(1)** and **group(4)** entries in the *HP-UX Reference*.

Depending on the needs of your installation, consider using the **chmod** command to change the protection mode of the user's login directory and files. A commonly used mode value is 755 (octal). This provides read, write, and execute (search) permission for the file's owner but provides only read and execute (search) permission to all others.

If you are adding a terminal for this user, refer to the "Adding/Removing Peripheral Devices" section to set up the terminal and add entries to the **/etc/ttytype** and **/etc/inittab** files.

## Setting a New User Password

When a new user login is created, the new user does not have a password but may log in without one. Depending on the security needs of your installation, you can:

- Ask the user to create a password.
- Create a password and notify the user.

- Force the user to create a password when logging in for the first time.

Procedures for the last two choices follow.

### Creating a Password for the User

To set a password for the user, log on to the system as **root**. Then type

```
    passwd username
```

and respond to the system's prompt for a new password. More detailed information about this command is in the **passwd(1)** entry in the *HP-UX Reference*. This command sets the new user's password to the password you typed.

### Forcing the User to Create a Password

You can set a parameter that forces a user to create a password the first time he/she logs in to the system. To do this, put two periods in the optional **aging** field in the user's **/etc/passwd** entry. The aging field is separated from the password field with a comma (see the **passwd(4)** entry for details). Thus a typical entry for a user without a password is:

```
    john:,..:105:77:J Jackson,production:/mnt/john:/bin/sh
```

The system displays

```
    Your passwd has expired.  Choose a new password.
```

followed by the system password prompts:

```
    Changing password for username: {password expected here}
    New password:  {new password expected here}
    re-enter new password:  {repeat above}
    login:
```

The user cannot log on to the system without creating a new password.

## Removing a User from the System

To remove a user from the system,

1. Remove the user's $HOME directory and any other directories and files to release the disk space for other users. The easiest way to remove the user's files is to type:

   ```
   find / -user username -exec rm {} \;
   ```

   Note that this **removes all** the user's files and directories so be sure that no other users need them.

2. Delete the user's entries from the **/etc/passwd** and **/etc/group** files.

3. Delete the user's login name from all the **/etc/group** entries (in case the user is in multiple groups).

4. If you also wish to remove the terminal associated with that user, delete the terminal's entries from the **/etc/ttytype** and **/etc/inittab** files. Refer to Chapter 3, "System Start Up and Shut Down" in this manual, and **inittab(4)** in the *HP-UX Reference* for details.

## Suspending a User from the System

To prevent a user from using the system for a time, you can temporarily revoke his/her login privileges by modifying that user's entry in **/etc/passwd**. If you replace the user's password with an asterisk (*), he/she cannot use the system until you delete that asterisk and assign a new password.

For example, to prevent Jane from using the system, change the login entry as follows:

```
jane:*:101:5:Jane :/mnt/jane:/bin/sh
```

## Overview of the I/O System

At the center of the HP-UX system is the **kernel**, the part of the operating system that deals directly with the hardware. The kernel provides low-level functions for the rest of the operating system and insulates it from the details of the hardware.

Compiled inside the kernel are software programs known as **device drivers** that control I/O for a particular device or class of devices. In order for the kernel to communicate with an I/O device, the appropriate driver must be linked into the kernel. Drivers are optional parts of the kernel that may be added or omitted when the kernel is built using the **uxgen** utility program.

Since each driver may have to handle more than one device, you need to assign a unique number to each device that the driver supports. This allows the driver to distinguish between various devices. This number is called the **logical unit** (LU) number and is normally assigned starting with 0. If a system has 10 disk drives on it, each disk is assigned an LU number from 0 to 9. You also need to specify the hardware address of each I/O device so the driver can correctly locate the devices it supports. You specify the LU number and hardware address of I/O devices in the S800 file that generates the kernel during **uxgen**.

## I/O Paths

I/O addressing occurs over a bus that forms an I/O path to and from memory and I/O device modules. Figure 5-1 illustrates the I/O path on Series 800 computers.

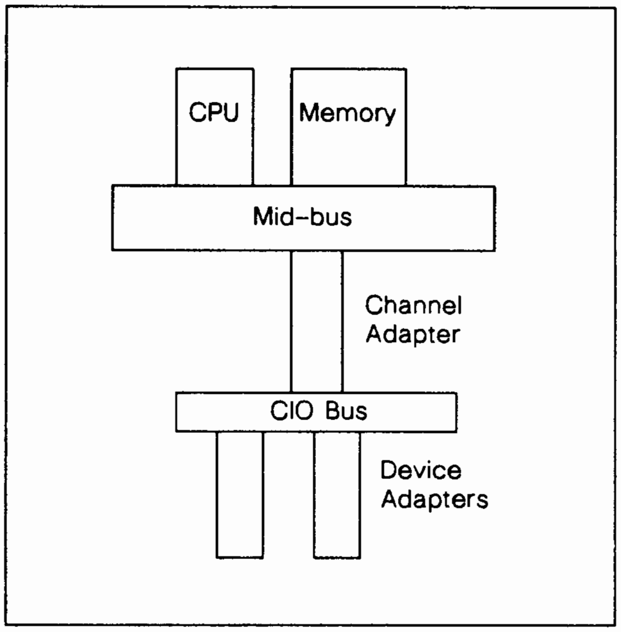


**Figure 5-1. I/O Path**

Notice that peripherals hook up to **device adapters** on computers implementing Mid-bus/CIO architecture. Device adapters communicate with the CIO bus, through a channel adapter and to the Mid-bus to access the CPU and memory.

## Device Drivers

When you run **uxgen** to recompile the kernel, information about the device drivers and hardware addresses is provided by the **S800** file (the full pathname for this file is **/etc/conf/gen/S800**). The **S800** file contains the names of the device drivers and the hardware addresses matched to them. In other words, it is with the **S800** file that I/O hardware and software are linked. The key to reconfiguring I/O lies in understanding the **S800** file. The role of the **S800** file is shown in Figure 5-2.

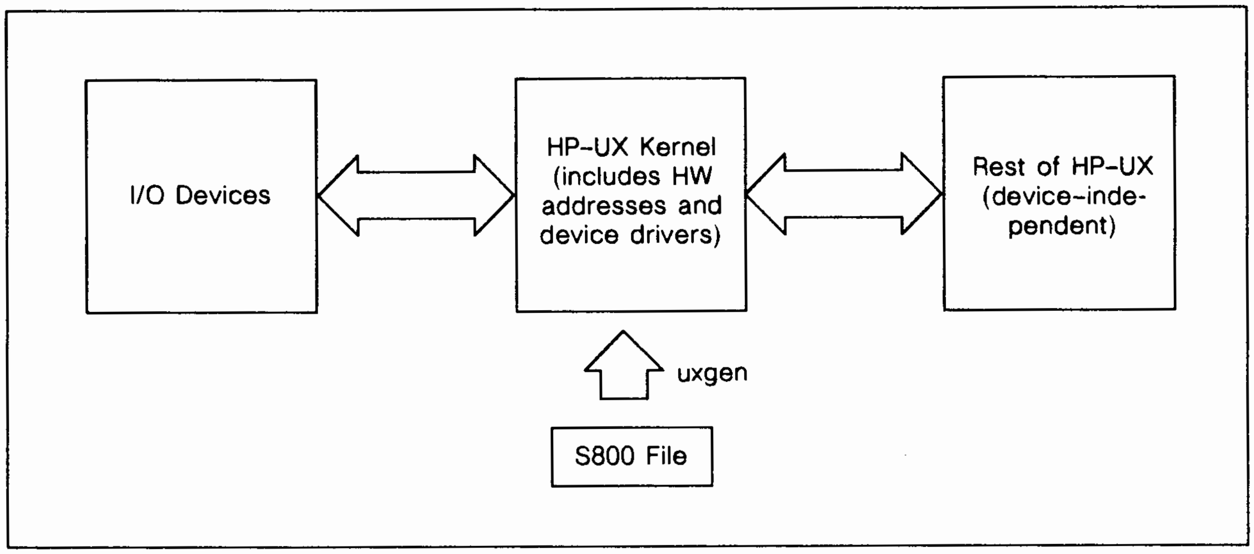

I/O Devices
HP-UX Kernel (includes HW addresses and device drivers)
Rest of HP-UX (device-independent)
uxgen
S800 File
LG200036_056

**Figure 5-2. The S800 File in Relation to the System**

For the kernel to communicate with a device, it must have the information listed below. The type of I/O transfer, block or character, is discussed in the section "Block and Character Special Files" later in this chapter.

- Name of the I/O driver
- Type of I/O transfer (block or character)
- LU number of the device
- Device address

The kernel recognizes a driver and an associated device by numbers called major and minor numbers. Each driver configured in the system is assigned a **major** number. Drivers that support both block and character I/O are assigned two major numbers: a block major number and a character major number. The system assigns major numbers when the kernel is built using **uxgen**. The kernel uses the major number and the device type (block or character) to locate the driver routine that services the I/O request.

Then the driver uses the minor number which includes the logical unit number to service the request. **Minor** numbers pass information to the driver regarding how to handle data and how to configure the device. Minor numbers are described in detail for each driver later in this chapter.

## Logical Unit Numbers

Logical Unit (lu) numbers allow one device driver to handle many devices. A different lu number is assigned to each device supported by a particular driver. The lu numbers for a driver start at 0, and continue upward in sequence.

For example, in the HP-IB block for CIO slot 0, lu numbers 0 through 3 are assigned for devices driven by **disc0**. If more devices are to be driven by **disc0**, they are assigned lu numbers in sequence. In the HP-IB block for CIO slot 6, more devices are assigned to the **disc0** driver; these devices receive lu numbers 4, 5, 6 and 7 as shown in the following portion of the S800 file:

```
cio_ca0 address 4{
        hpib0           address 6 {
                        disc0    lu 4 address 0;
                        disc0    lu 5 address 1;
                        disc0    lu 6 address 2;
                        disc0    lu 7 address 3;
        }
}
```

---

**NOTE**

**When modifying the S800 file, do not assign duplicate lu numbers for the same device driver. For example, if you assign lu 1 for disc0, do not assign another lu 1 for disc0. If you assign duplicate lu numbers for partitions other than args, dump, and swap, the uxgen program will issue error messages and abort when you try to recompile the kernel.**

---

## Special Files

**Special files** are also called **device files**. These files link devices with the kernel, and allow you to change the device drivers and hardware addresses compiled into the kernel. These files allow the HP-UX operating system to be device independent. The operating system handles each device as if it were a file, referencing the device by name. Thus, all I/O operations in HP-UX are performed through the file system. One or more special files are created for each I/O device in the system. Figure 5-3 shows how the special files link devices with the kernel.

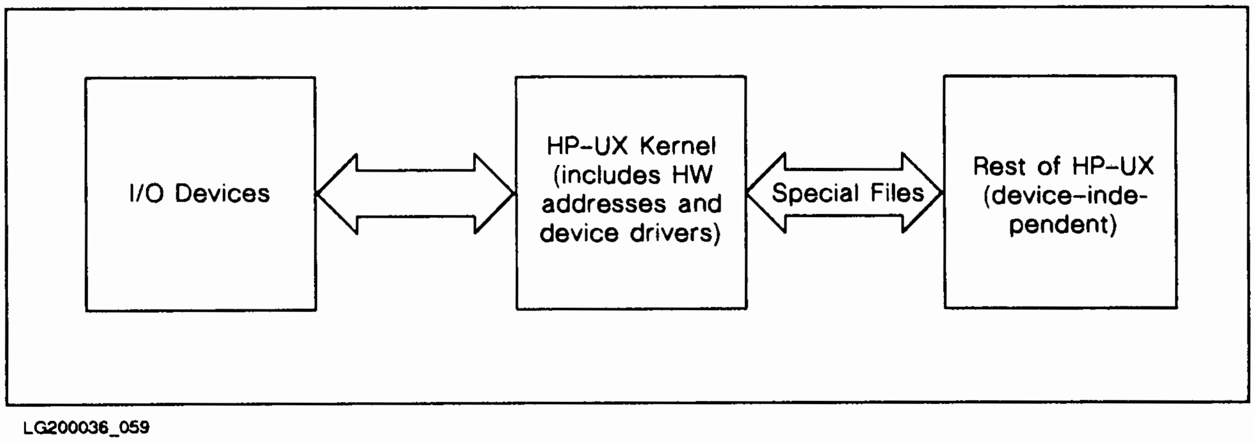

I/O Devices
HP-UX Kernel (includes HW addresses and device drivers)
Special Files
Rest of HP-UX (device-independent)
LG200036_059

**Figure 5-3. Special Files Link the Kernel with the Rest of HP-UX**

By convention, special files reside in the **/dev** directory. Instead of containing data as in normal files, they include the major and minor numbers and a flag indicating "block versus character" mode of transfer.

You must be logged in as **root** to create special files. You can use the following commands to create special files:

- **insf** (INstall Special Files) creates all default special files for the system.
- **mksf** (MaKe Special File) to make a special file for a specific device.
- **mknod** to create special files only if you are familiar with the HP-UX I/O system and know the major and minor numbers of the device.

Determining the major number for each driver and how to encode the bit fields in the minor number can be tedious. **insf**, **mksf**, and **lssf** (LiSt Special Files) simplify the task of creating and listing special files. They let you make and list special files without having to know the major and minor numbers.

To make the system easier to administer, HP-UX provides a set of default special files for each I/O device. The **insf** command lets you create default special files for I/O devices using these default special files. After system generation, the I/O configuration (driver names, major numbers, hardware addresses, etc.) is recorded for the kernel in the **devices** file in **/etc/conf/S800**. You should then copy the **devices** file to **/etc** and run **insf**.

You should use **insf** to create special files because it:

- Creates special files with standard default pathnames.
- Creates multiple special files when needed. Specifies the following for each special file:
  - block or character I/O transfer
  - LU number
  - multiple sections for disk files
  - different options for a particular device (for example, "raw", "rewind on close", "in vs out", etc.)
- Sets file permissions for system security.
- Sets file ownership when needed.
- Creates diagnostic special files when needed.

If the default set of special files is not adequate, you can create a new special file using **mksf**. The **mksf** command creates a single special file when you supply the driver name, the pathname, the logical unit number, and other optional parameters (for example, raw, rewind, caps, etc.) The procedures for creating special files are presented for several devices later in this chapter.

To create special files for all I/O devices specified in **/etc/devices**, enter:

```
cd /dev
insf
```

To create a special file for an HP-IB disk (**disc0**) with logical unit 5, unit 0, section 9, path /dev/dsk/c5d0s9, enter the following:

```
cd /dev/dsk
mksf -d disc0 -l 5 -u 0 -s 9
```

More information appears in the sections entitled "Creating disc0 Special Files," "Creating disc1 Special Files," and "Creating disc2 Special Files" in this chapter.

The **lssf** command describes any existing special file. For example, the following command displays information about the special file **c5d0s9**:

```
lssf /dev/dsk/c5d0s9
disc0 lu 5 unit 0 section 9 address 8.6.1 /dev/dsk/c5d0s9
```

## Block and Character Special Files

This section describes block and character special files. The description is given to help you determine whether to create a **block special file** or a **character special file** when adding another supported peripheral device.

**Block special files** transfer data of fixed blocks through system buffers (used as cache) to speed up I/O transfer. Devices that use block special files are random access storage devices such as tape drives and disks.

**Character special files** typically transfer data blocks of varying sizes without using any system buffer cache. Typically, the following devices use character special files: terminals, printers, plotters, digitizers, magnetic tape drives, cartridge tape drives, and, on occasion, disk mass storage devices. Character I/O transfer is also called "raw" I/O transfer, and sometimes character devices are called "raw" devices.

Certain devices have block and character special files. The system accesses ordinary files and directories with block special files, using the buffer cache for speed. In database management, direct access is needed. A database manager can use a character special file to access an area of the disk that the file system is not currently using.

For mountable file systems, the device file used for communicating with the device referencing the file system must be a block special file. There must also be a character special file for a mountable file system. The character special file is used by the **fsck** command in file system check and maintenance and by **newfs** to create a file system on a disk.

## Hardware Addressing

Each peripheral in your system has a unique **hardware address**. The hardware address tells the computer where to locate a particular peripheral. Figure 5-4 shows typical hardware addresses.

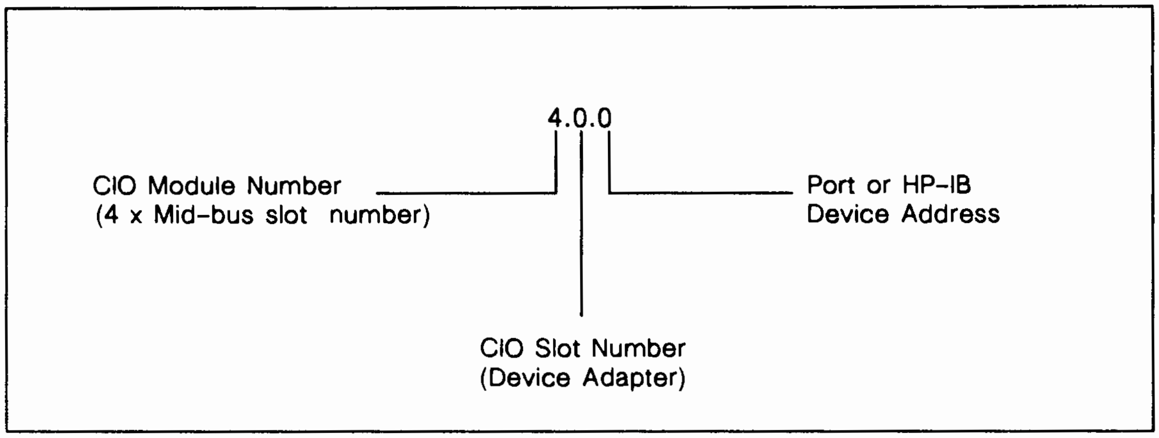


**Figure 5-4. Sample Hardware Addresses**

As Figure 5-4 shows, the three parts to a CIO hardware address specify a path through the I/O system.

In the example, the hardware address says in effect: "Go to the Channel Adapter represented by module number 4 (the card in Mid-bus slot 1). Then go to the Device Adapter in CIO slot 0. Finally, go to the device attached to this Device Adapter which has the HP-IB address of 0."

Channel Adapter cards are installed in Mid-bus slots. The Channel Adapter serves as an interface between the Mid-bus and the rest of the I/O system. Specifically, the Channel Adapter performs the conversion between the Mid-bus and the lower speed Channel Input/Output (CIO) bus. The CIO bus is the general-purpose I/O bus on CIO systems. The Device Adapter (DA) cards are attached to the CIO bus. The DA cards allow the computer to communicate with specific kinds of peripherals (for example, HP-IB cards for HP-IB devices). I/O addressing is shown in Figure 5-5.

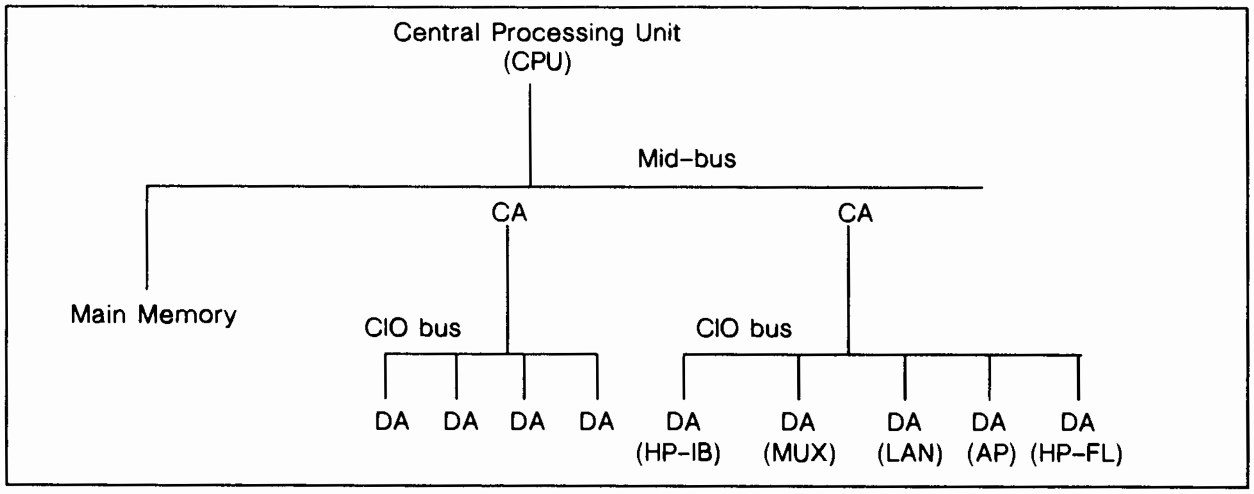


**Figure 5-5. I/O Addressing**

## Module Number — First Part of the Hardware Address

The location of the Channel Adapter determines the Module Number, which is the first part of the hardware address. To find the Module Number, find out which Mid-bus or Precision Bus slot the Channel Adapter is installed in. Then multiply the slot number by 4. If a Channel Adapter is in Mid-bus slot 2, for example, the Module Number is 8 (4 x 2).

Permissible locations of the Channel Adapter vary on specific Series 800 computers.

## CIO Slot Number

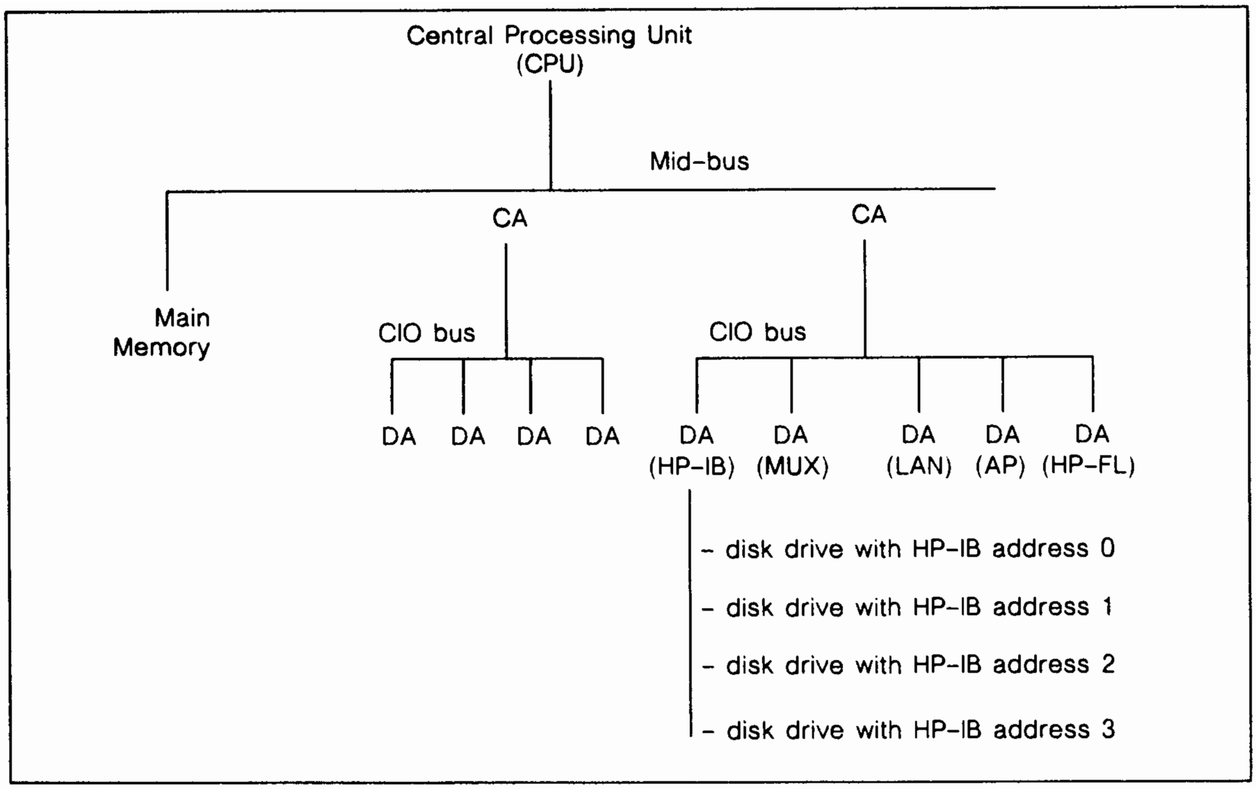
The second part of the hardware address is easier to figure out than the module number. This number is the CIO slot number of the Device Adapter card. For example, an HP-IB card in CIO slot 2 means that all the devices attached to it will have 2 as the second part of their hardware addresses.

For a specific I/O configuration, only one kind of Device Adapter card can be installed in a given slot. For example, a system may be configured so that only a MUX card can go in CIO slot 5. The default I/O configuration is suitable for most systems. However, you can change the configuration by editing the **S800** file, and regenerating the kernel with the **uxgen** command. Changing the configuration is covered later in this chapter.

## HP-IB Address or Port Number

The HP-IB address or port number is the third part of the address. Since a Device Adapter can have several devices attached to it, this number indicates which device you mean to address. It would be the HP-IB address (with disk drives and magnetic tapes) or the port number (with terminals). You set the HP-IB address on the device itself.

Figure 5-6 shows how HP-IB addresses differentiate between the disk drives attached to an HP-IB card.



Central Processing Unit (CPU)
Mid-bus
CA
CA
Main Memory
CIO bus
CIO bus
DA
DA
DA
DA
DA (HP-IB)
DA (MUX)
DA (LAN)
DA (AP)
DA (HP-FL)
- disk drive with HP-IB address 0
- disk drive with HP-IB address 1
- disk drive with HP-IB address 2
- disk drive with HP-IB address 3

**Figure 5-6. HP-IB Addressing**

## CIO Adapters

Most peripheral devices on CIO systems are connected to the computer through CIO device adapters (such as HP-IB, HP-FL, LAN, or other interface cards). CIO device adapters connect to the CIO bus which connects to the Mid-bus through a CIO channel adapter. A few devices, such as graphics devices, connect directly to the Mid-bus through interface cards. On some Series 800 computers (such as Models 840 and 825), the processor and memory controller are connected directly to the Mid-bus. On other models (such as the Model 850), the Mid-bus is connected to a bus converter that connects to the System Main Bus (SMB). The processor and memory controller are connected to the SMB, in this case.

The Mid-bus slots are labeled in hexadecimal (0 to f). Each slot can contain up to four independent hardware addresses known as module numbers, so slot 2 controls modules 8 to 11. Slot numbers are marked on the Mid-bus card cage. For each Mid-bus slot, a pair of numbers indicate the range of modules which exist for that slot (for example, 0-3, 4-7, 8-11, 12-15, ... 60-63). Modules are numbered in decimal. You need to specify the proper module numbers in the io statement in the **uxgen** input file to configure specific peripheral devices.

The CIO channel adapter is an interface between the Mid-bus and the CIO bus. This allows multiple CIO card cages to be connected to the HP 9000 Series 800 computer. The last two CIO card cage slots (slots C and D) are reserved for HP use on Model 840 computers (slot C contains the Access Port card and slot D contains the buffer card). The Access Port card for the Model 850 must be installed in slot 4 of CA(D).

Models 825, 835, and 850 do not use the buffer card. An Access Port is optional on Models 825 and 835. If you have one, it must reside in slot 0. When configuring HP-UX, use the decimal address of the CIO card slot. The **uxgen** utility interprets all I/O addresses as decimal numbers.

Model 850 computers contain two Mid-bus units: Mid-bus 0 and Mid-bus 1. These units are connected to the System Main Bus (SMB) through bus converter cards. The SMB is a high-speed processor to memory bus. The bus converter connects the high-speed processor memory bus to the Mid-bus and the associated I/O hardware.

If only a single bus converter is used, it must be for Mid-bus 0. Mid-bus 0 contains module numbers 0 to 63; if the bus converter for Mid-bus 1 is installed, Mid-bus 1 contains module numbers 64 to 127. The bus converter occupies the first module of each Mid-bus (modules 0 and 64). Therefore, the first slot on Mid-bus 0 and Mid-bus 1 contains module numbers 4 to 7 and 68 to 71, respectively. On Model 850 computers, valid module numbers for channels are 4, 8, 68, and 72 (additional slot numbers are available through CIO expanders).

Refer to the *Installation and Configuration Guide* for your computer to find out more about specific devices and to determine which devices connect to particular CIO device adapters. Some devices are further divided into parts. For example, disks may have multiple units (disk and cartridge tape) and each unit is divided into sections.

## Software Addressing

To communicate with an I/O device, the appropriate driver and the corresponding device hardware addresses must be configured into the kernel. This is done with the io statement in an input file which is processed by **uxgen** as the kernel is being built.

A sample **io** statement for a CIO system is shown below:

```
 io {
   cio_ca0 address 8 {
      hpib0 address 0 {
         disc0 lu 0 address 0;
         disc0 lu 1 address 1;
      }
      hpib0 address 1 {
         disc0 lu 2 address 0;
         disc0 lu 3 address 1;
      }
      mux0 lu 0 address 2;
   }
}
```

The preceding **io** statement configures the kernel with four disk drives. Two HP-IB cards are connected to the same CIO channel adapter (module address 8 - Mid-bus slot 2). Each HP-IB (CIO slot 0, 1) has two disk drives connected at HP-IB addresses 0 and 1. Each disk is assigned a logical unit number (0 - 3).

The **mux** driver controls both the device adapter and terminals on devices connected to the MUX card. These devices have software modules but their addresses are obtained differently. Only the hardware address is entered in the special file. Terminals such as the HP2392 are examples of these types of devices.

Notice that only two levels of addressing are required in **io** statements for peripherals on Precision Bus architecture: the module number and the HP-IB or port address. This statement configures the kernel with four disk drives: two are connected to an HP-IB card in slot 1 (address 4 = 1 x 4) and two are connected to an HP-IB in slot 4 (address 16 = 4 x 4).

## Special Files Needed by HP–UX

Some special files are needed by HP-UX for normal operation. They are automatically built when the command insf is executed. The following information explains what special files the system needs and how to recreate them if they are accidentally removed. Normally, you should not remove these special files.

**syscon**
**systty**
**console**

HP–UX uses these files to access the system console. If necessary, recreate these as follows:

```
cd /dev
insf -d cn
```

**diag0** Used by HP–UX diagnostics. If necessary, recreate it as follows:

```
cd /dev
insf -d diag0
```

**dmem** Used by HP–UX diagnostics. Recreate it as follows:

```
cd /dev
insf -d dmem
```

**kmem** Virtual memory access to kernel.
**mem** Physical memory access to kernel.
**null** "bit bucket"

You can recreate all three special files as follows:

```
cd /dev
insf -d mm
```

**swap** Used to access 'swap' area. Recreate it as follows:

```
cd /dev
insf -d sw
```

**tty** Used to access a user terminal. Recreate it as follows:

```
cd /dev
insf -d sy
```

## Pseudo–Terminal Drivers pty0/pty1

Pseudo–terminal drivers **pty0** and **pty1** support pseudo–terminals. When building a kernel, the number of pseudo–terminals is specified in the **npty** statement in the **uxgen** input file. Refer to the **pty(7)** entry in the *HP-UX Reference* for more information about pseudo–terminals. The number of pseudo–terminals is determined by your application program needs and subsystem requirements. Refer to the subsystem installation manual for more information. For example, if you have the ARPA Services/9000 product, the remote login feature (**rlogin**) requires the **ptyx** special files.

## Special Files for ptys

A pseudo-terminal requires two drivers, a master (**pty0**) and a slave (**pty1**). Each driver requires a special file. The only parameter used in making a pseudo-driver special file is the **pty** number. The **pty** numbers start with 0 and increase to the number of pseudo-terminals used minus one (**#ptys** minus 1). There are two naming conventions used for **pty** special files, see **insf(1M)**. Some HP-UX commands depend on these naming conventions; therefore, the command **insf** should be used to make the special files. The **insf** command generates two special files for each driver and links the special files together.

To create a default set of special files for pseudo-terminals, perform the following:

1. Enter the following:

   ```
   cd /dev
   insf -d pty0
   ```

   This creates the following special files:

   ```
   ptyp<number>
   ptyq<number>
   ptyr<number>
                               (where <number> is in hexadecimal, from 0 to f)
   ptym/ptyp<number>
   ptym/ptyq<number>
   ptym/ptyr<number>
   ```

   Each pair of files with the same number is linked together (for example, **pty0** and **ptym/pty0** are linked together).

2. Enter the following:

   ```
   cd /dev
   insf -d pty1
   ```

   This command creates the following special files:

   ```
   ttyp<number>
   ttyq<number>                (where <number> is in hexadecimal, from 0 to f)
   ttyr<number>

   pty/ttyp<number>
   pty/ttyq<number>
   pty/ttyr<number>
   ```

   The files with the same number from each set are linked together.

To create a single master **pty** file:

```
mksf -d pty0 -m pty_number filename
```

To create a single slave **pty** special file, enter the following:

```
mksf -d ptyl -m pty_number filename
```

## Adding/Removing Peripheral Devices

This section describes how to add to or remove supported peripheral devices from HP 9000 Series 800 computers.

Before adding a peripheral device, you should be thoroughly familiar with the HP-UX I/O system and the standard software driver and special file naming conventions. The I/O system, software drivers, and special files are described in the section "Overview of the I/O System" in this chapter. The naming conventions are discussed on the **insf(1m)** manual page in the *HP-UX Reference*. In addition, you should be familiar with **uxgen** and the system generation process before reconfiguring the kernel (see Chapter 7).

**NOTE**

**You may also use SAM to add or remove peripherals to the system using menus. Refer to** *System Administration Basics* **for more information on SAM.**

Follow these steps to add peripheral devices to your system. The tasks are summarized below and shown in the flowchart in Figure 5-7. Specific software installation procedures are explained later in this chapter.

1. Using the guidelines presented in the *Installation and Configuration Guide* for your computer and the installation manual supplied with the peripheral device, determine the best place (in terms of HP-IB bus addresses, shared sets of I/O resources, expected usage, etc.) to locate the peripheral.

2. Determine whether the kernel contains the correct drivers for the peripherals. Check **/etc/devices** file for a list of the installed drivers.

3. If the kernel contains the correct drivers, go on to step 4. If not, edit the S800 file in **/etc/conf/gen** and execute **uxgen S800**.

4. Determine if the special file necessary to communicate with the peripheral device already exists on your HP-UX system. The default special files reside in the **/dev** directory and follow the naming conventions explained on the **insf(1m)** page in the *HP-UX Reference*.

5. If the special file exists, go to step 6. If the appropriate special file does not exist for the device in question, you have to create one. You can use either the **insf** or the **mksf** command to create the special file. Information and examples of special files are shown in the sections on adding specific peripherals. For example,

creating a special file for a tape drive is found in the "Adding a Tape Drive" section.

6. Shut down the system using the **shutdown** command if you have to **uxgen** the kernel, or if you have to add an interface card. Shut down and power down the computer only if you have to add an interface card.

7. Connect the peripheral device and interface card to the location/address specified in the S800 file.

**CAUTION**

**Never install or remove an interface card while the computer is powered up.**

If the peripheral device requires an interface card, set the appropriate switches on the card and install the card in the computer. Then set any required switches on the device and connect it to the computer (or interface card).

If you change the switch settings on an HP-IB device, reset the peripheral before attempting to address it. Refer to the appropriate peripheral operations manual for how to reset the particular peripheral. If installing an HP 27140 MUX card, no switches need to be set. After installation is complete, note the slot number and the HP-IB address if the device uses the HP-IB interface card.

8. Test the peripheral device using the peripheral selftest. Required testing depends on the specific peripheral that was added.

9. Power up and reboot the system. You can further test the peripheral now.

Refer to Table 5-1 for the names of the drivers and the general types of devices they support.

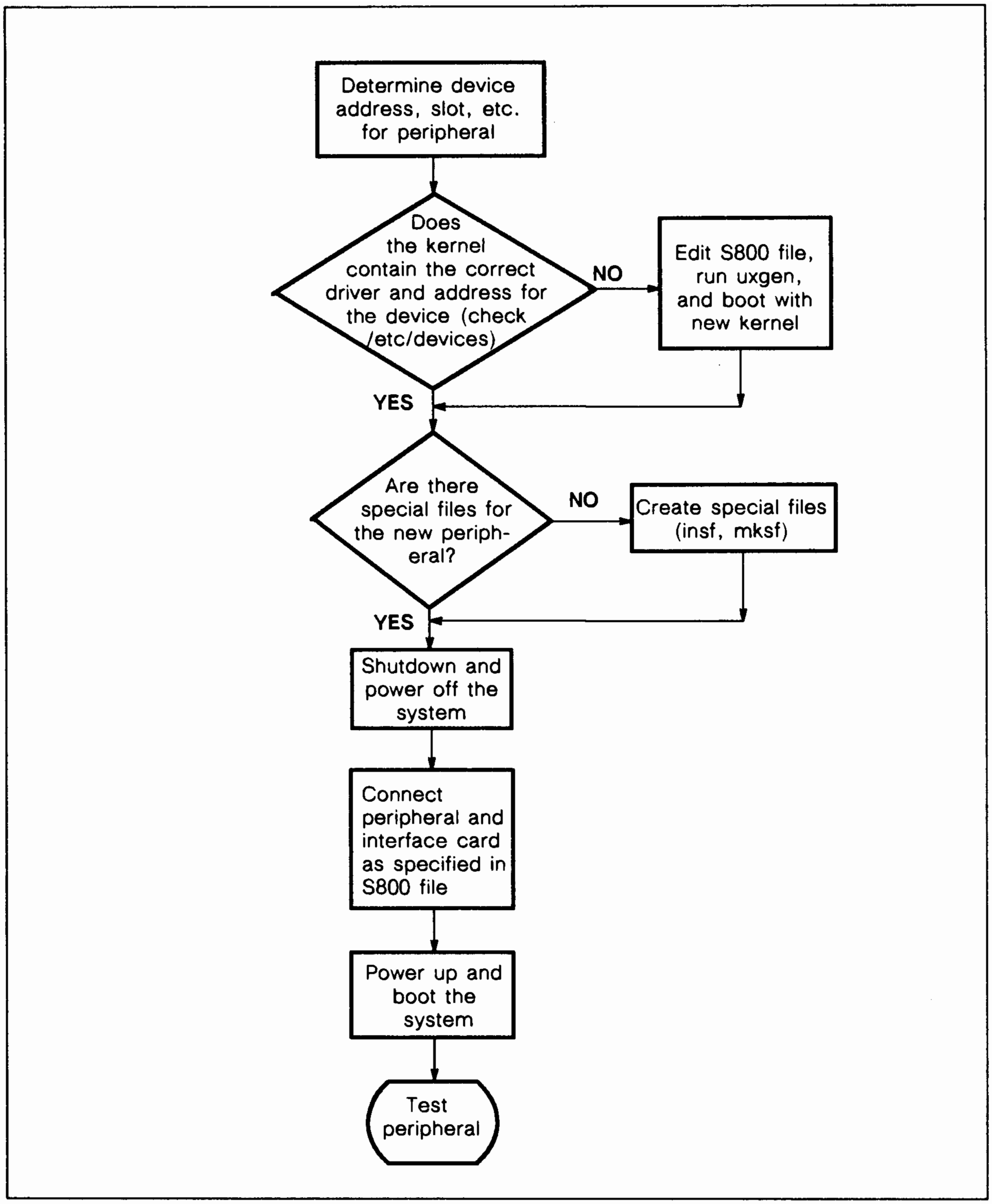

Determine device address, slot, etc. for peripheral
Does the kernel contain the correct driver and address for the device (check /etc/devices)
NO
Edit S800 file, run uxgen, and boot with new kernel
YES
Are there special files for the new periph- eral?
NO
Create special files (insf, mksf)
YES
Shutdown and power off the system
Connect peripheral and interface card as specified in S800 file
Power up and boot the system
Test peripheral

**Figure 5–7. Flowchart for Adding New Peripherals**

Table 5-1. I/O Drivers

| Driver Name | Supported Devices |
|---|---|
| disc0 | CS/80 and SS/80 devices (disks and cartridge tapes) connected via HP-IB interface (27110 card) |
| disc2 | CS/80 devices (disks) connected via an HP-FL interface (27111 card) |
| gpio0 | CIO general-purpose parallel I/O interface (27114 card) |
| graph0 | 98720A or 98550A graphics controller and its HIL modules |
| graph2 | 98730 (Models 825 and 835 only) |
| hil0 | Various input devices such as keyboards, mice, control knobs, and touchscreens (associated with hil, hilkbd, and hilknob drivers) |
| hpib0 | HP-IB Device Adapter Manager to CIO/HP-IB interface |
| instr0 | All other HP-IB peripherals (plotters, printers, general HP-IB instruments) |
| ite | Internal terminal emulator for color display terminals |
| lan0 | CIO Networking (27125 card) |
| lpr0 | 256x line printers (using CIPER protocol) |
| lpr1 | 2932/2934 and 2235 line printers (using Amigo protocol) |
| mux0 | RS-232 (serial) peripherals (27140 card); includes terminals, printers, and plotters |
| tape1 | 1/2 inch magnetic tape drives |

## Adding a CIO Expander

CIO expanders or expansion bays that increase system I/O capabilities are available for Series 800 computers. Figure 5-8 shows part of an S800 file with a CIO expander (added to a Model 825 computer). This section explains how to add an expander.

1. At the HP-UX prompt, issue the command:

   ```
   cd /etc/conf/gen
   ```

2. Copy the existing S800 file to a different name so you will have a backup copy in case of problems. For example:

   ```
   cp S800 S800BACKUP
   ```

3. Edit the S800 file so it has the correct drivers and hardware addresses for the new I/O configuration. (Refer to sample S800 file after this section.)

4. Recompile the kernel using **uxgen** using the edited S800 file as input:

   ```
   /etc/uxgen S800
   ```

5. Copy the old kernel (**/hp-ux**) and the old devices file (**/etc/devices**) in case the new kernel does not boot. For example:

```
cp /hp-ux /SYSBCKUP
cp /etc/devices /etc/DEVBCKUP
```

6. Change the working directory:

```
cd /etc/conf/S800
```

7. Move **hp-ux** to **/hp-ux** and **devices** to **/etc/devices**:

```
mv hp-ux /hp-ux
mv devices /etc/devices
```

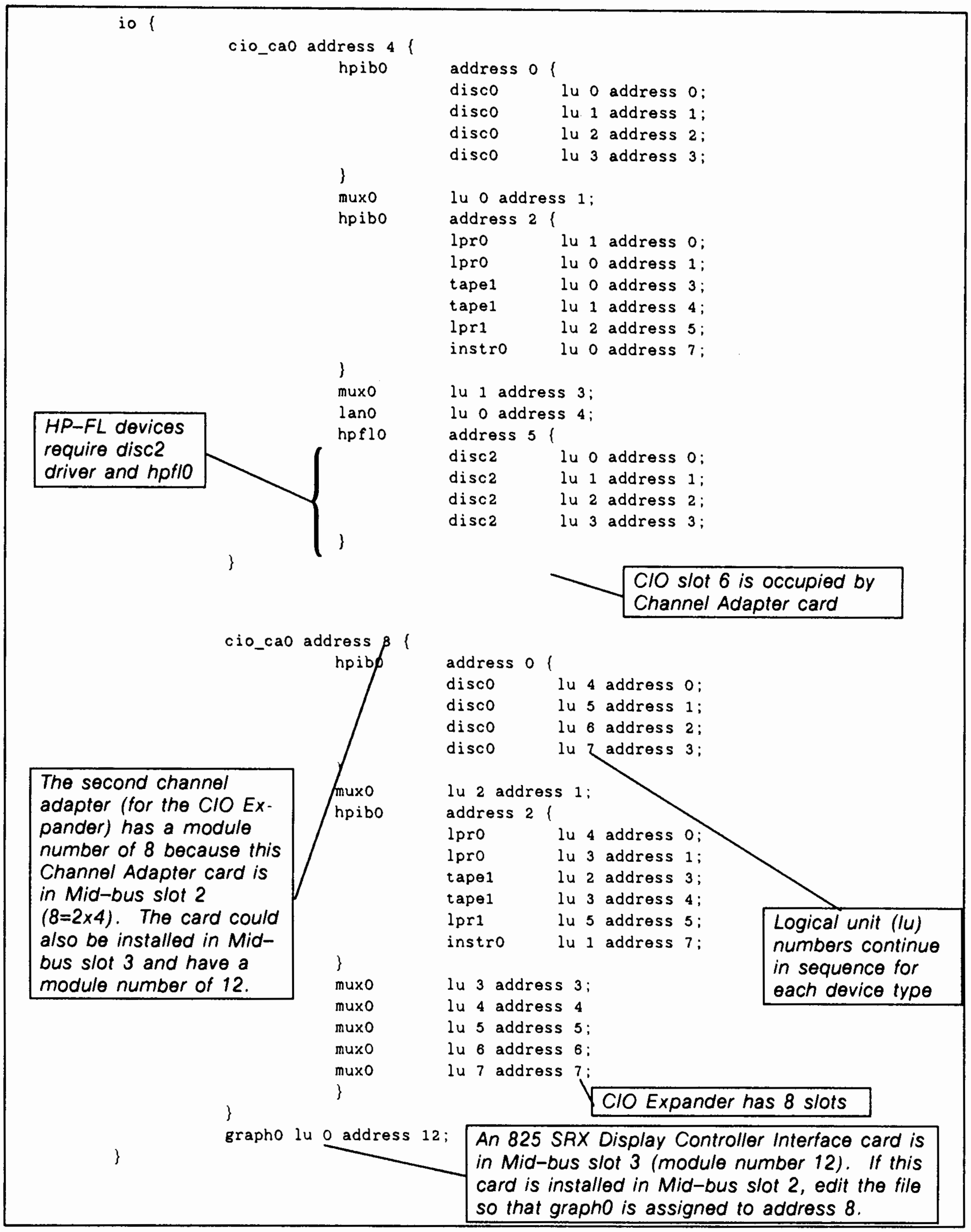
8. Create the special files for the new configuration:

```
cd /dev
/etc/insf
```

9. Shut down the system and turn the power off.

10. Attach the expander and install the CIO cards.

11. Turn on the system and reboot.



```
        io {
                cio_ca0 address 4 {
                        hpib0       address 0 {
                                    disc0       lu 0 address 0;
                                    disc0       lu 1 address 1;
                                    disc0       lu 2 address 2;
                                    disc0       lu 3 address 3;
                        }
                        mux0        lu 0 address 1;
                        hpib0       address 2 {
                                    lpr0        lu 1 address 0;
                                    lpr0        lu 0 address 1;
                                    tape1       lu 0 address 3;
                                    tape1       lu 1 address 4;
                                    lpr1        lu 2 address 5;
                                    instr0      lu 0 address 7;
                        }
                        mux0        lu 1 address 3;
                        lan0        lu 0 address 4;
                        hpfl0       address 5 {
                                    disc2       lu 0 address 0;
                                    disc2       lu 1 address 1;
                                    disc2       lu 2 address 2;
                                    disc2       lu 3 address 3;
                        }
                }

                cio_ca0 address 8 {
                        hpib0       address 0 {
                                    disc0       lu 4 address 0;
                                    disc0       lu 5 address 1;
                                    disc0       lu 6 address 2;
                                    disc0       lu 7 address 3;
                        }
                        mux0        lu 2 address 1;
                        hpib0       address 2 {
                                    lpr0        lu 4 address 0;
                                    lpr0        lu 3 address 1;
                                    tape1       lu 2 address 3;
                                    tape1       lu 3 address 4;
                                    lpr1        lu 5 address 5;
                                    instr0      lu 1 address 7;
                        }
                        mux0        lu 3 address 3;
                        mux0        lu 4 address 4
                        mux0        lu 5 address 5;
                        mux0        lu 6 address 6;
                        mux0        lu 7 address 7;
                        }
                }
                graph0 lu 0 address 12;
        }
```

HP–FL devices require disc2 driver and hpfl0
CIO slot 6 is occupied by Channel Adapter card
The second channel adapter (for the CIO Expander) has a module number of 8 because this Channel Adapter card is in Mid–bus slot 2 (8=2x4). The card could also be installed in Mid–bus slot 3 and have a module number of 12.
Logical unit (lu) numbers continue in sequence for each device type
CIO Expander has 8 slots
An 825 SRX Display Controller Interface card is in Mid–bus slot 3 (module number 12). If this card is installed in Mid–bus slot 2, edit the file so that graph0 is assigned to address 8.

LG200036_058

**Figure 5–8. Portion of S800 File Showing CIO Expander Added to Model 825 Computer**

## Adding Terminals and Modems

The following sections explain how to add terminals to your system, how to configure the terminals, and tell you about special considerations to be aware of during this process. The number of terminals and ports you can add depends on the model and configuration of your computer system. Your HP representative can discuss the appropriate number for your system to optimize performance, provide sufficient disk space, and effectively use available memory.

---

**NOTE**

**You can also use SAM to add terminals and modems using menus. Refer to** *System Administration Basics* **for more information.**

---

### Terminal Configuration Information

The *Installation and Configuration Guide* supplied with your computer discusses the hardware aspects of connecting a terminal to your system. This section offers the software configuration information you need.

After connecting the hardware, terminals must be configured so they can communicate with HP-UX. If a particular configuration option is not available on your HP terminal, the option is already properly chosen (as a default value) by the terminal.

Use the *Installation and Configuration Guide* to determine the typical terminal and data communications parameters. The manual supplied with the terminal describes how to use the function keys to configure the terminal. Generally, you press a key that chooses the "terminal configuration" option and alter the appropriate fields by answering prompts from the terminal's configuration program.

Except when using the terminal as a system console (discussed below), you may use any **baud rate** that the terminal can handle. The baud rate setting on the terminal **must** match the baud rate parameter in the **getty** command located in the terminal's entry in the **/etc/inittab** file as discussed below. Otherwise, it must reference a **gettydefs** entry that cycles the baud rate to select the correct one.

If you are using the terminal as the system console, set the terminal's baud rate at 9600. After the system is installed and running, you may change some of the configuration parameters to suit your own needs. Further information is provided in the next section ("Special Considerations for Terminals") and in the **getty(1M)**, **gettydef(4)**, and **inittab(4)** entries in the *HP-UX Reference*.

### Setting Up the System Console

Like any terminal, you need to install the device to be used as the system console and make sure it is addressed in the **io** statement of the S800 file. In addition, a specific line in the S800 file configures a terminal or graphics device as the system console. If your console is on a MUX card, port 0 of that MUX (usually mux 0) is reserved for the console; it must be present and set to 9600 baud. The S800 file shows:

```
console on mux0;    /* defaults to lu 0 */
```

If your console is a graphics interface, lu 0 of that interface is reserved for the console; it must be present. The S800 file must specify:

```
console on ite;
```

and include the **ite** driver with the line:

```
include ite;
```

## Special Considerations for Terminals

For terminals, as well as modems, printers, or X.25 packet assembler/disassembler (PAD) that connect via RS-232 cable, consider the following:

- Determine if a MUX card needs to be added or moved. On systems using CIO/Mid-bus architecture, several MUX cards are already configured in the CIO slots. To change the configuration, you need to reconfigure the system using **uxgen**.
- Select which of the six ports to use for the device. Attach the device to an available port on the terminal port distributor (TPD, also known as the junction panel).

## /etc/inittab Entries for Terminals

To add a terminal to the system, you must perform the steps described in the following "Terminal Installation Summary" section and add entries to the **/etc/inittab** file. This file displays a login prompt on configured terminals. The file, **/etc/inittab**, is described in Chapter 3 of this manual. This section discusses entries specific to terminals.

Most **/etc/inittab** entries have the form:

```
id:rstate:action:/etc/getty -txxx specialfilename N # comment field
```

The first three fields (shown as **id**, **rstate**, and **action**) are discussed in **init(1M)** and **inittab(4)** in the *HP-UX Reference*. The typical values for these fields are: **id** = unique two character string, **rstate** = 2, and **action** = **respawn** (for continuous scanning of **inittab**).

The two-character string **id** is arbitrary but must be unique for each entry. It is used to refer to the same entry/process in other states. **rstate** indicates the run levels the **getty** will be run in, see **init(1M)**. The **respawn** flag specifies that the command in the command field (such as **getty**) is to be re-invoked once the process terminates (typically, when a user logs off the system).

The fourth field contains the **/etc/getty** command for a terminal; it can contain commands for other functions. It can be followed by up to three parameters. The first parameter, **-txxx**, is the optional time-out option for use with modems. The second parameter, **specialfilename**, is the filename (**tty0p4**), not the complete pathname (**/dev/tty0p4**), of the terminal's or modem's character special file. The named file must reside in the **/dev** directory. The third parameter to **getty**, represented by **N**, specifies a speed indicator for **getty**; a value of 9600 is common for "hardwired" (9600 baud terminal) lines, a value of 1200 is common for dial-up (300/1200 baud modem) lines. Graphics devices ignore any speed indicator. The **N** parameter indexes into the **/etc/gettydefs** file. It is possible to set up a terminal to cycle through 22 different baud rates by formatting your **gettydefs** file properly. This file supplies information on the speed and terminal settings and what the login prompt should be. If the user enters a "break" on the terminal keyboard, it is an indication that the terminal speed is incorrect. The **getty** process will move to the next entry indicated in the **gettydefs** file.

On a multiuser system, be certain to set up **/etc/inittab** terminal entries **for each terminal** connected to the system. For example, to add a terminal on **/dev/tty0p4** the **/etc/inittab** entry would be:

```
c4:2:respawn:/etc/getty tty0p4 9600 #terminal at rob's desk
```

Note that the **id** field **c4** corresponds to the last two digits of the special file (**tty0p4**) for the terminal on which **getty** is invoked. This convention is often used with "continuous" (respawn) **getty** processes that get killed in the single-user run level but is **not** required syntax: any two-character string will suffice. **Respawn** causes the "login:" prompt to be displayed again after a user logs out. Refer to the "System Boot and Login" section in Chapter 3, and to the **getty(1M)**, **gettydefs(4)**, and **inittab(4)** entries in the *HP-UX Reference* for further details.

## Terminal Installation Summary

1. You may attach terminals to each port available on the junction box (except the one that is reserved for use with the Access Port).

2. For each terminal to be installed, you need to use the appropriate device file for the terminal.

   The device file format for terminal is:

   ```
   /dev/ttyXpY
   ```

   where: X = logical unit number and
   Y = port number of the MUX card.

   For example, to access a terminal attached to port 3 on the first MUX card (slot 1), use **/dev/tty0p3**. Note that port 0 of the first MUX card (slot 1) is usually configured as the system console.

3. For each terminal enabled, a terminal entry must be made in **/etc/inittab** (this file is described earlier in this chapter). The easiest way to add the new entry is to duplicate an existing entry line and then change the device filename and ID.

4. Implement the change in **/etc/inittab**.

5. Enter: `telinit q`

   Make sure that a **getty** was started on that port. If a question mark (?) appears in the **getty** column, something has gone wrong. Refer to Chapter 8, "Troubleshooting" for more information.

## Modem Installation Summary

1. Install the phone line.

2. Connect the modem to a MUX card port using a modem cable.

3. Connect the modem to the phone line.

4. Verify that the device port access type is set correctly (refer to Chapter 5 of the *Asynchronous Serial Communications Programming Manual* - 92453-90002).

5. Set up a **getty** entry in **/etc/inittab**. For example:

```
e0:2:respawn:/etc/getty -h ttyd2p0 1200     # modem 555-2463
```

## Adding a New Multiplexer Card

Follow these steps to add a new multiplexer. Be sure that you are familiar with **uxgen** and the system generation process. Otherwise, read the "System Generation Process" section in Chapter 7.)

1. Enter: `cd /etc/conf/gen`

2. Make a copy of the S800 file under a different name as a backup. For example,

```
cp S800 S800BACKUP
```

   Edit the S800 file to add the new multiplexer (**mux0**) to the **io** statement.

3. Run **uxgen** using the edited file as input:

```
/etc/uxgen S800
```

4. Copy the current kernel (**/hp-ux**) in the **root** (/) directory and the current devices file (in **/etc/devices**). Remember their names. You can reboot from the old kernel and restore the old devices file in case the new kernel does not boot.

```
cp /hp-ux /SYSBCKUP
cp /etc/devices /etc/DEVBCKUP
```

5. Change working directory to S800 by entering:

```
cd /etc/conf/S800.
```

6. Move **hp-ux** to **/hp-ux** and **devices** to **/etc/devices**:

```
mv hp-ux /hp-ux
mv devices /etc/devices
```

7. Make the special files for the new multiplexer. One way to do this is to enter the following:

```
cd /dev
insf -d mux0 -l <lu>
```

8. Shut down the system and turn off the system power.

9. Install the multiplexer.

10. Turn on the system power and reboot.

## Driver mux0

The driver **mux0** controls the 27140A 6-port multiplexer. See Figure 5-9 for the port device file minor number format.

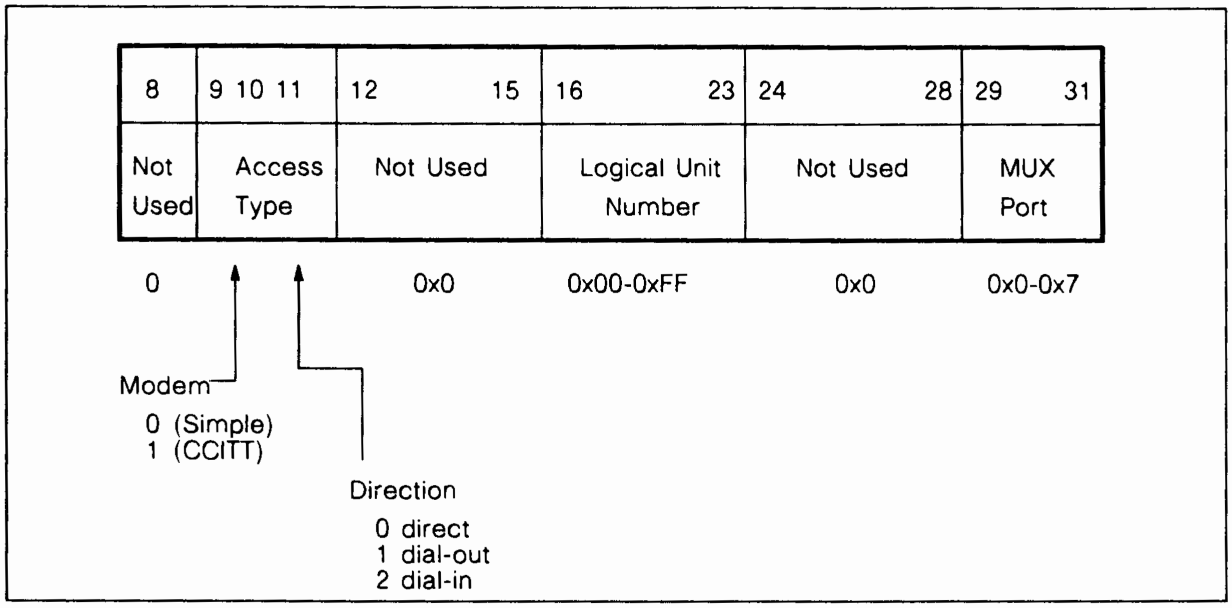


**Figure 5-9. Port Device File Minor Number Format**

### Addressing

Each multiplexer card must be connected to a CIO channel adapter. When building the kernel, the address of each multiplexer must be entered in the **uxgen io** statement. The address to enter is the CIO slot number (in decimal).

```
io {
   cio_ca0 address 8 {
      mux0    lu 0 address 1;
      mux0    lu 1 address 3;
      mux0    lu 2 address 8;
      mux0    lu 3 address 9;
      mux0    lu 4 address 10;
      mux0    lu 5 address 11;
   }
}
```

The preceding **io** statement would configure the kernel for 6 multiplexers at CIO slot numbers 1, 3, 8, 9, 10, 11. Each multiplexer is connected to the CIO channel adapter at module address 8 (Mid-bus slot 2).

Each multiplexer is assigned a logical unit number (0 - 5). Since each multiplexer contains 6 ports, this configuration can support 36 RS-232 lines.

### Special Files for mux0

Two drivers (**mux0** and **mux1**) control the multiplexer card. Although only **mux0** is specified in the **uxgen io** statement, the other driver (**mux1**) is automatically loaded into the kernel. Driver **mux1** is used to download firmware to the multiplexer and for running diagnostics. For each **MUX** card installed in the system, the following device files must be created:

- One device file for each port on the **MUX**, a total of six (6) for each card.
- One device file for the **MUX** for firmware downloading.
- One device file for the **MUX** for running diagnostics.

The special file for **mux0** includes the following parameters:

| | |
|---|---|
| **logical unit number** | Each multiplexer is assigned a logical unit number (LU). |
| **port number** | The port number (0 - 6) is specified in the special file. |
| **port type** | A port can be 'call-in', 'call-out' or 'hardwired'. |
| **CCITT flag** | This flag causes the port to follow the CCITT recommendations. |

The special file for **mux1** includes the following parameters:

| | |
|---|---|
| **logical unit number** | Each multiplexer is assigned a logical unit number (LU). |

## Adding Graphics Terminals

An HP-UX system can have a maximum of four graphics interfaces. You need to determine the appropriate number for your system to optimize performance, provide sufficient disk space, and effectively use available memory.

To install a graphics terminal:

1. You need to use **mksf** to create the appropriate special file for each graphics terminal to be installed. The special file format for a graphics terminal is:

   ```
   /dev/ttyiX
   ```

   Where `x` is the logical unit number.

   For example, to access the graphics terminal on the first graphics card, use **/dev/ttyi0**.

2. For each graphics terminal, you need to make an entry in **/etc/inittab** (this file is described earlier in this chapter). The easiest way to add an entry is to duplicate an existing entry line in **/etc/inittab** and change the device filename and ID.

3. Implement the change in **/etc/inittab**.

   Enter: `telinit q`

   This causes **init** to reread the **inittab** file and spawn the new **getty**. It will not disrupt users.

   Enter: `ps -el`

   Make sure that a **getty** was started on that graphics terminal. If a question mark appears in the **getty** column, something has gone wrong. Refer to Chapter 8, "Troubleshooting" for more information.

## Adding a New Graphics Card

The default number of graphics interfaces in the HP-UX system is one. If you want to have more than one, you need to regenerate the kernel using **uxgen**. Be sure that you are familiar with **uxgen** and the system generation process. Otherwise, read the "System Generation Process" section in Chapter 7.

To add a new graphics interface:

1. Issue the command:

   ```
   cd /etc/conf/gen
   ```

2. Copy the S800 file using a different name for a backup copy in case you need it. For example:

   ```
   cp S800 S800BACKUP
   ```

   Edit the S800 file to add the new graphics interface (**graph0** or **graph2**) to the **io** statement. Make sure the line "`include ite;`" is present if you want to use the graphics device as a terminal.

3. Run **uxgen** using the edited file as input:

   ```
   /etc/uxgen S800
   ```

4. Copy the old kernel (**/hp-ux**) in the root (/) directory and the old devices file (**/etc/devices**). Write down the names of these files in case the new kernel does not boot. For example:

   ```
   cp /hp-ux /SYSBCKUP
   cp /etc/devices /etc/DEVBCKUP
   ```

5. Change the working directory to S800 using the command:

   ```
   cd /etc/conf/S800
   ```

6. Move **hp-ux** to **/hp-ux** and **devices** to **/etc/devices**:

   ```
   mv hp-ux /hp-ux
   mv devices /etc/devices
   ```

7. Make the special files for the new graphics interface:

   ```
   cd /dev
   /etc/insf -d graph0 -l <lu>
   ```

   where `graphn` is **graph0** (for A1017A cards) or **graph2** (for A1047A cards).

8. Shut down the system and turn off the system power.

9. Install the graphics interface card.

10. Turn on the system power and reboot.

## Driver graph0

The driver module **graph0** supports the 98720, 98550, and 98730 (Models 825 and 835 only) graphics controllers, and the Human Interface Link (HIL) modules connected to them.

There is no single driver for **graph0**; instead, several drivers (such as **framebuf, hil, hilkbd,** and **ite**) control the graphics interface. When the module **graph0** is specified in the **uxgen io** statement, **hil, hilkbd,** and **framebuf** are automatically loaded into the kernel. Driver **framebuf** is used to control the graphics display itself. Driver **hil** controls various HIL devices, such as bar code readers, digitizers, and mouse devices. Driver **hilkbd** controls HIL keyboard devices. Driver **ite** is an Internal Terminal Emulator (ITE) that allows the graphics display to act as a terminal device. ITE is loaded into the kernel if the line "`include ite;`" is specified in the **/etc/conf/gen/S800** file.

### Addressing

Each graphics interface is connected directly to the Mid-bus. When building the kernel, the address of each graphics interface must be entered in the **uxgen io** statement. You can only configure **graph0** or **graph2** into the kernel but not both. Enter the address of the Mid-bus module number (in decimal). For example:

```
io{
            graph0   lu 0   address 12;
            graph0   lu 1   address 16;
            graph0   lu 2   address 20;
}
```

The preceding **io** statement configures the kernel for three graphics controllers at Mid-bus module_addresses 12, 16, and 20 (Mid-bus slots 3, 4, and 5). Each graphics controller is assigned a logical unit number (0 through 2).

## Special Files for graph0

As noted in the section "Driver graph0," several drivers (**framebuf, hil, hilkbd,** and **ite**) are part of the module **graph0**. You need to build special files for each of the drivers, either by running **mksf** for each driver or by running **insf** on the module **graph0**. Figures 5-10 through 5-13 show the formats for these drivers.

All the special files associated with **graph0** share the following parameter:

**logical unit number** Each graphics interface is assigned a logical unit number (LU).

In addition, special files associated with the **hil** driver have the following parameter:

**link address number** This is the HIL link address.

The default set of special files for a graphics interface can be created using the **insf** command for the **graph0** module. The following commands

```
cd /dev
insf -d graph0 -l <lu>
```

create the following special files in the **/dev** directory:

| | | |
|---|---|---|
| `crt<lu>` | `rw-rw-rw-` | for driver **framebuf** |
| `hilkbd<lu>` | `rw-rw-rw-` | for driver **hilkbd** |
| `hil_<lu>.<addr>` | | |
| `link addresses 1-7` | `rw-rw-rw-` | for driver **hil** |
| `ttyi<lu>` | `rw-rw-rw-` | for driver **ite** |

To make a single special file, you can use the **mksf** command. For **framebuf, hilkbd,** and **ite** special files, the commands are

```
cd /dev
mksf -d <driver>
```

where `<driver>` is either **framebuf, hilkbd,** or **ite**, respectively, followed by any of the options:

| | |
|---|---|
| **-l** *<lu>* | the logical unit number |
| *filename* | the name of the special file |

For **hil** special files, the commands are

```
cd /dev
mksf -d hil
```

followed by any of the options:

| | |
|---|---|
| **-l** *<lu>* | the logical unit number |
| **-a** *address* | the HIL link address |
| *filename* | the name of the special file |

| 8 15 | 16 23 | 24 31 |
|---|---|---|
| unused | logical unit | unused |
| 0x0 | 0x00-0xFF | 0x0 |

**Figure 5-10. Driver framebuf Minor Number Format**

| 8 15 | 16 23 | 24 27 | 28 31 |
|---|---|---|---|
| unused | logical unit | link address | unused |
| 0x0 | 0x00-0xFF | 0x01-0x07 | 0x0 |

**Figure 5-11. Driver hil Minor Number Format**

| 8 15 | 16 23 | 24 27 | 28 31 |
|---|---|---|---|
| unused | logical unit | link address must be 8 | unused |
| 0x0 | 0x00-0xFF | 0x08 | 0x0 |

**Figure 5-12. Driver hilkbd Minor Number Format**

| 8 15 | 16 23 | 24 31 |
|---|---|---|
| unused | logical unit | unused |
| 0x0 | 0x00-0xFF | 0x0 |

**Figure 5-13. Driver ite Minor Number Format**

## Driver graph2

The driver module **graph2** supports the 98730 connected to the A1047A graphics controller (Models 825 and 835 only), and the Human Interface Link (HIL) modules connected to them. There is no single driver for **graph2**; instead, several drivers (such as **framebuf, hil, hilkbd**, and **ite**) control the graphics interface. When you specify **graph2** in the **uxgen io** statement, **hil, hilkbd**, and **framebuf** are automatically loaded into the kernel. Driver **framebuf** controls the graphics display itself. Driver **hil** controls various HIL devices, such as bar code readers, digitizers, and mouse devices. Driver **hilkbd** controls HIL keyboard devices. Driver **ite** is an Internal Terminal Emulator (ITE) that allows the graphics display to act as a terminal device. ITE is loaded into the kernel if the line "`include ite;`" is in the **/etc/conf/gen/S800** file.

The **graph2** driver controls two types of interface cards: a full-size graphics card (A1047A) and half-sized display interface cards that connect to the graphics card. One full-sized card supports up to two display interface cards and thus connects to up to two graphics display terminals. (Each graphics display terminal requires a display interface card.)

### Addressing

When building the kernel, you must indicate only the address of the graphics card in the **uxgen io** statement. (The driver determines how many display interface cards are present.) The full-sized graphics card connects directly to the Mid-bus and its address is the one you need to specify in the **io** statement. You can only configure either **graph0** or **graph2** into the kernel, but not both. Specify the Mid-bus module number (in decimal). For example:

```
    io{
                .
                .
                graph2   lu 0   address 12;
                .
                .
    }
```

The preceding **io** statement configures the kernel for one graphics controller at Mid-bus module address 12 (Mid-bus slot 3). Without changing this statement, you can attach one or two display interface cards to the A1047A graphics card.

### Special Files for graph2

As noted previously, several drivers (**framebuf, hil, hilkbd,** and **ite**) are part of the module **graph2**. You need to build special files for each of the drivers, either by running **mksf** for each driver or by running **insf** on the module **graph2** for each display interface card you are installing. Figures 5-10 through 5-13 (shown previously) describe the minor number formats for these drivers.

All the special files associated with **graph2** share the following parameter:

**logical unit number** Each graphics interface is assigned a logical unit number (LU).

In addition, special files associated with the **hil** driver have the following parameter:

**link address number** This is the HIL link address.

You can create the default set of special files for a **graph2** display interface using the **insf** command for each interface you are installing. Specify `lu 0` as the logical unit number in the following command if you have one display interface. If you have two, create special files for the second by executing **insf** again using `lu 1`.

```
cd /dev
insf -d graph2 -l <lu>
```

The following special files are created in the **/dev** directory:

```
crt<lu>                     rw-rw-rw-         for driver framebuf
hilkbd<lu>                  rw-rw-rw-         for driver hilkbd
hil_<lu>.<addr>
  link addresses 1-7        rw-rw-rw-         for driver hil
ttyi<lu>                    rw-rw-rw-         for driver ite
```

Refer to the section "Special Files for graph0" for how to create special files for **graph2** using **mksf**. The procedure is the same.

## Adding a Tape Drive

This section describes the procedure for adding a tape drive to the HP 9000 Series 800 computer system. It assumes that the interface card was properly installed and that the tape drive is connected to the system and is operating properly. To add a tape drive, you should be familiar with **uxgen** and the system generation process. If you are not, read the "System Reconfiguration Process", section in Chapter 7.

5. 4. 6. 7980

1. Determine the CIO slot, the HP-IB address, tape density, and tape drive device filename. Refer to the *Installation and Configuration Guide* for your computer for information on a particular tape drive.

2. Ensure that the HP-IB address of the tape drive has been properly set.

3. Change the working directory to the **gen** directory:

   ```
   cd /etc/conf/gen
   ```

4. Make a copy of the S800 file under a different name as a backup. For example:

   ```
   cp S800 S800BACKUP
   ```

   Edit the S800 file to add the new tape drive to the **io** statement.

5. Run **uxgen** using the edited file as input:

   ```
   /etc/uxgen S800
   ```

6. Save the old kernel (**/hp-ux**) and old devices file (**/etc/devices**) in the root file system. Remember their names. You can reboot from the old kernel and restore the old devices file in case the new kernel does not boot.

   ```
   cp /hp-ux /SYSBCKUP
   cp /etc/devices /etc/DEVBCKUP
   ```

7. Enter: `cd /etc/conf/S800`

8. Move **hp-ux** to **/hp-ux** and **devices** to **/etc/devices**:

   ```
   mv hp-ux /hp-ux
   mv devices /etc/devices
   ```

9. Make the special files for the new tape drive. One way to do this is to execute

   ```
   cd /dev
   insf -d tape1 -l <lu>
   ```

10. Shut down and reboot.

## Driver tape1

Driver **tape1** controls the 79xx tape drives (including the 7974, 7978, 7979, 7980, and 7980xc tape drives). Details required to add a tape drive and to create a special file for driver **tape1** are explained in following sections.

### Addressing

Each tape drive connects to an HP-IB card (27110). Multiple tapes and other types of devices may be attached to the same card. However, each device on the same card must have a unique address between 0 and 7.

When building the kernel, the address of each tape drive must be included in the **uxgen io** statement. For example:

```
io {
   cio_ca0 address 8 {
      hpib0 address 0 {
        tape1 lu 0 address 0;
        tape1 lu 1 address 1;
        }
      hpib0 address 1 {
        tape1 lu 2 address 0;
        tape1 lu 3 address 1;
      }
   }
}
```

The preceding **io** statement configures the kernel for four tape drives. Two HP-IB cards connect to the same CIO channel adapter (module address 8, Mid-bus slot 2). Each HP-IB (CIO slot 0, 1) has two tape drives at HP-IB addresses 0 and 1. Each tape is assigned a logical unit number (0 – 3).

## Special Files for tape1

Figure 5–14 shows the driver **tape1** minor number format.

| 8 | 9 | 10 | 11 | 12 | 13 14 | 15 | 16 20 | 21 23 | 24 – 31 |
|---|---|---|---|---|---|---|---|---|---|
| Trans-parency | RTE Mode | IR inh. | UCB Mode | Rewind on Close | Density | Data Com-pression | Not Used | Logical Unit Number | Not Used |
| 0=Normal<br>1=Diagnostic | 0=Off<br>1=On | 0=NO<br>1=Yes | 0=Off<br>1=On | 0=Yes<br>1=No | 00=800<br>01=1600<br>10=6250 | 0=off<br>1=on | 0 | 0 - 7 | |

**Figure 5–14. Driver tape1 Minor Number Format**

Tape special files include the following parameters:

| | |
|---|---|
| **character vs block** | Character special files read from and write to the tape without going through the file system. This method is called 'raw'. This is the normal way for accessing tape drives. Block special files are used to read mounted file systems. Users generally do not access tape drives as block devices. |
| **logical unit number** | Each tape drive is assigned a logical unit number (LU). LU numbers start at 0 and increase by 1 for each tape drive. For example, only one tape drive can be logical unit 0. |
| **no–rewind flag** | If this flag is not set when the special file for a tape drive is closed, the tape drive will automatically be rewound. If set, the tape drive is not repositioned. |
| **style** | If the no rewind flag is set and the tape drive special file is opened 'read-only', there are two industry standard methods (called styles) of repositioning the tape. The first style, used on AT&T systems, positions the tape after the EOF mark which follows the data just read. The other style, used on Berkeley systems, leaves the tape alone (no repositioning). |
| **bits per inch (bpi)** | Some tape units support multiple densities. For example, the 7974 supports 800 and 1600 bpi, the 7979 supports 1600 bpi, and the 7978 or 7980 supports 1600 and 6250 bpi. For tape drives, such as the 7974, 7978, or 7980, which support multiple densities, the desired tape density is specified in the special file. |
| **compatibility** | RTE compatibility - no alteration or movement of tape during a close operation. |
| **transparent flag** | This flag is used by HP-UX diagnostics. |
| **wait** | To speed up I/O operations, the tape drive has a mode called 'immediate reporting'. When 'immediate reporting' is enabled, the tape drive indicates that a write operation is complete after all data is transferred to the data buffer, although the data is not completely written on tape. Disable this mode by setting the 'wait' flag. Note that setting this flag degrades system performance considerably. |
| **compression** | This flag enables tape drives that support data compression (such as 7080xc). If set to 1, compression is on. Use only for data compression devices. |

A default set of special files for a tape may be created using the **insf** command. Executing

```
insf -d tape1 -l <lu>
```

creates the following special files in the current directory:

```
mt/<lu>l
800 bpi, block, rw- rw- rw-

mt/<lu>m
1600 bpi, block, rw- rw- rw-

mt/<lu>h
6250 bpi, block, rw- rw- rw-

mt/<lu>ln
no rewind, 800 bpi, block, rw- rw- rw-

mt/<lu>mn
no rewind, 1600 bpi, block, rw- rw- rw-

mt/<lu>hn
no rewind, 6250 bpi, block, rw- rw- rw-

rmt/<lu>l
800 bpi, character, rw- rw- rw-

rmt/<lu>m
1600 bpi, character, rw- rw- rw-

rmt/<lu>h
6250 bpi, character, rw- rw- rw-

rmt/<lu>ln
no rewind, 800 bpi, character, rw- rw- rw-

rmt/<lu>mn
no rewind, 1600 bpi, character, rw- rw- rw-

rmt/<lu>hn
no rewind, 6250 bpi, character, rw- rw- rw-


rmt/<lu>hc
6250 bpi, -compressed, character, rw- rw- rw-

diag/mt/<lu>
```

To make a single special file, use the **mksf** command

```
cd /dev
mksf -d tape1
```

followed by any of the following options:

| | |
|---|---|
| **-l** *lu* | the logical unit number |
| **-r** | raw; use character entry, not block |
| **-b** *integer* | set density (800, 1600, 6250) |
| **-n** | set no-rewind flag |
| **-u** | set Berkeley style (**-u** and **-a** are mutually exclusive) |
| **-a** | set AT&T style |
| **-c** | set RTE compatibility flag |
| **-C** | enable data compression |
| **-t** | set transparent flag (for diagnostics only) |
| **-w** | set wait flag and disable immediate reporting |
| *filename* | the filename |

## Adding a LaserJet or Serial Printer

To add a serial printer to your system, follow these steps:

1. Connect the serial printer to an available port on one of the muxes (do not use the one that is reserved for use with the Access Port).
2. Check the file **/etc/inittab** to be sure that no getty is running on the port to which the printer is connected.
3. Create the appropriate device file for the LaserJet or other serial printer.

   Although you can name the device file whatever you want, the suggested device filename format for serial printers is:

   ```
   /dev/ttyXpY
   ```

   where: X = logical unit number, and
   Y = port number of the MUX card.

   Use **mksf** to create the device file. For example, to create a device file for a printer on mux0 port 2:

   ```
   /etc/mksf -d mux0 -l 0 -p 2 -h /dev/tty0p2
                        ↑    ↑       ↑
                    mux # port # device file
   ```

4. Check to see if the device file is correct. The following example verifies that the device file was created when you executed **mksf.**

   ```
   /etc/lssf /dev/tty0p2

   mux0 lu 2 port0 hardwired address 8.8 /dev/tty0p2
   ```

   where `8.8` is the hardware path.

5. For each printer enabled, make an entry in **/etc/inittab** (this file is discussed earlier in this chapter and in Chapter 3). The easiest way to add the new entry is to copy an existing entry line in the file and then change the device filename and ID. An example entry for a serial printer would be

   ```
   e0:2:off:/etc/getty -h tty0p2
   ```

   Note that "off" in the third field indicates that there is no getty running on the port.

6. Implement the change in **/etc/inittab** by forcing **init** to read the **/etc/inittab** file by entering:

   ```
   telinit q
   ```

7. This step is optional. To test whether the printer works *before* setting up spooling, you may connect the printer for direct printing by editing the **/etc/rc** file. Add the following information to the file:

   ```
   nohup sleep 2000000000 < /dev/tty0p2
   stty -parenb -ienqak cs8 9600 -cstopb -0local ixon opost onlcr tab3
     < /dev/tty0p2
   ```

You may then test whether or not the printer works by printing a file directly. For example:

```
cat /etc/passwd > /dev/tty0p2
```

8. To implement spooling, set up the spooler as explained in "Setting Up the LP Spooler" in Chapter 4.

## Adding an HP-IB Printer

The following is a procedure for installing an HP-IB printer. It is assumed that the interface card has been properly installed and the appropriate hardware configuration correctly completed. If necessary, refer to the *Installation and Configuration Guide* and to the peripheral device service manual for your computer for more information.

1. Ensure that the interface card is inserted in the appropriate CIO slot, that the switch settings are correct, and that the cable is of the proper length and connected.

2. Ensure that the printer is configured correctly. (The HP-IB address is usually 1 for the first CIPER printer and 5 for the first Amigo printer.)

3. Determine the appropriate device filename. The default printer device filenames are **/dev/lp0** for the first printer and **/dev/lp1** for the second printer. These are CIPER printers (256x). The default device filename for an Amigo printer is **/dev/lp2**.

4. Set up the spooler as explained in "Setting Up the LP Spooler" in Chapter 4.

Although all printers on HP-UX are character devices, they can be used in different "modes"; you can communicate with printers using drivers that either interpret or do not interpret the data. The printer characteristics are defined in the associated driver minor number. The minor number format for line printers is shown in a separate subsection below.

The following procedure describes how to use **uxgen** to add a line printer to the kernel configuration.

If the existing kernel configuration suits your needs, running **uxgen** is not necessary. Before adding a line printer, be sure that you are familiar with **uxgen** and the system generation process. Otherwise, read the "System Reconfiguration Process" section in Chapter 7.

1. Change current working directory to **gen** by entering:

   ```
   cd /etc/conf/gen
   ```

2. Back up the S800 file. For example,

   ```
   cp S800 S800BACKUP
   ```

   Edit the S800 file to add the new line printer to the **io** statement.

3. Run **uxgen** using the edited file as input.
4. Save the old kernel (**/hp-ux**) and old **devices** file (**/etc/devices**) in the root file system. Remember their names. In case the new kernel does not boot, you can boot from the old kernel.
5. Change the current working directory to **S800** by entering:

```
cd /etc/conf/S800
```

6. Move **hp-ux** to **/hp-ux** and **devices** to **/etc/devices**

```
mv hp-ux /hp-ux
mv devices /etc/devices
```

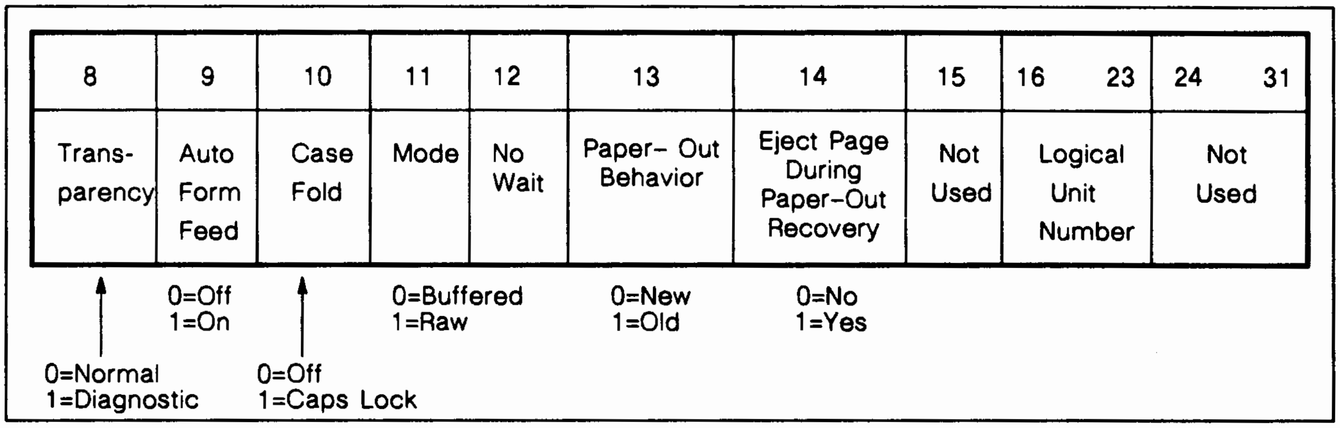
7. Make the special files for the new line printer. One way is to use the default special file as shown below.

```
cd /dev
insf -d <driver> -l <lu>
```

8. Shut down and reboot.

## Driver lpr0 or lpr1

The driver **lpr0** controls a number of line printers including model numbers 2563A, 2564B, 2566B, 2567B. The driver **lpr1** controls 2932A, 2934A (Amigo), and 2235 line printers. Figure 5-15 shows the driver **lpr0** or **lpr1** minor number format.



| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 23 | 24 31 |
|---|---|---|---|---|---|---|---|---|---|
| Trans-parency | Auto Form Feed | Case Fold | Mode | No Wait | Paper- Out Behavior | Eject Page During Paper-Out Recovery | Not Used | Logical Unit Number | Not Used |
| 0=Normal 1=Diagnostic | 0=Off 1=On | 0=Off 1=Caps Lock | 0=Buffered 1=Raw | | 0=New 1=Old | 0=No 1=Yes | | | |

**Figure 5-15. Driver lpr0/lpr1 Minor Number Format**

Line printer special files include the following parameters:

| | |
|---|---|
| **logical unit number** | Each line printer is assigned a logical unit number (lu). LUs are explained in the "Overview of the I/O System" section. |
| **raw flag** | When this flag is set, the line printer does not perform special processing of characters. |
| **no form-feed flag** | When this flag is set, the driver assumes that the printer cannot interpret a form–feed character. |
| **caps flag** | When this flag is set, all lowercase characters are shifted to uppercase. |
| **transparent flag** | This flag is used by HP-UX diagnostics. |
| **old behavior flag** | When this flag is set, the line printer aborts when it runs out of paper. |
| **eject page** | When this flag is set, the printer ejects a page when you add paper to it. |
| **no wait for error on open flag** | When this flag is set, the line printer forfeits the print job if it detects an error at open time. |

## Addressing

Each line printer must be connected to an HP-IB card (27110B). Multiple line printers and other device types (for example, tape drives) may exist on the same card; however, each device must have a unique HP-IB address, and that bus address must be between 0 and 7. Mixing high-speed devices (such as disks) with low-speed devices (such as printers) is not recommended.

When building the kernel, the address of each line printer must be entered in the **uxgen io** statement. A sample **io** statement is shown below.

```
io {
   cio_ca0 address 8 {
      hpib0 address 0 {
         lpr0 lu 0 address 3;
         lpr0 lu 1 address 4;
         lpr1 lu 4 address 5;
      }
      hpib0 address 1 {
         lpr0 lu 2 address 3;
         lpr0 lu 3 address 4;
         lpr1 lu 6 address 5;
      }
   }
}
```

The preceding **io** statement configures the kernel for six line printers. Two HP-IB cards are connected to the same CIO channel adapter (module address 8 - Mid-bus slot 2). Each HP-IB card (CIO slot 0, 1) has three line printers connected at HP-IB addresses 3, 4, and 5. Each line printer is assigned a unique logical unit number (0, 1, 2, 3, 4, and 6).

You can create a default special file for a line printer using the **insf** command. Enter:

```
insf -d lpr0 -l <lu>
```

or

```
insf -d lpr1 -l <lu>
```

to create the following special file in the current directory.

```
lp<lu> owner lp, group bin, rw- --- ---
```

The owner and group are set to **lp** and **bin** so that the line printer spooler commands (cancel, disable, enable, lp, lpstat) have access to the line printer.

To make a special file with a different filename or options, the command **mksf** may be used. The syntax for **mksf** is

```
mksf -d <lpr>
```

where `<lpr>` is either lpr0 or lpr1 followed by any of the options listed below:

| | |
|---|---|
| **-l** *lu* | the logical unit number |
| **-t** | set the transparent flag (for diagnostics only) |
| **-c** | set the caps flag |
| **-n** | set the no form-feed flag |
| **-r** | set the raw flag |
| **–o** | set abort on paper–out |
| **–e** | set eject page after paper–out |
| *filename* | the filename |

If you use **mksf**, be sure to change the owner and group of the special file to **lp** and **bin**, respectively, to allow the spooler to access the printer.

You can alter printing characteristics such as indent, lines per page, backspace handling, and number of pages to eject on open or close. Generally, you change these using a filtering program. You can also use **/usr/bin/slp**. Refer to the **slp(1)** man page and note constraints and warnings before using this command.

## Adding Disks and Cartridge Tape Drives

The section describes the procedure for adding a new disk or cartridge tape. This procedure is only needed if the default HP-UX device configuration is not suitable. Before you attempt this procedure, be sure that you are familiar with **uxgen** and the system generation process. If you are not, read the "System Reconfiguration Process" section in Chapter 7.

1. Change the working directory to directory **gen**:

   ```
   cd /etc/conf/gen
   ```

2. Make a copy of the S800 file under a different name as a backup. For example:

   ```
   cp S800 S800BACKUP
   ```

Edit the S800 file to add the new disk drive to the **io** statement.

3. Run **uxgen** using the edited file as input:

```
/etc/uxgen S800
```

4. Save the current kernel (**/hp-ux**) and devices file (**/etc/devices**) in the root file system. Remember their names. If the new kernel does not boot, you can boot from the copy.

```
mv /hp-ux /hp-ux/SYSBCKUP
mv /etc/devices /etc/DEVBCKUP
```

5. Change the working directory to directory S800:

```
cd /etc/conf/S800
```

6. Move **hp-ux** to **/hp-ux** and **devices** to **/etc/devices**

```
mv hp-ux /hp-ux
mv devices /etc/devices
```

7. Make the special files for the new disk. For example:

```
cd /dev
insf -d <disc> -l <lu>
```

where *<disc>* is **disc0** for HP-IB discs and **disc2** for HP-FL disks.

8. Shut down the system and reboot.

## Driver disc0

The **disc0** driver controls a broad range of CS/80 and SS/80 disks and cartridge tape for HP-IB units. Details required to create a special file for **disc0** are explained in this section. Figure 5-16 shows the driver **disc0** minor number format.

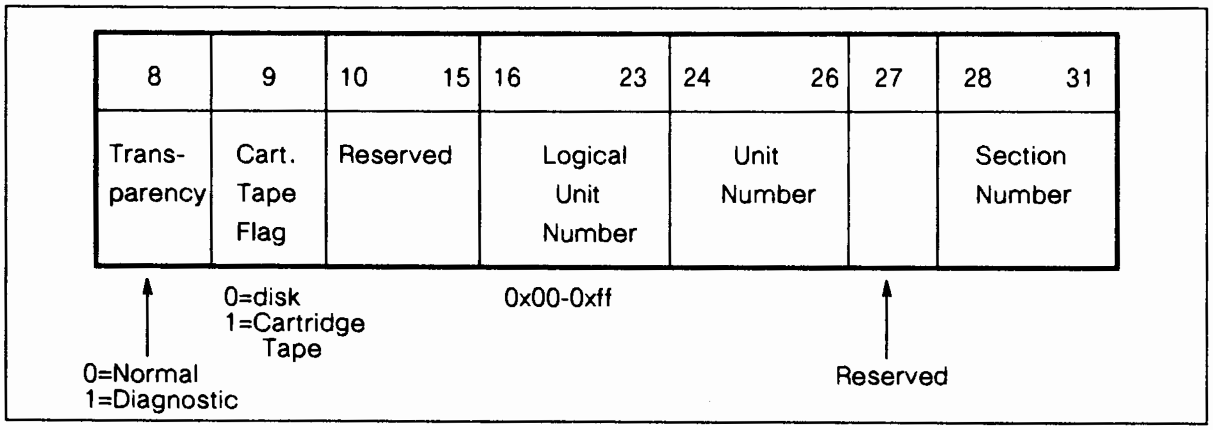


**Figure 5-16. Driver disc0 Minor Number Format**

### Addressing

Each disk must be connected to an HP-IB card (27110B). Multiple disks may exist on the same card; however, no other type of device (e.g., tape drive) may be connected to the same card. When building the kernel, the address of each disk drive must be entered in the **uxgen io** statement. Enter the HP-IB address of the disk drive. You are limited to four disks per HP-IB card.

```
      io {
        cio_ca0 address 8 {
           hpib0 address 0 {
              disc0 lu 0 address 0;
              disc0 lu 1 address 1;
           }
           hpib0 address 1 {
              disc0 lu 2 address 0;
              disc0 lu 3 address 1;
           }
        }
     }
```

The preceding **io** statement would configure the kernel for four disk drives. Two HP-IB cards are connected to the same CIO channel adapter (module address 8 - Mid-bus slot 2). Each hpib (CIO slot 0, 1) has two disk drives connected at HP-IB addresses 0 and 1. Each disk is assigned a logical unit number (0 - 3).

### HP-IB Disk Sections

Each disk is divided into parts called 'sections'. The appropriate section number is one of the parameters specified in creating a disk special file. Refer to the section on "Creating a New File System" in Chapter 7 for more information about disk sections.

### Special Files for Driver disc0

Disk special files include the following parameters:

**character vs block** Character special files read from or write to the disk without going through the system buffer cache. This method is called 'raw'. Some reasons for doing this are

1. Information may be accessed that is not normally available through the file system (super block, inode table, etc.) for programs such as **fsck**.
2. Large blocks of data (1 MB or more) may be read in one system call which is faster in certain cases than using the file system.
3. File systems may be diagnosed and repaired using **fsck**.

All special files for disks should always be write protected from general users to prevent them from corrupting data unknowingly.

Block special files are used to read/write mounted file systems.

|  |  |
|---|---|
| **logical unit number** | Each disk or cartridge tape is assigned a logical unit number (LU). |
| **unit number** | Each disk or cartridge tape is assigned a unit number at the factory. The unit number is not the HP-IB address. Some disks have a built-in cartridge tape (e.g., 7914). Generally, the unit number will be 0, unless multiple devices share the same HP-IB address (controller). |
| **section number** | Each disk or cartridge tape is divided into numbered parts or *sections* by the software. Refer to the "Creating a New File System" section in Chapter 7 for more information about disk sections. |
| **cartridge tape flag** | Cartridge tapes and disks are accessed identically by the driver. To prevent accidental writing onto a disk rather than a cartridge tape, this flag in the special file must be set to indicate that the special file is used for cartridge tapes only. |
| **transparent flag** | This flag is used by diagnostics and the **mediainit** utility. |

Create a default set of special files for a disk using the **insf** command. Executing

```
insf -d disc0 -l <lu>
```

creates the following special files in the current directory:

```
dsk/c<lu>d0s<section>
            sections 0 to 11, group sys, block entry, rw- r-- ---

rdsk/c<lu>d0s<section>
            sections 0 to 11, group sys, character entry, rw- r-- ---

ct/c<lu>d<unit>s2
            units 0 and 1, block entry, rw- rw- rw-

rct/c<lu>d<unit>s2
            units 0 and 1, character entry, rw- rw- rw-

diag/dsk/c<lu>d<unit>
            units 0 and 1, transparent, character entry, rw- --- ---
```

To make a single special file, you can use the **mksf** command. The command syntax is **mksf -d disc0** followed by any of these options:

|  |  |
|---|---|
| -l *lu* | Logical unit number |
| -t | Set transparent flag (for diagnostics only) |
| -c | Set cartridge tape flag |
| -u *unit* | Unit number (e.g., unit 0 – disk, unit 1 – tape) |
| -s *section* | Section number |
| *filename* | File name |
| -r | Use character entry, not block (raw) |

## Driver disc2

The driver "disc2" controls HP-FL disks. Steps required to create a special file for **disc2** are presented in this section. Figure 5-17 shows the driver **disc2** minor number format.

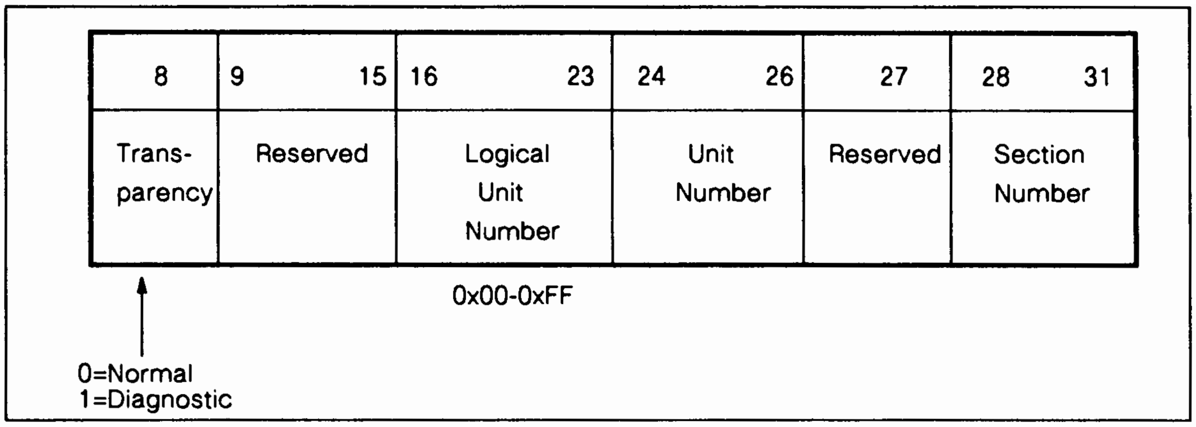


**Figure 5-17. Driver disc2 Minor Number Format**

## Addressing

Each HP-FL disk must be connected to an HP-FL or 27111A card (Refer to the *HP 27111A HP-FL Interface Card Installation and Service Manual*, 27111-90001 for more information.) Multiple disks may connect to the same card; however, no other type of device (such as a tape drive) may be connected to the same card. When building the kernel, the address of each disk drive must be entered in the **uxgen io** statement. Enter the HP-FL address of the disk drive. You are limited to eight disks per HP-FL card.

```
io {
   cio_ca0 address 8 {
      hpfl0 address 0 {
         disc2 lu 0 address 0;
         disc2 lu 1 address 1;
      }
      hpfl0 address 1 {
         disc2 lu 2 address 0;
         disc2 lu 3 address 1;
      }
   }
}
```

The preceding **io** statement configures the kernel for four disk drives. Two HP-FL cards are connected to the same CIO channel adapter (module address 8 - Mid-bus slot 2). Each HP-FL card (CIO slot 0, 1) has two disk drives connected at HP-FL addresses 0 and 1. Each disk is assigned a logical unit number (0 - 3).

### HP-FL Disk Sections

Each disk is divided into parts called ‘sections’. The appropriate section number is one of the parameters specified in creating a disk special file. Refer to the section on “Creating a New File System” in Chapter 7 for more information about disk sections.

### Special Files for Driver disc2

Disk special files include the following parameters:

**character vs block** Character special files read from or write to the disk without going through the system buffer cache. This method is called ‘raw’. Some reasons for doing this are:

1. Programs, such as **fsck**, can access information that is not normally available through the file system (superblock, inode table, etc.).
2. Large blocks of data (1 MB or more) may be read using one system call which is faster in certain cases than using the file system.
3. File systems may be diagnosed and repaired using **fsck**.

Character special files should always be write protected from general users to prevent them from corrupting data unknowingly.

Block special files are used to read/write mounted file systems.

| | |
|---|---|
| **logical unit number** | Each disk is assigned a logical unit number (LU). |
| **unit number** | Each disk is assigned a unit number at the factory. The unit number is not the HP-FL address. Generally, the unit number is 0, unless multiple devices share the same HP-FL address (controller). |
| **section number** | Each disk is divided into parts by the software (this is not a physical partitioning). Refer to the "Creating a New File System" section in Chapter 7 for more information about disk sections. |
| **transparent flag** | This flag is used by diagnostics. |

You can create a default set of special files for a disk using the **insf** command. Executing

```
insf -d disc2 -l <lu>
```

creates the following special files in the current directory:

```
dsk/c<2000+lu>d0s<section>
          sections 0 to 15, group sys, block entry, rw- r-- ---

rdsk/c<2000+lu>d0s<section>
          sections 0 to 15, group sys, character entry, rw- r-- ---

diag/dsk/c<2000+lu>d<unit>
          units 0 and 1, transparent, character entry, rw- --- ---
```

To make a single special file, you can use the **mksf** command. The command syntax is **mksf -d disc2** followed by any of these options:

| | |
|---|---|
| **-l** *lu* | Logical unit number |
| **-t** | Set transparent flag (for diagnostics only) |
| **-u** *unit* | Unit number (for example, unit 0 - disk, unit 1 - tape) |
| **-s** *section* | Section number |
| *filename* | Filename |
| **-r** | Use character entry, not block (raw) |

## Converting Disks from HP-IB to HP-FL Interfaces

Some HP disk drives connect to the system through an HP-IB interface card. Others connect through HP-FL (fiber-optic link) interface cards. The **disc2** driver controls HP-FL disks as explained previously in the section "Driver disc2." HP-FL disks provide higher throughput and greater flexibility when a lot of disk access is required, for example, on a system disk. Currently, only the HP 7936 and the HP 7937 are HP-FL disks.

This section explains how to upgrade HP-IB disks to HP-FL disks. You should be familiar with **uxgen** and system reconfiguration before you attempt this procedure. If you are not, see Chapter 7.

Follow these steps to convert HP-IB to HP-FL disks:

1. Change to the **gen** directory:

   ```
   cd /etc/conf/gen
   ```

2. Copy the current **S800** file to create a backup:

   ```
   cp S800 S800BCKUP
   ```

3. Edit the **S800** file to specify the HP-FL driver as explained in the section "Driver disc2." For example, if the HP-FL disk is the system disk, change the part of the file that specifies kernel device locations, as follows:

   ```
   args     on   disc2   lu   0   section 1;
   console  on   mux0    /*defaults to lu 0*/
   dumps    on   disc2   lu   0   section 1;
   root     on   disc2   lu   0   section 0;
   swap     on   disc2   lu   0   section 1
                 disc2   lu   1   section 1;
                   .
                   .
   ```

   (The **disc0** driver was changed to **disc2**.)

   Then change the **io** statement. HP-FL interface cards can hold up to eight disk drives and need to be accessed by the **disc2** driver. The parts of the **io** statement that change, for example, follow:

```
hpfl0  address      0 {
        disc2       lu 0 address 0;
        disc2       lu 1 address 1;
        disc2       lu 2 address 2;
        disc2       lu 3 address 3;
        disc2       lu 4 address 4;
        disc2       lu 5 address 5;
        disc2       lu 6 address 6;
        disc2       lu 7 address 7;
                    .
                    .
```

*The hpib0 driver was changed to hpfl0.*

*The disc0 was changed to disc2 and more entries were added.*

   Renumber the HP-IB disk LU numbers, if you have HP-IB disks also connected to the system.

```
hpib0   address     6 {
         disc0      lu 0 address 0;
```

*The logical unit numbers have changed*

```
          disc0       lu 1 address 1;          for hpib0.
          disc0       lu 2 address 2;
          disc0       lu 3 address 3;
```

4. Run **uxgen** using the edited S800 file as input:

   ```
   /etc/uxgen S800
   ```

5. Save the current kernel (**/hp-ux**) and devices file (**/etc/devices**) in the root directory using different names. (If the system does not boot with the new kernel, you can use these files to boot the old kernel.)

   ```
   mv /hp-ux /hp-ux/SYSBCKUP
   mv /etc/devices /etc/DEVBCKUP
   ```

6. Change directories to **/etc/conf/S800:**

   ```
   cd /etc/conf/S800
   ```

7. Move **hp-ux** to **/hp-ux** and **devices** to **/etc/devices**:

   ```
   mv hp-ux /hp-ux
   mv devices /etc/devices
   ```

8. Make the special files for HP-FL disk in **/dev**:

   ```
   cd /dev
   insf -d disc2
   ```

9. Copy the current **/etc/checklist** into a backup file:

   ```
   cp /etc/checklist /etc/cklstbck
   ```

10. Edit **/etc/checklist** for the new device configuration. For example, **/etc/checklist** entries for a 7937 disk might look as follows:

    ```
     /dev/dsk/c2000d0s0     /        defaults   0  1  # root
     /dev/dsk/c2000d0s3     /tmp     defaults   0  2  # /tmp
     /dev/dsk/c2000d0s4     /usr     defaults   0  3  # /usr
     /dev/dsk/c2000d0s5     /extra   defaults   0  4  # extra space
     /dev/dsk/c2000d0s10    /mnt     defaults   0  5  # /mnt
    ```

    Ensure that the device files for disc2 are on the system.

11. Shut down the system with the **shutdown** command, and power it off.

12. Change the hardware on the disks and HP-FL interface. This requires changing the disk power supply, replacing the HP-IB interface card, adding the HP-FL card to the CIO card cage, and installing the fiber-optic cable. (Normally, your Customer Engineer makes these hardware changes.)

13. Power up and reboot the system.

    To change back to HP-IB disks, you only need to change the controller board and not the power supply. Both HP-IB and HP-FL controllers operate with the new power supply.

## General-Purpose Instruments

This section describes how to install general-purpose instruments.

### Driver instr0

The driver **instr0** controls general-purpose HP-IB instruments and supports the Device I/O Library (DIL). The supported plotters and digitizers are examples of the HP-IB devices that use driver **instr0**. Figure 5-18 shows the driver **instr0** minor number format.

| 8 | 9 15 | 16 23 | 24 26 | 27 31 |
|---|---|---|---|---|
| Trans-parency | Reserved | Logical Unit Number | Reserved | Instrument Address |
| 0=Normal<br>1=Diagnostic | | 0x00-0xFF | | |

**Figure 5-18. Driver instr0 Minor Number Format**

#### Addressing

Each HP-IB instrument must be connected to an HP-IB card (27110B). Before building the kernel with **uxgen**, you must include an **instr0** entry in the **io** statement for each HP-IB card that is connected to instruments. Many instruments can be connected to one HP-IB card; however, you only need one instance of **instr0** for each HP-IB card in the **io** statement. Any address between 0 and 7 can be used for **instr0**, because the address is not used by **instr0**. The actual address for each HP-IB instrument is contained in the special file (one per instrument). For example:

```
io {
   cio_ca0 address 8 {
      hpib0 address 0 {
         instr0 lu 0 address 0;
      }
      hpib0 address 1 {
         instr0 lu 1 address 0;
      }
   }
}
```

The preceding **io** statement would configure the kernel for two HP-IB cards that may contain any number of HP-IB instruments. Two HP-IB cards are connected to the same CIO channel adapter (module address 8 - Mid-bus slot 2). Each HP-IB card (CIO slot 0, 1) has one or more HP-IB instruments which are controlled by the **instr0** driver. Each **instr0** driver has a logical unit number assigned which is used in special files to identify the address of the instrument bus.

### Special Files for Driver instr0

Driver **instr0** special files include the following parameters:

| | |
|---|---|
| **logical unit number** | Each driver **instr0** is assigned a logical unit number, identifying the HP-IB device adapter. |
| **address** | The HP-IB device adapter address (0-31) is entered in each special file. Address 31 means 'raw' – no auto-addressing. |
| **transparent flag** | This flag is used by HP diagnostics. |

You can create a default set of special files for HP-IB instruments on a single HP-IB using the **insf** command. Enter

```
insf -d instr0 -l <lu>
```

to create the following special files in the current directory:

```
hpib/<lu>a<address>
           address 0 to 30, rw- rw- rw-
hpib/<lu>
           no address, rw- rw- rw-
diag/hpib/<lu>
           no address, transparent, rw- --- ---
```

To make a single special file, you can use the **mksf** command. The syntax for **mksf** is

```
mksf -d instr0
```

and any of the following options:

| | |
|---|---|
| **-l** *lu* | the logical unit number |
| **-a** *address* | the instrument address |
| **-t** | set transparent flag (for diagnostics only) |
| **-r** | set raw flag (no associated HP–IB address) |
| *filename* | the filename |

## Driver gpio0

The driver **gpio0** controls the general–purpose I/O device adapter (27114A), called the AFI card. Figure 5–19 shows the driver **gpio0** minor number format.

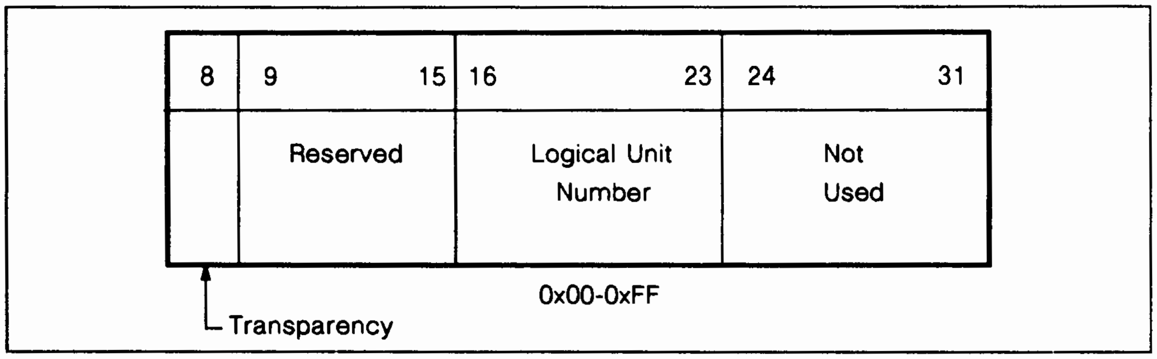


**Figure 5–19. Driver gpio0 Minor Number Format**

### Addressing

Each **gpio** card must be connected to a CIO channel adapter. When building the kernel, the address of each **gpio** card must be entered into the **uxgen io** statement. The address entered is the CIO slot number of each **gpio** card (in decimal). For example:

```
io {
   cio_ca0 address 8 {
      gpio0 lu 0 address 0;
      gpio0 lu 1 address 1;
   }
}
```

The preceding **io** statement would configure the kernel for two **gpio** cards at CIO slots 0 and 1. Each **gpio** card is connected to the same CIO channel adapter (module address 8 - Mid-bus slot 2). Each **gpio** card is assigned a logical unit number (0 and 1). The logical unit number is used in special files to identify the **gpio** card.

The **gpio** card must be in one of the first 7 slots of the CIO backplane. Otherwise, the card will not respond.

### Special Files for Driver gpio0

Driver **gpio0** special files consist of only the logical unit number parameter.

You can create a default special file for a **gpio** card using the **insf** command. Enter

```
insf -d gpio0 -l <lu>
```

to create the following special file in the current directory.

```
gpio<lu> rw- rw- rw-
diag/gpio<lu> rw- --- ---
```

To make a special file with a different filename, use the **mksf** command. The syntax for **mksf** is as follows.

```
mksf -d gpio0 -l  <lu> <filename>
```

## Removing Peripheral Devices

When a peripheral device is removed from the HP-UX system, there is no need to do anything to the software. If the device is a terminal, you may want to edit **inittab** to remove the entry for that terminal. If the device is a line printer, you may want to remove the appropriate entry from **/etc/rc** and/or **/etc/mklp**.

## Access Port

The primary purpose of the Access Port (AP) is to allow for remote diagnostics on Series 800 computers. With the AP and a modem, the Hewlett-Packard Response Center can dial into the AP and obtain status information. The AP also allows for resetting the computer or initiating a transfer of control. (You can also transfer control on Model 825 computers by turning the key switch.)

---

**NOTE**

**The AP card is standard equipment on Model 840 and 850 computers but is optional on Model 825 and 835 computers.**

---

As System Administrator, you need to be aware of the location of the AP card. You may need to be familiar with AP commands and operating modes. This information is presented briefly here with an example of the console display.

### Location

The AP card is in slot 12 of the CIO backplane on a Model 840 computer and is required hardware. If an AP is installed on the Model 825, it must be in slot 0 of the CIO backplane. On the Model 850, the AP card is in slot 4 of the CIO backplane D.

### Access Port Operating Modes

The Access Port can operate in three modes:

- **Independent system port mode** - In this mode, two independent terminals can be connected to the two independent session ports on the AP card. One terminal is the local console and the other is the modem connection.
- **Control mode** - In this mode, the local console communicates with the AP. Press the yellow button on the front panel of a Model 840 computer or turn the key switch on a Model 825 and 850 computer and press CTRL-B on the system console to activate Control mode.
- **Parallel console mode** - The remote console is in parallel with the local console. This allows the person at the remote console to boot the system. To set up the remote console, use the ER command (explained in the section "Access Port Commands"), enter the password, and access questions. The AP adjusts the baud rates accordingly.

### Access Port Commands

The following commands can be used if the AP configuration is lost, to change modes, or when you need remote support through the AP port. You type these commands at the system console when the AP is in control mode from the **CM>** prompt.

| | |
|---|---|
| CA | allows the data configuration of the AP. It lists the current configuration and requests changes. |
| CO | switches the session to console mode and is only valid from the local console. |
| DI | disconnects the modem line cleanly from the modem port. |
| DR | disables remote modem port by disallowing CTRL-B from the remote port and turns off the LED on the front panel. |
| DS | disables the hex display on the front panel from being displayed at the bottom of the console display during control mode. |
| ER | enables remote modem port by allowing CTRL-B from the remote port and lights the LED on the front panel. |
| ES | allows the hex display on the front panel to be displayed at the bottom of the console display during console mode. |
| HE | HELP command; provides a list of valid AP commands. |
| RS | reset; forces a power ON or hard reset which results in total loss of memory and system reboot. Displays a warning if the system is running. |
| TA | test AP; invokes the AP selftest. You can specify a parameter to initiate a nondestructive (use parameter 0, the default) or destructive selftest (use parameter 1). If you use 1, a warning is issued and the AP state is lost. |
| TC | transfer of control; performs a soft reset of the system. It transfers control of the operating system to a routine that flushes memory and dumps the system onto disk for analysis. |
| TE | tell; allows messages to be passed between the local and remote consoles. It allows messages to be sent without causing the AP command interpreter to generate errors. |
| SE | connects to session mode from AP command mode. Only valid from the remote modem port. |

## Changing and Creating System Run Levels

One of the tools you need as the System Administrator is the ability to move properly from one system run level to another. A run level is a system state (or configuration) where only a selected group of processes are allowed. You may find it useful to create new system run levels for particular tasks or applications specific to your installation. The material in this section covers some protection issues associated with these capabilities followed by guidelines for changing the run level of the system and creating new system run levels.

As discussed in the "System Boot" section in Chapter 3, the System Administrator (or anyone with the **root** user capabilities) may change the system's run level by executing the **init** command. Also, anyone having write permission to the file **/etc/inittab** can create new run levels or redefine existing run levels. Even if a user lacks the capability to invoke **init**, having the ability to redefine existing run levels in **/etc/inittab** could wreak havoc upon your system. Make sure that the permissions for this file are correct (that is, 0644).

## Changing the System Run Level

This section explains how to change the system run level.

### Entering Run-Level S

Many of the system maintenance tasks you perform as System Administrator require the system to be running in single-user mode. In single-user mode (run-level s), only the **root** user can access the system at the system console; the only processes that can run on the system are the shell and any processes the **root** user invokes.

When taking the system from any numbered run level (run levels 0 thru 6) to the single-user run level (run-level s), use the **shutdown** command instead of **init** s to change the system run level. The **shutdown** command kills all non-essential daemon processes and brings the system safely to run-level s.

To enter run-level s, type in:

```
shutdown graceperiod
```

This warns all users that they have a number of seconds, specified by *graceperiod*, to log off, issues the sync system call to flush buffers to disk, kills all processes, and safely brings the system to run-level s.

### Changing Run Levels

The following is a general procedure for changing the system from one run level to another. You must be logged in as the superuser to change the system run level.

1. If any users are on the system, warn them before you change run levels. Changing to another run level while users are logged on will kill (terminate) their processes if **/etc/inittab** does not list the run level you are moving to in the rstate field for their **getty**. Use the **write** or **wall** commands to communicate with the users. Note that the **wall** (write all) command immediately sends your message to the terminal of each user on the system and, in the process, interrupts whatever they are doing (but does not stop execution). Avoid using **wall** unless you feel it is necessary.

   If each **getty** entry has the new run level in its **rstate** field, or if the **rstate** field is empty (implies all run levels), you don't need to ask them to log off; their processes will not be killed.

2. After users are off the system, force the system to write the contents of its I/O buffers to the file system by typing:

   ```
   sync
   ```

3. Next, change to the desired run level by typing:

   ```
   /etc/init new_run-level
   ```

   where *new_run-level* is the run level you wish to enter. If any outstanding user processes should not be present in the new run level, you must have the appropriate **/etc/inittab** entries to kill them.

The **cron** process, and other processes initiated by **/etc/rc**, are only initiated the first time the system enters run-level 2.

## Creating New System Run Levels

Before creating a new run level, take the following precautions:

- Copy the original **/etc/inittab** file and save using a different name (such as **/etc/orig_inittab**). If anything goes wrong, you will still have a working version of the file.

**NOTE**

**Make sure to copy** /etc/inittab **before changing it. The** initdefault **line must remain as shipped or you may not be able to boot your system.**

- You can override the autoboot sequence and come up in single-user mode (**hpux -is**) to test your changes.

To create new run levels, use one of the HP-UX text editors to make entries in **/etc/inittab**. These entries define how the system operates in its new run level. Each one line entry in **/etc/inittab** should contain:

- A one or two-character ID used to identify a process or process group
- A list of run levels to which each entry applies
- An action to be performed, such as **respawn**
- The command that will be executed when that run level is entered

Refer to **init(1M)** and **inittab(4)** in the *HP-UX Reference* for a more complete description of **inittab**'s run-level entries. Once **/etc/inittab** contains all of the entries for the new run level, save the file and exit the text editor. As before, warn all users to log off the system and follow the other procedures in the "Changing the System Run Level" section above **before changing run levels**.

In a few cases, such as when a new run level is quite similar to an existing run level, you can move freely between run levels provided that entering the new run level does not kill (user or system) processes that may have begun in the previous run level. If the new run level is not specified in the **rstate** field of the **/etc/inittab** entry for their

**getty**, the process will be killed. Watch for side effects, however. Consider the case where a user logs off, you then change run levels (for example, from run-level 2 to run-level 4) and when the user attempts to log back in, he/she cannot because an **/etc/getty** entry does not exist in the run-level 4 definition.

Whenever the system enters the newly defined or created run level, its actions are similar to those described in the System Boot section (of the "System Boot" and "Login" sections in Chapter 3) except that the commands executed are those identified by the new run level number.

## Example /etc/inittab

The following is an example **/etc/inittab** for a system that has a system console and five terminals. Run-level s is single-user run level. Run-level 2 is a multiuser run level, with a **getty** on every terminal. Run-level 3 is a test run level, with a **getty** on both the system console and the System Administrator's terminal (**/dev/tty0p1**) and "kill" entries for the other terminals. A System Administrator who preferred to work at his/her own terminal (rather than the system console) could use this run level.

```
is:2 :initdefault:
bl:  :bootwait:/etc/bcheckrc </dev/syscon >/dev/syscon 2>2&1
bc:  :bootwait:/etc/brc 1>/dev/syscon 2>&1
sl:  :wait:(rm -f /dev/syscon; ln /dev/systty /dev/syscon;) 1>/dev/console
rc:  :wait:/etc/rc <dev/syscon >/dev/syscon 2>&1
n1:  :off:
n2:  :off:  These are comment lines
n3:  :off:
co:  :respawn:/etc/getty console  console
01:23:respawn:/etc/getty tty0p1   9600
02:2 :respawn:/etc/getty tty0p2   9600
03:2 :respawn:/etc/getty tty0p3   9600
04:2 :respawn:/etc/getty tty0p4   3
05:2 :respawn:/etc/getty tty0p5   400
```

## Backing Up the File System

System backup is an important administrative task on any multiuser HP-UX system. Making backup copies of files allows you to recover lost data if there is a hardware failure, a system **crash**, or if you accidentally remove or corrupt a file. Backups can be made on cartridge tape or magnetic tape.

System backup takes a considerable amount of time and relies heavily on system resources. Users may perceive a slower response time if you try to back up a system during the day. Ideally, backups should be done during off-hours.

You can set the system up to perform backups automatically. For example, you can use the **crontab** command to schedule backups during non-prime hours. This way, **cron**, the HP-UX clock daemon periodically invokes backup programs customized for your environment. Refer to the *HP-UX Reference* manual pages for more information on using these commands.

To minimize the chance of data loss, backup media should be stored in a different location from the main file system. "Data safes," specially designed air-tight, water and fire-proof containers for mass storage media, are available from many computer accessory manufacturers. If a file or the entire file system is lost or destroyed, you can recover by restoring the latest version of your system backup.

---

**NOTE**

**You may also use SAM to back up the system with menus instead of individual commands. Refer to** ***System Administration Basics*** **for more information.**

---

### Backup Strategies and Trade-offs

The method, frequency, and extent of the backup operation depends on how much you use your system and how much data you feel you can afford to lose. Complete backups (as compared with partial ones) take significant time and the media costs involved can be high.

#### Daily Backups

One backup strategy is to make complete backups of the file system on a daily basis. Restoring the file system from a full backup consists of restoring the most recent backup 9-track tape or cartridge tape. While relatively expensive in terms of media, system resources and the time required to make full daily backups, the time and effort spent recovering the system is minimized.

#### Incremental Backups

An **incremental backup** contains only files that have changed since the last full or incremental backup. Incremental backups almost always require less time and less backup media than full backups.

### Mixing Full and Incremental Backups

The ability to use both full and incremental backups leads to a useful backup strategy that is cost-effective and time-efficient without sacrificing the integrity of the file system. This method involves making full backups of the file system once a week and supplementing with daily incremental backups.

For example, if you make a full backup of the file system on Monday and incremental backups on other weekdays. Each incremental backup contains only those files that have changed since a previous full or incremental backup. Further assume that, on Thursday, the file system is destroyed. The file system may be reconstructed by first restoring Monday's full backup of the file system and then restoring the files from all incremental backups. Note that the file system is now restored to the end of Wednesday's incremental backupand only work done on Thursday was lost.

You can write a shell script to do incremental backups, taking advantage of the time-stamp feature of the file system. (Make sure that the system clock is set accurately!) The backup script may do either a full or an incremental backup. You can customize these scripts to suit your system needs.

The following backup schedule is recommended:

- Full backup monthly or bimonthly
- Weekly incremental backup
- Daily incremental backup
- Take a full backup and store it in a permanent archive library.

Incremental backups must be made regularly. You should save incremental backups until one or two weeks have passed since the last full backup.

## Backup and Restore Methods

You must back up the file system onto tape. You can either:

- Perform a regular system backup using a script of your design.
- Backup selected files using the dump and restore commands to copy selected files to a backup media.

HP recommends using the **fbackup** and **frecover** commands for file system backup. The **fbackup** command backs up file systems onto the backup media. The **frecover** command recovers all or selected files from the backup media. Refer to the **fbackup(1M)** and **frecover(1M)** in the *HP-UX Reference* for complete information on using these commands.

Note that for limited or small-system backup, you may also use the **tar(1)** and **cpio(1)** commands. Refer to the *HP-UX Reference.*

## Using the fbackup Command

The **fbackup** command is a versatile and powerful backup utility. It provides for both full and incremental backups. You may assign various level numbers (from 0 to 9) to different types of backups. This allows you to implement as complex a backup strategy as required for your needs. For example, you can assign a full backup - level 0, a weekly backup - level 3, and a daily backup - level 8. **fbackup** attempts to back up live file systems (that is, those with active files) up to five times (by default) or as many times as specified using the *-c* option with the **fbackup** command. To specify a configuration file see the description of **fbackup** in the *HP-UX Reference* manual.

The syntax of the command, showing commonly used options, is as follows:

```
/etc/fbackup -f device [-f device]... -5 -u -g graphfile -c config
```

where:

**-f** *device* specifies the name of an output device file. Multiple files can be specified to increase the amount of media that you can use for backup. So, if you have more than one magnetic tape drive, you can specify more than one drive and load tapes onto all drives specified in the command. The command backs up onto the first drive specified first, then the second, etc... then returns to the first and continues cyclically (provided that the tape was changed - otherwise, it stops and waits).

**-0-9** indicates the backup level (a single-digit number) where level 0 is a full backup and higher numbers are incremental backup levels.

**-u** updates **/usr/adm/fbackupfiles/dates** to contain the backup level, when backup occurred, and the graphfile (see -g option for definition).

**-g** *graphf* refers to a *graphfile,* that is, a text file that contains a list of files included or excluded from the backup. If a directory is included, it is expanded to back up the entire subtree. Graphfile entries are preceded either by **i**, a space, and a pathname to include them in the backup or by **e**, a space, and a pathname to exclude all of them from the backup. A sample graphfile is shown later in this section.

**-c** *config* specifies a *config* (configuration) file that specifies values for the following parameters (only commonly used ones shown here):

- name of a file to be executed when a fatal error occurs, for example:

  ```
  error     /usr/adm/fbackupfiles/error
  ```

- number of times to try to back up an active file:

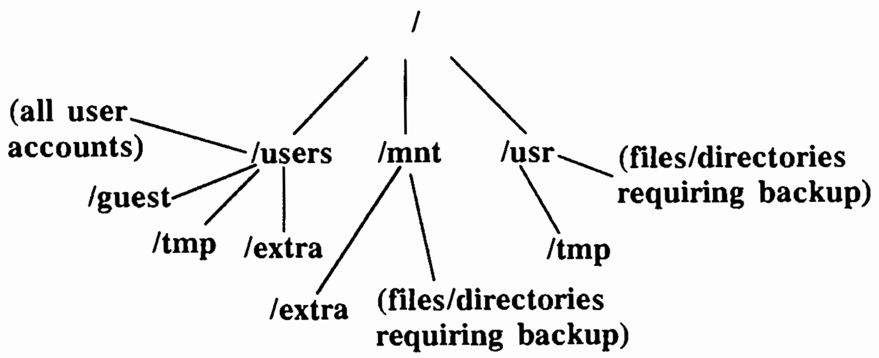
  ```
  maxretries     5    (default is 5)
  ```

- number of file reader processes to use

  ```
  readerprocesses    2   (default is 2; may have up to 6)
  ```

## Sample Graphfile

As System Administrator, you can ease backup procedures by creating a *graphfile* that is specified in the backup commands. For consistency and ease of identification, name this file: **/user/adm/fbackupfiles/graphfile**. Since the basic directory structure of your system should remain the same, you may use the same graphfile for all backups. For example:, given the following directory structure:


/
(all user accounts)
/users
/mnt
/usr
(files/directories requiring backup)
/guest
/tmp
/extra
/tmp
/extra
(files/directories requiring backup)

Here is a sample graphfile you could use to back up the system:

```
i /usr
e /usr/tmp
i /mnt
e /mnt/extra
i /users
e /users/guest
e /users/tmp
e /users/extra
```

The sample graphfile when specified in an **fbackup** command will recursively backup all of **/usr** except **/usr/tmp**, all of **/mnt** except **/mnt/extra**, and **/users** except for **/users/guest**, **/users/tmp**, and **/users/extra**. You need to customize a graphfile that includes the appropriate directories in your environment.

## Examples

The following examples show how to backup all directories, subdirectories, and files implied recursively in the text file **/usr/adm/fbackupfiles/graphfile** to 9-track tape at **/dev/rmt/0h** using the **fbackup(1M)** command. The **/usr/adm/fbackupfiles/config** file specifies a shell script for the system to execute (**/usr/adm/fbackupfiles/error**) if an error is encountered during backup.

To do a full backup:

```
/etc/fbackup -u0f /dev/rmt/0h -g /usr/adm/fbackupfiles/graphfile -c
/usr/adm/fbackupfiles/fb_config
```

To do a weekly backup:

```
/etc/fbackup -u3f /dev/rmt/0h -g /usr/adm/fbackupfiles/graphfile -c
/usr/adm/fbackupfiles/fb_config
```

To do a daily backup:

```
/etc/fbackup -u8f /dev/rmt/0h -g /usr/adm/fbackupfiles/graphfile -c
/usr/adm/fbackupfiles/fb_config
```

To back up information onto cartridge tape, pipe the output of **fbackup** to **tcio(1M)**. For example, the following command writes backed up information to **stdout** which is then input to **tcio(1M)**:

```
/etc/fbackup -uf - -i /mnt/user/src -e /mnt/user/src/file.0 | tcio -oer /dev/rct
```

The **-f** with the filename - (stdout) option send the output to **stdout** which is then sent by **tcio** to the output device called **/dev/rct**.

## File System Recovery

---

**NOTE**

**If your file system is destroyed, take the time to understand the circumstances that caused the problem to help prevent having to repeat this procedure.**

---

If the entire file system is destroyed or if the **frecover** command does not function properly, you first need to get the system up and running.

Be sure that it is not a hardware problem with your disk. If the disk operates properly and you have a backup, restore the backup data to the corrupted file system after using **mkfs** or **newfs** to remake the file system. This should be done only if you have a current backup tape because ALL contents of the file system will be lost after completion of **newfs** or **mkfs**.

If you have not performed a backup and the error is in one of the root sections, you must reinstall the system using the original distribution medium. Follow the instructions in *Installing and Updating HP-UX*.

---

**NOTE**

**If you have updated your system since you first installed HP-UX, contact your local Hewlett-Packard Sales Office to obtain a current installation tape. Otherwise, you will have to do one (or more) updates before you can begin restoring files.**

---

Once you have reinstalled the original system and it is operating properly, use the **frecover** command to retrieve the most recent full and incremental backups from the backup device.

The **frecover** command options are similar to those for **fbackup**. You can also define a graphfile to indicate which files should be recovered, and a config file to customize the behavior of the command. For example, the following command recovers all files specified in **/usr/adm/fbackupfiles/graphfile** and executes a shell script listed in **/usr/adm/fbackupfiles/fr_config** if the recovery process encounters an error:

```
/etc/frecover -x -g /usr/adm/fbackupfiles/graphfile -c
/usr/adm/fbackupfiles/fr_config -f/dev/rmt/0h
```

The following command recovers all files on the backup media and organizes them into the directories they were backed up from:

```
/etc/frecover -r -f/dev/rmt/0h
```

You may also use SAM to recover an index of files or specific files from tape.

6

# System Accounting

HP-UX System Accounting allows you to accomplish accounting tasks through a number of commands. This chapter describes these commands and contains the following sections:

- Overview of System Accounting
- Daily Usage and Installation
- Disk Space Usage Accounting
- Connect Session Accounting
- Process Accounting
- Charging Fees to Users
- Accounting Summary and Report
- Updating the Holidays File
- Fixing Corrupted Files
- Sample Accounting Shell Scripts

Refer also to Appendix B "System Accounting Files" which contains brief definitions of all the files used by System Accounting.

---

**NOTE**

**Most of the material in this chapter assumes greater knowledge of HP-UX than is required of most users. If you are unfamiliar with the concepts and terminology in this chapter, review Chapter 9, "System Management Concepts."**

---

## Goals of System Accounting

Multiuser HP-UX allows concurrent sharing of computer resources among many users: several users can be logged in, sharing disk space, memory, and the CPU. On multiuser systems, HP-UX System Accounting provides the means to:

- Monitor disk space usage for individual users
- Record connect session data (logins/logouts)
- Collect resource utilization data (such as memory usage and execution times) for individual processes
- Charge fees to specific users
- Generate summary files and reports that can be used to analyze system performance and bill users for resource use

## Overview of System Accounting

This section generally describes the features of System Accounting. Key terms are defined, commands are introduced, system data flow is described, and finally, you are shown the login and directory structure of System Accounting.

### Definitions

The following terms are specific to System Accounting and are described in the following paragraphs.

Prime connect time
Non-Prime connect time
Process Accounting Records
Total Accounting Records

#### Prime/Non-Prime Connect Time

Prime time is the time during the day when the computer system is most heavily used, from 9:00 am to 5:00 pm. Non-prime time is the remaining time during the day when the system is less heavily used.

When reporting computer time usage, System Accounting distinguishes between prime and non-prime time usage. You can specify prime and non-prime time on your system by editing the file **/usr/lib/acct/holidays**. (For details on the **holidays** file, see the section "Updating the Holidays File" in this chapter.)

**NOTE**

**Prime time is in effect only on weekdays (Monday through Friday); non-prime time is in effect weekends (Saturdays and Sundays) and on any holidays specified in the** holidays **file.**

### Process Accounting Records

Once System Accounting is installed and turned on, the following occurs: when a process terminates, the kernel, by default, writes a process accounting record into the current process accounting file, **/usr/adm/pacct**. You may specify a file other than **pacct** as the process accounting file with the **accton** command.

A process accounting record contains resource-usage data for a single process; it summarizes how much of the various resources the process used during its lifetime. Process accounting records contain:

- The user ID of the process's owner.
- The name of the command that spawned the process.
- The time it took for the process to execute.

For more detail on the contents and format of process accounting records, see **acct(4)** in the *HP-UX Reference*.

### Total Accounting Records

These records, created by various accounting commands, contain summary accounting information for individual users. They provide basic information for many reports generated by System Accounting and contain:

- The ID and user name of the user for whom the total accounting record was created
- The total number of processes the user spawned during the accounting period for which the total accounting record was created
- Fees for special services provided to this user.

The exact contents and format of total accounting records may be found in **acct**. In addition, commands covered later in this chapter show how these records are created and used by System Accounting.

## Introduction to System Accounting Commands

System Accounting provides many commands to accomplish tasks such as creating, displaying, removing, merging, summarizing, and reporting data.

Accounting commands are organized in the following groups:

- Installation
- Disk usage accounting
- Connect session accounting
- Process accounting
- Fee charging
- Summarizing and reporting accounting information

These commands reside in the directory **/usr/lib/acct**. Descriptions of these command groups, along with a brief synopsis of each command, are given below.

An additional utility called the System Activity Reporter (**sar**) is available to help monitor and report on system activity. Refer to Chapter 8 for more information on **sar**.

### Installation Commands

Two commands turn System Accounting on or off. One command turns accounting on when HP-UX is powered up and the other turns it off when the system is shut down. They also cleanup some files.

- **startup** - Starts accounting when HP-UX is switched to multiuser mode. Invoked from **/etc/rc**.
- **shutacct** - Turns off accounting when HP-UX is shut down with the **/etc/shutdown** shell script.

### Disk Usage Accounting Commands

In general, these commands show disk space usage (in blocks) for individual users. They also produce total accounting records.

- **acctdusg** and **diskusg** - Show how many blocks of disk space users are consuming. They differ in how they produce the information. **acctdusg** gets its input from a list of pathnames created by **find**, and **diskusg** looks at the inodes of the file system to create its output.
- **acctdisk** - Produces total accounting records and its input is supplied (either directly or indirectly) from **acctdusg** or **diskusg**.
- **dodisk** - Produces total accounting records with the **diskusg** and **acctdisk** commands. **dodisk** is normally invoked by **cron** and, by default (no options), will do disk accounting on special files in **/etc/checklist**.

### Connect Session Accounting

Independent of System Accounting, the **login** and **init** programs record connect sessions by writing records into **/etc/wtmp**. Accounting commands can display or fix this file, and produce total accounting records.

- **fwtmp** displays the information contained in **wtmp** (converts binary records to formatted ASCII records first).
- **wtmpfix** normalizes connect session records in **wtmp** that span date/time changes (see **date(1)**). It also validates login names in connect session records.
- **acctcon1** summarizes **wtmp** in ASCII readable format, producing one line per connect session.
- **acctcon2** takes input from **acctcon1** and produces total accounting records as output.
- **prctmp** displays session record file which is the accounting standard input. (The session record file is normally **/usr/adm/acct/nite/ctmp**.)

## Process Accounting

When process accounting is turned on, the kernel writes a process accounting record to **/usr/adm/pacct** whenever a process terminates. A number of accounting commands exist that summarize and report this accounting information. In addition, certain commands turn process accounting on or off and ensure that **pacct** doesn't become too large.

- **accton** with a filename turns process accounting on and the kernel writes records to the specified file. Without a filename, process accounting turns off and the kernel stops writing records to **pacct**.

  **accton** with the system call **acct**, turns process accounting on or off. This command is called with **startup** script. In addition, only a superuser can execute **accton**.
- **ckpacct** checks the size of the process accounting file **pacct**. If **pacct** becomes too large, then a new **pacct** file is created via **turnacct switch**. If disk space becomes critically short, then process accounting is turned off until sufficient space is available. This command is normally invoked by **cron**.
- **turnacct on/off/switch** performs one of three functions, depending on which argument (on, off, or switch) is specified. **turnacct on** turns process accounting on by calling **accton** with the default filename **/usr/adm/pacct**; **turnacct off** turns process accounting off by calling **accton** with no filename; **turnacct switch** renames the current **pacct** file (so that it is no longer the current process accounting file) and creates a new, empty **pacct** file.
- **acctcom** searches and displays process accounting records contained in **pacct** (or any specified file).
- **acctcms** takes one or more files in the form of **acct** (including **pacct**) as input, and produces summary accounting information by command, as opposed to by process.
- **acctprc1** adds log-in names corresponding to user IDs and produces readable process accounting information (sorted by user IDs and log-in names), mainly for input to **acctprc2**.
- **acctprc2** takes input from **acctprc1** and produces total accounting records.

### Charging Fees

Occasionally, you may want to charge a user for services performed. For example, fees for the amount of disk space used. The **chargefee** command allows you to charge fees to specific users.

### Summarizing and Reporting Accounting Information

These commands summarize and report data created through the command groups described previously. They represent the highest level of accounting commands.

- **prtacct** converts and displays total accounting records in ASCII readable format.
- **acctmerg** combines or adds the contents of separate total accounting files (up to 9 besides standard input) into a single total accounting file. Allows merging of disk, process, and connect session total accounting records.
- **runacct** is the main accounting shell script. Normally invoked daily by **cron**, this command processes disk, connect session, process, and fee accounting information and produces summary files and reports. It accomplishes its task by proceeding through various states. In each successive state, it invokes accounting commands to perform a specific task. For example, in one state, total accounting records for connect sessions are created; in another, disk, connect session, process, and fee total accounting records are merged to create one total accounting file.
- **prdaily** is invoked by **runacct** and formats a report of the previous day's accounting data; the report is stored in the file

  **/usr/adm/acct/sum/rptmmdd**

  where **mmdd** is the month and day of the report. **runacct** may also be used to display a report of the current day's accounting information.
- **monacct** is invoked once a month (or accounting period), and summarizes daily accounting files and produces summary files for the accounting period.
- **acctcms** reads one or more files, normally in the form of **acct**. It adds all records for processes that executed identical commands, sorts them, and writes them to the standard output.

**NOTE**

**The installation commands do not appear in Figure 6-1, because they are not directly involved in the data creation process; they merely ensure that it happens. The commands runacct and prdaily are shown as having no input. This isn't exactly true; they do have input, but they get their input by executing other accounting commands. In essence, their input is basically the same as that of the other command groups.**

## System Data Flow

At this point, you have the rudimentary knowledge necessary to understand how System Accounting works; you know some important definitions and should basically know what the various commands do. The purpose of this section is to help you visualize how the different commands work together to create accounting data.

Figure 6-1 illustrates how accounting data is created. The diagram is broken into five separate parts, each one representing the data flow for a given command group. The following conventions are used:

| Symbol | Description |
|---|---|
| **source => dest** | Wide arrows represent **transfer of data** from a source to a destination. The source is at the start of the arrow; the destination, at the point. For example, the inodes of the file system are the source of information used by **diskusg**, which in turn is the source of disk usage reports that are inputs to **acctdisk**. |
| **cause —> object** | Thin arrows represent cause-effect relationships. The cause lies at the start of the arrow; the object affected lies at the point. For example, **turnacct on** invokes **accton** which then signals the kernel to begin writing process accounting records to **pacct**. |
| **files** | Boxes with rounded corners represent files or groups of files. In a more general sense, they represent the input to and output from the various commands. |

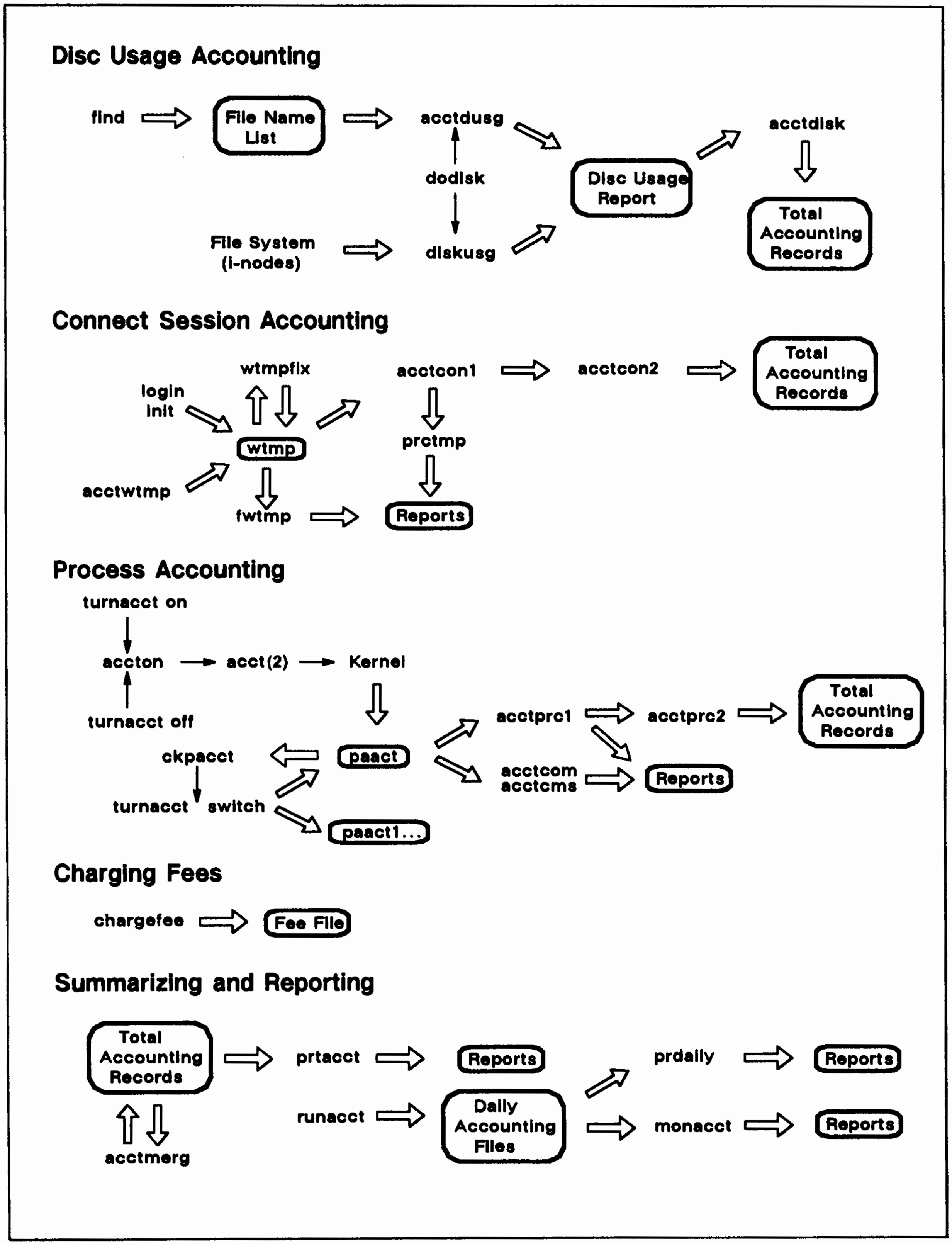


Figure 6-1. System Accounting Data Flow Diagram

## Login and Directory Structure

You now know the basics, but before you begin learning the day-to-day usage of accounting commands you must know how to log in. In addition, you should know the accounting directory structure where the various commands, directories, and files are located. These two topics are discussed here.

### Logging In

The login name for System Accounting is **adm**; the user ID for **adm** is 4. The **adm** login is a member of the **group adm**, and the group **adm** has a group ID of 4, also.

The home directory for the **adm** login is **/usr/adm**. You log in to System Accounting the same way you do for any account; simply supply the login name to the HP-UX login prompt:

```
login: adm
```

---

**NOTE**

**The integrity of accounting data files must be maintained if System Accounting is to generate accurate reports. For this reason, it is highly recommended that a password be used with the adm login.**

---

### Directory Structure

System Accounting uses a multilevel directory structure to organize its many accounting files. Each directory in this structure stores related groups of files, commands, or other directories. (See Appendix B, "System Accounting Files" for definitions of the accounting data files.)

Figure 6-2 illustrates this structure, and descriptions of each directory follow:

- **/usr/adm** contains all active data-collection files, such as **pacct** and **fee**.
- **/usr/adm/acct** contains the **nite**, **sum**, and **fiscal** directories described below.
- **/usr/adm/acct/nite** stores data files that are processed daily by **runacct**.
- **/usr/adm/acct/sum** contains cumulative summary files updated by **runacct**.
- **/usr/adm/acct/fiscal** contains periodic (monthly) summary files created by **monacct**.
- **/usr/lib/acct** stores System Accounting commands.
- **/etc** contains **wtmp** and shell scripts **rc** and **shutdown**.

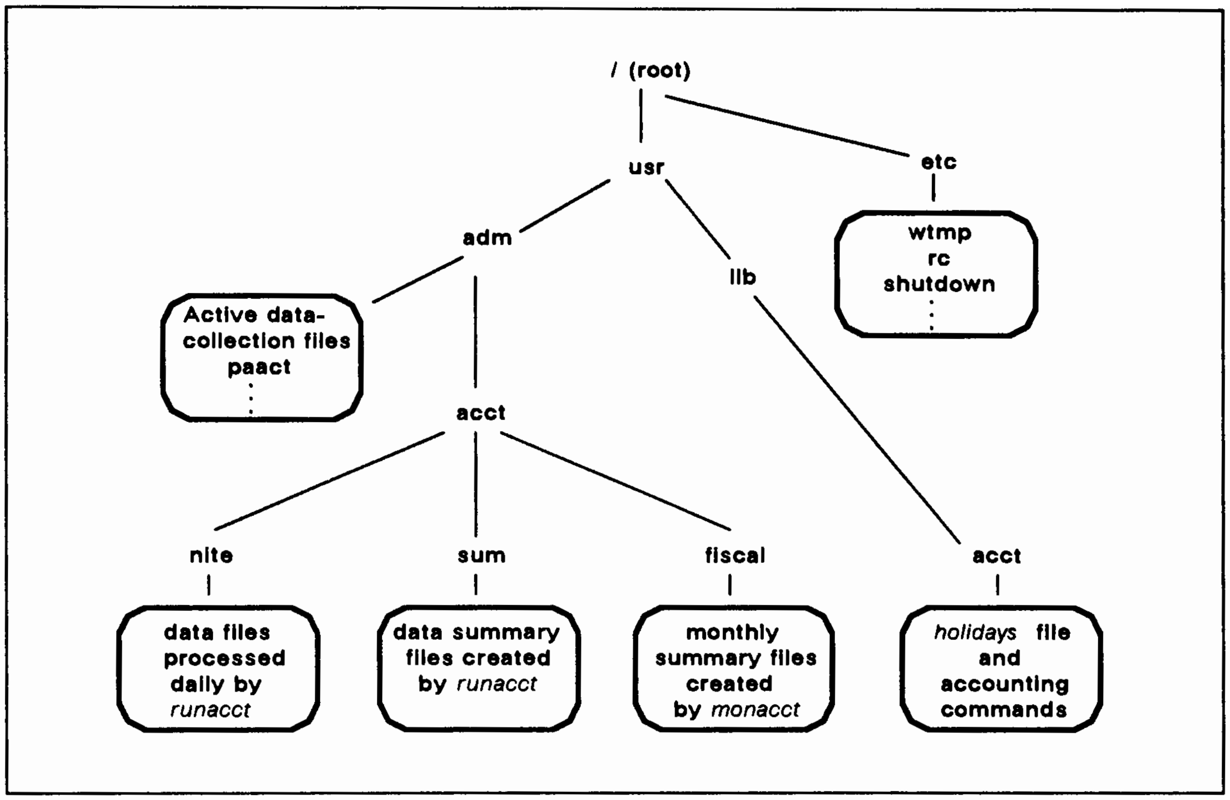

/ (root)
usr
etc
adm
lib
wtmp
rc
shutdown
Active data-collection files
paact
acct
nite
sum
fiscal
acct
data files processed daily by *runacct*
data summary files created by *runacct*
monthly summary files created by *monacct*
*holidays* file and accounting commands

**Figure 6–2. System Accounting Directory Structure**

## Daily Usage and Installation

Now that the basics have been covered, you can start learning how to use System Accounting daily. The purpose of this section is to show you:

1. How System Accounting automatically creates daily and monthly accounting data and reports.

2. What you must do to get System Accounting running on your system.

After reading this section, you should be able to install System Accounting on your system. Once properly installed, it automatically generates daily and monthly accounting data and reports.

### Summary of Daily Operation

The daily operation of System Accounting is summarized by the following steps:

1. When HP-UX is switched into multiuser mode, the system initialization shell script **/etc/rc** executes the accounting command **startup**. **startup** starts accounting and performs the following functions:

   a. Calls **acctwtmp** to add a boot record to **wtmp**. This record is marked by storing "acctg on" in the device name field of the **wtmp** record.

   b. Turns process accounting on via **turnacct on**. **turnacct on** executes **accton** with the filename argument **/usr/adm/pacct**.

   c. Removes work files left in the **sum** directory by **runacct**.

2. A report of the previous day's accounting information can be created by running **prdaily**. After **runacct** has been executed, **prdaily** generates valid reports.

3. The **ckpacct** command is executed every hour via **cron** to ensure that the process accounting file **pacct** doesn't become too large. If **pacct** grows past a set maximum number of blocks, **turnacct switch** is invoked, which creates a new **pacct** file. (Other conditions may also limit the size of the process accounting file or turn process accounting off; for more details, see the discussion of **ckpacct** in the "Process Accounting" section.) The advantage of having several small **pacct** files is that **runacct** may be restarted faster if a failure occurs while processing these records.

4. The **chargefee** program charges fees to users. It adds records to the file **fee**. These records are processed during the next execution of **runacct** and merged in with total accounting records.

5. **runacct** is executed via **cron** each night. It processes the active **fee** file and the process, connect session, and disk total accounting files. It produces command and resource-usage summaries by login name.

6. When the HP-UX system is turned off with **shutdown**, the **shutacct** command is executed to stop System Accounting and perform the following:

   a. Write a termination record to **wtmp** via the command **acctwtmp**. This record is marked by having "acctg off" in the device name field.

   b. Turn process accounting off by calling **turnacct off**.

## How to Install System Accounting

Not all systems require accounting services. For this reason, HP-UX System Accounting is an option. If you want to use accounting, you must install it yourself. The installation procedure follows.

1. Making **startup** in **/usr/lib/acct** executable (mode 755).

2. Updating **/etc/shutdown**.

3. Setting PATH for accounting commands.

4. Creating **crontab** entries.

5. Edit **/etc/rc** to remove the comment characters preceding the **startup** command entries.

Each of these steps must be carried out to ensure that System Accounting automatically creates daily and monthly accounting information. Detailed descriptions of each step follow.

## The Startup Script

The system initialization shell script **/etc/rc** contains entries that start System Accounting automatically when the system is switched into multiuser mode. These entries check to see if the file **/usr/lib/acct/startup** is executable and if so, execute it. Initially, they are disabled with the comment character (#) in column 1. To enable accounting, remove the # character from these lines. The entries in **/etc/rc** are shown below.

```
#                    if [ -x /usr/lib/acct/startup ]
#                    then /usr/lib/acct/startup
#                    fi
```

If you want to use accounting, log in as **root** and enter:

```
chmod 755 /usr/lib/acct
```

## Updating /etc/shutdown

To ensure that accounting is turned off when the system is brought down via **shutdown**, ensure that the **shutacct** command is in the **shutdown** shell script. The call to **shutacct** should be placed in **shutdown** after all processes are killed by the **/etc/killall** command. By calling **shutacct** after **killall**, process accounting information can be captured for the processes that were terminated by **killall**. The entries for the **shutacct** command should be as follows:

```
if [ -f /usr/lib/acct/shutacct ]
then
     /usr/lib/acct/shutacct
echo "\n Accounting stopped."
fi
```

**NOTE**

**If you do not use** /etc/shutdown **to bring your system down, then use some other means such as** turnacct off **or** shutacct **by itself to turn system accounting off before shutting down your system.**

## Setting PATH for Accounting Commands

Finally, you should set the PATH shell variable in **/usr/adm/.profile** so that System Accounting knows where to look for commands. The following paths should be added to the current paths.

```
PATH=/usr/lib/acct:/bin:/usr/bin:/etc:/usr/adm
```

## Creating crontab Entries

To automate the daily and monthly creation of accounting data, create or add to a **crontab** file that **cron** may use to automatically run certain accounting commands. This process consists of the following steps:

1. Log in to System Accounting as user **adm**.
2. Use an editor to create a **crontab** file containing the accounting commands that are to be run automatically by **cron**. The entries you should include are shown after these steps.
3. Execute the **crontab** command, specifying the file created in step 2 as input. This step ensures that the **crontab** file created in step 2 will be scanned by **cron** every minute. After invoking this command, the **crontab** file will be stored in the file:

   ```
   /usr/spool/cron/crontabs/adm
   ```
4. This file is stored without write permission for general users. At this point, you are finished creating **crontab** entries. To change entries, re-edit the **crontab** file created in step 2 and use the **crontab** command again.

Include the following entries, accompanied by a description of each, in the **crontab** file created in step 2:

```
0 4 * * 1-6 /usr/lib/acct/runacct 2> /usr/adm/acct/nite/fd2log
```

**runacct**, the main accounting shell script, should be executed daily (during non-prime hours) to generate daily accounting reports. This entry executes **runacct** at 4:00 AM every Monday through Saturday. Error messages will be redirected to the file **/usr/adm/acct/nite/fd2log**, if any errors occur while **runacct** executes.

```
0 2 * * 4 /usr/lib/acct/dodisk
```

**dodisk** creates total accounting records that summarize disk space usage for individual users. This entry runs **dodisk** at 2:00 AM every Thursday morning.

```
5 * * * * /usr/lib/acct/ckpacct
```

To prevent the process accounting file **pacct** from becoming too large, execute **ckpacct** hourly. This entry invokes **ckpacct** at five minutes into every hour.

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

The **monacct** command merges monthly accounting data. This entry generates a monthly total report and total accounting file. **monacct** is executed at 5:15 AM on the first of every month.

---

**NOTE**

**The dates and times shown in the crontab entries above are only suggestions; you may tailor crontab entries to suit your needs. However, if you use different entries than those shown here, be sure that monacct is run at a time when runacct has sufficient time to finish.**

---

## Disk Space Usage Accounting

System Accounting monitors disk space usage for individual users. In this section, disk space usage accounting commands are explained. Before reading this discussion, you may want to review the "File System Implementation" section of Chapter 9.

Disk usage commands have two main functions: they report disk usage (in blocks) for individual users and create disk total accounting records (supplied as inputs to commands such as **prtacct** or **runacct**).

### Reporting Disk Space Usage

The **acctdusg** and **diskusg** commands report disk usage for individual users; both commands show the number of disk blocks allocated to specific users. However, each command has slightly different options. In addition, each differ in how they produce accounting information.

#### acctdusg

The **acctdusg** command takes from standard input a list of pathnames, usually created by the **find** command. For each file in the list, **acctdusg** identifies the owner of the file, computes the number of blocks allocated to the file, and adds this amount to a running total for the file's owner. When finished looking through the list, **acctdusg** displays the information accumulated for each user: user ID, user name, and number of blocks used.

This command is useful for reporting disk usage for specific users or files. For example, suppose you want to know how many blocks of disk space you are using: your user ID is **351**, user name is **bill**, and your home directory is **/users/pseudo/bill**. The following illustrates how you would use the **find** and **acctdusg** commands to show this information.

```
$ find /users/pseudo/bill -print > bills.files
$ acctdusg < bills.files
00351   bill       30
$ rm bills.files
```

In the above example, **bill** is using 30 blocks of disk space. The series of commands shown could easily have been combined into one line, such as:

```
$ find $HOME -print | acctdusg
00351   bill       30
```

The next example shows how to use **acctdusg** to generate disk usage information for all files in the system:

```
$ find / -print | acctdusg
00350   fred   11
00351   bill   30
00352   mike   17
00353   sarah  13
00354   molly  18
00000   root   3
00004   adm    36
00001   bin    2434
```

Two options are included with **acctdusg**:

**-u noowners** If **-u** is given, then pathnames of the files for which no owner is found are written into the file **noowners**. This option could potentially find users who are trying to avoid disk charges.

**-p pfile** The password file **/etc/passwd** is the default file used by **acctdusg** to determine ownership of files. If the **-p** option is used, then **acctdusg** uses **pfile** instead. This option is not needed if your password file is **/etc/passwd**.

The shell script **grpdusg** provided in the section "Sample Accounting Shell Scripts" displays disk accounting information for users in a given group. It illustrates the use of the **-u** option with **acctdusg**.

## diskusg

The diskusg command reports disk usage information in the same format as **acctdusg** reporting user ID, user name, and total disk blocks used. However, **diskusg** generates disk accounting information by looking through the inodes of a specified disk special file (see the "Inodes" and the "File System Implementation" sections in Chapter 9 for more information on inodes and special files.) This makes **diskusg** faster than **acctdusg**.

The syntax of the **diskusg** command is:

```
diskusg [options] [files]
```

It generates a disk usage report from data in **files**, if specified; otherwise standard input is used. **diskusg** is normally invoked with the **files** argument. When specified, **files** are the special filenames of the devices containing the inode information used by **diskusg** to generate its report. **files** is normally a special file for a disk section from the **/dev** directory. The following options may be used with **diskusg**:

-s This tells **diskusg** that:

1. input is in **diskusg** output format, and
2. that all lines for a single user should be combined into a single line.

This option merges data from separate files, each containing the output from using **diskusg** on different devices.

| | |
|---|---|
| **-v** | This option is useful for finding users who are trying to avoid disk space accounting charges. When this option is specified, **diskusg** writes records to **stderr** (standard error output) showing the special filename, inode number, and user ID of files that apparently have no owner. |
| **-i fnmlist** | With this option, **diskusg** ignores data on file systems whose name is in **fnmlist**. **fnmlist** is a list of file systems separated by commas or enclosed within quotes. |
| **-p pfile** | This is the same as the **-p** option of **acctdusg**. |
| **-u ufile** | This option produces the same output as the **-v** option. The difference between the two options is that **-v** writes its output to **stderr**; this option writes its output to **ufile**. |

The output of **diskusg** is normally used by **acctdisk** to create disk total accounting records. In addition, **diskusg** is normally called by **dodisk**.

The following example creates disk usage information for all users whose files reside on the disk whose device file is **/dev/dsk/c0d0s0**. Note that the file system used in this example is the same as was used in the previous **acctdusg** example.

```
$ diskusg /dev/dsk/c0s0d0
0       root        10616
1       bin           778
4       adm            96
350     fred           14
351     bill           32
352     mike           20
353     sarah          16
354     molly          22
355     horatio         2
501     guest           2
```

The differences between **diskusg** and **acctdusg** are best illustrated by comparing their outputs. Note that:

1. **acctdusg** places leading zeros on user IDs; **diskusg** doesn't.

2. **acctdusg** counts files **only under each users $HOME directory**. Files that users own in directories other than their home directory (for example, files in the **/tmp** directory) are counted as files with no owner.

3. Two extra users **horatio** and **guest** show up in the output of **diskusg** when compared with the output from

   ```
   find / -print | acctdusg
   ```

   This occurred because the directories of these two users were empty; therefore, no disk usage totals were generated. However, **diskusg** looked at inodes and saw that **horatio** and **guest** were actually using two blocks each for the directories themselves.

4. If two or more users have links to a particular file, then **acctdusg** will prorate disk space usage for the file between each user. For example, if three users had a link to a 300-block file, each user would be charged for 100 blocks of this file.

## Creating Total Accounting Records

Two commands are used to create total accounting records: **acctdisk** and **dodisk**.

### acctdisk

The **acctdisk** command takes from standard input records of the format produced by **acctdusg** and **diskusg**. From these records, **acctdisk** produces disk total accounting records that may be inputs to **prtacct** or **runacct**.

The following would write disk total accounting records to the file **disktacct** for all users in the group **pseudo**:

```
find / -group pseudo -print | acctdusg | acctdisk > disktacct
```

The next example would generate disk total accounting records for all users who have files on the disk specified. The total accounting records are written to the file **disktacct**.

```
diskusg /dev/dsk/c0d0s0 | acctdisk > disktacct
```

**acctdisk** has no options and is normally invoked by **dodisk**.

### dodisk

The **dodisk** command, normally invoked by **cron**, creates disk total accounting records for daily usage by System Accounting. The syntax for **dodisk** is:

```
dodisk [-o] [ files ... ]
```

In the default case, **dodisk** creates disk total accounting records on the special files whose names are stored in **/etc/checklist**; the special filenames are supplied as input to **diskusg**, which pipes its output to **acctdisk**, which in turn creates total accounting records.

If the **-o** option is used, **dodisk** creates total accounting records more slowly by using the log-in directory.

If **files** are used, disk accounting will be done on these file systems only. If the **-o** option is used, then **files** should be mount points of mounted file systems; if omitted, they should be the special filenames of mountable file systems.

**NOTE**

**See the "Daily Usage and Installation" section of this chapter for more information on how** dodisk **should be invoked by** cron.

It is possible for malicious users to defeat disk space accounting by giving their files away to other users with **chown** or **chgrp** (by default, all users can execute them). To avoid this, run the accounting commands periodically or take away the **chown** privilege from specific groups. To let only the superuser use the change-ownership abilities, add the following line to **/etc/rc**:

```
setprivgrp -n CHOWN
```

To let one or more groups of users use the change-ownership abilities, add a line for each group to **/etc/rc**, similar to the following:

```
setprivgrp groupname CHOWN
```

---

**NOTE**

**Taking away the change-ownership ability may cause problems when running some commands or applications.**

---

## Connect Session Accounting

Whenever a user logs in or out of HP-UX, the program **login** records the connect session in the file **/etc/wtmp**. Records in **wtmp** contain the following information:

- The terminal name on which the connect session occurred.
- The login name of the user.
- The current time/date at login or logout.
- Other status information (see **utmp(4)** for details).

System Accounting provides commands that allow you to write records to **wtmp**, to display and manipulate **wtmp**, and to create total accounting records from **wtmp**. These commands are covered in this section.

### Writing Records to wtmp

The **acctwtmp** command writes records to **wtmp** and is normally invoked by **startup**. **shutacct** records when System Accounting was turned off. The format of the command is:

```
acctwtmp "reason" >> /etc/wtmp
```

where **reason** is a string describing the reason for writing the record to **wtmp**. Because **acctwtmp** does not directly write records to **wtmp**, you must append the output from **acctwtmp** to the **wtmp** file **/etc/wtmp**.

The **reason** string may be any combination of letters, numbers, spaces, and the dollar sign ($), but may not exceed 11 characters and must be enclosed in double quotes.

## Displaying Connect Session Records

To display the contents of **wtmp**, use the **fwtmp** command. When no options are specified, **fwtmp** takes **wtmp** records from standard input and writes them to standard output as ASCII readable records. The output of this command can either:

1. be edited, via a HP-UX editor such as **vi**, and then rewritten to **wtmp** using special **fwtmp** options described below; or
2. supplied as input to commands which convert the information to total accounting records.

The syntax of **fwtmp** is:

```
fwtmp [-ic]
```

The options can be used in any combination. The following table describes what the different combinations do.

| Option | Description |
|---|---|
| **-ic** | Denotes that input is in ASCII readable form to be converted to binary. This is essentially the opposite of using **fwtmp** without any options. |
| **-i** | Both input and output are in ASCII format. This is the same as performing an ASCII to ASCII copy. |
| **-c** | Both input and output are in binary format, a binary to binary copy. |

The following example shows **fwtmp** output followed by descriptions of each column in the report:

```
$ fwtmp < /etc/wtmp
           system boot  0   2 0000   0000  479472540   Mar 12 03:49:00 1985
root   co  console      0   7 0000   0000  479475173   Mar 12 04:32:53 1985
           acctg on     0   9 0000   0000  479493135   Mar 12 09:32:15 1985
mike   a1  ttya1      352   7 0000   0000  479493590   Mar 12 09:40:00 1985
mike   a1  ttya1      352   8 0011   0000  479496000   Mar 12 10:20:00 1985
sarah  07  tty07      353   7 0000   0000  479518335   Mar 12 16:32:15 1985
bill   10  tty10      351   7 0000   0000  479521475   Mar 12 17:24:35 1985
sarah  07  tty07      353   8 0011   0000  479522478   Mar 12 17:41:18 1985
bill   10  tty10      351   8 0011   0000  479526487   Mar 12 18:48:07 1985
       co  console      0   8 0011   0000  479526488   Mar 12 18:48:08 1985
           acctg off    0   9 0000   0000  479526493   Mar 12 18:48:13 1985
           system boot  0   2 0000   0000  479389800   Mar 12 05:00:00 1985
```

| Column | Description |
|---|---|
| 1 | The login name of the user who logged in or out. |
| 2 | **/etc/inittab** id (this is usually the number of the line on which the connect session took place). |
| 3 | The name of the device on which the connect session occurred. |
| 4 | Process ID of the user who logged in or out. |
| 5 | Entry type. This field contains information on the type of record. For example, it shows whether the record is a login record (entry type=7), logout record (entry type=8), or if the record was written by **acctwtmp** (entry type=9). See **utmp(4)** for more details on this field. |
| 6,7 | Exit status for connect session. See **login(1)** and **utmp(4)** for details. |
| 8 | Time that entry was made (in elapsed seconds since January 1, 1970). |
| 9 | The equivalent of column 8 in date/time format showing month, day, time of day (in 24-hour format), and year. |

## Fixing wtmp Errors

When a user logs into HP-UX, the **login** program stores the value seven (7) in the entry type field of the connect session record. When that user logs out, an entry of eight is recorded. You can see this by examining the sample output created by **fwtmp** in the previous section. Note that in the example, login records precede their corresponding logout records in chronological order.

Occasionally, this time-stamped ordering becomes inconsistent: logout records might precede login records. (This occurs when the date and time are reset while users are still logged in.) When this happens, the commands that create connect session total accounting records will not work properly.

Fortunately, there is a command that fixes corrupted **wtmp** files: **wtmpfix**. **wtmpfix** takes as input **wtmp** binary records and corrects the time/date stamps to be consistent; its standard output is also binary **wtmp** records. Its syntax is:

```
wtmpfix [files]
```

If you specify **files** then input is taken from **files**. You can use a dash (-) in place of **files** to indicate standard input. **Note that if you specify wtmp as both input to and output from this command, wtmp will be destroyed**. Therefore, take care not to destroy **wtmp**. The following shows how to properly fix **wtmp** using **wtmpfix**:

```
$ wtmpfix /etc/wtmp > wtmp.temp
$ fwtmp -c < wtmp.temp > /etc/wtmp
$ rm wtmp.temp
```

## Creating Total Accounting Records

This final set of connect session accounting commands is used to create connect session total accounting records. Before reading any further, you may want to review Figure 6-1 (in the "System Data Flow" section).

### acctcon1

The **acctcon1** command converts a sequence of login/logoff records (**wtmp** format) read from its standard input to a sequence of records, one per login session. Its input is normally redirected from **wtmp**; its output is in columnar ASCII format and can be supplied as input to **prctmp** or **acctcon2**.

The use of **acctcon1** is illustrated below by first displaying the contents of **wtmp** with **fwtmp**, and then using **acctcon1** to create connect session summary file. The columnar data produced by **acctcon1** is described after the report.

```
$ fwtmp < /etc/wtmp
                system boot    0 2 0000 0000   479472540  Mar 12 03:49:00 1988
root      co    console        0 7 0000 0000   479475173  Mar 12 04:32:53 1988
                acctg on       0 9 0000 0000   479493135  Mar 12 09:32:15 1988
mike      a1    ttya1        352 7 0000 0000   479493590  Mar 12 09:40:00 1988
mike      a1    ttya1        352 8 0011 0000   479496000  Mar 12 10:20:00 1988
sarah     07    tty07        353 7 0000 0000   479518335  Mar 12 16:32:15 1988
bill      10    tty10        351 7 0000 0000   479521475  Mar 12 17:24:35 1988
sarah     07    tty07        353 8 0011 0000   479522478  Mar 12 17:41:18 1988
bill      10    tty10        351 8 0011 0000   479526487  Mar 12 18:48:07 1988
          co    console        0 8 0011 0000   479526488  Mar 12 18:48:08 1988
                acctg off      0 9 0000 0000   479526493  Mar 12 18:48:13 1988
$ acctcon1 < /etc/wtmp
520095488    353     sarah    1665    2478   479518335  Tue Mar 12 16:32:15 1988
521012224    352     mike     1258    2045   479493590  Tue Mar 12 09:40:00 1988
520095488    351     bill        0    5012   479521475  Tue Mar 12 17:24:35 1988
521011712      0     root    41047    6488   479475173  Tue Mar 12 04:32:53 1988
```

Descriptions of the columnar data produced by **acctcon1** follow:

| Column | Description |
|---|---|
| 1 | Shows the device address (in decimal equivalent of major/minor device address) at which the connect session occurred. |
| 2 | Gives the user ID for the connect session record. |
| 3 | Displays the login name for the user. |
| 4 | Shows the number of prime connect time seconds that were used during the connect session. |
| 5 | Shows non-prime connect seconds. |
| 6 | The connect session starting time (in seconds elapsed since January 1, 1970) is displayed here. |
| 7 | The remaining columns convert column six to date/time format. |

In addition to its normal usage, **acctcon1** has four options:

| Option | Description |
|---|---|
| **-p** | This option tells **acctcon1** not to produce one record per connect session. Instead, **acctcon1** simply echoes its input one line per **wtmp** record showing line name, login name, and time (in both seconds and day/time format). Using this option is similar to using **fwtmp**, except that this option doesn't show status information, whereas **fwtmp** does. |
| **-t** | **acctcon1** maintains a list of users logged in. When it reaches the end of its input, **acctcon1** emits a session record for each line that still appears to be active. It normally assumes that its input is a current file, so that it uses the current time as the ending time for each session in progress. The **-t** flag causes it to use, instead, the last time found in its input, thus assuring reasonable and repeatable numbers for non-current files. |
| **-l file** | This option places a line usage summary report in **file**. This report shows each line's name, number of minutes used, percentage of total elapsed time, number of sessions charged, number of logins, and number of logins and logoffs. This report may be used to track line usage, identify bad lines, and find software/hardware oddities. Note that hang-up, termination of **login(1)**, and termination of the login shell each generate logoff records; therefore, the number of logoffs is often three to four times the number of connect sessions. Shown below is an example of the line use file **(lineuse)** created from the same **wtmp** file used in the previous **acctcon1** example; the standard output of **acctcon1** has been redirected into the file **ctmp**. |
| **-o file** | Using the **-o** option (for example, **acctcon1 -o foverall**) loads **file** with an overall record for the accounting period, including starting time, ending time, number of reboots, and number of date changes. |

Example:

```
$ acctcon1 -t -l lineuse < /etc/wtmp > ctmp
$ cat lineuse
TOTAL DURATION IS 899 MINUTES
LINE        MINUTES  PERCENT      # SESS      # ON      # OFF
console       856       95           1          1          1
tty07          69        8           1          1          1
ttya1          40        4           1          1          1
tty10          84        9           1          1          1
TOTALS       1049       --           4          4          4
```

### prctmp

The **prctmp** command is simple. Its only function is to put headings on the output created by **acctcon1**: **prctmp** makes a readable report from the output of **acctcon1**.

Input to the **prctmp** command is from standard input; therefore, to create a **prctmp** report from **acctcon1** information, you may simply pipe the output from **acctcon1** into **prctmp** as follows:

```
$ acctcon1 < /etc/wtmp | prctmp
```

Command **prctmp** responds by generating a report with a header showing the time when the report was run and the page number and other appropriate headings over the columns of output from **acctcon1**.

### acctcon2

The **acctcon2** command creates connect session total accounting records from standard input created by **acctcon1**. In other words, to create connect session total accounting records, simply send the output from **acctcon1** to the input of **acctcon2**.

The total accounting records created by **acctcon2** are sent to standard output. To store these records, you must redirect standard output. The following command line shows how to write total accounting records from the connect session record file (**wtmp**) into the file **ctacct**:

```
$ acctcon1 < /etc/wtmp | acctcon2 > ctacct
```

## Process Accounting

Process accounting commands provide the means to accumulate execution statistics such as memory usage, CPU time, number of input/output transfers for individual processes. This section describes how to:

1. Turn process accounting on,
2. Turn process accounting off,
3. Make sure that the process accounting file (**pacct**) does not become too large,
4. Display process accounting records,
5. Generate a command summary report, and
6. Create total accounting records from the process accounting file.

You might find it helpful to look at the System Data Flow Diagram (shown previously in Figure 6-1) when reading this section.

## Turning Process Accounting On

Process accounting is turned on automatically when the system initialization shell script **/etc/rc** is executed. However, if you have to turn on process accounting manually, there are two commands available: **turnacct on** and **accton**. After process accounting has been turned on, the kernel writes a process accounting record for every terminating process into the current process accounting file (**pacct** by default).

### turnacct on

The most common command used to activate accounting is **turnacct on** and only the superuser or the **adm** login may execute this command. **turnacct on** assumes that the process accounting file is the default file **pacct**. The action of **turnacct on** can be summarized as follows:

1. Verifies that the process accounting file **pacct** exists.
2. If **pacct** does not exist, then creates a new **pacct** file.
3. Turns process accounting on by invoking **accton** with the filename argument **pacct**.

To execute this command, simply enter **turnacct on** to the HP-UX prompt.

### accton

Only the superuser and the **adm** login can execute **accton**. When invoked with a filename argument, **accton** turns on process accounting and makes the specified filename the current process accounting file. For example,

```
$ accton /usr/adm/pacct
```

tells the kernel to start writing process accounting records to the file **/usr/adm/pacct**. The next example activates process accounting and makes the current process accounting file **/usr/adm/XX107**:

```
$ accton /usr/adm/XX107
```

Make sure that the file specified exists; otherwise, **accton** will fail.

Note that in the System Data Flow diagram (shown previously in Figure 6-1), **accton** is shown calling another routine, **acct**. **acct** is the system call that actually tells the kernel to start writing process accounting records. Refer to the *HP-UX Reference* for details on **acct(2)**.

## Turning Process Accounting Off

Two commands turn process accounting off: **turnacct off** and **accton** (with no filename argument). They tell the kernel to stop writing records to the current process accounting file.

---

**NOTE**

**If you have updated the** /etc/shutdown **shell script as described in the section "How to Install System Accounting," you may seldom use these commands because by adding the** shutacct **command to** /etc/shutdown, **the system automatically turns process accounting off.**

---

### turnacct off

Only the superuser or **adm** login may execute **turnacct off** to turn process accounting off.

```
$ turnacct off
```

### accton

Only the superuser or **adm** login may execute **accton**. Executing **accton** without a filename turns process accounting off.

```
$ accton
```

In the System Data Flow Diagram (shown previously in Figure 6-1), **accton** tells the kernel to stop writing process accounting records with the system call **acct**.

## Checking the Size of pacct

On a multiuser system, many processes can execute during a single hour. Therefore, process accounting files have the potential to become quite large. System Accounting has built-in mechanisms to ensure that the default process accounting file **pacct** does not become too large. The two commands used for this purpose are: **turnacct switch** and **ckpacct**.

---

**NOTE**

turnacct off **and** accton **work only on the default process accounting file,** pacct.

---

## ckpacct

The **ckpacct** command is normally invoked by **cron** every hour to ensure that the current process accounting file **pacct** has not become too large.

```
ckpacct [blocks]
```

If the size of **pacct** exceeds the **blocks** argument, 1000 by default if **blocks** is not specified, then **turnacct switch** is executed, which renames the current **pacct** file and creates a new **pacct** file.

---

**NOTE**

**If the amount of free space falls below a certain threshold,** ckpacct **will automatically turn off process accounting via** turnacct off. **When free space goes over the specified limit, process accounting will be reactivated. The kernel may also enforce a size limit on the size of** pacct. **This will take precedence over the limit set by** ckpacct. **See** acctsh(1M) **and** acct(2) **in the** *HP-UX Reference* **for more details.**

---

## turnacct switch

The **turnacct switch** command creates a new **pacct** file when the current **pacct** file is too large. The action of **turnacct switch** is summarized as follows:

1. Process accounting is temporarily turned off.
2. The current **pacct** file is renamed to **pacctincr**, where **incr** is a number that starts at **1** and increments by one for each **pacct** file created via **turnacct switch**.
3. Because the old **pacct** file was renamed, a new, current **pacct** file is created.
4. Process accounting is restarted; the kernel starts writing records to the new **pacct** file.

The following example illustrates the effect of the **turnacct switch** command when it is executed from the **adm** home directory **/usr/adm**. Comment lines begin with a pound sign(#) and are included in the example only as explanatory material.

```
$ #
$ # First, list all the process accounting files
$ # (at this point, there is only one).
$ #
$ ll pacct*
-rw-rw-r-- 1 adm          adm      2196     Mar 21 12:44 pacct
$ #
$ # Now execute turnacct switch, which will rename the current
$ # pacct file to pacctl and will create a new pacct file.
$ #
$   turnacct switch
$ #
$ # Now verify this by listing all process accounting
$ # files again.
$ #
$ ll pacct*
-rw-rw-r-- 1 adm          adm        72     Mar 21 12:46 pacct
-rw-rw-r-- 1 adm          adm      2196     Mar 21 12:44 pacctl
$ #
$ # The current process accounting file is pacct.  The previous
$ # process accounting file is now named pacctl.
$ #
```

## Displaying Process Accounting Records

The **acctcom** command displays records from any file containing process accounting records and normally these would be from the **pacct** files (**pacct, pacctl, pacct2, ...**).

```
      acctcom [[options][file]] ...
```

If no **file** is specified, **acctcom** defaults to the current **pacct** file. Input may also be taken from standard input. Some of **acctcom**'s options allow you to select specific records; other options control the format of the report.

The information contained in this section is organized as follows:

- Definitions for the **acctcom** columnar data.
- Command options that control report format.
- Options that allow selection of particular records.
- Sample **acctcom** reports to help understand **acctcom** options.

## Definitions of Information Produced by acctcom

The **acctcom** command generates a columnar report with descriptive headings. Each line of the report contains execution statistics that a particular process accumulated during its lifetime. The following table defines the standard columns in the report, that is, the columns that are displayed when none of **acctcom**'s options are specified.

| Column Header | Definition |
|---|---|
| COMMAND NAME | Command or program that spawned the process. Whenever you enter a command, you are spawning a process. For example, if you enter the command `$ ll /usr/lib/acct` you create a process with the command **ll**. If a command requiring superuser privileges is executed, a number sign (#) appears before the command name. |
| USER | Login name of the user who created the process. |
| TTYNAME | Terminal from which the process was executed. If the process was not executed from a known terminal (for example, if it was executed via **cron**), then a question mark(?) appears in this column. |
| START TIME | Time (**hh:mm:**ss) that the process began executing. |
| END TIME | Time (**hh**:**mm**:ss) that the process finished executing. |
| REAL (SECS) | Number of elapsed seconds from **START TIME** to **END TIME**. |
| CPU (SECS) | CPU processing time used during execution. |
| MEAN SIZE(K) | Estimate (in kilobytes) of memory usage during execution. This estimate is determined by the current process's memory usage at each system clock interrupt. It is, therefore, subject to statistical sampling errors. Only the memory resident pages of a process are counted; no pages in the swap space are counted. Shared code and data is divided among the processes using it. The size is divided by the number of processes sharing the code or data. |

The following table defines columns that are not displayed on the standard report, but which may be displayed using **acctcom** options.

| Column Header | Definition |
|---|---|
| F | For a process created by **fork** which does not do an **exec**, this column takes the value **1**; commands which require superuser privileges show a **2**; superuser commands which do a **fork** without an **exec** show a **3**; otherwise, this column shows a **0**. |
| STAT | Displays system exit status. (This is **not** the status returned by **exit** to a parent process during **wait**). Normally when a process terminates, this field is **0**. If a command terminates abnormally, then a value other than zero is shown. For example, if you interrupt a command with the **DEL** key, this column contains a **2**. |
| HOG FACTOR | The hog factor is computed as **CPU** time divided by **REAL** time and it provides a relative measure of the available CPU time used by the process during its execution. For example, a hog factor of less than 0.50 indicates that the process spent less than half of its time using the CPU. A hog factor of 0.75 indicates that a process spent 75% of its time using the CPU. |
| KCORE MIN | Provides a combined measurement of the amount of memory used (in kilobytes) and the length of time it was used (in minutes). It is computed as follows:<br><br>KCORE MIN = CPU (SECS) * MEAN SIZE(K) / 60 |
| CPU SYS | Total CPU time spent executing operating system code, such as system calls writing to disk. |
| USER (SECS) | User CPU time spent executing a process's code. |
| CPU FACTOR | Total CPU time is comprised of **system** and **user** CPU time:<br><br>CPU (SECS) = CPU SYS + USER (SECS)<br><br>The CPU factor shows the ratio of user CPU time to total CPU time:<br><br>CPU FACTOR = USER (SECS) / (CPU SYS + USER (SECS))<br><br>For example, if a command has a CPU factor of 0.35, that means that the CPU spent 35% of its time executing user code and 65% performing system functions. |
| CHARS TRNSFD | Number of characters (bytes) read and/or written by the command. |
| BLOCKS R/W | Number of file system blocks read and/or written as a result of executing this command. This number is not directly related to CHARS TRNSFD and may vary each time the command is |

executed, because **BLOCKS R/W** is affected by directory searches made before opening files, other processes accessing the same files, and general file system activity.

## Report Format Options

When no report format options are specified, **acctcom** produces a report containing only default information as described previously in "Definitions of Information Produced by acctcom." Optional information may be displayed with the following:

| Option | Description |
|---|---|
| **-a** | Displays the following average statistics at the end of the report:<br>■ Total number of commands processed (**cmds=xxx**)<br>■ Average real time per process (**Real=x.xx**)<br>■ Average CPU time per process (**CPU=x.xx**)<br>■ Average USER CPU time per process (**USER=x.xx**)<br>■ Average SYS CPU time per process (**SYS=x.xx**)<br>■ Average characters transferred (**CHARS=x.xx**)<br>■ Average blocks transferred (**BLK=x.xx**)<br>■ Average CPU factor (**USR/TOT=x.xx**)<br>■ Average HOG factor (**HOG=x.xx**) |
| **-b** | Displays process records starting with the most recently executed command. |
| **-f** | Prints the **fork/exec** flag (**F** column) and process exit status (**STAT** column) on the report. |
| **-h** | Displays the optional **HOG FACTOR** column, instead of the standard mean memory size column **MEAN SIZE(K)**. |
| **-i** | The optional I/O counts **CHARS TRNSFD** and **BLOCKS R/W** replace the standard **MEAN SIZE(K)** column in the report. |
| **-k** | Replaces the standard **MEAN SIZE(K)** column with **KCORE MIN**. |

| | |
|---|---|
| **-m** | Shows the default column **MEAN SIZE(K)** on the report. This option is used to include **MEAN SIZE(K)** when it was bumped by another option. For example, |

```
$ acctcom -km
```

produces a report showing both **KCORE MIN** and **MEAN SIZE(K)**.

| | |
|---|---|
| **-r** | Includes the optional **CPU FACTOR** column in the report. |
| **-t** | Shows separate system and user CPU times (**CPU SYS** and **USER (SECS)** respectively). |
| **-v** | Suppresses column headings at the top of the report. |
| **-q** | This option is the same as the **-a** option, except that individual process accounting records are not displayed, only the averages are displayed. |
| **-o ofile** | Copy the input process accounting records to **ofile**. |

## Record Selection Options

The options described here allow you to select the records included in the report produced by **acctcom**. The table shown below defines and provides examples for each option:

| Option | Description |
|---|---|
| **-l line** | Display only processes that were executed from the user terminal **/dev/line**. For example, |

```
$ acctcom console
```

displays records only for the processes that were created from the terminal **console**.

**-u user** Show only the processes belonging to **user**. **user** can be any of the following:

- A user ID (for example, **acctcom -u 355**)
- User name (**acctcom -u horatio**)
- A number sign # (**acctcom -u#**)
- A question mark ? (**acctcom -u?**)

If # is specified as the user name, then **acctcom** displays only the commands that require superuser privileges. If ? is given as the user, then only the processes with unknown process IDs are displayed. Remember to escape the # and ? characters (using a backslash), because they have meaning to the shell.

As an example, the following two commands are equivalent:

```
$ acctcom -u 0
$ acctcom -u root
```

**-g group** Show only processes belonging to **group**. **group** may be specified as either a group name or group ID. For example, suppose the group **pseudo** with group ID **300** is defined in **/etc/group**, the following two commands are equivalent:

```
$ acctcom -g 300
$ acctcom -g pseudo
```

**-s time** Select processes existing at or after **time**. Time is given in 24-hour format

**hr**[:**min**[:**sec**]].

The following example would display all the processes that existed at or after 3:30 pm:

```
$ acctcom -s 15:30
```

**-e time** Select processes that existed at or before **time**. Time is supplied in 24-hour format

**hr**[:**min**[:**sec**]]

The next example would display all the processes that existed between midnight and 12:15 am:

```
$ acctcom -e 0:15
```

**-S time** Select processes **starting** at or after **time** where **time** is in 24-hour format. The following example displays all the processes that **started** at 1:30:42 pm or after:

```
$ acctcom -S 13:30:4
```

**-E time** Display only processes that **terminated** at or before **time**, where **time** is in 24-hour format

**hr**:[**min**[:**sec**]].

Note that both the **-S** and **-E** options with the same **time** argument displays only processes that existed at the specified **time**. For example, to see all the processes that existed at exactly 30 minutes past noon, enter:

```
$ acctcom -S 12:30 -E 12:30
```

**-n pattern** Show only commands matching **pattern**. **pattern** may be a regular expression as described in **ed**, except that **+** means one or more occurrences. For example, to display all processes created by executing **ls**, enter:

```
$ acctcom -n ls
```

To display all the commands that start with **acct**, enter:

```
$ acctcom -n acct
```

To see all the commands that contain the letter **m** in their spelling, enter:

```
$ acctcom -n "*m*"
```

**-H factor** Display only processes whose hog factor exceeds **factor**. For example,

```
$ acctcom -H 0.85
```

displays all processes that spent over 85% of their execution time in the CPU. You can use this option to find processes that are "hogging" the CPU.

**-O time** Show only processes whose system CPU time exceeds **time**, specified in seconds. The following example determines which processes took more than 8.25 seconds of CPU time to execute:

```
$ acctcom -O 8.25
```

Use this option to determine which processes are making heavy use of the operating system calls.

**-C sec** Show only processes whose total CPU time (**SYS** + **USER**) exceeds **sec** seconds. The next example displays all the processes that used over 5.28 seconds of CPU time to execute:

```
$ acctcom -C 5.28
```

**-I chars** Display only processes transferring more characters than the limit given by **chars**. For example,

```
$ acctcom -I 10240
```

displays all the processes that transferred greater than 10 kilobytes of characters (10240 = 10 X 1024 bytes).

## Sample Reports

The following sample report illustrates the use of **acctcom** without options.

```
$ acctcom
ACCOUNTING RECORDS FROM:  Thu Mar 21 12:52:26 1985
COMMAND                                     START     END        REAL     CPU      MEAN
NAME              USER        TTYNAME       TIME      TIME       (SECS)   (SECS)   SIZE(K)

#accton           root        console       12:52:26  12:52:26   0.12     0.10     19.00
ls                sarah       tty07         14:04:08  14:04:08   0.28     0.23     16.50
ckpacct           adm         ?             14:30:00  14:30:05   5.13     1.45     24.00
pwd               bill        tty10         15:09:07  15:09:07   0.48     0.22     22.50
find              sarah       tty07         18:51:37  18:51:39   2.73     0.15     26.50
tabs              root        console       19:10:18  19:10:18   0.92     0.13     23.50
stty              root        console       19:10:19  19:10:19   0.88     0.08     26.00
mail              bill        tty10         19:10:21  19:10:22   1.78     0.23     28.50
news              root        console       19:10:23  19:10:23   0.73     0.12     23.00
acctcom           adm         ttya0         19:53:16  19:53:38  22.58     2.55     28.50
```

Display all the processes created between 7:00 PM and 7:30 PM by the **root** user. In addition, include the optional CPU factor and average statistics in the output.

```
$ acctcom -S 19:00 -E 19:30 -u root -ah
START AFT: Thu Mar 21 19:00:00 1985
END BEFOR: Thu Mar 21 19:30:00 1985
COMMAND                                     START     END        REAL     CPU      HOG
NAME              USER        TTYNAME       TIME      TIME       (SECS)   (SECS)   FACTOR

tabs              root        console       19:10:18  19:10:18   0.92     0.13     0.14
stty              root        console       19:10:19  19:10:19   0.88     0.08     0.09
news              root        console       19:10:23  19:10:23   0.73     0.12     0.16
cmds=3  Real=0.84  CPU=0.11  USER=0.02  SYS=0.09  CHAR=26.12 BLK=11.50
USR/TOT=0.19  HOG=0.13
```

Sample reports are helpful, but the best way to learn the various **acctcom** options is to use them. Take a few minutes and experiment with this command; it is very powerful and can provide you with much useful information if used properly.

## Command Summary Report

The **acctcms** command takes process accounting records as input but instead of reporting on the individual processes, **acctcms** generates a report on commands that generated the process accounting records. The action of **acctcms** can be summarized as follows:

1. Command **acctcms** looks through the input process accounting records and accumulates execution statistics for each unique command name. This information is stored in internal summary format one record per command name.
2. Depending on **acctcms** options, the command summary records created in step 1 are sorted.
3. The command summary records are written to standard output in the internal summary format mentioned in step 1. (To get an ASCII, readable report of this information, you would use the **-a** option described later.)

The syntax of the **acctcms** command is:

```
acctcms [options] files
```

where **files** is a list of input process accounting files for which the command summary report is to be generated.

## Producing a Readable Report

By default, the output of **acctcms** is in internal summary record format. If displayed to the terminal, it will not be readable. To get a readable report, use the **-a** option.

The **-a** option produces a report with descriptive column headings. Total and average (mean) execution statistics for each command are displayed, one line per command .along with total and average statistics over all commands in the report. Descriptions of the columnar data produced by **acctcms** follow:

| **Column Header** | **Description** |
|---|---|
| COMMAND NAME | Name of the command for which execution statistics are summarized. Unfortunately, all shell scripts are lumped together under the name **sh**, because only object modules are reported by the process accounting system. |
| NUMBER CMDS | Total number of times that the command was invoked. |
| TOTAL KCOREMIN | Total of kcore minutes accumulated for the command. (See the "Definitions of Information Produced by acctcom" for a more accurate description of kcore minutes.) |
| TOTAL CPU-MIN | Total CPU time the named command has accumulated. |
| TOTAL REAL-MIN | Total accumulated real-time seconds. |
| MEAN SIZE-K | Average of memory (in kilobytes) consumed by the command. |
| MEAN CPU-MIN | Average CPU time consumed per command invocation. The following equation shows how it is computed: MEAN CPU-MIN = TOTAL CPU-MIN / NUMBER CMDS |
| HOG FACTOR | Average hog factor over all invocations of the command. It is computed as: HOG FACTOR = TOTAL CPU-MIN / TOTAL REAL-MIN |
| CHARS TRNSFD | Total number of characters transferred by the command. Note that this number may sometimes be negative. |
| BLOCKS READ | Total count of the physical blocks read and written by the command. (See the section "Displaying Process Accounting Records acctcom" for details on the significance of this total.) |

**NOTE**

**When only the -a option is specified, the report is sorted in descending order on the TOTAL KCOREMIN column: commands using more TOTAL KCOREMIN are shown before those using fewer TOTAL KCOREMIN. This report gives a relative measure of the amount of memory used over time by the various commands: commands toward the start of the report are making more use of memory resources than are commands toward the end of the report.**

## Other Options

In addition to the **-a** option, you may use several other options to control the format of the report generated by **acctcms**. Some options specify which field to sort the report on; other options control the printing of prime/non-prime time usage. The following table defines these options and illustrates their use.

| Option | Description |
|---|---|
| **-c** | Sort the commands in descending order on **TOTAL CPU-MIN**, rather than the default **TOTAL KCOREMIN**. You may use this report to determine which commands are consuming the most CPU time. |
| **-n** | Sorts the report in descending order on the column named **NUMBER CMDS**. Commands toward the start of this report are the ones used most frequently; commands toward the end are used least often. |
| **-j** | All commands invoked only once are combined on one line of the report; this line is denoted by having "*****other**" in the **COMMAND NAME** column. This option is useful for shortening a report that has many one-invocation commands. |
| **-o** | Used only with the **-a option**, **-o** causes the report to be generated only for commands that were executed during non-prime time (as specified in the **holidays** file). You can use this option to get a non-prime time command summary report. |
| **-p** | Also used only with the **-a option**, **-p** elicits a report only for commands that were executed during prime time (as specified in **holidays**). This option is used to get a prime time command summary report. |
| **-apo** | When the options **-o** and **-p** are combined with **-a**, a combination prime and non-prime time report is produced. The output of this report is the same as that produced by **-a** alone, except that the **NUMBER CMDS**, **TOTAL CPU-MIN**, and **TOTAL REAL-MIN columns are divided into** |

**two columns, one for prime time totals, the other for non-prime time. (Prime time columns have a (P)** header, while non-prime time columns are headed by **(NP)**.)

-s Specifies that any named input files following the -s on the command line are already in internal summary format. This option is useful for merging previous **acctcms** reports with current reports. The following example uses -s to create a command summary report from previous process accounting files (named by the users) and the current process accounting file. The final ASCII report is stored in the file **asciisummary**.

```
$ acctcms pacct? > oldsummary
$ acctcms pacct > newsummary $ acctcms -as oldsummary
newsummary > asciisummary
```

## Sample Report

The ASCII reports produced by **acctcms** contain more than 80 characters per line and when displayed on an 80-column display, the lines wrap around on the screen. In addition, if the report is printed on an 80-column printer, some of the rightmost columns will be lost. Therefore, be sure to use either:

1. a printer with compressed print, so that all of the report will fit on standard computer paper; or

2. a printer with enough columns to display all the information such as a 132-column printer.

The following example generates a command summary report for the current process accounting file (no file is specified, so the current **pacct** file is assumed). By giving the **-j** option, all the commands that were executed only once are grouped under the command name *****other**. Note also that total execution statistics for all commands are grouped under the command name **TOTALS**.

```
      $ acctcms -aj

                            TOTAL  COMMAND  SUMMARY
CMD       NO.   TOTAL       TOTAL   TOTAL    MEAN     MEAN    HOG      CHARS    BLOCKS
NAME      CMDS  KCORE-      CPU-    REAL-    SIZE-K   CPU-    FACTOR   TRNSFD   READ
                MIN         MIN     MIN               MIN

TOTALS    61    17.63       0.38    164.49   46.25    0.01    0.00     104553   1027
acctcms   17    12.13       0.16      0.35   76.72    0.01    0.45      49192    306
sh         8     2.43       0.09    152.86   26.79    0.01    0.00       9043    163
more       3     0.73       0.02     10.50   31.00    0.01    0.00      21618     83
ll         6     0.61       0.04      0.11   16.50    0.01    0.33       5715     95
acctcom    4     0.58       0.02      0.07   28.50    0.01    0.30      15319     42
***other   9     0.54       0.02      0.14   25.26    0.00    0.16        459    161
cat        4     0.19       0.01      0.35   22.97    0.00    0.02       3112     52
rm         2     0.11       0.00      0.02   22.22    0.00    0.29          0     29
chmod      2     0.10       0.00      0.01   22.00    0.00    0.35          0     15
accton     2     0.08       0.00      0.02   19.00    0.00    0.29          0     22
sed        2     0.08       0.01      0.04   14.50    0.00    0.13         73     38
echo       2     0.05       0.00      0.02   20.00    0.00    0.16         22     21
```

## Creating Total Accounting Records

The **acctprc1** and **acctprc2** commands create total accounting records from process accounting files. The output of **acctprc1** becomes input to **acctprc2** which produces the total accounting records. These commands are normally invoked by **runacct** to produce daily accounting information.

### acctprc1

The **acctprc1** command reads process accounting records from standard input, adds login names corresponding to the user ID of each record, and then writes for each process an ASCII line showing:

- The ID of the user that created the process
- The user's login name
- Prime CPU time in ticks (a "tick" is one hundredth of a second)
- Non-prime CPU time, also in ticks
- Mean memory size (in pages)

The format of **acctprc1** is:

```
    acctprc1 [ctmp]
```

Input must be redirected from a process accounting file.

The following example creates a file, **asciiptacct**, containing ASCII process accounting information that can be used to create process total accounting records. This file is created from the current process accounting file **pacct**.

```
    $ acctprc1 <pacct >asciiptacct
```

Normally, **acctprc1** gets log-in names from the password file **/etc/passwd**, which is sufficient on systems where each user has a unique user ID. However, on systems where different users share the same user ID, the **ctmp** file should be specified; it helps **acctprc1** distinguish different log-in names that share the same user ID.

When specified, **ctmp** is expected to contain a list of login sessions of the form created by **acctcon1**, sorted by user ID and log-in name.

### acctprc2

The **acctprc2** command reads from standard input records of the form created by **acctprc1**, summarizes the records by user ID and name, and writes the sorted summaries to standard output as total accounting records. The following example creates total accounting records for all processes in the current process accounting file **pacct**; the total accounting records are stored in the file **ptacct**.

```
    $ acctprc1 <pacct cctprc2 >ptacct
```

## Charging Fees to Users

System Accounting provides the capability to charge users on the system with the **chargefee** command. **chargefee** allows you to charge generic **units** to a specific log-in name. The syntax of this command is:

```
chargefee loginname number
```

where **number** is the number of units to be charged to a particular user, and **loginname** is the log-in name of the user to whom **number** units are to be charged.

**NOTE**

number **can be any whole number from -32768 to 32767; when charging fees, keep in mind that the sum of each user's fees must also be within this range.** chargefee **accumulates fee charge records in the file** /usr/adm/fee. **These records are then merged with other accounting records via** runacct.

Examples:

The following example charges 25 units to the user whose login name is **horatio**:

```
$ chargefee horatio 25
```

If you mistakenly charged 247 units to the user **horatio**, you can credit the account by entering a negative amount:

```
$ chargefee horatio -247
```

If your site has a fixed, one-time installation fee of 25 units and **horatio** is a new user, you may use **chargefee** to charge **horatio** for the 25 units.

## Accounting Summary and Report

This final group of commands summarizes and reports accounting information. Certain commands display and merge total accounting files, while others generate the daily and monthly reports used to analyze system performance and bill users for resource usage. The following commands are discussed in this section:

- **prtacct**--displays total accounting records.
- **acctmerg**--merges total accounting files.
- **runacct**--generates daily summary files and reports.

- **prdaily**--displays the daily summary files and reports created by **runacct**.
- **monacct**--creates monthly summary files and reports.

## Displaying Total Accounting Records

The **prtacct** command displays the contents of a process accounting file. Its format is

```
     prtacct file "heading"
```

where:

| | |
|---|---|
| **file** | is the name of the total accounting file to be displayed |
| **"heading"** | is a comment to be included in the standard report header produced by **prtacct** |

The format of the **prtacct** report is described next and is followed by an example.

### Report Format

The **prtacct** command produces a columnar report with one line per total accounting record. Descriptive column headings are included in the report. Definitions of each column follow:

| Column Header | Description |
|---|---|
| UID | ID of the user for whom the total accounting record was created. |
| LOGIN NAME | Login name of the owner of the total accounting record. |
| CPU (MINS) | Total CPU time (in minutes) that the user has consumed. This column is divided into prime and non-prime columns (PRIME and NPRIME respectively). Information in these columns is created through process accounting commands. |
| KCORE-MINS | Cumulative measure of memory and CPU time a user has consumed (see "Definitions of Information Produced by acctcom" for a more precise definition). Information in this column is also divided into PRIME and NPRIME columns and is created through process accounting commands. |
| CONNECT (MINS) | Identifies real time used (in minutes). In essence, what this column identifies is the amount of time that the user was logged in to the system and is also divided into PRIME and NPRIME columns. The connect session accounting commands are the source of this information. |
| DISK BLOCKS | Total number of disk blocks allocated to the user and is created via disk space accounting commands. |
| # OF PROCS | Total number of processes spawned by the user and is created via the process accounting commands. |

| | |
|---|---|
| # OF SESS | Total number of times the user logged in. Connect session accounting commands create this data. |
| # DISK SAMPLES | How many times disk accounting was run to obtain average number of disk blocks listed in the **DISK BLOCKS** column. |
| FEE | Number of fee units charged via **chargefee**. |

Example:

The following example displays disk total accounting records. First the total accounting records are created via disk space accounting commands; then they are displayed using **prtacct**. When examining this report, take note of the following:

1. The similarities between this and the sample report produced by **diskusg** (see "Disk Space Usage Accounting").

2. Only the columns relating to disk space usage have nonzero values, because the total accounting records were created only from disk space usage accounting commands.

```
   $ while read rdev filesystem jnk
   > do
   > diskusg $filesystem >dtmp.`basename $filesystem`
   > done < /etc/checklist

   Mar 26 17:01 1985  disc TOTAL ACCOUNTING RECORDS Page 1

    LOGIN   CPU   (MINS)  KCORE   MINS  CONNECT (MINS)  DISK    # OF  # OF #DISK  FEE
UID NAME   PRIME  NPRIME  PRIME  NPRIME  PRIME  NPRIME  BLOCKS  PROCS SESS  SAM-
                                                                           PLES
0   TOTAL   0      0       0       0      0       0     11598    0     0    10     0
0   root    0      0       0       0      0       0     10616    0     0     1     0
1   bin     0      0       0       0      0       0       778    0     0     1     0
4   adm     0      0       0       0      0       0        96    0     0     1     0
350 fred    0      0       0       0      0       0        14    0     0     1     0
351 bill    0      0       0       0      0       0        32    0     0     1
352 mike    0      0       0       0      0       0        20    0     0     1     0
353 sarah   0      0       0       0      0       0        16    0     0     1     0
354 molly   0      0       0       0      0       0        22    0     0     1     0
355 horatio 0      0       0       0      0       0         2    0     0     1     0
501 guest   0      0       0       0      0       0         2    0     0     1     0
```

## Merging Total Accounting Files

Normally executed by **runacct**, the **acctmerg** command merges separate total accounting files into a single file. All the total accounting records for a particular user name and ID are merged together to form one record. This command is useful for merging disk, connect session, and process total accounting files together to form a single, comprehensive total accounting file.

The **acctmerg** command reads standard input and up to nine additional files, all in total accounting record format (or an ASCII version thereof). Its syntax is

```
    acctmerg [options] [file] ...
```

where:

| | |
|---|---|
| **options** | control the report format and the manner in which records are merged. |
| **file** | is one of up to nine files (in addition to standard input) that are to be merged into a single total accounting file, written to standard output. |

### acctmerg Options

The following options may be used with **acctmerg** to control the report format and the manner in which the total accounting records are merged:

**Option** **Description**

**-a** **acctmerg** normally produces output as total accounting records. The **-a** option produces output in ASCII. Note that the output generated by using this option is the same as the report produced by prtacct, except that no report headings or totals are displayed--only the columnar data is shown.

**-i** In default, **acctmerg** assumes that its input files contain total accounting records. If **-i** is specified, then **acctmerg** expects input files to be in the ASCII format created by the **-a** option.

**-p** This option simply echoes input records--no merging or processing is done. The output is displayed in the format produced by the **-a** option.

**-t** Produces a single total accounting record that summarizes all input records. To see the ASCII version of this record, you must use the **-t** and **-a** options together:

```
$ acctmerg -t -a <totacctrecs>
```

Note that **-t** and **-a** may be specified in any order, but they must be specified separately as shown.

**-u** Normally, **acctmerg** merges records that have the same user ID and user name. **-u** merges records based on user ID only (and disregards the user name).

**-v** This option produces output in verbose ASCII format. The same report is produced as the **-a** option, except that floating point numbers are displayed in more precise notation:

```
<mantissa>e<exponent>
```

The **-a**, **-v**, and **-i** options are useful if you wish to edit total accounting records. For example, suppose that you have created a total accounting file (**ptacct**) containing process total accounting records, and you want to make some adjustments to these records. The following sequence could be used to make "repairs" to this file:

```
$ acctmerg -v -a <ptacct >ptacct.ascii
edit ptacct.ascii as desired ...
then copy the changes back to ptacct
$ acctmerg -i <ptacct.ascii >ptacct
```

Example:

The following example creates disk, process, and connect session total accounting records, merges them together, and stores the merged file in the file **mergedfile**.

```
$ # First, create disc space usage total accounting records (dtacct)...
$ #
$ while read rdev fs junk
> do
> diskusg $fs >dtmp.`basename $fs`
> done < /etc/checklist
$ diskusg -s dtmp.* |sort +0n +1 |acctdisk >dtacct
$ #
$ # Now create connect session total accounting records (ctacct)...
$ #
$ acctcon1 </etc/wtmp |acctcon2 >ctacct
$ #
$ # Create process total accounting records (ptacct)...
$ #
$ >ptacct
$ for pfile in pacct*
> do
> acctprc1 <$pfile |acctprc2 >>ptacct
> done
$ #
$ # Now merge all the total accounting files (?tacct) into
$ # a single total accounting file (tacct)...
$ #
$ acctmerg dtacct ctacct <ptacct >tacct
```

## Creating Daily Accounting Information

**runacct** is the main daily accounting shell procedure. It is normally initiated via **cron** during non-prime hours. **runacct** processes disk, connect session, process, and fee accounting files. It prepares cumulative summary files for use by **prdaily** and/or for billing purposes. This section discusses the following aspects of **runacct**:

- Files processed by **runacct**.
- The states that **runacct** progresses through while executing.

- Recovery from **runacct** failure.
- Restarting **runacct**.
- Reports produced by **runacct**.

## Files Processed by runacct

The following files, processed by **runacct**, are of particular interest. Filenames are given relative to the directory **/usr/adm/acct**.

- **nite/lineuse** contains usage statistics for each terminal line on the system. This report is especially useful for detecting bad lines. If the ratio of logoffs to logins on a particular line exceeds 3 to 1, then there is a good possibility that the line is failing.
- **nite/daytacct** contains total accounting records from the previous day.
- **sum/tacct** contains accumulated total accounting records for each day's total accounting records (**nite/daytacct**) and can be used for billing. It is restarted each month or fiscal period by the **monacct** shell script.
- **sum/daycms** is produced by **acctcms**. It contains the daily command summary. The ASCII version of this file is in **nite/daycms**.
- **sum/cms** holds the accumulation of each day's command summaries (**sum/daycms**). A new **sum/cms** file is created each month by **monacct**. The ASCII version of this file is in **nite/cms**.
- **sum/loginlog** maintains a record of the last time each user logged in.
- **sum/rprtMMDD** is the main daily accounting report created by **runacct**. This report can be printed via **prdaily**.

The **runacct** command takes care not to damage files in the event of errors. A series of protection mechanisms are used that attempt to recognize errors, provide intelligent diagnostics, and terminate processing in such a way that **runacct** can be restarted with minimal intervention. To accomplish these goals, the following actions are performed by **runacct**:

- **runacct**'s progress is recorded by writing descriptive messages to the file **nite/active**.
- All diagnostics output during the execution of **runacct** are redirected to the file **nite/fd2log**.
- If the files **lock** and **lock1** exist when **runacct** is invoked, an error message will be displayed and execution will terminate.
- The **lastdate** file contains the month and day that **runacct** was last run and is used to prevent more than one execution per day.
- If **runacct** detects an error, it writes a message to **/dev/console**, sends mail to **root** and **adm**, removes locks, saves diagnostics files, and terminates execution.

## The States of runacct

To allow **runacct** to be restartable, processing is broken down into separate re-entrant **states**. As **runacct** executes, it records its progress by writing the name of the most recently completed state into the file called **/usr/adm/statefile**. After processing for a state is complete, **runacct** examines **statefile** to determine which state to enter next. When **runacct** reaches the final state (CLEANUP), the lock and lock1 files are removed, and execution terminates. The following table describes **runacct**'s states:

| **State** | **Action** |
|---|---|
| SETUP | **turnacct switch** is executed. The process accounting files, **pacct?**, are moved to **Spacct?.MMDD**. The **/etc/wtmp** file is moved to **nite/wtmp.MMDD** with the current time added on the end. |
| WTMPFIX | **nite/wtmp.MMDD** is checked for correctness by **wtmpfix**. Some date changes will cause **acctcon1** to fail, so **wtmpfix** attempts to adjust the time stamps in the **nite/wtmp.MMDD** file if a date change record appears. |
| CONNECT1 | Connect session records are written to **ctmp**. The **lineuse** file is created, and the **reboots** file, showing all of the boot records found in **nite/wtmp.MMDD**, is created. |
| CONNECT2 | **ctmp** is converted to connect session total accounting records in the file **ctacct.MMDD**. |
| PROCESS | The **acctprc1** and **acctprc2** programs are used to convert the process accounting files, **Spacct?.MMDD**, to the 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; if you don't, runacct will not finish.** |
| MERGE | Merge the process and connect session total accounting records to form **nite/daytacct**. |
| FEES | Merge in any ASCII **tacct** records from the file **fee** into **nite/daytacct**. |
| DISK | On the day after the **dodisk** shell script runs, merge **nite/disktacct** with **nite/daytacct**. |
| MERGETACCT | Merge **nite/daytacct** with **sum/tacct**, the cumulative total accounting file. Each day, **nite/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 summary file **sum/cms**. Produce ASCII and internal format command summary files. |

| | |
|---|---|
| USEREXIT | Any installation-dependent (local) accounting programs can be run in this state. For example, you might want to execute commands that generate daily billing data for individual users (the shell script **acctbill** in the section "Sample System Accounting Shell Scripts" could be used for this purpose). To have local accounting programs executed by **runacct**, simply enter the commands in runacct in the code for the **USEREXIT** state of **runacct**. |
| CLEANUP | Clean up the temporary files, run **prdaily** and save its output in the file **sum/rprtMMDD**, remove the locks, then exit. |

## Recovering from Failure

It is possible that **runacct** might fail and terminate abnormally. The primary reasons for **runacct** failure are:

- A system "crash".
- Not enough disk space remaining in **/usr**
- A corrupted **wtmp** file.

If the **nite/activeMMDD** file exists, check it first for error messages. If the **nite/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 be restarted.

**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** before restarting **runacct**.

**ERROR: turnacct switch returned rc=?**

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

**ERROR: Spacct?.MMDD already exists**

File **setups** probably has run. Check the status of files, then run **setups** manually.

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

You must perform the SETUP step manually, because the daily **wtmp** file already exists.

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

**wtmpfix** detected a corrupted **wtmp** file. See the section "Fixing Corrupted Files" for details on fixing **wtmp** errors.

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

**acctcon1** encountered a bad **wtmp** file. Again, see the section "Fixing Corrupted Files" on how to fix the 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** is normally run via **cron** only once per day. However, if an error occurs while executing **runacct** (as described above), it may be necessary to restart **runacct**. **runacct** has the following syntax:

```
runacct [ mmdd [ state ]]
```

When called without arguments, **runacct** assumes that it is being invoked for the first time on the current day; this is how **runacct** is invoked by **cron**. The argument **mmdd** is necessary if **runacct** is 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 entry **state** on the command line.

For example, to start **runacct**, you would enter:

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

To restart **runacct** on the 26th day of March:

```
$ nohup runacct 0326 2> /usr/adm/acct/nite/fd2log &
```

To restart runacct at state **WTMPFIX** on June 1st:

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

### Daily Reports

**runacct** generates five basic reports upon each invocation. Brief descriptions of each report follow. Detailed descriptions of the reports are found in the following section, "Displaying runacct Reports -- prdaily."

- **Daily Line Usage Report**--summarizes connect session accounting since the last invocation of **runacct**. It provides a log of system reboots, power failure recoveries, and any other records dumped into **/etc/wtmp** via **acctwtmp**. In addition, it provides a breakdown of line utilization.
- **Daily Resource Usage Report**--summarizes resource usage per individual and basically merges all the total accounting records and displays the records, one per user.
- **Daily Command Summary**--summarizes resource usage data for individual commands since the last invocation of **runacct**. The data included in this report is

useful in determining the most heavily used commands; you can use these commands' characteristics of resource utilization when "tuning" your system.

This report is sorted by TOTAL KCOREMIN, an arbitrary but often good measure of load on a system.

- **Monthly Total Command Summary**--This report is the same as the Daily Command Summary, except that the Daily Command Summary contains command summary information accumulated since the last invocation of **runacct**, while the Monthly Total Command Summary summarizes commands from the start of the accounting period to the current date. In other words, the monthly report reflects the data accumulated since the last invocation of **monacct**.
- **Last Login**--simply gives the date each user last logged in to the system. This could be a good source for finding likely candidates for the archives, or getting rid of unused login directories.

## Displaying runacct Reports

After **runacct** finishes executing, it stores a report of the current day's accounting in the file **/usr/adm/acct/sum/rptmmdd**, where **mmdd** is the month and day that the report was generated. The **prdaily** command is used to display the contents of any daily report file created by **runacct**. Its syntax is

```
    prdaily [-l] [-c] [ mmdd ]
```

where:

| | |
|---|---|
| **mmdd** | is an optional report date. If no date is specified, **prdaily** produces a report of the current day's accounting information. If you include a date, specify any reasonable (previous) date for which to report accounting information. |
| **-l** | prints a report of exceptional usage by login name for a specified date. This option determines which users are consuming excessive amounts of system resources. The limits for exceptional usage are kept in the file **/usr/lib/acct/ptelus.awk** and can be edited to reflect your installation's requirements. |
| **-c** | is valid only for the current day's accounting. The **-c** option produces a report of exceptional resource usage by command. This option determines which commands are using excessive amounts of system resources. The limits for exceptional usage are maintained in the file **/usr/lib/acct/ptecms.awk** and may be edited to reflect your system's needs. |

The reports produced by **runacct** were described briefly in the previous subsection. Now the reports are discussed in more detail.

### Daily Line Usage Report

In the first part of the daily line usage report, the **FROM/TO** banner should alert you to the period reported on. The times are the date-time that the last report was generated by runacct, and the date-time that the current report was generated. It is followed by a log of system reboots, shutdowns, power failure recoveries, and any other records dumped into **wtmp** by the **acctwtmp** command.

The second part of the report is a breakdown of line utilization. The **TOTAL DURATION** shows how long the system was in a multiuser state. The columns of the report are defined in the following table:

| Column | Description |
|---|---|
| LINE | The terminal line or access port being reported on. |
| MINUTES | The total number of minutes that the line was in use during the accounting period. |
| PERCENT | The percentage of **TOTAL DURATION** that the line was in use:<br><br>PERCENT = (MINUTES / TOTAL DURATION) * 100 |
| # SESS | Shows the number of times that this port was accessed for a **login** session. |
| # ON | Historically, this column displayed the number of times that the port was used to log a user on; but since **login** can no longer be executed explicitly to log in a new user, this column should be identical to **# SESS**. |
| # OFF | This column reflects not only the number of times a user logged off, but also any interrupts that occurred on the line. This column comes into play when **# OFF** exceeds **# ON** by a large factor. This usually indicates that the multiplexer modem cable is going bad, or that there is a bad connection somewhere. The most common cause of this is an unconnected cable dangling from the multiplexer. |

You should monitor **wtmp** because it is the file that connect session accounting is taken from. If it grows rapidly, execute **acctcon1** to determine which line is the noisiest. If the interruptions occur frequently, general system performance and real-time will be affected.

### Daily Resource Usage Report

This report gives a by-user breakdown of system resource usage. The format of this report is the same as that produced by the **prtacct** command. See the Report Format table for the **prtacct** command for definitions of the columnar data found in this report.

### Daily and Monthly Command Summary

These two reports are the same, except that the Daily Command Summary reports information only for commands executed since the last invocation of **runacct**; the Monthly Command Summary contains information on commands executed since the last invocation of **monacct**.

The output of this report is identical to that produced by **acctcms**. For definitions of the data found in this report, see the discussion of **acctcms** in the "Process Accounting" section.

### Last Login

This report simply shows the last date and time that each user logged in. The longer it has been since a particular user logged in, the more likely it is that the user's files could be archived, or maybe even that the user could be removed from the system.

## Creating Monthly Accounting Reports

Monthly accounting summary files and reports are created with the **monacct** command. The resulting output is stored in the directory **/usr/adm/acct/fiscal**. After creating its monthly reports, it removes the old daily accounting files from the directory **/usr/adm/acct/sum** and replaces them with new summary accounting files.

The **monacct** command should be invoked once each month or accounting period. Its syntax is

```
monacct number
```

where:

**number** indicates the accounting period (01=January, 12=December). If **number** is not specified, **monacct** assumes that it is being invoked for the current month; this default is useful if **monacct** is executed via **cron** on the first day of each month (as described in the "Daily Usage and Installation" section).

Descriptions of the files created in the **acct/fiscal** directory follow:

**cms*** contains the total command summary file for the accounting period denoted by *. The file is stored in internal summary format. Therefore, to display this file, use the **acctcms** command. The following example shows how to display this file for the month of June:

```
$ acctcms -a -s /usr/adm/acct/nite/fiscal/cms06
```

**fiscrpt*** contains a report similar to that produced by **prdaily**. The report shows line and resource usage for the month represented by *. The following would display the fiscal accounting file for the month of November:

```
$ cat /usr/adm/acct/nite/fiscal/fiscrpt11
```

**tacct*** is the total accounting file for the month represented by *. To display this file, use the **prtacct** command. The following would display the total accounting summary file for the month of January:

```
$ prtacct /usr/adm/acct/fiscal/tacct01 "JANUARY TOTAL ACCOUNTING"
```

## Updating the Holidays File

The file **/usr/lib/acct/holidays** contains information that System Accounting needs to distinguish between prime and non-prime time. It contains the following information:

1. **Comment Lines**. Comment lines are entered by placing an asterisk (*) as the first character in the line; they may appear anywhere in the file.

2. **Year Designation Line**. This line should be the first non-comment line in the file and must appear only once. It has the following format:

   ```
   year     PrimeTime    Non-PrimeTime
   ```

   The line consists of three four-digit numbers (leading blanks and tabs are ignored). The first number designates the year; the second denotes the time (in 24-hour format) that prime time starts; the third gives the time that prime time ends and non-prime time starts.

   For example, to specify the year as 1988, prime time at 9:00 a.m., and non-prime time at 4:30 p.m., the following entry would be appropriate:

   ```
   1988   0900   1630
   ```

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

3. **Company Holiday Lines**. These entries follow the year designation line. Company holidays are days when few people should be using the computer. Therefore, System Accounting assumes that non-prime time is in effect during the entire 24 hours of a specified holiday.

Company holiday lines have the following format:

```
day_of_year    Month    Day    Description of Holiday
```

The **dayofyear** field is a number in the range 1 through 366, corresponding to the day of the year for the particular holiday (leading blanks and tabs are ignored). The remaining fields are simply commentary and are not used by other programs.

---

**NOTE**

**As delivered, the** holidays **file contains valid entries for Hewlett-Packard's prime/non-prime time, and holidays. You should check this file and edit it as necessary to reflect your organization's requirements.**

---

## Fixing Corrupted Files

System Accounting files can become corrupted or lost. Some of these files may simply be ignored or restored from the file system backup. However, certain files must be fixed in order to maintain the integrity of System Accounting.

### Fixing wtmp Errors

The **wtmp** files seem to create most of the problems in daily operation of System Accounting. When the date is changed and HP-UX is switched into multiuser mode, a set of date change records is written into **/etc/wtmp**. The **wtmpfix** command 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 not be caught by **wtmpfix** and will cause **acctcon1** to fail. The following steps show how to "patch" a damaged **wtmp** file.

```
$ cd /usr/adm/acct/nite
$ fwtmp <wtmp.MMDD >wtmp.temp
    (Using an editor, delete corrupted records or          )
    (delete all records from beginning up to the date change)
$ fwtmp -ic <wtmp.temp >wtmp.MMDD
$ rm wtmp.temp
```

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 your installation is using System Accounting to charge users for system resource usage, the integrity of **sum/tacct** is important. If **sum/tacct** becomes corrupted, check the contents of **sum/tacctprev** with the command **prtacct**. If it appears correct, then the latest **sum/tacct.MMDD** should be patched up, and **sum/tacct** should then be recreated. A simple patch procedure would be:

```
$ cd /usr/adm/acct/sum
$ acctmerg -a -v <tacct.MMDD >tacct.temp
    (Using an editor, remove the bad records and)
     (write duplicate UID records to another file)
$ acctmerg -i <tacct.temp >tacct.MMDD
$ acctmerg tacctprev <tacct.MMDD >tacct
$ rm tacct.temp
```

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

## Sample Accounting Shell Scripts

The following are sample accounting shell scripts.

### grpdusg

This shell script displays disk space usage totals for the users who are members of a specified group. The syntax of this command is:

```
grpdusg groupname
```

where **groupname** is the name of the group for which disk space accounting information is to be generated.

For example,

```
$ grpdusg pseudo
```

generates disk space usage information for all the users in the **group, pseudo**. A listing of the shell script is shown below.

```
# Check for the group-name parameter.
#
if      [ $# -ne 1 ]
then    echo "Usage:  grpdusg group-name"
exit 1
fi
echo    "One moment please..."
#
# Use the find command to find all the files whose owners are members
# of group-name.  Pipe the output from find into acctdusg which will
# accumulate disc space usage information for the users in group-name.
# NOTE:
#       - accounting data is temporarily stored in _${1}_tmp
#       - error messages are stored temporarily in _${1}_err
#       - if files exist that have no owners, then the names of
#         these files are stored in _no_owners
#
fn=_${1}_
find / -group $1 -print 2>${fn}err |acctdusg -u _no_owners >${fn}tmp
#
# Remove the _no_owners file if its size is not greater than zero.
#
 if      [ -s _no_owners ]
 then    echo "Files having no owners exist--check _no_owners"
 else    rm _no_owners
 echo "All files have owners-- _no_owners not created"
 fi
#
# Use echo and awk to display disc usage totals for this group.
#
 echo "Disc space usage information (group is ${1}):"
 awk 'BEGIN {print "UID______USER NAME_______________BLOCKS"}
 { sum += $3 ;                  # add up total disc blocks used
 print $0 }                   # display information for user
 END { print "TOTAL DISC SPACE USAGE= ", sum, "blocks" }' ${fn}tmp
#
# Remove temporary files, then exit.
#
 rm ${fn}*
```

## acct_bill

The **acct_bill** command **acct_bill** takes as input a total accounting file and produces as output billing totals for all users found in the input file. The syntax of **acct_bill** is:

```
acct_bill [ mmdd ]
```

If the optional **mmdd** is not specified, then **acct_bill** takes as input the current day's total accounting file (**acct/nite/daytacct**); if **mmdd** is given, then input is taken from the total accounting file for the date specified by **mmdd** (**acct/sum/tacctmmdd**). Output is written to the file **billsmmdd**, where **mmdd** is the date given with the command, or the current date if **mmdd** was not specified with the command.

### Examples

To generate billing information for the current day, simply enter:

```
$ acct_bill
```

and the billing information will be stored in the file **acct/sum/billsmmdd**, where **mmdd** is the current date.

To create billing information for January 23rd, you would enter:

```
$ acct_bill 0123
```

after which the billing information would be stored in the file called **acct/sum/bills0123**.

To automatically generate daily billing totals for all users, you should call **acct_bill** without the date argument from the

```
USEREXIT state of runacct.
```

### Output Produced by acct_bill

The output of **acct_bill** contains one line per user and has the following format:

```
user_ID    user_name    billing_amount
```

where **user_ID** and **user_name** identify the user who is being billed, and **billing_amount** shows the total amount that the user is to be charged.

**billing_amount** is computed by multiplying **accounting coefficients** (found in the shell script) by columns of the report generated by **prtacct**. Assuming that billing amounts are in dollars, the coefficients (as they are currently shown) produce the following billing amounts:

- Ten cents for every minute of prime CPU time consumed.
- Five cents for every minute of non-prime CPU time consumed.
- One-half cent for every prime kcore minute used.
- Two-tenths of a cent for every non-prime kcore minute.
- One-half cent for every prime connect time minute.
- Two-tenths of a cent for every non-prime connect minute.

- Two-and-a-half cents for every block of disk space used.
- Two-and-a-half cents for every process spawned by the user.
- Ten cents for every connect session.
- Each fee unit charged via **chargefee** counts as one cent.

You should experiment with this command by altering the coefficients to see how **billing_amount** is affected. After gaining confidence with this shell script, you can alter the coefficients to suit your installation's needs. A listing of the shell script is shown below.

```
_date=`date +%m%d`
_outfile=/usr/adm/acct/sum/bills
_infile=/usr/adm/acct
#
# Set _infile and _outfile, based on whether or not MMDD was given
#
if      [ $# -eq 0 ]
then                    # Generate billing data for current day.
         _infile=${_infile}/nite/daytacct
         _outfile=${_outfile}${_date}
else                    # Create billing data for date given (MMDD).
         _infile=${_infile}/sum/tacct${1}
         _outfile=${_outfile}${1}
fi
#
# Create a file containing the ASCII equivalent of the input total
# accounting file (tacct_ASC.tmp_).  The file can then be supplied
# as input to awk, which will generate billing data for each user.
#
_tmpfile=tacct_ASC.tmp_
acctmerg -a -t <$_infile >$_tmpfile # output TOTAL amount first
acctmerg -a <$_infile >>$_tmpfile   # append users' total accounting record
#
#Using awk, compute billing totals for each user in the total accounting file
#
awk 'BEGIN {
          # ****************************************************
          #    A C C O U N T I N G   C O E F F I C I E N T S
          # ****************************************************
          cpu_P =0.10  # 0.10 monetary units per minute of prime CPU time
          cpu_NP=0.05  # 0.05 monetary units per non-prime CPU minute use
          kcm_P =0.005 # for prime kcore minutes consumed
          kcm_NP=0.002 # for non-prime kcore minutes used
          con_P =0.005 # prime connect (real) time
          con_NP=0.002 # non-prime connect time used
          blk =  0.025 # number of blocks used
          prc =  0.025 # number of processes spawned
          ses =  0.10  # number of connect sessions
          fee =  0.01  # 100 charge units per monetary unit
          # ****************************************************
     }
   # Start computing billing amounts for each user.
   { _sum= cpu_P*$3 + kcm_P*$5 + con_P*$7      # compute prime usage
     _sum+= cpu_NP*$4+ kcm_NP*$6+ con_NP*$8    # add non-prime usage
     _sum+= blk*$9 + prc*$10 + ses*$11 + fee*$13  # add remaining amounts
     printf "%-8s %-10s %10.3f", $1, $2, sum   # display user total
   }' $_tmpfile >$_outfile # write output from awk to appropriate file
rm $_tmpfile                # remove the temporary ASCII file
```

7

# System Reconfiguration

You can reconfigure, or change, various attributes of your HP-UX operating system. Some of the attributes you can modify are I/O hardware configurations, kernel device locations, internal table sizes, kernel software modules, and other system parameters. For example, additional space can be allocated for swapping if the default specification is insufficient for your applications. A special application can sometimes run more efficiently by tuning selected system parameters. A kernel software module can be included to make additional functions available to applications. These tasks are described in this chapter in "The System Reconfiguration Process" section. Also included in this chapter is detailed information on configuring swap space, controlling disk use, and creating a new file system.

**NOTE**

**You may also use SAM to configure the kernel using menus. Refer to** *System Administration Basics* **for more information.**

## System Reconfiguration Summary

Reconfigure the system by running the **uxgen** program to create a new HP-UX kernel. The **uxgen** program creates the kernel with information supplied in an input file, produces the appropriate files for your configuration, and executes **make** to build the new kernel.

The steps for using **uxgen** to generate the system are given below. However, before generating a system, you should be familiar with the section called "The System Reconfiguration Process" in this chapter.

1. Log in as the superuser, **root**.
2. Change the current working directory to **/etc/conf/gen**.
3. You may either create your own **uxgen** input file or use the default S800 file. To create your own input file, start with the supplied S800 file which specifies the number and type of I/O devices to be configured; it also specifies swap space and system parameter configuration. Be sure to save the supplied S800 file as a backup. For example, you can do this as follows:

```
cp S800 S800BACKUP
```

4. Execute **uxgen** on the input file (either S800 or a file that you created). For example:

   ```
   /etc/uxgen S800
   ```

   The above command entry creates the directory **../S800** which contains the new kernel and associated files.

   - If you have any optional kernel software modules, such as subsystems, perform the necessary installation procedures *before* generating a new kernel. You need to add the appropriate statements to the **uxgen** input file to configure the optional software modules into the final kernel image (that is, the **hp-ux** file). Refer to the installation procedures included in the documentation provided with the optional kernel modules.

5. Back up the existing kernel by typing:

   ```
   mv /hp-ux /SYSBCKUP
   mv /etc/devices /etc/DEVBCKUP
   ```

   Remember the backup kernel filename. You may need to boot from this file if the new kernel does not boot properly.

6. Test the new kernel as follows.

   a. Copy the new kernel to the root directory (/). For example:

   ```
   mv ../S800/hp-ux /hp-ux
   ```

   b. Copy the new **devices** file to the **/etc** directory. For example:

   ```
   cp ../S800/devices /etc
   ```

   c. Write all recently changed files to disk using **sync**. Otherwise, data corruption could occur. Type (three separate occurrences):

   ```
   sync
   sync
   sync
   ```

   d. Reboot the system.

   If the new kernel was configured correctly, so that it boots and runs properly, you can proceed to step 7.

   If the new kernel does not boot, use the boot ROM manual mode to access the backup kernel. From the ISL prompt enter "**hpux SYSBCKUP**" where **SYSBCKUP** is the name of the backup kernel. After the system comes up, examine the **uxgen** input file, fix any problems found, and repeat this step.

7. If you added swap devices to the system, enable them by placing appropriate entries in the **/etc/checklist** file and then executing

   ```
   /etc/swapon -a
   ```

   The **-a** argument in the previous command makes all devices marked as swap devices (with "sw") available for swapping. The format of the **/etc/checklist** file is described later in this chapter.

8. If you added new file systems that you want **fsck** to check when you boot the system, you need to add them to **/etc/checklist**.

9. If you added new devices to the system, you need to create new special files for them. Refer to Chapter 5 for information on creating special files.

10. Return the system to the multiuser state.

## The System Reconfiguration Process

System reconfiguration is generating a new kernel tailored to your installation. To do this, you change the S800 file and then execute the **uxgen** command with the S800 file as input. The input file tells **uxgen** what files to include in the kernel, table sizes, and the supported hardware I/O configuration. If **uxgen** finds no errors in the input file, it executes the **make** program. The **make** program processes the **Makefile** created by **uxgen** to generate the new kernel.

### Directories Required for System Generation

The directory structure required to generate a system is pictured in Figure 7-1. At the top is the **/etc** directory. The **conf** directory is supplied with the HP-UX system in the **/etc** directory. You can move the **conf** directory (and all the subdirectories and files in it) to another part of your file system, and use that directory for system generation. But many system administration utilities (such as **update**) expect the **conf** directory to be in **/etc**. Therefore, it is a good idea to symbolically link that new directory to **/etc**. This is particularly useful if you need to free up some space in the root file system. For example, if you move all the files in **conf** to a new file system using the name **/newdisc/conf**, you have to **cd** to **/newdisc/conf** to begin system generation. The examples in this chapter assume that the **conf** directory is **/etc/conf**. If you moved the **conf** directory, substitute the new directory name wherever **/etc/conf** is referenced.

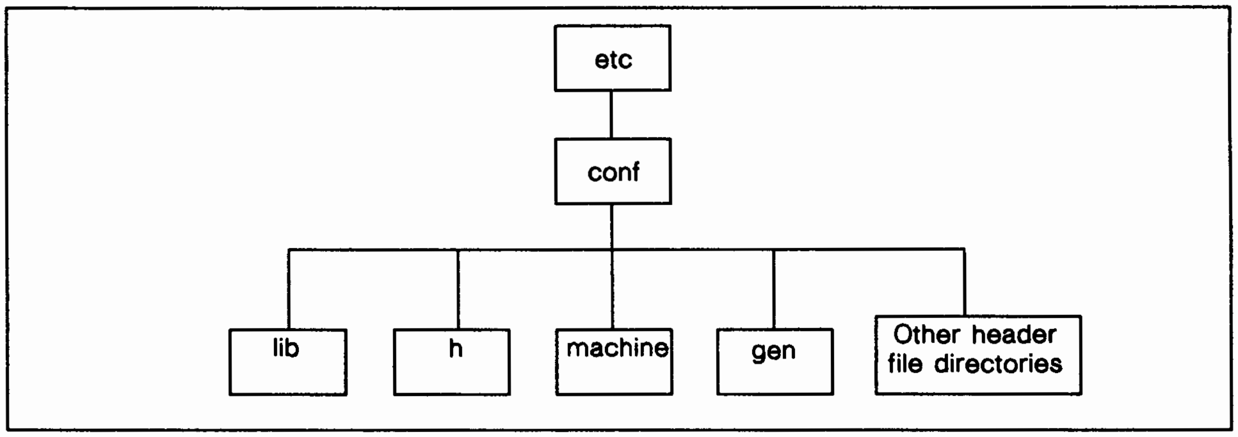


**Figure 7–1. System Generation Directory Structure**

The **gen** directory contains the input file to **uxgen** and the **Makefile** which compile the kernel. For your convenience, an input file template is supplied with your system using the name **S800** and is described in "uxgen Input File" later in this chapter.

In the **gen** directory, **Makefile** contains information required to compile the kernel. It has dependencies that rely on the library and header file directory paths (**../lib, ../h, ../machine**, etc.).

The **lib** directory contains all library files required to compile the kernel. It contains relocatable object code modules.

The **h, machine,** and other header file directories contain all system header files (for example, file.h, user.h, etc.). The system header files define system constants and data structures.

## Running uxgen

When the directory structure and required files are in place, change to the **gen** directory before running **uxgen**, as follows:

```
cd /etc/conf/gen
```

Run **uxgen** using the following syntax:

uxgen *inputfile*

For example:

```
uxgen S800
```

The **uxgen** program first checks the S800 file for errors. If there are no errors, **uxgen** creates the ../S800 directory; if it does not exist, **uxgen** reads the input file, creates the required files, and schedules the **make** command. The **make** command executes the **Makefile** created by **uxgen**.

If everything is successful, then an HP-UX executable kernel is created in the **conf/S800** directory. Note that you are still in the **gen** directory after executing **uxgen**.

The **devices** file created by **uxgen** contains a description of the currently supported hardware I/O configuration. This file is important during system generation because it is used by **insf**, **mksf**, and **lssf** in making and working with special files. You must move this file to the **/etc** directory after installing a new kernel. See Figure 7-2 for the directory structure after executing **uxgen**.

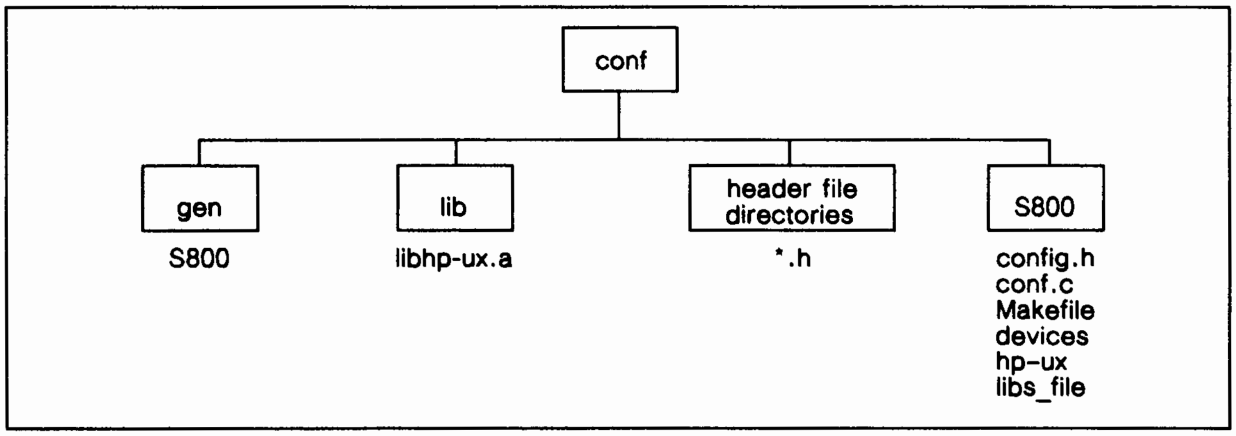


**Figure 7–2. Directory Structure After Running uxgen**

## Installing a New Kernel

To install a new kernel, you must backup the old kernel (**/hp-ux**), if you have not already done so. For example, you may copy **/hp-ux** to **/SYSBCKUP** so that you can always reboot to a working system. Move **../S800/hp-ux** to **/hp-ux** and then reboot:

```
mv /hp-ux /SYSBCKUP
mv /etc/devices /etc/DEVBCKUP
mv ../S800/hp-ux /hp-ux
mv ../S800/devices /etc/devices
shutdown
```

If the new kernel will not boot, reboot from the previous version by booting manually. If section 0 is the root partition, type:

```
ISL> hpux SYSBCKUP
```

To bring the system up (with the new kernel) in single-user mode, enter:

```
ISL> hpux -is
```

If you changed the partition for root to a section other than 0, the above commands will not work (refer to the section "Using Alternate Partitions for the Root and Swap File Systems"). Refer also to "Booting the System" in Chapter 3 for the full command syntax you need to use.

## uxgen Input File

The **uxgen** input file contains information that defines permissible I/O hardware configurations, table sizes, configuration parameters, and optional configurable kernel software modules. A default template file called S800 is supplied with your system.

You can modify the S800 file to specify permissible mid-bus or CIO card slot locations, as well as devices that attach to the CIO cards. You may also specify the HP-IB bus addresses for these devices.

A driver that is not specified in the I/O configuration is not loaded. This allows you to tailor the kernel specifically to your configuration and free additional memory and disk space by eliminating unnecessary drivers.

The ability to specify device locations leads to increased performance and greater flexibility. Typically, you include the device locations for the root file system, swap space, dumps, and console.

The table size parameters specify table sizes for messages, shared memory, semaphores, files, processes, etc.

Optional configurable kernel software modules can be selected and included in the final kernel. These configurable kernel software modules provide additional capabilities to an HP-UX system. Consult the documentation provided with the software for information on how to configure the modules using the **uxgen** input file.

The rest of the S800 file includes miscellaneous parameters that specify information such as the time zone and limits on accounting file sizes.

## Sample S800 File

The sample S800 file is an abridged file. The **#include** statement on the first line of the file is identical to the C language **#include** statement. An **#include** statement is used to include the contents of another file. Within the S800 file, you must include a file called **/etc/master** which contains verification rules that **uxgen** uses to check the I/O configuration. The I/O configuration statements are only for adding HP-supplied drivers. You may notice redundant statements in both the **/etc/master** and S800 files. Information specified in the S800 file can override statements in the **/etc/master** file. Within either the **/etc/master** or S800 file, the last occurrence of a statement overrides any previous occurrence of that statement. C-style comments are permitted in the **uxgen** input file.

A sample S800 file is shown below. In this example, the file **/etc/master** contains the information on each I/O device driver, and is included in the S800 file in the **#include** statement on the first line. The section following the **#include** statement specifies kernel device locations. Networking software statements appear next and are commented out. The comment delimiters are automatically removed when and if you install networking software. The next section includes kernel parameters such as the maximum number of users and processes permitted on the system. The **io** statement in the last section defines the I/O hardware configuration.

```
#include "/etc/master"
args    on      disc0   lu 0 section 1;
console on      mux0;   /* defaults to lu 0 */
dumps   on      disc0   lu 0 section 1;              (kernel device locations)
root    on      disc0   lu 0 section 0;
swap    on      disc0   lu 0 section 1
                disc0   lu 1 section 1;

/*include       nsrfa0;*/
/*include       nsnsipc0;*/                     (networking include statements)
/*include       nsdiag0;*/
/*include       nfs;*/

acctresume      4;                              (kernel parameters --
acctsuspend     2;                              explained later in this chapter)
bufpages        0;
dst             1;
itebuflines     100;
maxdsiz         0x8000;
maxssiz         0x1000;
maxtsiz         0x8000;
maxuprc         25;
maxusers        32;
msgmap          100;
msgmax          8192;
msgmnb          16384;
msgmni          50;
msgseg          1024;
msgssz          8;
msgtql          40;
nbuf            0;
ncallout        "(16 + NPROC)";
netmeminit      0;
netmemmax       "(1536 * NETCLBYTES)";
netmemthresh    -1;
nfile           "(16 * (NPROC + 16 + MAXUSERS) / 10 + 32 + 2 * NETSLOP)";
```

```
nflocks         200;
ninode          "((NPROC + 16 + MAXUSERS) +32)";
nproc           "(20 + 8 * MAXUSERS)";
npty            60;
ntext           "(24 + MAXUSERS + NETSLOP)";
semaem          16384;
semmap          10;
semmni          10;
semmns          60;
semmnu          30;
semume          10;
semvmx          32767;
shmmni          100;
shmmax          0x4000000;
shmseg          12;
timeslice       "(HZ/10)";
timezone        420;
unlockable_mem  0;

io {
     cio_ca0 address 8 {
          hpib0   address 0 {
                disc0   lu 0 address 0;
                disc0   lu 1 address 1;
                disc0   lu 2 address 2;
                disc0   lu 3 address 3;
          }
          mux0    lu 0 address 1;
          hpib0   address 2 {
                lpr0    lu 1 address 0;
                lpr0    lu 0 address 1;
                tape1   lu 0 address 3;
                tape1   lu 1 address 4;
                instr0  lu 0 address 7;
          }
          mux0    lu 1 address 3;
          /*lan0    lu 0 address 4;*/
          gpio0   lu 0 address 5;
          hpib0   address 6 {
                disc0   lu 4 address 0;
                disc0   lu 5 address 1;
                disc0   lu 6 address 2;
                disc0   lu 7 address 3;
          }
          hpib0   address 7 {
                lpr0    lu 2 address 1;
                tape1   lu 2 address 3;
                instr0  lu 1 address 7;
          mux0    lu 2 address 8;
          mux0    lu 3 address 9;
          mux0    lu 4 address 10;
          mux0    lu 5 address 11;
     }
}
```

## Drivers

A driver is a software module that controls a particular hardware module for the following types of hardware: channel adapters, device adapters, and I/O devices. There are four types of software modules: a Channel Adapter Manager (CAM) that drives the channel adapter, a Device Adapter Manager (DAM) that drives the device adapter, and a Device Manager (DM) and a Logical Device Manager (LDM) that both drive the physical devices. A DM manages devices. An LDM provides the system call interface to the kernel and makes calls to the DM to manage the physical devices. For example, the device adapter driver **hpib0 is** a single **hpib0** manager (DAM), whereas the **lpr** driver is composed of two managers, **lpr0** (LDM) for the logical device and **lpr10** (DM) for the physical device.

The use of driver names simplifies the **io** statement that specifies the I/O hardware configuration.

The **uxgen** utility has no inherent knowledge about drivers or the module managers corresponding to a driver name. Utility **uxgen** learns about drivers and managers via statements in the **/etc/master** file which defines the relationship between drivers and managers. Most interrelated managers have a parent/child relationship. The CAM, DAM, DM, LDM hierarchy requires explicitly stated driver dependencies. These dependency statements allow **uxgen** to flag errors in the S800 file. For example, driver **mux0** is not allowed to be connected to the HP-IB card. The statements describing these dependencies are discussed later in the "**uxgen** Statements" section.

## uxgen Statements

The input statements to **uxgen** are described using the Backus Naur Form (BNF). The BNF notation and the identifier format are illustrated in Figure 7-3.

```
<io>  ::=  {  <io-forest>  }

<io-forest> ::= <io-tree>
              | <io-forest>  <io-tree>

<io-tree>   ::= <io-leaf>
              | <io-branch>  {  <io-forest> }

<io-branch> ::=
              <module-id>  address  <integer>

<io-leaf>   ::=
              <module-id> lu <integer>  address  <integer>

Identifier
```

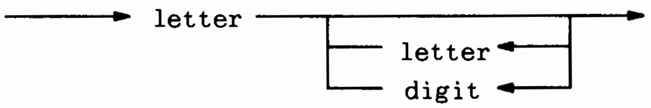


**Figure 7–3. Input File Statement and Identifier Formats**

Identifiers used in the input statements are made up of one or more characters where the first character is alphabetic (upper- or lowercase) and the remaining characters are alphanumeric. Identifiers may be any length but iotree_names are limited to 15 characters.

### args

The **args** statement specifies the disk and section to which the argument list for an **exec** request is written.

Examples:

```
     args on disc0 lu 0 section 1;
     args on disc0 lu 9 section 2;

     BNF:

          <args-stmt> ::= args on <module-id> lu <integer> section <integer>;
```

## dumps

The **dumps** statement specifies on which disk and section to write the operating system image if the operating system detects a fatal error (panic). It is often set up to dump to the primary swap space.

Examples:

```
          dumps on disc0 lu 0 section 1;
          dumps on disc0 lu 9 section 2;

     BNF:

          <dumps>  ::= dumps on <module-id> lu <integer> section <integer>;
```

## io

The **io** statement describes the location and addresses of I/O hardware devices. You need to specify each hardware device that requires a driver in the **io** statement. Each device entered in the **io** statement links the appropriate managers to the kernel.

Nested braces { } define a tree structure that relates to the hardware configuration. Two hardware types are described: leaf and branch. That is, some hardware connects directly to the CIO bus and other hardware connects through a card. To distinguish between leaf elements, each device that uses a particular driver must be assigned a unique logical unit number (starting from 0). For example, each **disc0** or **disc2** device must specify a different LU number. Similarly, each **mux0** statement requires a different LU number:

```
     hpib1 address 4 {        (this is the branch or the card to which devices attach)

        disc0     lu 0 address 0;          (these are the leaves)
        disc0     lu 1 address 1;          (2nd disc0 disk has unique lu of 1)
        tape1     lu 0 address 2;          (but  tape1 can have lu of 0 since it attaches)
        tape1     lu 1 address 3;           to a different driver, tape1)

     }

     mux0     lu 0 address 1;              (each mux has a different lu number)
     mux0     lu 1 address 3;
     mux0     lu 2 address 8;
     mux0     lu 3 address 9;
     mux0     lu 4 address 10;
     mux0     lu 5 address 11;
```

The **address** parameter specifies the location of a piece of hardware. This address parameter has different meanings depending on where it appears in the **io** statement.

Addresses need not be sequential and permissible addresses vary depending on which model computer you have.

Figure 7-4 relates hardware addresses to the **io** statement.

**Mid-bus/CIO Addresses**

```
io {
    cio_ca0 address 8 {<------------------ First level, CIO module number
       hpib0 address 0 {<----------------- Second level, CIO slot number
          disc0  lu 0 address 0; <-------- Third level, port or HP-IB
          disc0  lu 1 address 1;             device address
       }
       mux 0  lu 1 address 3;
                 .
                 .
```

**Figure 7-4. io Statement**

Each slot can hold a card that contains up to four modules (a module usually consists of memory or I/O). Some cards contain only one module but the addressing scheme allows for up to four. Thus, you determine first-level hardware addresses (also called the physical module number) by multiplying the slot number by 4 and adding the module number to it.

Valid hardware addresses at the first level on Model 825, 835, and 840 computers are 4, 8, 12, 16, 20, 24, and 28. Valid hardware addresses at the first level on Model 850 computers are 4, 8, 12, 16, 20, and 24 on mid-bus A and 68, 72, 76, 80, 84, and 88 on mid-bus B.

The second address level corresponds to the slot number for CIO device adapter cards. The channel adapter can handle a maximum of 16 CIO device adapter cards, but fewer physical slots are actually available. The Model 840 supports 12 physical slots on its channel adapter; the Model 825 supports 7 physical slots. The Model 850 computer supports 4 native channels (addresses 4, 8, 68, and 72) each supporting 5 physical CIO slots. All channel adapters on expanders support 8 slots. Device adapter addresses on all models start at zero.

The third address level corresponds to device-specific addresses for peripherals attached to HP-IB cards or MUX cards, for example.

Addresses are static, except for the address of the CIO channel adapter on Mid-bus/CIO architecture. For example, if a printer is specified in the **io** statement at bus address 1, then the printer must be at bus address 1. The system does not probe the hardware to locate the printer. On the other hand, the CIO channel adapter is probed to see if it is present.

The "Hardware Addressing" section in Chapter 5 contains additional relevant information.

**Mid–bus/CIO Example:**

```
      /* This system consists of one CIO channel adapter (cio_ca0) at
         address 8.  The channel adapter has two HP-IB cards (named hpib0)
         and one MUX (mux0) connected to it. Two disk drives are connected to
         the first HP-IB card, and two tape drives and a printer are connected
         to the second HP-IB card.

      */

      io {
           cio_ca0 address 8 {
                  hpib0 address 0 {
                      disc0 lu 0 address 0;
                      disc0 lu 1 address 2;
                  }
                  mux0 lu 0 address 1;
                  hpib0 address 2 {
                      tape1 lu 0 address 6;
                      tape1 lu 1 address 7;
                      lpr0  lu 0 address 5;
                  }
           }
       }
```

```
BNF on Mid–bus/CIO Architecture:

 <io>             ::= io { <io-forest> }

 <io-forest>      ::= <io-tree> |   <io-forest> <io-tree>

 <io-tree>        ::= <io-leaf> |   <io-branch> { <io-forest> }

 <io-branch>      ::= <module-id> address <integer>

 <io-leaf>        ::= <module-id> lu <integer> address <integer>;
```

## filesystem

The **filesystem** statement specifies a file system and the libraries associated with it. A file system, as used here, is a logical set of kernel functions that deals with a specific file system type, such as **nfs** (not to be confused with a file system on disk). Refer to the documentation supplied with each file system to find out how to configure the file system into the kernel.

Example:

```
      filesystem  nfs   lib    libnfs.a;

      BNF:

           <filesystem-stmt>  ::= filesystem <identifier> lib <lib-id>;
```

## include

The **include** statement causes a file system, a pseudo-driver, or a subsystem to be included in the kernel.

Examples:

```
include pty0;
```

```
BNF:

     <include-stmt> ::= include <identifier>;
```

## remove

The **remove** statement removes a file system, a pseudo-driver, or a subsystem from a kernel configuration. It overrides any previous **include** statement concerning the file system, pseudo-driver, or subsystem.

Examples:

```
remove pty0;
```

```
BNF:

     <remove-stmt> ::= remove <identifier>;
```

## root

The root statement specifies on which disk and section the root file system is located.

Examples:

```
root on disc0 lu 0 section 0;
```

```
BNF:

     <root>  ::= root on <module-id> lu <integer> section <integer>;
```

### subsystem

The **subsystem** statement specifies a subsystem and the libraries associated with it. A subsystem is an optional configurable kernel software module. Refer to the documentation supplied with each subsystem to determine how to configure the subsystem into the kernel.

Examples:

```
subsytem    subsys1    lib      libsubsys1.a;
```

```
BNF:

     <subsystem-stmt> ::= subsystem <identifier> lib <lib-id>;
```

### swap

The **swap** statement indicates on which disk and section the swap area is located. Note that multiple swap areas may be specified. The first swap area specifies the "primary swap area" which is always enabled. Other swap areas must be enabled using **swapon(1M)**. This is normally done at boot time by a command in **/etc/rc**. **swapon** reads the **/etc/checklist** file to enable all devices labeled as swap devices. Therefore, you must add additional swap devices to the **checklist** file.

Examples:

```
swap on disc0 lu 0 section 1;
swap on disc0 lu 0 section 1 disc0 lu 1 section 1;
swap on disc0 lu 1 section 0
        disc0 lu 2 section 0;
```

```
BNF:

      <swap>              ::= swap on <disc-list>;
           <disc-list>    ::= <disc-spec>
                          |   <disc-list> <disc-spec>
           <disc-spec>    ::= <module-id> lu <integer> section <integer>
```

## System Parameters

HP-UX allows you to configure certain operating system related parameters (attributes). These parameters determine how your system manages memory, limits table sizes, and determines other system limits.

**NOTE**

**HP-UX system parameters should not be modified unless you fully understand the ramifications of doing so. Each parameter is described in detail in "Appendix D: System Parameters". Read the entry for the system parameter thoroughly before you attempt to change the parameter.**

The format for specifying system parameters in the input file to **uxgen** is:

```
parameter (integer or formula) ;
        "anychars";
```

where **parameter** is one of the parameters described in Appendix D.

You can configure all attributes except **maxusers.**

- **accounting code parameters** - used by system accounting. Parameters: **timeslice, acctsuspend, acctresume.**
- **time information** - the system calculates the time from Greenwich Mean Time and allowing for differences due to daylight savings time. Parameters: **dst, timezone**.
- **limiter for system resource allocation** - calculates the default values of other global system parameters. Parameter: **maxusers**.
- **file system parameters** - number of open files, number of inodes in the system, number of file system buffer-cache buffer headers, and number of file locks, number of shared text descriptors, number of memory pages in the buffer cache. Parameters: **nfile, ninode, nbuf, nflocks, ntext, bufpages**.
- **process maximums** - the maximum number of processes per user and per system and the maximum number of shared text descriptors. Parameters: **maxuprc, nproc, ntext**.
- **maximum number of kernel timeouts** - maximum number of timeouts that can be scheduled by the kernel at a time. Parameter: **ncallout**.
- **user process size limits** - maximum data, stack, and text size. See the description below. Parameters: **maxdsiz, maxssiz, maxtsiz**.
- **pseudo-terminals** - maximum number of pseudo-terminals. Parameter: **npty**.
- **interprocess communication parameters** - used by the message, semaphore, and shared memory code. Parameters: **msgmap, msgmax, msgmnb, msgmni, msgseg, msgssz, msgtql, semaem, semmap, semmni, semmns, semmnu, semume, semvmx, shmmax, shmmni, shmseg**.
- **memory parameters**
  - memory guaranteed to be available for virtual memory or system overhead. Parameter: **unlockable_mem**.
  - memory available for each Internal Terminal Emulator (ITE) in the form of the number of lines of emulated terminal screen. Parameter: **itebuflines**.
  - networking subsystem memory parameters. These parameters effect your kernel only if the networking system is configured. Parameters: **netmeminit, netmemmax, netmemthresh, netisr.priority.**
  - memory available to the I/O system in bytes. Parameter: **iomemsize**.

### User Process Size Limits

Each of the system parameters is detailed in Appendix D. The parameters associated with process size interact with each other and need to be explained as a group.

Your process address space consists of text space, data space, stack space, and possibly some shared memory segments. The parameters, **maxtsiz, maxdsiz,** and **maxssiz** control your process's space. For example, **maxssiz** limits the stack size and stops infinitely recursive programs.

If you change these parameters, make sure they can still work together. Refer to the "Memory Management" section in Chapter 9 for a description of user process space.

## Configurable Parameters

Following are the tunable parameters found in the **uxgen** input file. Appendix D includes more complete descriptions of these parameters.

### acctresume

The **acctresume** statement specifies the percentage of file system space that must be free to re-enable process accounting after it was suspended because of insufficient free space (see **acctsuspend**).

Example:

```
acctresume 4;

BNF:

     <acctresume-stmt>  ::= acctresume <integer>;
                          |   acctresume "<anychars>";
```

### acctsuspend

The **acctsuspend** statement specifies the percentage of file system space that must be free to allow process accounting.

Example:

```
acctsuspend 2;

BNF:

     <acctsuspend-stmt> ::= acctsuspend <integer>;
                          |   acctsuspend "<anychars>";
```

### bufpages

The **bufpages** statement specifies the number of memory pages allocated to the buffer cache. Each page is allocated 2048 bytes of memory. If **bufpages** is set to 0, the kernel will allocate two buffer pages for every buffer header defined by **nbuf**. If both are 0, 10 percent of available memory is allocated to buffer space. HP recommends changing the size of the buffer cache using this statement, not using **nbuf**.

Example:

```
bufpages   0;
```

```
BNF:

     <bufpages-stmt>  ::= bufpages <integer>;
                        |  bufpages "<anychars>";
```

### dst

The **dst** statement specifies whether or not to convert to daylight savings time. 0 means do not use daylight savings time; 1 means use USA daylight savings time.

Examples:

```
dst 0;
dst 1;
```

```
BNF:

     <dst-stmt> ::= dst <integer>;
```

### iomemsize

The **iomemsize** statement specifies the amount of memory in bytes available to the I/O system.

Example:

```
iomemsize "(32+2*NUM_IOTREE_RECS)*NBPG";
```

```
BNF:

     <iomemsize-stmt>  ::= iomemsize <integer>;
                        |  iomemsize "<anychars>";
```

### itebuflines

The **itebuflines** statement specifies the number of lines of emulated terminal screen memory (or scrolling) for each Internal Terminal Emulator (ITE) configured into the system. Refer to the "**itebuflines**" section in Appendix D for more information.

Example:

```
itebuflines 100;
```

```
BNF:

     <itebuflines-stmt>  ::= itebuflines <integer>;
```

## maxdsiz

The **maxdsiz** statement specifies the maximum process data segment size (in 2048-byte pages). Refer to the "**maxdsiz**" section in Appendix D for more information.

Example:

```
maxdsiz 0x8000;

BNF:

          <maxdsiz-stmt>  ::= maxdsiz <integer>;
                            |   maxdsiz "<anychars>";
```

## maxssiz

The **maxssiz** statement specifies the maximum process stack size (in 2048-byte pages).

Example:

```
maxssiz 0x1000;

BNF:

          <maxssiz-stmt>  ::= maxssiz <integer>;
                            |   maxssiz "<anychars>";
```

## maxtsiz

The **maxtsiz** statement specifies the maximum process shared text segment size (in 2048-byte pages).

Example:

```
maxtsiz 0x8000;

BNF:

          <maxtsiz-stmt>  ::= maxtsiz <integer>;
                            |   maxtsiz "<anychars>";
```

## maxuprc

The **maxuprc** statement specifies the maximum number of processes that a user may have.

Example:

```
maxuprc 25;

BNF:

          <maxuprc-stmt> ::= maxuprc <integer>;
                           |   maxuprc "<anychars>";
```

## maxusers

The **maxusers** statement defines the macro MAXUSERS (for example, "#define MAXUSERS 8"). It determines the size of system tables. The actual limit of the number of users depends on the license version of HP-UX purchased. You can examine the license version using the **uname(util)** .

Rather than varying each configurable parameter individually, it is easier to specify certain parameters using a formula based on the maximum number of expected users (for example, **nproc** "20 + 8 * MAXUSERS;"). Thus, if you increase the maximum number of users on your system, you only need to change the **maxusers** statement.

This statement must appear before any other statement that uses the MAXUSER macro. For example, **nproc** "20 + 8 * MAXUSERS;" must follow "maxusers 8;".

Examples:

```
    maxusers 8;
    maxusers 32;

    BNF:

          <maxusers> ::= maxusers <integer>;
```

## msgmap

The **msgmap** statement specifies the number of message pool map entries. This is a resource map used to find fragmented memory space in the pool of message space.

Example:

```
    msgmap 100;

    BNF:

          <msgmap-stmt>  ::= msgmap <integer>;
                          |   msgmap "<anychars>";
```

## msgmax

The **msgmax** statement specifies the maximum number of bytes in a message.

Example:

```
    msgmax 8192;

    BNF:

          <msgmax-stmt>  ::= msgmax <integer>;
                          |   msgmax "<anychars>";
```

## msgmnb

The **msgmnb** statement specifies the maximum number of bytes for all messages that are queued in a single message queue.

Example:

```
msgmnb 16384;

BNF:

     <msgmnb-stmt>  ::= msgmnb <integer>;
                     |   msgmnb "<anychars>";
```

## msgmni

The **msgmni** statement specifies the number of message queue identifiers.

Example:

```
msgmni 50;

BNF:

     <msgmni-stmt> ::= msgmni <integer>;
                    |   msgmni "<anychars>";
```

## msgseg

The **msgseg** statement specifies the number of units (each msgssz bytes long) available for messages.

Example:

```
msgseg 1024;

BNF:

     <msgseg-stmt>  ::= msgseg <integer>;
                     |   msgseg "<anychars>";
```

## msgssz

The **msgssz** statement specifies the size (in bytes) of each unit of memory used for messages.

Example:

```
msgssz 8;

BNF:

     <msgssz-stmt>  ::= msgssz <integer>;
                     |   msgssz "<anychars>";
```

## msgtql

The **msgtql** statement specifies the number of message headers.

Example:

```
msgtql 40;
```

```
BNF:

     <msgtql-stmt>  ::= msgtql <integer>;
                     |   msgtql "<anychars>";
```

## nbuf

The **nbuf** statement specifies the number of file system buffer cache buffer headers. If **nbuf** is set to 0, the kernel allocates one buffer header for every two buffer pages defined by **bufpages**. If both are 0, 10 percent of available memory is allocated to buffer space. HP recommends using **bufpages** to configure buffer cache size.

Example:

```
nbuf 0;
```

```
BNF:

     <nbuf-stmt>  ::= nbuf <integer>;
                   |   nbuf "<anychars>";
```

## ncallout

The **ncallout** statement specifies the number of timeouts that may be pending.

Example:

```
ncallout "(64 + NPROC)";
```

```
BNF:

     <ncallout-stmt> ::= ncallout <integer>;
                      |   ncallout "<anychars>";
```

## netisr_priority

The **netisr_priority** statement specifies the real-time process priority for networking. For complete information on setting this parameter, refer to the *LAN, NS, ARPA Services/9000 Series 800 Node Manager's Guide*.

Example:

```
netisr_priority    0;
```

```
BNF:

     <netisr_priority-stmt>  ::= netisr_priority <integer>;
```

### netmeminit

The **netmeminit** statement specifies the number of bytes of memory to be preallocated at system initialization time. For information on setting this parameter, refer to the *LAN, NS, ARPA Services/9000 Series 800 Node Manager's Guide.*

Example:

```
netmeminit    0;
```

```
BNF:

     <netmeminit-stmt>  ::= netmeminit <nonnegative integer>;
                         |   netmeminit "<anychars>";
```

### netmemmax

The **netmemmax** statement specifies the maximum number of bytes of physical memory that networking will ever use. For more information on setting this parameter, refer to the *LAN, NS, ARPA Services/9000 Series 800 Node Manager's Guide.*

Example:

```
netmemmax    "(1536 * NETCLBYTES)";
```

```
BNF:

     <netmemmax-stmt>  ::= netmemmax <nonnegative integer>;
                        |   netmemmax "<anychars>";
```

### netmemthresh

The **netmemthresh** statement specifies the maximum number of bytes of physical memory at which the networking subsystem switches from a full memory reservation scheme to an open memory reservation scheme. For more information on setting this parameter and networking memory reservation schemes, refer to the *LAN, NS, ARPA Services/9000 Series 800 Node Manager's Guide.*

Example:

```
netmemthresh    -1;
```

```
BNF:

     <netmemthresh-stmt>  ::= netmemthresh <integer>;
                           |   netmemthresh "<anychars>";
```

### nfile

The **nfile** statement specifies the maximum number of open files.

Example:

```
nfile "(16 * (NPROC + 16 + MAXUSERS)/10 + 32 + 2 * NETSLOP)";
```

```
BNF:

     <nfile-stmt>  ::= nfile <integer>;
                    |   nfile "<anychars>";
```

### nflocks

The **nflocks** statement specifies the maximum number of file/record locks.

Example:

```
      nflocks 200;

      BNF:

           <nflocks-stmt>  ::= nflocks <integer>;
                            |   nflocks "<anychars>";
```

### ninode

The **ninode** statement specifies the maximum number of open in-core inodes.

Example:

```
      ninode "(NPROC + 16 + MAXUSERS + 32)";

      BNF:

           <ninode-stmt>  ::= ninode <integer>;
                           |   ninode "<anychars>";
```

### nproc

The **nproc** statement specifies the maximum number of processes that may simultaneously exist.

Example:

```
      nproc "(20 + 8 * MAXUSERS)";

      BNF:

           <nproc-stmt>  ::= nproc <integer>;
                          |   nproc "<anychars>";
```

### npty

The **npty** statement specifies the number of ptys (pseudo–ttys).

Example:

```
      npty 60;

      BNF:

           <npty-stmt>  ::= npty <integer>;
                         |   npty "<anychars>";
```

## nstlbe

The **nstlbe** statement enables or disables use of the two-level software TLB mechanism by specifying the number of software TLB entries requested.

Example:

```
nstlbe 1024;

BNF:

     <nstlbe-stmt>  ::= nstlbe <integer>;
                      |   nstlbe "<anychars>";
```

## ntext

The **ntext** statement specifies the maximum number of active shared text descriptors.

Example:

```
ntext "(24 + MAXUSERS + NETSLOP)";

BNF:

     <ntext-stmt>  ::= ntext <integer>;
                    |   ntext "<anychars>";
```

## semaem

The **semaem** statement specifies the maximum amount a semaphore may be adjusted due to a process terminating.

Example:

```
semaem 16384;

BNF:

     <semaem-stmt>  ::= semaem <integer>;
                      |   semaem "<anychars>";
```

## semmap

The **semmap** statement specifies the number of semaphore map entries.

Example:

```
semmap 10;

BNF:

     <semmap-stmt> ::= semmap <integer>;
                    |   semmap "<anychars>";
```

## semmni

The **semmni** statement specifies the number of semaphore identifiers.

Example:

```
semmni 10;

BNF:

     <semmni-stmt>  ::= semmni <integer>;
                    |   semmni "<anychars>";
```

## semmns

The **semmns** statement specifies the maximum number of semaphores for the system.

Example:

```
semmns 60;

BNF:

     <semmns-stmt>  ::= semmns <integer>;
                    |   semmns "<anychars>";
```

## semmnu

The **semmnu** statement specifies the maximum number of processes that can have pending "semaphore undo" requests on a semaphore.

Example:

```
semmnu 30;

BNF:

     <semmnu-stmt> ::= semmnu <integer>;
                   |   semmnu "<anychars>";
```

## semume

The **semume** statement specifies the maximum number of semaphores on which a process may have pending "semaphore undo" requests.

Example:

```
semume 10;

BNF:

     <semume-stmt>  ::= semume <integer>;
                    |   semume "<anychars>";
```

## semvmx

The **semvmx** statement specifies the maximum value of a semaphore.

Example:

```
semvmx 32767;
```

```
BNF:

     <semvmx-stmt>   ::= semvmx <integer>;
                      |   semvmx "<anychars>";
```

## shmmax

The **shmmax** statement specifies the maximum number of bytes in a shared memory segment.

Example:

```
shmmax 0x4000000;
```

```
BNF:

     <shmmax-stmt>   ::= shmmax <integer>;
                      |   shmmax "<anychars>";
```

## shmmni

The **shmmni** statement specifies the maximum number of shared memory segments.

Example:

```
shmmni 100;
```

```
BNF:

     <shmmni-stmt>   ::= shmmni <integer>;
                      |   shmmni "<anychars>";
```

## shmseg

The **shmseg** statement specifies the maximum number of shared memory segments that can be simultaneously attached to a process.

Example:

```
shmseg 12;
```

```
BNF:

     <shmseg-stmt>   ::= shmseg <integer>;
                      |   shmseg "<anychars>";
```

## timeslice

The **timeslice** statement specifies the number of 10 millisecond intervals in each timeslice of round-robin scheduling. A value of -1 disables round-robin scheduling.

Example:

```
timeslice "HZ/10";
```

```
BNF:

     <timeslice-stmt>    ::= timeslice <integer>;
                          |   timeslice "<anychars>";
```

### timezone

The **timezone** statement specifies the local time zone in number of minutes west of Greenwich.

Example:

```
timezone 480;

BNF:

     <timezone>       ::= timezone <integer>;
```

### unlockable_mem

The **unlockable_mem** statement specifies the number of bytes of memory that cannot be locked.

Example:

```
unlockable_mem 0;

BNF:

     <unlockable_mem-stmt>  ::= unlockable_mem <integer>;
                            |   unlockable_mem "<anychars>";
```

## Swap Space Configuration

Swap space is an area on your disk (or disks) reserved for the virtual memory system (see "Memory Management" in Chapter 9). The system uses this space during pageouts, and during process swaps when memory is in short supply. Each user process in the system must have enough swap space allocated to back up its current virtual memory needs.

By default, when the system boots, the primary swap device is always opened and enabled. If the open fails, the system displays the panic message "Can't open primary swap device." This is usually because the default primary swap device was changed, and the new device is either off line, or nonexistent. It is advisable to always have the primary swap device on the root disk (if space permits). This reduces the number of variables that are required to get your system up and running, since only one disk must be present. Additional swap space may be added using the **swapon** command and by adding entries for additional swap areas in **/etc/checklist**. The **swapon** command is usually invoked from the **/etc/rc** file when going to the multiuser state.

### Determining the Amount of Swap Space Needed

Swap space holds the process image of all processes that are running. If you do not have enough swap space, HP-UX will not execute your program. It is also possible to receive a swap space error when any of the following occurs: executing **fork** or **exec**, using increasing amounts of stack space, creating shared memory segments, or increasing memory usage through the **brk** system call. Depending on the situation, HP-UX will either return an error (for example ENOMEM) or kill your process and send you the message:

```
Sorry, pid PIDNUM was killed due to no swap space
```

where *PIDNUM* is the identification number of the process that was killed.

There are three ways to choose the amount of swap space that your system will need. They are listed in the order of increasing difficulty:

1. Use the system default. This is typically section 1 of your root disk. This may or may not be enough space depending on the number of users and the type of applications you are running. A moderate system generally requires a second swap partition. The kernel, as shipped, has a second swap partition specified as section 1 of drive 1, on the first HP-IB card. The partition is commented out in the **S800** file. To use the additional swap sold partition, edit the **S800** file and regenerate the kernel. Enable the second swap section by adding to **/etc/checklist** and executing **swapon -a** in **/etc/rc**.
2. Estimate the amount of swap space you need by adding the swap space required by your largest applications to the default swap space provided. Determine the swap space required by your application from either the applications documentation or by using some of the methods outlined in option 3 (parts A, B, and C) given below. You can determine the default swap space size, and other candidate swap partitions by looking into the **/etc/disktab** file.
3. Calculate your swap space requirements with the following formula and the worksheet provided.

If you run large processes, or if you have many users on your system, you may need to use the formula given below. It is difficult to determine some of the numbers to use in the formula. Use the formula only if options one and two do not work.

**Swap Space Computation**:

A. SUM(all shared code sizes) of all running processes as shown by **ps -el**. Do not count the pagedaemon, swapper, or statdaemon. The **file** command tells if a file contains shared text, and the size of the text may be found with the **size** command.

B. SUM(all data and stack sizes) of all processes. By using the **size** command, you can calculate the size of initialized data and uninitialized data (BSS). This represents only part of the total swap space requirements of the process. In addition, you must calculate the amount of dynamic heap and stack space that the program might require. If you are familiar with the program's runtime logic, you might be able to calculate this by looking at requests made to **sbrk** or **malloc**.

   You can approximate by running the program with a typical input stream, and determine the total virtual memory size in number of pages. The virtual memory size can be obtained by running **ps -el** and looking in the SZ (size in 2048 byte pages) field for the program you are interested in. You may then subtract the code size calculated in step A from this to get the total data and stack size.

C. SUM (sticky code sizes) for all sticky code files that will be executed, but are not currently being used by any processes. Typically editors fall into this category.

D. SUM (all existing shared memory segment sizes) for shared memory segments created by users via **shmget**. **Ipcs** can be used to show active shared memory segments.

E. Size of the scratch area used by **exec** to hold arguments. The default size of this area is 256 Kbytes; it can be changed by using **uxgen**.

F. Fragmentation and overhead. Because data and stack segments can grow dynamically, the system uses an algorithm which allocates extra space for these segments to allow growth. The system allocates swap space in units. The units are controlled by the configuration parameters **dmmin** and **dmmax** as shown in Appendix D. The first unit allocated is in 1Kb-blocks and the number of blocks is determined by **dmmin**. Each successive unit is twice as large as the previous one, until **dmmax** is reached. From this point, all additional units are the same size, determined by **dmmax**. This algorithm results in a wide divergence of efficiency, depending on the application programs being run. The fragmentation is the difference between the allocated space as calculated above, and the requested size (i.e., virtual data and stack sizes). The process swap space overhead is used when the process is swapped out. It is used to hold the image of system process-related information that is swapped to the disk.

Swap space = A + B + C + D + E + F.

## Swap Space Computation Worksheet

**A.** For shared code, fill out the code space needed by the process.

| Process ID | Code size. |
|---|---|
| ________________ | ________________ |
| ________________ | ________________ |
| ________________ | ________________ |
| ________________ | ________________ |
| ________________ | ________________ |

**B.** For each shared process listed above, fill out the data and stack space needed by the process. For each nonshared process, add the process's code size to its data size and enter the amount (i.e., total from executing size).

| Process ID | Data size.<br>(minimum data space = dmmin x 1Kb block<br>default = 32Kb) |
|---|---|
| ________________ | ________________ |
| ________________ | ________________ |
| ________________ | ________________ |
| ________________ | ________________ |
| ________________ | ________________ |

**C.** For each sticky bit file that was executed since power-up, but not currently used, fill out the code space.

| Process ID | Code size. |
|---|---|
| ________________ | ________________ |
| ________________ | ________________ |
| ________________ | ________________ |
| ________________ | ________________ |
| ________________ | ________________ |

**D.** For each shared memory segment, give the shared segment size.

| Shared Memory | Segment Size. |
|---|---|
| ________________ | ________________ |
| ________________ | ________________ |
| ________________ | ________________ |
| ________________ | ________________ |
| ________________ | ________________ |

**E.** Scratch area used for arguments during **exec**. Default is 256 Kb:

________________

________________

**F.** Fragmentation and overhead. Compute as outlined in previous discussion or use an estimate. A suggested estimate is 6 Mb. ______________________________

TOTAL AMOUNT OF SWAP SPACE NEEDED IS:

A + B + C + D + E + F = Total swap space

____ + ____ + ____ + ____ + ____ + ____ = ______________________________

Example:

Let's assume that you are going to run four FORTRAN compiles with optimization. This example shows how to calculate the additional swap space that might be required to handle this. Let's start with one compile and gather some statistics.

Note that the amount of swap space required by HP-UX compilers depends on the size of the program being compiled. Also, compiling with optimization may consume almost twice as much swap space. The following example may not be indicative of how much swap space your compile environment would use. It is suggested that you overestimate swap space initially. Later, reduce it as you become familiar with the requirements of your compile environment.

```
% fc -o test.f >& out &
[1] 3663
% ps -l
F S   UID   PID  PPID  C PRI NI   ADDR  SZ   WCHAN TTY       TIME COMD
1 S   867 22055     1  1 168 20 a0a308 121 1515308 ttyd3p4   0:07 csh
1 S   867  3663 22055  0 158 20 5f3900  28  4c8024 ttyd3p4   0:00 fc
1 R   867  3664  3663204 229 20 d2ae306963         ttyd3p4   0:25 f77comp
1 R   867  3680 22055 49 190 20 f9d260 182         ttyd3p4   0:00 ps

% file /usr/lib/f77comp
/usr/lib/f77comp:      s800 shared executable

% size /usr/lib/f77comp
5720($MILLICODE$) + 141600($LIT$) + 1726120($CODE$) + 8($CODE20$) +
576($UNWIND$MILLICODE$) + 47152($UNWIND$) + 1824($UNWIND$) + 16($UNWIND20$) +
132($RECOVER$) + 164192($GLOBAL$) + 93032($DATA$) + 9424($DATA$) +
8($PFACOUNTER$) + 24400($BSS$) = 2214204
```

From the above information, you can determine that

- The code size for /usr/lib/f77comp is 5720+1726120+8= 1731848 bytes
- The data size for /usr/bin/f77comp is 93032+9424= 102456 bytes
- The BSS size for /usr/bin/f77comp is 24400
- (BSS and initialized data = 102456 + 24400 = 126856 bytes)

Looking at the **ps** output for this FORTRAN compile, 6963 pages or 14260224 bytes are in use. Subtract the code size from this and that leaves 12528376 bytes for the total data, heap, and stack. The difference between this data size and the fixed (obtained from **size** above), is the amount of dynamic space used by the FORTRAN compiler for this program.

| A) Process ID | Code Size |
|---|---|
| *3664* | 1731848 |

| B) Process ID | Data size |
|---|---|
| *3664* | 12528376 |

Therefore, you need 1.7 Mb of swap space for the compiler text (of which there will be only one copy) and 12.5 Mbyte of swap space for the per process data. This would mean that the system would need approximately 1.7+4 * 12.5 ( or 51.7 Mb) of swap space for the four compiles to execute in parallel. Then add this number to the default swap space to obtain the new swap space requirements.

## Determining Swap Space Configuration

After estimating the amount of swap space required, you must designate sections in the system as swap space. The allocation of swap space is on a partition basis. You cannot further divide a section and use only a part of it for swap space. Sizes of the sections on each drive mayOnce you be found in the **/etc/disktab** file. For instance the amount of space in section one of a 7935 is 48560 blocks or approximately 48 Mb (1024 Kb blocks). Section 1 is the default primary swap area on the root disk.

If you have only one disk and require more than the default amount of swap space, you must choose either to use an additional section for swapping or to change the current primary partition to a larger section. If you have more than one disk, you should probably choose another section on a different disk for a second (or third) swap section. This enables the system to take advantage of the interleaved swap feature. Your swap space will be spread across multiple spindles, and thus reduce excessive head movement on any one disk drive.

As mentioned earlier, the standard kernel is configured with an extra swap partition (in addition to the primary) on drive 1, section 1, on the first disk HP-IB card. If you have a disk at that address, then you may enable swapping to that area by adding an entry to the **/etc/checklist** file. A sample entry for the **disc0** driver is:

```
      /dev/dsk/c1d0s1 /swap1  swap  defaults  0 0
```

Next, execute the **swapon -a** command (from **root** login) to add the extra swap space into the current running system.

Remember, when choosing a section on any device, you must take care not to choose a section which currently contains or overlaps a file system that you want to keep. The swapping code allows the superuser to start swapping on a file system which is currently mounted. This would cause critical damage to that file system, cause the system to panic, or you would be unable to mount the file system.

To change the default swap space layout, you must use **uxgen** and modify its input file. You should know the syntax and exact procedures for executing **uxgen** and changing kernels before going any further. Refer to the "System Reconfiguration Process" section given earlier in this chapter.

The first device specified to **uxgen** with the **swap** statement is the primary swap device for swapping. This primary partition is enabled during the boot up sequence and MUST be available. All subsequent entries are enabled only after the **swapon** system call is made to add them. This is usually done using a **swapon** command invoked from **/etc/rc**. The **swapon** command with the **-a** option, looks in the **/etc/checklist** file to determine which partitions to add as additional swap space. A "type" field of "swap" in the **/etc/checklist** file specifies a swap partition. So **swapon** adds partitions of type "swap." Note, that once a partition is added as a swap partition, you have to reboot the system to remove it.

Examples:

In the following examples, the **disc0** driver is used and the **uxgen** configuration file has the following setup:

LU 0 – 3 are located on the first disk HP-IB card, and represent HP-IB addresses 0 – 3.

LU 4 – 7 are located on the second disk HP-IB card, and represent HP-IB addresses 0 – 3.

Example 1. There is only one disk and you want to add a second swap partition on the same device. For this example, assume that the root device is a 7935, and that section 10 is the extra swap partition.

The swap section of the uxgen input file would look as follows:

```
swap on disc0 lu 0 section 1
        disc0 lu 0 section 10;
```

Note that one semicolon terminates the list.

This tells **uxgen** to make the primary swap partition entry (in the kernel's swdev_t table) section 1 of the root disk and to make an additional swap partition entry as section 10 of the same disk.

The **/etc/checklist** file would need the following entry to enable the additional swap partition:

```
/dev/dsk/c0d0s10  swap1 swap  defaults 0 0
```

Note that the first and second fields (**/dev/dsk/c0d0s10** and **swap1)** are filled in here as a reminder that this section is being used for swapping.

Example 2. Assume that you have only one disk and want to change the default swap partition to a larger partition. Again, assume that the root disk is a 7935, and section 10 of the root disk contains the primary swap partition.

The swap statement of the uxgen input file would look as follows:

```
swap on disc0 lu 0 section 10;
```

This tells **uxgen** to make the primary swap space entry (in the kernel's swdev_t table) section 10 of the root disk.

Example 3. This example shows a more complicated configuration. Assume that a 7914 is the root device and that there is another 7914 connected to a different HP-IB card. The second 7914 is at HP-IB address 1. In this example, the primary swap space is on section 1 of the root disk (the default). The goal is to make the entire second 7914 one large swap device (section 2 is the entire disk).

The swap statement of the **uxgen** input file would look as follows:

```
swap on disc0 lu 0 section 1
        disc0 lu 5 section 2;
```

This tells **uxgen** to make the primary swap space entry (in the kernel's swdev_t table) section 1 of the root disk and to make an additional swap space entry as section 2 of the second HP-IB card at HP-IB address 1.

The **/etc/checklist** file would need the following entry to enable the additional swap space:

```
/dev/dsk/c5d0s2    /swap1  swap  defaults 0 0
```

Example 4. In this example, add another 7935 disk to Example 3, using section 4 as additional swap space. This 7935 is on the same HP-IB card as the second 7914 and is at HP-IB address 0 (drive 0).

The swap section of the **uxgen** input file would look as follows

```
swap on disc0 lu 0 section 1
        disc0 lu 4 section 4
        disc0 lu 5 section 2;
```

The **/etc/checklist** file would need the following entries to enable the additional swap partitions:

```
/dev/dsk/c4d0s4   swap1   swap  defaults 0 0
/dev/dsk/c5d0s2   swap2   swap  defaults 0 0
```

When the swap configuration is changed in the **uxgen** input file, you need to run **uxgen** and build another kernel. The new kernel has the new swap space configuration. Next, you need to update the **/etc/checklist** file to contain entries with the type field of "swap" for each of the additional swap areas that were specified to **uxgen**. The above examples include the **/etc/checklist** entries which are needed for each case. When you boot the new kernel, you can then add (using **swapon**) all the new swap areas that were specified to **uxgen**.

## Controlling Disk Use

As System Administrator, you should keep track of the amount of disk space available to users and the distribution of free disk space across file systems. The following procedure helps evaluate disk use and identify future disk needs.

1. Execute the **du** command regularly (weekly or biweekly) and keep the output in a file for later comparison. This method lets you spot users who are rapidly increasing their disk usage.
2. Use the **find** command to locate large or inactive files. For example, the following command writes in the file **agingfiles** the names of files neither written nor accessed in the last 90 days:

   ```
   find / -mtime +90 -atime +90 -print > agingfiles
   ```

3. Use the **df** command to list the amount of free disk space on a volume. The **df** command returns the number of free blocks left on the file system available for the ordinary user (not superuser); it does not count the number of blocks reserved by **minfree**. If you wish to see a more detailed report of disk usage, type:

   ```
   df -t
   ```

An example of output from **df -t** is:

```
/    (/dev/dsk/c0d0s10):  93918 blocks       135875 i-nodes
                          715904 total blocks  163840 total i-nodes
                          550394 used  blocks   27965 used  i-nodes
                          10 percent minfree
```

In the example, 93918 blocks (512-byte blocks) are available to the ordinary user, and 165510 512-byte blocks (715904 - 550394 = 165510) are available to the superuser.

4. Some files, if present, are written to automatically by certain HP-UX commands (to monitor system use and for general administration). Some files are created automatically by the commands that require them. These files are called **logging files**. If not periodically checked and truncated, these files simply continue to grow.

   Clean the logging files daily or weekly. Here are some typical logging files:

   - **/etc/wtmp** (binary) - history of logins, logouts, and date changes.
   - **/etc/btmp** (binary) - history of failed login attempts. You need to create this file to keep a record of failed login attempts.
   - **/usr/adm/sulog** (ASCII) - history of use of the **su** command.
   - **/usr/lib/cron/log** (ASCII) - history of actions by **cron**.
   - **/usr/adm/shutdownlog** (ASCII) - history of system shutdown.
   - **/usr/adm/messages** (ASCII) - history of messages.
   - **/usr/adm/diaglog** (binary) - collection of diagnostic messages.
   - **/usr/spool/mqueue/syslog** (ASCII) - mail message queue.
   - **uucp** log files (all ASCII):

     **/usr/spool/uucp/ERRLOG**
     **/usr/spool/uucp/LOGDEL**
     **/usr/spool/uucp/LOGFILE**
     **/usr/spool/uucp/SYSLOG**

It is easiest to set up **cron** to truncate these files periodically or when they grow too large.

The following log files are available for you to review but their contents should not be destroyed:

- **/etc/utmp** (binary) - current login status.
- **/etc/mnttab** (ASCII) - mounted devices (not the official list kept by the system in the kernel) in almost the same format as **/etc/checklist**.
- **/usr/adm/rebootlog** (ASCII) - history of rebooting.

## Creating a New File System

If you run out of space on a file system, you may either remove files to gain space or create another file system. The following sections provide the background information and steps necessary to create a new file system.

**NOTE**

**You can also use SAM to create file systems on a disk, modify the** /etc/checklist **file, and for listing and enabling swap devices. Refer to** *System Administration Basics* **for more information.**

### Need for a New Disk

The first thing to consider when creating another file system is whether or not you need an additional disk. If you have an empty section on an existing disk, you can create a file system there; otherwise, you need to add a new disk to your system.

By convention, disks are divided into sections so the disk driver can easily manage disk space. Each disk usually consists of several sections but one section could be the entire disk. The file **/etc/checklist** should reflect which disk sections are currently in use. Ensure that this file is always up to date.

The size of each section for each disk type is described in the file **/etc/disktab**. The overall sectioning scheme is shown in Figure 7-5. Depending on the type of disk, all or part of this scheme is used. For example, for an HP 7935 disk, all 15 sections are defined; for an HP 7912 disk, only sections 0 through 9 and 12 through 15 are defined.

In addition to the size of the sections, **/etc/disktab** maintains the default block and fragment size for certain sections. These sections have predefined uses for first-time installation.

## CAUTION

**When determining which sections to use, DO NOT allocate file systems on sections that overlap. For example, if you allocate a file system on section 11 on the 7935 disk, you then cannot allocate sections 2, 3, 4, 5, 8, 9, 10, 12, and 13 (see Figure 7-5) for other file systems. You can only put file systems in sections 1, 0, 6, 7, 14, or 15 (and some of these sections overlap).**

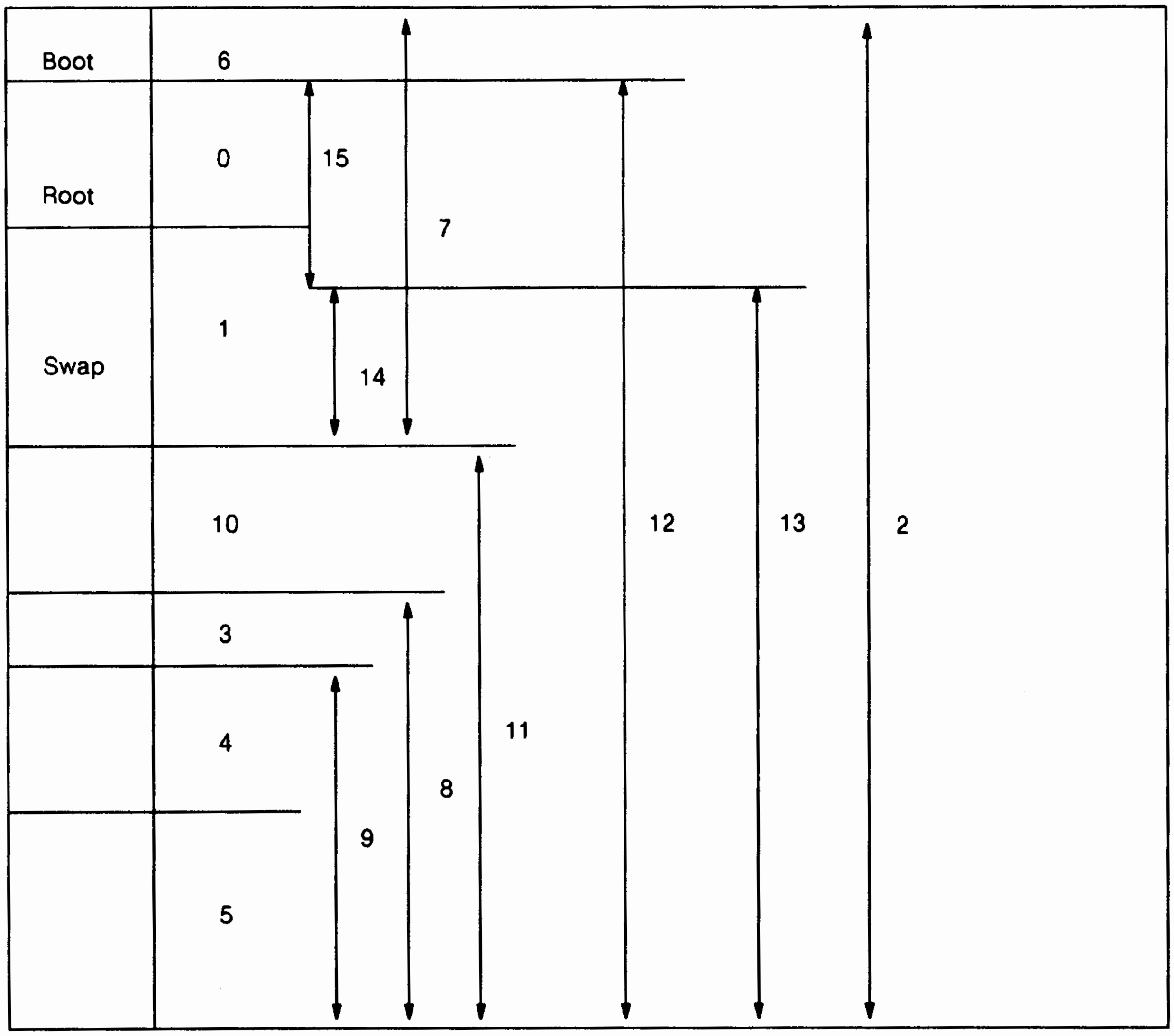


**Figure 7-5. Disk Sectioning Scheme**

Find a section that is not being used and does not overlap an existing section on the current disk. If such a section exists and it provides enough space for the new file system, create the file system in that section. Be sure to take into account that the primary swap device is sometimes left out of **/etc/checklist**. Ensure that you do not create a file system in space reserved for swapping. If there is no section available, you need to add a new disk.

When a section is available, either on the current disk or on an additional disk, you may proceed to create a new file system.

## File System Creation Steps

The steps involved in creating a new file system are listed below. Each of these steps is discussed in the following paragraphs.

1. Make sure that a device file exists for the disk where the file system is to reside.

2. Determine the parameters required for the new file system.

3. Run the appropriate file system creation process.

4. Update the **/etc/checklist** file.

5. Mount the new file system.

### Step 1

Ensure that the appropriate special files for the particular disk and section exist. If not, you must create the special file. Several programs are available for making special files: **insf**, **mksf**, and **mknod**. Programs **insf** and **mksf** are the ones normally used. Using **mknod** requires a thorough understanding of the I/O system; the other programs provide a friendlier user interface.

### Step 2

After you have chosen the disk and section where the file system is to reside, run either **/etc/newfs** or **/etc/mkfs**. The **/etc/newfs** command is a front end to **/etc/mkfs**; it is easier to use as long as the file **/etc/disktab** has an entry for the desired disk type. Both require the use of a special file.

The **/etc/newfs** command uses default values defined in **/etc/disktab**. Not all of the sections specified in **disktab** have defaults. You should examine the file to see if there are defaults and if they are suitable for your system configuration. These values may or may not be appropriate for your needs. The defaults may be overridden by supplying arguments to **newfs**. Refer to the **newfs** and **mkfs** descriptions in the *HP-UX Reference* for details. You should examine the following defaults:

1. Block and fragment size.

2. Parameter **minfree**.

3. Bytes per inode.

### Blocks and Fragments

Blocks and fragments represent the classical time/space trade-offs. The larger the block size, the greater the access speed. However, more disk space is wasted. You may use one of the following suggested block and fragment combinations (block size/fragment size).

1. **/tmp** is usually 8K/8K to allow quick access. Most files in this directory are temporary. Therefore, wasting space is not a problem here.

2. **/usr** is 8K/1K which has the median trade-off between speed and space utilization.

3. **/mnt** is usually a 4K/1K file system because files that reside here are typically small and remain for a long time.

### minfree

The value in **minfree** is the percentage of disk space reserved for the superuser when the file system fills up. It allows the superuser to reserve space for system use. The file system throughput degrades as the number of choices for free blocks is reduced. By setting **minfree** at 10%, which is the default, you are ensuring that the file system throughput will not degrade significantly.

### Bytes Per Inode

This parameter dictates the relationship between the number of data bytes on the disk and the number of inodes allocated on the disk. If you increase the number of bytes per inode, you are asking for fewer inodes. The default is to create one inode for every 2048 bytes of data space.

### Step 3

The following example creates a file system on an HP 7935 disk for section 11. The file system will have a block size of 8K and fragment size of 1K, with the default values for **minfree** and **bytes per inode**.

```
newfs -b 8192 -f 1024 /dev/rdsk/c3d0s11 hp7935
```

To set **minfree** and bytes per inode to some value, see the following example:

```
newfs -b 8192 -f 1024 -m 20 -i 4096 /dev/rdsk/c3d0s11 hp7935
```

The above example sets **minfree** at 20% and 4096 bytes per inode. Refer to Table 7-1 for a table of parameters to **newfs** and **mkfs**. This table shows the typical range and the defaults, if any.

Table 7-1. Typical Parameter Values

| Parameter | Range | Default |
|---|---|---|
| size | N/A | none |
| sectors/track | N/A | Varies with disk type |
| tracks/cylinder | Greater than 0 | Varies with disk type |
| block size | 4K or 8K | Varies with section |
| fragment size | DEV_BSIZE to block-size (in 1K increments) | Varies with section |
| cylinders/group | 1 to 32 | 16 |
| % free space | 0 to 100 | 10% |
| revolutions/second | N/A | Varies with disk type |
| bytes per inode | Greater than 1 | max (2048, frag size) |

## Redundant Superblocks

The HP-UX file system uses the concept of cylinder groups to organize related files in an efficient way. Cylinder groups contain redundant copies of the superblock which may be accessed if the primary superblock is destroyed. Note that **newfs** lists the alternate superblock locations. Record these locations and keep them available. If the primary superblock becomes corrupted, use an alternate superblock with **fsck** to repair your system.

## Step 4

After creating the file system, you need to update **/etc/checklist**. There are certain items you should be aware of.

1. If you do not want the file system mounted (or maintained with **fsck**) when the system is booted, place "ignore" in the type field. If you are going to have this file system mounted all the time, determine during which pass of **fsck** to check the file system to maintain the best performance. You should organize **/etc/checklist** so that sections on separate disks and of approximately equal sizes are checked in the same pass. Always check the root file system by itself first. From there, you can check any other file systems. Throughput is degraded if you check two sections on the same disk during the same pass.

2. Use the character device with **fsck** for file system maintenance (root is the exception). The character device is much faster than the block device. Character devices support larger requests, and do not require a memory-to-memory copy. The root file system is checked with the block device file because it is always mounted. **Never run fsck on a mounted file system other than the root**. Because certain data structures are copied into memory at mount time, **fsck** may change these structures without notifying the system. This is why you need to reboot your system if an error is found and fixed by **fsck** on the **root** volume. The **fsck** command writes its changes to disk, not to the memory-resident data structures.

### Step 5

Create a directory where the new file system will be mounted using the **mount** command. For example:

```
mkdir /mnt1
mount /mnt1
```

On Series 800 computer systems, **mkfs** does not write a boot block on the disk. A 2-megabyte section holds the boot information. **hpux-boot** writes this information into section 6 during the initial installation process. It can then be copied to other disks using **dd** to other disks. A copy of the data written into section 6 is also kept in **/usr/lib/uxbootlf**.

## The Checklist File

The **/etc/checklist** file is an ASCII file that lists all file systems and swap devices. It is described in detail in **checklist(4)** in the *HP-UX Reference*. The system refers to the checklist file in the following cases:

- If you boot the system (to a run level that runs **/etc/rc**, and then **/etc/rc** executes **mount -a**) or execute the **mount -a** command, all file systems (type "hfs") listed in **/etc/checklist** are automatically mounted.
- If you boot the system (to a run level that runs **/etc/rc**, and then **/etc/rc** executes **swapon -a**) or execute **swapon -a**, all **swap devices** (type "swap") listed in **/etc/checklist** are enabled.

Include one or more blanks between fields within entries in **/etc/checklist**. The format of an entry in **/etc/checklist** consists of the following fields:

*special_file directory type options backup_freq pass_number comment*

where:

| | |
|---|---|
| *special_file* | A required field that specifies a block special filename. |
| *directory* | Name of the root of the mounted file system; directory pathname on which the file system is mounted. |
| *type* | One of the following: |

| | |
|---|---|
| **hfs** | hierarchical file system |
| **nfs** | networked (remote) file system |
| **swap** | makes *special_file* into swap space when you also issue the **swapon(1M)** command. |
| **ignore** | marks an unused partition (one that **fsck** and **mount** will ignore) |

If *type* is **nfs**, **ignore**, or **swap**, **fsck(1M)** does not check the entry and **mount(1M)** does not attempt to mount the entry as a file system.

| | |
|---|---|
| *options* | One or more options (separated by commas): |

| | |
|---|---|
| **defaults** | use all default options (rw,suid) |
| **rw** | read-write (default) |
| **ro** | read-only |
| **suid** | set user ID execution allowed (default) |
| **nosuid** | set user ID execution not allowed |

| | |
|---|---|
| *backup_freq* | Reserved for future use by backup utilities; set to 0. |
| *pass_number* | Used by **fsck** to determine the order in which to check file systems (when -p is used): assign a pass number of 1 to the root file system and larger numbers to other file systems. Fsck ignores file systems with pass numbers of 0. |
| *comment* | Optional comment field beginning with # and ending with new-line. |

**Sample /etc/checklist Entries**

For systems installed on a 7933, 7935, or 7937 using the **disc0** driver (connected via an HP-IB interface):

```
/dev/dsk/c0d0s0     /       hfs defaults   0 1  # root disk
/dev/dsk/c0d0s3     /tmp    hfs defaults   0 2  # /tmp directory
/dev/dsk/c0d0s4     /usr    hfs defaults   0 3  # /usr directory
/dev/dsk/c0d0s5     /extra  hfs defaults   0 4  # extra space
/dev/dsk/c0d0s10    /mnt    hfs defaults   0 5  # /mnt directory
```

For systems installed on a 7936:

```
/dev/dsk/c0d0s0     /       hfs   defaults   0 1  # root
/dev/dsk/c0d0s1     /swap   swap  defaults   0 0  # swap
/dev/dsk/c0d0s3     /tmp    hfs   defaults   0 2  # /tmp
/dev/dsk/c0d0s4     /mnt    hfs   defaults   0 3  # /mnt
/dev/dsk/c0d0s5     /usr    hfs   defaults   0 4  # /usr
```

For Mid-bus/CIO systems installed on a 7914:

```
/dev/dsk/c0d0s0     /       hfs   defaults   0 1  # root
/dev/dsk/c0d0s1     /swap   swap  defaults   0 0  # swap
/dev/dsk/c0d0s3     /tmp    hfs   defaults   0 2  # /tmp
/dev/dsk/c0d0s9     /usr    hfs   defaults   0 3  # /usr
```

For systems installed on a 7937 connected via an HP-FL interface:

```
/dev/dsk/c2000d0s0  /       hfs defaults   0 1  # root
/dev/dsk/c2000d0s3  /tmp    hfs defaults   0 2  # /tmp
/dev/dsk/c2000d0s4  /usr    hfs defaults   0 3  # /usr
/dev/dsk/c2000d0s5  /extra  hfs defaults   0 4  # extra space
/dev/dsk/c2000d0s10 /mnt    hfs defaults   0 5  # /mnt
```

## Using Alternate Partitions for the Root and Swap File Systems

---

**CAUTION**

**Do not attempt this procedure unless you are an experienced system administrator and are familiar with the information in the following pages in the *HP-UX Reference*: uxgen(1M), isl(1M), hpuxboot(1M), newfs(1M), fbackup(1M), frecover(1M), lif(4), lifcp(1), mount(1M), and umount(1M).**

---

If you need more space for the root file system or the swap area, reconfigure the system by following these steps:

1. Generate a kernel with the new root (and/or swap) location specified in the **uxgen** input file. For example:

```
root  on   disc0   lu 0  section 3;

swap  on   disc0   lu 0  section 5;
```

2. Copy the existing root file system to the new location.
3. Update **hp-ux**, **/etc/checklist**, and **/etc/mnttab** in the new root file system.
4. Reboot the system but interact with ISL to select the new kernel and make sure that all changes were made correctly.
5. Modify the boot partition to automatically boot from the new root partition.

### Example

Following is a detailed example that changes the location of the **root** and **/tmp** file systems on disk 0.

Assume disk 0 is an HP 7935 with the following partitions:

```
/       on    /dev/dsk/c0d0s0          # 24 MB
swap    on    /dev/dsk/c0d0s1          # 49 MB
/tmp    on    /dev/dsk/c0d0s3          # 30 MB
/usr    on    /dev/dsk/c0d0s4          # 110 MB
/mnt    on    /dev/dsk/c0d0s5          # 54 MB
/extra  on    /dev/dsk/c0d0s10         # 132 MB
```

This example exchanges the root (/) and **/tmp** file systems because the root file system is full but the **/tmp** space is larger than needed. The example moves **root** to section 3 and **/tmp** to section 0.

**First, build a new kernel:**

1. Change directories:

```
cd /etc/conf/gen
```

2. Make a new **uxgen** input file and edit it:

| | |
|---|---|
| `cp S800 MY.S800` | *back up the existing uxgen input file* |
| `vi  MY.S800` | *edit the new uxgen input file* |

3. Change the line specifying the root location to:

```
root   on   disc0   lu 0 section 3;
```

   Save the new **uxgen** input file.

4. Now assuming that you have enough space to build the kernel:

| | |
|---|---|
| `uxgen MY.S800` | *execute uxgen using new input file* |

   This creates **/etc/conf/MY.S800/hp-ux**.

**Then, back up your system completely:**

```
/etc/fbackup -u0f /dev/rmt/0h -g /usr/adm/fbackupfiles/graphs/g1 -c
/usr/adm/fbackupfiles/fb_config
```

This example backs up all files and directories (recursively) listed in the text file **/usr/adm/fbackupfiles/graphs/g1** to tape at **/dev/rmt/0h**.

**Put the system into single-user mode:**

```
/etc/init 1
```

**Copy the root file system:**

1. Remove any unwanted files from **/tmp**.

2. Unmount the **/tmp** file system:

```
/etc/umount /dev/dsk/c0d0s3
```

3. Build a file system and create a directory for it:

```
/etc/newfs -n -v -b 8192 -f 1024 /dev/rdsk/c0d0s3 hp7935
mkdir /newroot
```

4. Mount the new file system:

```
/etc/mount /dev/dsk/c0d0s3 /newroot
/etc/mount > /graph
```

5. Edit the graphfile that frecover uses during backup:

   ```
   vi /graph
   ```

   Enter text, such as:

   ```
   i /
   e /newroot
   e /mnt
   e /usr
   e /extra
   ```

6. Now copy the root file system:

   ```
   cd /newroot
   /etc/fbackup -f - -g /graph | /etc/frecover -x -X -f -
   ```

7. Now create directories on which to mount the other file systems, such as those excluded in the graph file. From the directory **/newroot**, type:

   ```
   mkdir tmp extra mnt usr
   ```

**Update two files on the new root:**

1. Modify **/newroot/etc/mnttab** so the device name for / is c0d0s3 and the one for **/tmp** is c0d0s0.

2. Also change the device names in **/newroot/etc/checklist**.

**Bring the new kernel over:**

```
cp /etc/conf/MY.S800/hp-ux /newroot/hp-ux
```

**Reboot the system to test the new kernel and root file system:**

1. Type:

   ```
   sync
   sync
   sync
   /etc/reboot
   ```

2. Interrupt the automatic boot process. When you see:

   ```
   Autoboot from primary path enabled.
   To override, press any key within 10 seconds.
   ```

   press a key. The ISL program on partition 6 does not know that **hp-ux** was moved.

3. Boot from primary path. Answer y to the question:

```
Boot from primary boot path (Y or N)?>Y
```

4. Interact with IPL. Answer Y to the question:

   ```
   Interact with IPL (Y or N)?>Y
   ```

5. When the system displays the ISL prompt, type:

   ```
   hpux disc0(8.0.0;3)hp-ux
   ```

   The kernel (hp-ux) is now booted from the new root on section 3. The system should come up normally except that the old root file system still exists in **/tmp**. If the system does not come up normally, reboot without intervening to bring up the old kernel and root file system.

**Modify for autoboot:**

You can then modify the boot partition on disk to allow for automatic boot of the root file system from partition 3.

1. Change to a directory (*dirname*) with more than 2 MB of free space:

   ```
   cd dirname
   ```

2. Copy the lif image of the root partition to **my.boot** with a block size of 64K:

   ```
   dd  if=/dev/rdsk/c0d0s6  of=my.boot  bs=64K
   ```

3. Copy the old autoboot file into **my.AUTO** so you can edit it:

   ```
   lifcp  my.boot:AUTO  my.AUTO
   ```

4. Edit the autoboot file:

   ```
   vi my.AUTO
   ```

5. Write down the old boot path. Change the boot path in the autoboot file so it boots from Section 3:

   ```
   hpux  disc0(8.0.0;3)hp-ux
   ```

6. Delete the old autoboot file from the lif image:

   ```
   lifrm  my.boot:AUTO
   ```

7. Install the new autoboot file:

   ```
   lifcp -r -K2 -T-12289  my.AUTO  my.boot:AUTO
   ```

8. Check that the new AUTO appears below the old one:

   ```
   lifls  -l my.boot:
   ```

9. Copy the modified **my.boot** lif image back into the boot partition:

   ```
   dd  if=my.boot  of=/dev/rdsk/c0d0s6  bs=64k
   ```

10. Reboot the system:

```
/etc/reboot
```

The system should come up normally. If not, boot from the old copy of the root file system. Reboot the system. You need to interact with ISL and type in the old boot path.

**Make a new /tmp file system over the old root file system:**

1. Unmount the old root file system:

   ```
   /etc/umount /dev/dsk/c0d0s0
   ```

2. Make the **/tmp** file system:

   ```
   /etc/newfs -n -v -b 8192 -f 8192 /dev/rdsk/c0d0s0 hp7935
   ```

3. Mount the **/tmp** file system:

   ```
   /etc/mount /dev/dsk/c0d0s0 /tmp
   ```

4. Restore any files you saved from **/tmp**.

## Moving the Swap Area

This example exchanges the **/mnt** file system (Section 5) and the swap area (Section 1).

To move the swap area:

1. Build a new kernel.
2. Save files that reside in the area you want to make into the swap area.
3. Modify **/etc/checklist** and **/etc/mnttab**.
4. Reboot the system using the new kernel.
5. Restore any files you saved from the area now configured as the swap area.

**First, build a new kernel:**

1. Change directories;

   ```
   cd /etc/conf/gen
   ```

2. Make a new **uxgen** input file and edit it:

   ```
   cp S800  MY.S800              back up the existing uxgen input file
   vi MY.S800                    edit the new uxgen input file
   ```

3. Change the **uxgen** input file as follows:

```
args  on  disc0   lu 0  section 5;
dumps on  disc0   lu 0  section 5;
swap  on  disc0   lu 0  section 5;
```

Save the new **uxgen** input file.

4. Assuming there is enough space to build the kernel:

```
uxgen MY.S800
```

**Then back up /mnt:**

```
/etc/fbackup -0f /dev/rmt/0h -i /mnt
```

**Update two files:**

1. Modify **/etc/mnttab** to remove the line for **/mnt**
2. Modify **/etc/checklist** to remove the line for **/mnt**.

**Copy the new kernel:**

```
cp /etc/conf/MYS800 /hp-ux
```

**Reboot the system:**

```
sync
sync
sync
/etc/reboot
```

**Make a new /mnt file system:**

```
/etc/newfs -n -v -b 8192 -f 8192  /dev/rdsk/c0d0s1
```

**Edit two files again:**

1. Modify **/etc/mnttab** and specify **/dev/dsk/c0d0s1** as the device name for **/mnt**.
2. Modify **/etc/checklist** to specify **/dev/dsk/c0d0s1** as the device name for **/mnt**.

**Mount the /mnt file system:**

```
/etc/mount -a
```

Then restore any files you saved on tape from the former **/mnt** file system to the new **/mnt**:

```
/etc/frecover -xf /dev/rmt/0h -i /mnt
```

## Enabling Long Filenames

HP-UX file systems can be configured to support either long or short filenames. The standard filename length is 14 characters. Optionally, this limit can be increased to 255 characters. You are prompted to choose the long or short limit when you install your system (see installation instructions in Chapter 2). When configured with the standard limit, HP-UX file systems are compatible with earlier system releases that are not configured to accept long filenames, and with some other versions of UNIX.

---

**NOTE**

**You may use SAM to enable long filenames. Refer to** *System Administration Basics* **for more information.**

---

Generally, you install the system with long filenames (the 255 character limit) to gain flexibility in naming files. Also, files created on other systems that allow long filenames can be moved to your system without being renamed. Avoid long filenames if:

- You plan to use applications that read directory file information and do not use portable directory routines (like those described in **directory**(3c) in the *HP-UX Reference*). If these applications assume that directories are an array of fixed-size entries, they will not work with long filenames. To correct this, rewrite the application to correct the assumptions about directories using the **directory**(3c) routines. This way, the applications can properly parse the directory file information required by long filenames.
- Programs (with no source code available) that were developed for or compiled on releases of HP-UX that do not support long filenames will be run on the system.
- Other systems in your organization run versions of UNIX or HP-UX that impose a 14-character limit on filename length. In this environment, you may want uniformity across the systems so that files may be moved to different systems.

## Main Differences Between Long and Short Filenames

Generally, only programs that scan and read directories for files (like **ls** or **find**) are affected by your choice of long or short filenames. Other standard HP-UX commands and utilities work properly with either configuration.

To find out if a program uses the **directory**(3) routines, search for a call to a routine named "opendir" in the source code, or if the source code is not available, try this:

```
nm <program_name> | grep opendir
```

If this command results in a line that contains "opendir", then you know that your program uses the **directory**(3) routines.

The **nm(1)** command will not work if you have previously invoked the **strip(1)** command. The **strip** command removes the symbol table and line number information from object files. Refer to the *HP-UX Reference* for a full description of these commands.

In a file system that accepts only short filenames, the maximum length of a filename in a directory is 14 characters. The system accepts filenames longer than 14 characters, but truncates the filename after the fourteenth character. In a file system that accepts long filenames, the system returns an error message when a filename longer than 255 characters is passed to a system call.

Also, in a file system that accepts only short filenames, directory entries are always aligned on 32-byte boundaries because an entry always contains 32 bytes of information. In a file system that accepts long filenames, the directory entries may vary in size and are guaranteed only to be aligned on a 4-byte boundary.

The two varieties of HP-UX file systems are distinguished by different magic numbers in the superblock of the file system (see "File System Implementation" in Chapter 9 and **/usr/include/sys/fs.h** for a description of what is stored in a file system superblock).

### Applications and Long Filenames

**CAUTION**

**Make sure that your applications run on file systems that support long filenames before converting all of your file systems to avoid the risk of destroying files.**

If you are unsure whether your applications run on a system that supports long filenames, create a test file system, and then convert it using the **convertfs** utility. Create or copy sample programs and try to execute them on the test file system.

### Enabling Long Filenames on an Existing File System

If you have an HP-UX file system that allows only short filenames, use the **/etc/convertfs** utility to convert the file system to allow long filenames.

Follow these steps:

1. Back up your system before you use the **/etc/convertfs** utility. (You should do a backup before you perform any operation that alters the file system).
2. Shut your system down to single-user state (see "Shutting Down the System" in Chapter 3) as follows:

   ```
   cd /
   /etc/shutdown
   ```

3. Unmount all of your file systems:

   ```
   /etc/umount -a
   ```

4. Execute the **convertfs** utility:
```
        /etc/convertfs
```

You will receive these messages:

```
Warning:  Conversion to long filenames is irreversible and certain programs
may not work with long filenames.

Converting the file system will cause a system reboot.  The system should be
shut down into single user state and all non-root file systems should be
unmounted before this utility is run.

Do you wish to continue? [y/n]
```

If you have your system in single-user state and have all non-root file systems unmounted, answer **y**.

It will then ask you if you want to convert all of the normally mounted file systems listed in **/etc/checklist**. If you answer "no" to this prompt, **convertfs** will ask whether you want to convert each file system in **/etc/checklist**. Respond to the prompt for each file system.

The **convertfs** utility modifies the superblocks and reformats the directories in the file systems you want to convert. After modifying each file system, **convertfs** executes an **fsck** so that the file system can again be mounted.

**NOTE**

Although the **convertfs** utility allows just one (or a few) file systems to be selected for conversion to long filenames, HP recommends that you convert all or none of your normally mounted file systems in **/etc/checklist.** This prevents inconsistencies and undesired events (such as the overwriting of files) that can occur if you mix long and short filenames on the same system.

If you have converted the root file system, **convertfs** reboots the system so that the changes made to the file system superblock will not be overwritten by an update of the superblock in the system memory.

You can also execute **convertfs** with the name of the file system you want to convert:

```
    /etc/convertfs /dev/rdsk/c2d0s10
```

The **convertfs** utility converts the named file system without prompting for input.

After you reboot the system or remount the converted file systems, you may use long filenames on the converted file systems. **Newfs** and **mkfs** create new files of the same type as the root file system. If you converted the root file system, all new file systems you create allow long filenames. If you need a file system with short filenames, use the -S option to either **newfs** or **mkfs**. (Refer to either **newfs(1M)** or **mkfs(1M)** in the *HP-UX Reference* for more details.)

When you use **/etc/convertfs** to convert to long filenames, you cannot use the utility to convert back to short filenames. Find out if your applications perform properly with long filenames before you convert any of your file systems. To be sure, convert a temporary or scratch file system and run your applications on this test file system. If you must convert back to short filenames after using **/etc/convertfs**, the file system should not have any filenames longer than 14 characters. Use the **mv** (move) command to change filenames to be shorter than or equal to 14 characters. Also, the entire file system should be backed up on another media (such as a tape). Recreate the file system with short names using the -S option to **newfs** or **mkfs** and then recover the original files from the backup media. If the root file system needs to be converted back to short filenames, reinstall it from the installation tape. Be sure to save any files customized for your system so these files can be recovered after the reinstallation.

### Modifying Programs to Run with Long Filenames

Here are some potential problems and troubleshooting suggestions:

1. A program opens directories and reads the directory entries directly.

   Change the program to use directory library routines or use **getdirentries** system calls.

2. A program assumes that the maximum length of a filename (in a buffer) is 14 characters. For example, **char filename[14]** or **char filename[DIRSIZ]** (**MAXNAMLEN** should be used for the buffer size, if only a few buffers are involved).

   If you want to store more than a few filenames, enable the DIRSIZ_MACRO compilation flag. When DIRSIZ_MACRO is enabled, DIRSIZ is a macro instead of a constant of 14. The macro accepts an argument which is a pointer to a struct direct and returns the size of the filename rounded to a 4-byte boundary. You can then allocate more memory for the filenames.

3. A program includes **<dir.h>** and uses **struct direct**. The struct direct for systems that support long filenames is a variable length structure and the struct direct for systems that support only short filenames is a fixed-size structure.

   You can include **ndir.h** and use directory libraries.

4. A program assumes there is only one file system magic number. (The magic number for a system that supports long filenames is different from the magic number for a system that supports only short filenames.)

   Change the program to allow the new magic number for long filenames.

5. A program uses **MAXNAMLEN** and assumes it has a value of 14 (when you convert to long filenames, you need a **MAXNAMLEN** of 255).

Recompile the program.

6. A program uses **DIRSIZ** and assumes it is a constant of 14 (meaning the maximum filename length is 14 characters).

   Instead of **DIRSIZ**, use **MAXNAMLEN** to dictate the maximum filename length on a system that supports long filenames. (For systems that support only short filenames, use **DIRSIZ_CONSTANT** to dictate the maximum filename length.)

## Adding a New Disk

This section describes the considerations for adding a new disk to your system. Add a new disk to the system when you see that you are running out of disk space and you have purged or archived all files not currently in use. Refer to the *HP-UX Hardware Installation and Service Manual* and the *Disk Operating and Service Manual* for information on the physical installation of new disks. Refer to "Adding Disks and Cartridge Tape Drives" in Chapter 5 for steps required to add a new disk.

- You may reorganize the existing file systems so that heavily used file systems are broken up and spread across multiple devices.
- You may consider using a section on the new disk for additional swap space.
- To take advantage of the multiple pass capability of **fsck**, add an entry to **/etc/checklist** so the new file system is checked in the same pass. For example, both of the following file systems are checked during pass 3.

```
/dev/dsk/c0d0s4  /mnt  hfs     rw 0 3 #/mnt directory
/dev/dsk/c1d0s10 /usr  hfs     rw 0 3 #/usr directory
```

- Generally, you should not normally add a new disk as a single section. If you do, consider the following:

  1. A file system can only be mounted in one place in the overall file system hierarchy. A file system that covers the entire disk can only be mounted in one place. For example, if **/usr/spool** and **/mnt/usrtmp** become too large and you want to move them without changing their names, you would need two partitions.
  2. By using separate smaller sections, each devoted to a different file system, you can tune each file system differently with **tunefs** and **newfs** parameters on that file system to suit the needs of the users without affecting other file systems.
  3. If the file system is not frequently used, you will waste time running **fsck** on the entire disk.

## Mounting and Unmounting File Systems

You attach or *mount* additional and functionally independent file systems to an existing file system with the **mount** command. Remove or *unmount* independent file systems with the **umount** command.

---

**NOTE**

**You may also use SAM to mount and unmount file systems with menus. Refer to** *System Administration Basics* **for more information.**

---

You must create special files (if they do not yet exist) for the mass storage device containing the file system which is to be mounted. Refer to the "Adding/Removing Peripheral Devices" section in Chapter 5 for details. To mount a file system, you need to:

- Create a block special file
- Create a character special file
- Create a new file system on the disk (unless one already exists)

When mounted, the files in the new file system become part of the existing file system hierarchy. Files can then be created, modified, and deleted within the new file system. When you are finished with the files on that file system, it can be unmounted. Unmounting a file system removes its files from the file system hierarchy. The files themselves are untouched and remain on the mass storage medium; they may be accessed by mounting the file system again.

### To Mount a File System

Before attempting to mount a file system, be certain the mass storage device associated with the file system is powered up and on-line. If the file system is on a removable medium, insert the medium in the mass storage device at this time. Do not remove the medium until it is unmounted.

You mount a file system using the command:

```
/etc/mount sfname directory
```

where:

*sfname* is the name of the block special file associated with the device containing the file system to be mounted. For example:

```
/dev/dsk/c0d0s4
```

or the name of a remote file system in the form: *host:path*.

*directory* is the directory in which the file system is to be mounted.

If the file system was unmounted improperly or not checked for inconsistencies with **fsck**, **mount** will not be able to mount the file system. Run **fsck** on that file system before attempting to remount it.

## To Unmount a File System

Use the following procedure to unmount a file system:

1. Make sure that all files in the file system are closed and that no one is accessing any files in the file system. Attempting to unmount a file system that has open files (including your current working directory) causes the **umount** command to fail without unmounting the file system.

2. Enter the following:

   ```
   /etc/umount sfname
   ```

   where:

   | | |
   |---|---|
   | *sfname* | is the pathname of the block special file of the device containing the file system to be unmounted or the name of a remote file system in the form: *host:path*. |

   This command fails if there are open files on the file system you are attempting to unmount.

**NOTE**

**Always unmount a file system before removing it from its mass storage device. Removing a mounted file system from a mass storage device before unmounting it is likely to corrupt the file system.**

3. When the shell prompt is again displayed on your screen, it indicates that the file system is unmounted. If the file system is on a removable medium (such as a disk pack), you can now remove it safely. This assumes that no other sections on the device are mounted.

## Mounting/Unmounting File Systems

To mount all file systems with entries in **/etc/checklist**, type the following:

```
mount -a
```

To unmount all file systems in **/etc/checklist**, enter the following:

```
umount -a
```

## File Unmounting Problems – Open Files

Open files in a file system keep the file system from being unmounted. The following is a list of frequent causes of open files on a file system.

- Having your current working directory within a file system would cause an open-file failure in unmounting the file system.
- If a program stored on a file system is a shared program and the use count of that program is not zero, the file associated with the program is still open.

To determine which processes have open files on a device and who owns the process, use the following command:

```
fuser block_filename
```

Refer to **fuser(1m)** the *HP-UX Reference* for more information.

8

# Troubleshooting

This chapter describes how to deal with various situations that can occur such as a system panic, an unresponsive terminal, or a power failure. This chapter also explains some activities that diagnose the system and determine how it is behaving.

## System Panic

A system panic is a system crash, and the system must be rebooted. A system panic occurs when the operating system encounters an unexpected condition. When such a situation occurs, the system usually gives an error message commonly referred to as the panic message. System panic messages are normally displayed on the system console, giving information such as date, operating system release, system message, and stack trace.

The panic message is kept in a message buffer of the kernel. If a successful core file was left on the system after the panic occurred, it is possible to read the message buffer to examine the panic message. This is necessary if the system has been rebooted and the message is no longer on the system console.

You can use **adb** to read the kernel message buffer. Refer to the *HP-UX Reference* and the *ADB Tutorial* for details on **adb**. For example:

```
adb -k kernel corefile
msgbuf+0x8/s
```

The following is a sample panic message:

```
trap type 15 , pcsq.pcoq = 0.bbac, isr.ior = 119.1f4
 @(#)1.0 HP-UX (sys.A.B1/S800) #1:  Sat Jun 14 00:21:45 PDT 1988
 panic:  (display==0xb000, flags==0x0) Data segmentation fault

PC-Offset Stack trace (read across, most recent is 1st):
 0x0007566c  0x0007582c  0x000759ac  0x00012a88  0x0000a6ec  0x0000bbac
 0x0007a4f0  0x0005d934  0x00012fdc  0x0000a6ac
End of Stack
```

If you have a system panic, you should save the core file for analysis by HP. Use the **savecore** command (explained later in this chapter) to save the core file.

## Manual System Dump

If the operating system becomes inoperative and you are unable to regain control of the system, rebooting is always an option you may take. However, rebooting the system will not allow you to see what was occurring during the period when the system was inoperative. Another option is to create a system dump. This can be done with a method called "transfer of control".

If you have an Access Port in your system, you can use it to initiate transfer of control as discussed in the section called "Access Port" in Chapter 5. Transfer of control can only be done through the Access Port on Model 850 computers.

For a Model 840 computer, an alternative method involves setting the diagnostic switch on the Monitor card as follows:

1. Set the third (from the top) DIP switch on the system Monitor card to the right. All other switches should be set to the left.
2. Press the RESET button.
3. Make sure that the system is working correctly by observing the following LED display on the control panel:
    a. CB00 should be displayed momentarily.
    b. D004 should then be displayed.
    c. D904 should be displayed briefly.
    d. Autoboot procedure should follow.
4. After ISL, make sure that you push the third switch back to the left. Otherwise, you will not be able to reboot.
5. The system then reboots.

On a Model 825 computer, transfer control as follows:

1. Turn the key on the front panel to SOFT RESET/TC. The rightmost LED turns on, and all the others turn off.
2. The system then reboots. Then, use **savecore** to write the kernel from the swap area to a desired directory. Refer to the next section for how to use **savecore**.

## Using savecore

The **savecore** utility transfers a crash dump (core) from the swap area to a specified directory. This command is usually included in the **/etc/rc** file so that it is invoked after the system is rebooted. The entry in **/etc/rc** is typically as follows:

```
/etc/savecore /tmp/syscore
```

The entry shown assumes that the directory **/tmp/syscore** already exists.

Note that disk space might be a problem. When the core is written to a directory, the file system must have enough room for all of system memory (printed at bootup) and the kernel (to find out, use the long listing command: `ll /hp-ux`). Otherwise, the dump is truncated. If you have 24-Mb of memory, you need enough space for a 24-Mb core file plus space for the **hp-ux** file.

Writing of the dump file can be controlled with a file called **minfree**. This file can be included in the directory in which the core is to be written. This file contains the number of free 512-byte blocks to be retained in the file system when saving the crash dump. If there are fewer free blocks in the file system than the number specified in **minfree**, the core file will not be written and a message is posted to standard error.

## Power Fail Recovery

During a power failure, the internal battery backup (if your system has one) keeps your system memory active for at least fifteen minutes, depending on the system and its configuration.

When power returns, the selftest checks memory status bits to determine whether or not a power failure occurred. If one did occur, the pre-powerfail memory image is checked for integrity. If it is corrupted, the system is rebooted. If the image is intact, powerwait entries in **/etc/inittab** are launched and all other processes resume normally.

## /etc/passwd File Recovery

If the **/etc/passwd** file is corrupted, your system will not allow anyone to log in when the system is running in multiuser mode. A system prompt appears on the screen but the system responds to any entry with the message: "No Shell".

To edit **/etc/passwd**, first rename it (to **/etc/opasswd**, for example) in case you make a mistake or destroy the new file.

If you cannot log in to your system and you do not have any superuser shell running, reset the system and boot to single-user state. From this point, you can edit the password file.

## /dev/tty File Recovery

If the **/dev/tty** file is destroyed, you will not be able to complete your login sequence while in the multiuser mode. The symptom is as follows. The login prompt is displayed. When a valid login name is entered, the system will not prompt for a password and returns "login incorrect".

If you cannot login to your system and you do not have any superuser shell running, reset the system and boot to single-user state and restore **/dev/tty** by performing the following:

1. Recreate the **/dev/tty** file by:

   ```
   cd /dev
   insf -d sy
   ```

2. Set the file permission mode for directory **/dev**:

   ```
   chmod 755 /dev
   ```

3. Check the file systems for the correct file permission modes to prevent other users from changing or destroying important system files accidentally. All system files should have the same access permission as set in step 2.

## Dealing with an Unresponsive Terminal

Some common user errors, such as trying to display the contents of a binary file (for example, an executable program) on a terminal might leave the terminal in an unusable state.

If, for whatever reason, a terminal does not respond (or does not appear to respond) to commands, several solutions are available. The first is to login to another terminal and kill the shell process running on the first terminal. Use the **ps** command to find the process ID of the shell and then use the **kill** command to kill the process.

A second solution is to do a soft or hard reset of your terminal. If you do a hard reset, the tab settings might be cleared. To determine if you need to reset the tabs, type a tab. If the cursor goes to the beginning of the next line, you need to reset tabs. To reset the tabs, execute the **/usr/bin/tabs** command.

In some cases, the **tset** command will correct the problem. To use this, type the following: (You might not see anything echoed on the screen.)

```
CONTROL-J stty sane erase "^H" kill "^U" echo CONTROL-J
```

This sets the "erase" character to CONTROL-H and the "kill" character to CONTROL-U . When the screen and keyboard response returns, type:

```
tset
```

Your terminal should now exhibit correct behavior. This may not work if the terminal is in 8-bit mode.

As another option to this procedure execute the **tabs** command.

## Online Diagnostics

Online diagnostics in the HP-UX operating system allow you to diagnose your system in multiuser mode. Diagnostic functions can be performed without shutting down or rebooting the system. A protection system is also provided for the diagnostic system. Protection is the form of security levels assigned to selected users. Each security level allows users a particular set of diagnostic capabilities. Therefore, users can use only those diagnostic programs required for their applications.

HP-UX provides a set of diagnostic programs for supported I/O devices. Devices and the associated diagnostic programs can be added to your system. Details are given in Chapter 5 of this manual. The following is a sample list of diagnostic programs and the devices for which the programs can be used.

| Diagnostic Program | Devices | |
|---|---|---|
| **CS80DIAG** | CS/80 Disk Drives: | HP 7914, HP 7933, HP 7935, HP 7958, HP 7959 |
| **SS80DIAG** | SS/80 Disk Drives: | HP 9122, HP 9127 |
| **DIAG7478** | Tape Drives: | HP 7974, HP 7978 |
| **REELDIAG** | Tape Drives: | HP 7979, HP 7980 |
| **CIPERLPD** | Printers: | HP 2563, HP 2564, HP 2566, HP 2567 |
| **HPIBDIAG** | HP-IB Card:HP 27110 | |
| **HPFLDIAG** | HP-FL and Controller Card: | HP 27111 |
| **FLEXDIAG** | Fiberlink Exchange: | HP-FL disks |
| **MUXDIAG** | CIO Multiplexer Card: | HP 27140A |
| **LANDAD** | LAN IF Card: | HP 27125B |
| **AFIDAD** | AFI IF Card: | HP 27114 |
| **MEMDIAG** | System Memory: | HP 19748 |
| **GP3DDIAG** | 3-D Graphics Device: | 98720 |
| **GS2DDIAG** | 2-D Graphics Device: | 98550A, A1021A |
| **GPIODAD** | GPIO Devices: | HP 28651A |
| **HILDIAG** | HIL Master Link Controller | |
| **HPIBDAD** | HP-IB Device Adapter: | HP 28650A |

All diagnostic programs must be run from the Diagnostic User Interface program **/usr/diag/bin/sysdiag**. Refer to the *Online Diagnostics Subsystem Reference Manual* (09740-90020) for details.

## When to Run Online Diagnostics

Run online diagnostics during preventive maintenance and after system crashes. You might also want to run online diagnostics for a particular card or device when many diagnostic events (errors) occur that indicate hardware failure on that card or device. To find out if diagnostic events have occurred in the system, run the **decode** utility on file **/usr/adm/diaglog**.

The **/usr/adm/diaglog** file is the diagnostic log file. A pseudo-driver in the I/O subsystem called diag0 receives diagnostic events from all other I/O drivers in the system. Meanwhile, a daemon process, **delog**, reads an event from **diag0**, processes the event and writes it to **/usr/adm/diaglog**. In addition, **delog** accepts optional parameters to force it to generate a brief synopsis of each event. Because **/usr/adm/diaglog** contains the diagnostic information in binary form, the decode utility is used to convert this to readable information. Refer to **decode(1m)** in the *HP-UX Reference* for more information on using this utility.

The **/usr/adm/diaglog** file grows with time. When you boot the system, this file is removed because rc moves the file to OLDdiaglog. If the system has been running for a long time and disk space is getting scarce on **/usr**, you might want to move or remove the **diaglog** file.

If **delog** is killed for any reason, the message queue for **diag0** starts to fill. The queue can hold 30 messages before it fills up. When the 30th message is received, **diag0** prints a message on the system console to indicate that the message queue is full. This message is repeated each time a new message is processed and the queue has more than 29 messages. To resolve this problem, start **delog** manually.

## Diagnostic System Security

Access to the diagnostic system is structured so that users can access only those diagnostic files required for their applications. Each user can be assigned a diagnostic security system level. This assignment is contained in the file **/usr/diag/security**.

The format of the **security** file entries is as follows:

**login:n**

where:

| | |
|---|---|
| **login** | is the login name of the user. |
| **n** | is the security level between 0 and 3. |

A security level of 0 provides the most capability while a security level of 3 provides the least capability. Users not included in this file are assigned a level of 3. The **root** account is assigned a default level of 0. A sample **security** file is shown below.

```
root:0
rootc:0
glenn:1
bobk:2
roland:1
louie:2
```

## Diagnostic System Directory Structure

The file structure of the diagnostic system begins with directory **/usr/diag**. Directories under the diagnostic system root directory are described below.

| | |
|---|---|
| **/usr/diag/bin** | Contains executables for the Diagnostic User Interface program **sysdiag** and all other diagnostic programs. |
| **/usr/diag/cat/$LANG** | Contains message catalogs for **sysdiag** and other diagnostic programs (where $LANG denotes the value of the user's LANG environment variable). |
| **/usr/diag/etc** | Contains data files used by diagnostic programs. |
| **/usr/diag/install** | Contains data files describing installed diagnostic programs. |

Program **/usr/diag/bin/sysdiag** is set up with **root** as its owner. Its **setuid** bit is on. Note that it may be linked to a file called **/usr/diag/bin/dui**.

### Diagnostic Special Files

The diagnostic root directory **/usr/diag** also contains I/O device diagnostic special files. These files are different from the special files residing in **/dev**. However, they are also created by **insf** or **mksf** when special files are created for a particular I/O device. Refer to Chapter 5 for information on special files.

## System Activity Reporter

If you are having trouble with system performance, use the System Activity Reporter (SAR) to monitor and collect data on various aspects of the system and determine more information about how the system is behaving. Some of the information this utility can report on is:

- CPU utilization
- Buffer activity
- Block device activity
- TTY activity
- System call usage
- Swapping activity
- Frequency of file access
- Queue activity levels
- Interprocess communication levels

By monitoring these activity levels, you can find out where system usage is heaviest and adjust the system by altering system parameter values, if necessary.

### Sar Commands

The following commands are all part of the system activity reporter but normally, you only need to use the **sar** command:

| | |
|---|---|
| **sar(1M)** | calls **sadc** and formats requested information into a report. |
| **sadc(1M)** | a program that collects system activity data and sends the output to a file or to standard output. |
| **sa1, sa2(1M)** | are used if you want to automatically check system behavior. You can specify these commands in **crontab**. **sa1** stores data in binary form; **sa2** writes a daily report into a file. |

A related command is also available:

| | |
|---|---|
| **timex(1M)** | traces more specific system activity such as how the system is behaving during the execution of one command or a set of commands. |

These commands are all described in the *HP-UX Reference*.

## Sar Example

This example report prints disk activity for each disk drive. In the command line, 3600 is the interval in seconds (each hour) between reports, and 10 means that the data is sampled 10 times.

By executing the command:

```
sar -d 3600 10
```

the following sample report is produced.

## Sample sar Report

```
08:00:03 device    %busy  avque  r+w/s  blks/s  avwait  avserv

09:00:05 dsk-0       23    5.3      9      19   108.9    25.1
         dsk-5        3    1.5      1       2    15.4    29.6

10:00:04 dsk-0       15    4.2      5      11    89.5    27.6
         dsk-5       15    3.9      5      10    90.7    31.2

11:00:11 dsk-0       24    7.5      9      18   171.3    26.5
         dsk-5        1    1.2      0       1     7.9    38.3

12:00:08 dsk-0       81    8.3     29      58   206.2    28.1
         dsk-5        4    1.3      1       2    17.7    38.3

13:00:05 dsk-0       24    5.2      9      19   106.5    25.6
         dsk-5        2    1.5      1       1    16.5    31.5

14:00:05 dsk-0       60    7.5     20      42   189.9    29.4
         dsk-5       13    1.7      4       7    26.0    37.7

15:00:06 dsk-0       38    6.2     13      28   145.5    27.9
         dsk-5        6    1.5      2       4    16.2    32.4

16:00:04 dsk-0       19    4.4      7      14    88.7    26.0
         dsk-5        6    2.8      2       4    56.8    31.2

17:00:13 dsk-0       43    6.5     16      32   150.6    27.3
         dsk-5        8    2.0      2       5    33.6    33.2

18:00:04 dsk-0       30    3.2      6     119   103.5    47.4
         dsk-1       12    2.9      1     108   317.6   169.7
         dsk-5        2    1.3      0       1    12.6    44.7
```

The sample report contains the following fields:

- **timestamp** – The time stamp indicates when a particular activity was started on the system. The first time stamp on the sample report is 08:00:03; it indicates when the sar report was first generated.

| | |
|---|---|
| ■ **device** | – The device name. |
| ■ **%busy** | – The period of time that a device was busy servicing a transfer request. |
| ■ **avque** | – The average number of outstanding requests. |
| ■ **r+w/s** | – The number of data transfers to or from a device. |
| ■ **blks/s** | – The number of blocks transferred (in 512-byte units). |
| ■ **avwait** | – The average time (in milliseconds) that request wait in the queue. |
| ■ **avserv** | – The average time it took for a request to be serviced (for disks include seek, rotational, latency, and data transfer times). |

# System Management Concepts

This section discusses concepts you need to be familiar with to effectively manage an HP-UX system. It is not necessary for you to understand all of these concepts in depth. However, you should be aquainted with the concepts and where to locate additional information.

## The Superuser

The term **superuser** describes system users whose effective user ID equals 0. Users with effective user IDs of 0 are provided with powerful capabilities on HP-UX systems (hence the name "superuser"). Many commands and system calls can only be successfully executed by a superuser. For example, a superuser can

- Invoke any executable command in the system.
- Override any protection placed on user files.
- Modify any system configuration files.
- Add users to and remove users from the system.
- Perform other system functions.

For security reasons, some commands and system calls should only be executed by the superuser. It is unwise to tell users the **root** (or superuser) password. For example, someone using the **rtprio** command can demand a large part of CPU time and prevent others from using the system. In addition, the superuser has unrestricted access to the accounting directories and could use the **chown** or **chgrp** commands to defeat the accounting processes.

While it is not advisable to allow users full use of superuser commands, you can assign a subset of privileged commands to groups of users by using the **privileged group** feature. All user processes whose effective group ID matches the ID of the privileged group, or whose group access list contains the privileged group, can access those commands. See the **setprivgrp(1M)** entry in the *HP-UX Reference* for a list of privileges that you can assign to selected groups.

## Processes

A process is an environment in which a program executes. It includes the program's code and data, the status of open files, the value of all variables, and the current directory. Each process is identified by a unique integer value called the process ID.

A process consists of a single executing program at any given time. However, a process can create another process to execute another program concurrently and wait for its completion. A new process is created when a program executes either the **fork** or the **vfork** system call. The terms **parent process** and **child process** refer to the original process and the process which it created, respectively.

The **init** process is the parent of all processes on the system. The following sections explain the use of **fork, exec**, and **vfork** system calls for process creation from your programs.

### Using fork

When a child process is created with a **fork** system call, nearly all code and data (including virtual code and data) is copied from the parent to the child. Only shared code and data are not copied (the child process uses the same shared code as the parent process instead of creating a separate copy for itself). Thus, the child process is nearly identical to the parent process (with the exception of its process ID); it has exact copies of the parent's code, data, and current variable values.

When the **fork** system call is executed, the system must have enough free memory to duplicate the parent process or the call to **fork** fails. Once the child process is created, both processes begin execution from the completion of the call to **fork** (at the program statement immediately following the call to **fork**).

The **fork** system call returns the actual process ID of the child (a nonzero value) to the parent process, while the identical call in the child's copy of the code always returns zero. Since the process IDs returned by the **fork** system calls are distinguishable, each process can determine whether it is the parent process or the child process.

For example, suppose that a process consists of a program that tests the life of car batteries. The program has read 1000 data values from a voltmeter and is ready to print and plot the data. The program could have been written to do one task completely (such as printing the data) and then perform the other task. However, the programmer has included a **fork** system call in the program at a location after the data has been read.

When the program completes the statement containing the **fork** system call, two nearly identical processes exist. Each process examines the value returned by its **fork** system call to determine whether it is the child process or the parent process. Following the **fork** statement is a conditional branch statement that states: "If the process is the child process, it should print the data. If the process is the parent process, it should plot the data". Because of the inclusion of the **fork** statements and the conditional branch statement, both printing and plotting are done simultaneously. Each process has its own copy of the test data, so each can modify the data without affecting the other process.

## Using exec

The **exec** system call overlays separate code and data on top of the code and data of an exiting process. In this manner, a parent process can create a new process using **fork**, and subsequently execute an entirely different program via **exec**.

As an example, when writing a text editing program, you may want to permit the user to pause and list directories on the system before choosing a file to edit. One way of doing this is to **fork** a different process, and then immediately **exec** the program **ls**. You can also use **vfork** as explained in the next section.

## Using vfork

Copying a parent process's code and data to a child process can be time consuming when a large program or a large amount of data is involved. The **vfork** system call provides an alternate way to create a new process in situations where generating a separate copy of the parent process's code and data is not necessary. **vfork** differs from **fork** in that the child process borrows the parent process's memory and thread of control until the child executes either an **exec** or **exit** system call, or it terminates abnormally. The parent process is suspended while the child uses its resources.

In situations where the child process simply calls **exec**, the parent's code and data is not required by the child. If **fork** is used to create the child process, time is wasted copying the unneeded code and data. Depending on the size of the parent's code and data space, using **vfork** instead of **fork** can result in a significant performance improvement.

Like **fork**, **vfork** returns the actual process ID of the child process to the parent process and returns a zero to the child.

## Process Termination

A process terminates when any of the following occurs:

- The program that is executing in the process successfully completes.
- The process terminates itself by calling the **exit** or **_exit** system call.
- The process receives a signal for which the default action is taken (if the default action is fatal).

When a process terminates, all open files associated with the process are closed. System resources associated with the process are deallocated.

## Process Groups

A process group is a set of related processes, such as a parent process, its child processes and its children's child processes.

A process group is established when a process calls the **setpgrp** system call. The calling process becomes the **process group leader**; it and all of its future descendants (such as its child processes and grandchild processes) are members of only that process group. Process group membership is inherited by a child process. Descendants already in existence are not placed in the new process group. The **init** process is the parent process of all processes. It initially sets up process groups as it executes commands from the command field of **/etc/inittab**.

A signal sent to a process may also be sent to all other members of its process group. Typically, process groups are used to set up processes that can be signalled after an event. For example, process groups are sometimes used to ensure that when a process group leader is terminated, all members of its process group are also terminated.

### Terminal Affiliation

Process groups and process group leaders have significance in that a process group leader can become "affiliated" with a terminal. All standard input, standard output, and standard error generated by process group members is, by default, directed to the affiliated terminal (unless redirected). Affiliation is caused by an unaffiliated process group leader opening an unaffiliated terminal. Only a process group leader can become affiliated. At the time of affiliation, the process group leader cannot be affiliated with any other terminal and the terminal cannot be affiliated with any other process group. The terminal sends signals to the members of its affiliated process group in response to the interrupt or quit character, the BREAK key, or a modem hang-up signal.

A child process inherits terminal affiliation when it is created. Thus, if an unaffiliated process group leader creates a child process, the child process is unaffiliated, even if the parent process becomes affiliated later.

### Open Files in a Process

For a process to access files, it must first open them. HP-UX limits the number of files that one process can have open to 60. A process inherits all open files from the parent. Three files that are usually open are: **standard input (stdin)**, **standard output (stdout)**, and **standard error (stderr)**. When a process terminates, the system closes any files that this process has open.

## IDs

As previously mentioned, each process is assigned a process ID (a unique integer value) which identifies that process. The process also has associated with it a real user ID, a real group ID, an effective user ID, an effective group ID, a saved user ID, and a saved group ID.

A **real user ID** is an integer value that identifies the owner of a process. Similarly, a **real group ID** is an integer value that identifies the group to which the user belongs. The real group ID is a unique integer identifier that is shared by all members of a group. It enables members of the same group to share files without allowing access to these files by nongroup members. The real user ID and real group ID of a user are specified in the file **/etc/passwd** and are assigned to the user at login.

**Effective user and group IDs** allow the process executing a program to have the same privileges as the program file owner for the duration of its execution. The effective user ID and group ID are separate entities and can be set individually. The effective IDs are usually identical to the user's corresponding real IDs. However, a file can be protected so that when executed, the process's effective IDs are set equal to the real IDs of the program file owner. The new effective ID values remain in effect until the process is terminated or the effective IDs are reset.

When the process starts another program, the effective IDs are reset by an "overlaying" process. The effective IDs are reset by a call to the **setuid** system call or the **setgid** system call. The primary use of effective IDs is to allow a user to access/modify a data file and/or execute a restricted program. When the effective user ID is zero, the user can execute system calls as the superuser (described previously).

Each process also has a **group access list** associated with it. A group access list specifies all the groups to which a process belongs. A process can access the files of any group in this list as though that group were the process's effective group ID. The access list is assigned at login based on the group memberships specified in the file **/etc/logingroup**.

## File System Implementation

HP 9000 Series 800 computers implement a High-Performance File System (HFS). The *HP-UX User's Guide* supplied with your system discusses the structure of the file system at the user level and introduces some basic concepts and terms. This section expands on those concepts and provides a more detailed explanation of HFS. This information is useful when verifying, maintaining, and repairing HFS file systems. For details on how to create a file system, read the section "Creating a File System" in Chapter 7 of this manual.

The files in the HFS file system are stored on a formatted mass storage medium, usually a disk. Each file has a unique pathname. How files are stored in the HFS is explained in this chapter.

### Disk Layout

Each disk drive used for the file system has an 8K boot block. This area is reserved for system use and is not available to the file system. The rest of the disk holds the file system and swap area. Each file system begins with the primary copy of the superblock (described later) and consists of one or more cylinder groups. Each cylinder group contains a redundant copy of the superblock. See Figure 9-1 for the file system disk layout.

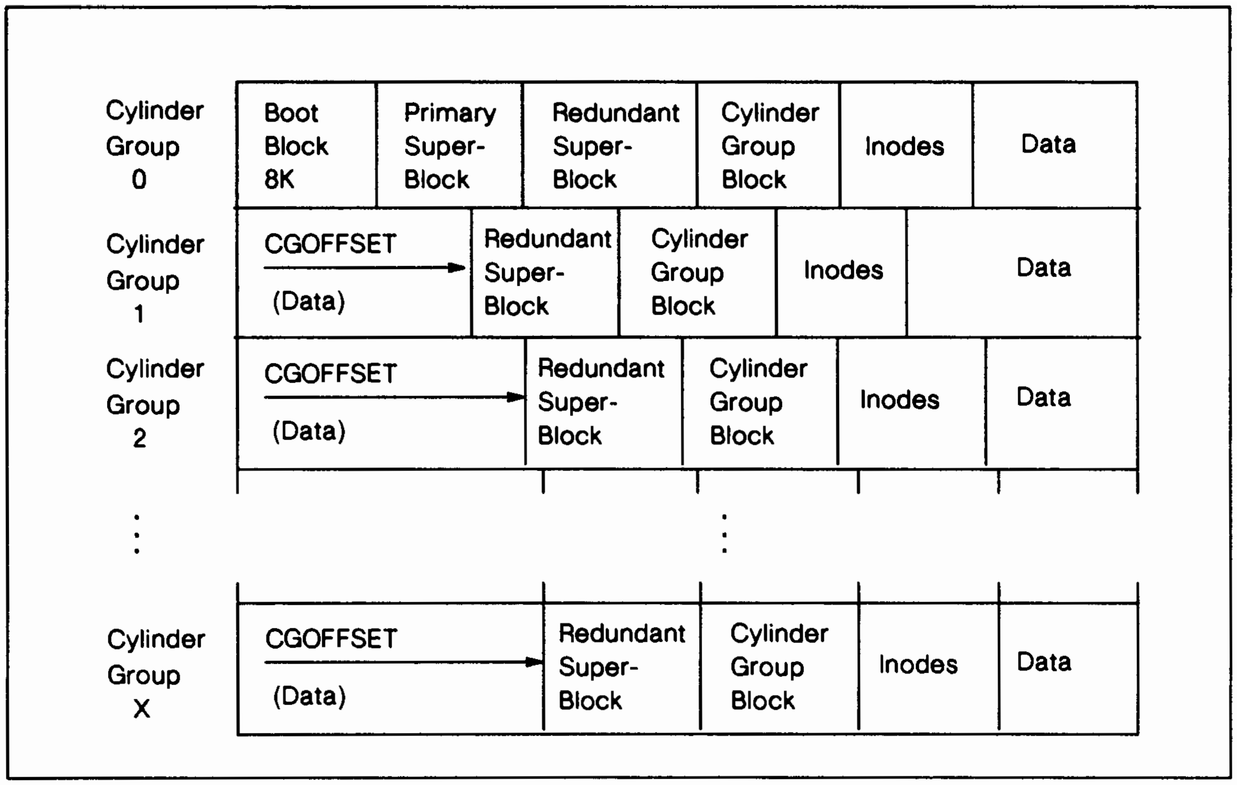


**Figure 9-1. File System Layout on Disk**

## The Cylinder Group

Each cylinder group contains a copy of the superblock, the cylinder group information structure, an inode table (explained later), and data blocks. The superblock, cylinder group information, and inode table are located in each cylinder group at varying offsets so that any single track or cylinder can be lost without losing all copies of the superblock.

If a superblock is lost, you can repair the file system using **fsck** with an alternate superblock. If major reconstruction is necessary, use **fsdb**. On a cylinder, the superblock, cylinder group block, and inode table are in contiguous blocks. All other blocks in the cylinder are used for data blocks.

A primary **superblock** is at the beginning of the file system, and a copy of the superblock is in each cylinder group. The superblock contains static information known at file system creation: block size, fragment size, disk characteristics, etc. (Fragments are portions of a block.) The fragment size is defined at file system creation. The primary superblock also keeps track of file system update information in its **summary information** area.

Eight Kbytes are reserved for each copy of the superblock. The layout of the superblock data structure is defined in the file **/usr/include/sys/fs.h**.

The **cylinder group information** contains the static and dynamic parameters of the cylinder group:

- Number of inodes and data blocks
- Pointers to the last used block, fragment, and inode
- Number of available fragments
- Used inode map
- free block map

A bit map in the cylinder group information keeps track of available data blocks and fragments. Data blocks are divided into 1-Kbyte, 2-Kbyte, 4-Kbyte, or 8-Kbyte fragments. Data block and fragment allocation are described in the "Data Storage" section.

The cylinder group information specifies a data structure size between 1 fragment and 1 block (a block can be either 4 Kbytes or 8 Kbytes). The actual size depends on the number of data blocks per cylinder group. The layout of the cylinder group information is defined in the file **/usr/include/sys/fs.h**.

The **inode table** contains per-file information (see Figure 9-2). A fixed number of inodes is allocated for each cylinder group when the file system is created. HFS uses a default that assigns more inodes per cylinder group than needed for average usage. Refer to the file **/usr/include/sys/inode.h** for more details on the inode structure.

A file system uses blocks of either 4 Kbytes or 8 Kbytes: for the rest of the discussion on inode pointers, the size of blocks are referred to as **fs_bsize**. You can replace the variable with 4K or 8K, depending upon what block size your file system uses.

NDADDR direct pointers are in an inode. If a file is larger than (NDADDR * fs_bsize) bytes, the file uses indirect blocks to hold its data. The (NDADDR+1)th pointer points to an indirect block. The next pointer (NDADDR + 2) points to a double indirect block. The (NDADDR +3)th pointer points to a triple indirect block. NDADDR is defined as 12 for HP-UX.

This limit is probably larger than most disks. Note that files cannot cross partition boundaries.

Inode pointers hold the address of a fragment. The address is interpreted as referencing a whole block or as referencing 1 or more fragments, depending on the number of bytes stored at the address. Whether a block or a fragment is used depends on the following information in the inode: the file size, file system block size, and the pointer's index number. A partial block (1 or more fragments) is allocated only at the end of a file. So, if there are 3 pointers to data, pointers 0 and 1 point to full blocks, but pointer 2 may point to a partial block.

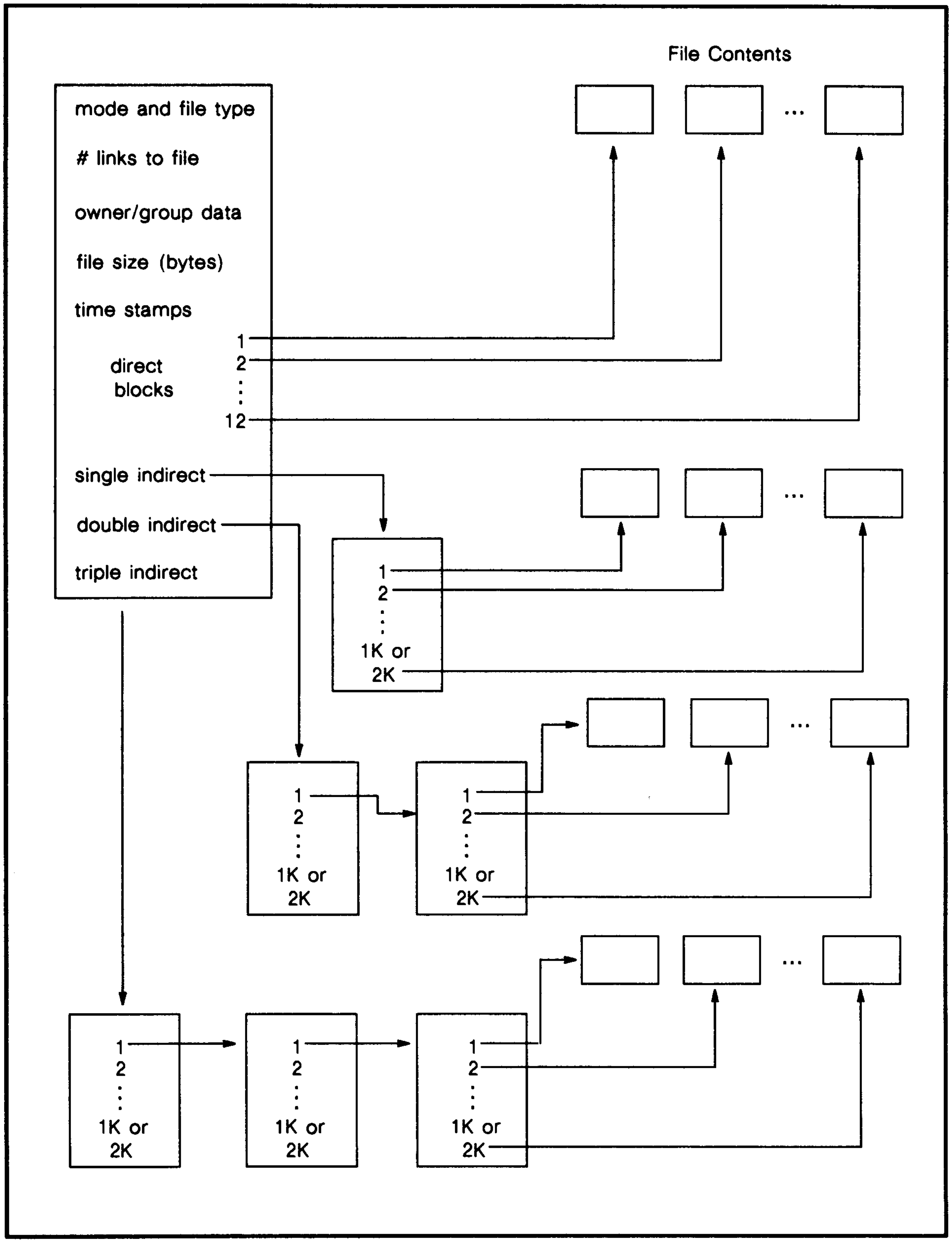


**Figure 9–2. Regular File Mapping Scheme and Inode Structure**

Figure 9-3 shows an example of a 20 Kbyte file stored in 8 Kbyte blocks with 1 Kbyte fragments. The number of blocks needed is 20 % 8 (file size % block size): 2 full blocks with a remainder of 4 fragments. Therefore, the first and second pointers point to full blocks, but the third pointer points to the remaining 4 fragments.

All indirect blocks are referenced only as full blocks; no part of the file is addressed at the fragment level beyond the 12 direct pointers.

If the file described by the inode is not a regular file, some fields of the inode are interpreted differently than shown in Figure 9-2.

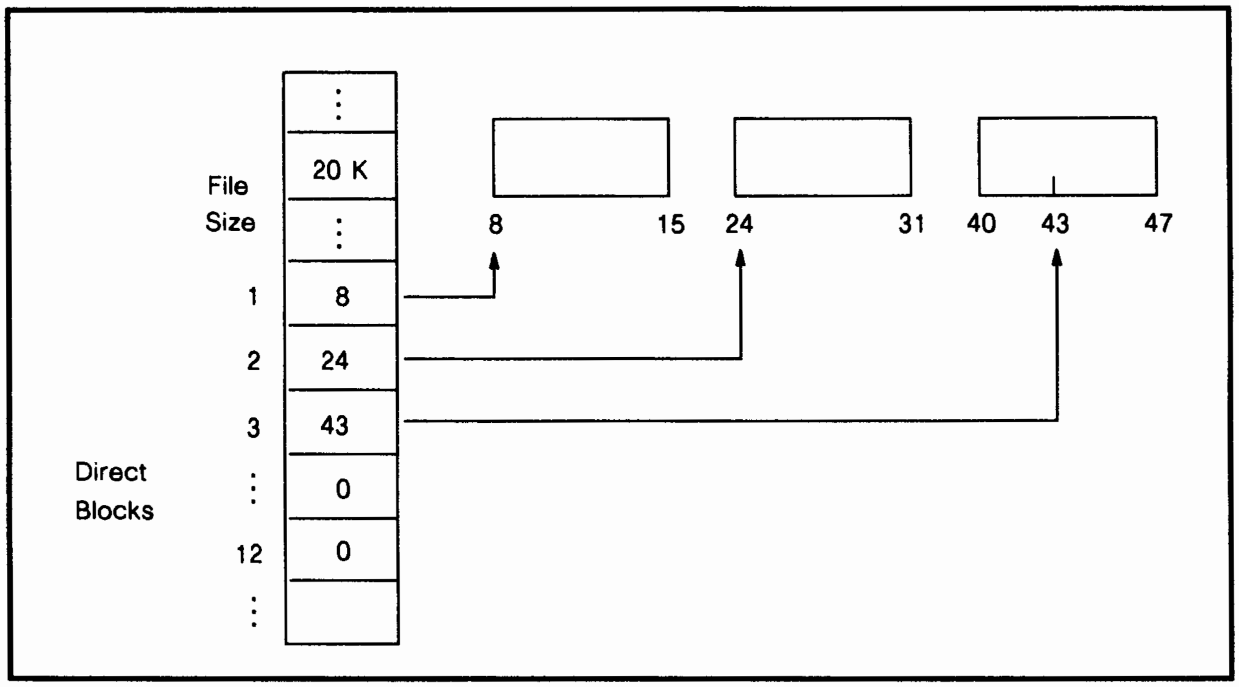


**Figure 9–3. Inode Addressing Example**

The differences are:

- FIFO and Pipes

The space reserved for indirect block pointers contain information about the current state of a FIFO or pipe.

- Character or block special files

The first direct block address is actually the major and minor number of the device. The rest of the direct block addresses are 0.

- Directory

The pointers point to regular file system data blocks, but the blocks contain specifically formatted data (see **/usr/include/sys/dir.h**).

- Symbolic Links

The first direct block address points to a regular file system data block that contains the pathname referred to by the symbolic link. (Refer to **symlink(4)** in the *HP-UX Reference* for more details on symbolic links.)

- Network special files

The pointers point to regular file system data blocks, but the blocks contain special data used by the Remote File Access (RFA) networking service.

## Data Storage

In each cylinder group, the superblock, the cylinder group block, and inode table are in contiguous blocks. The area before or after these blocks are used to store data of regular files, directories, pipes, and FIFOs (see Figure 9-1). For files referenced by indirect blocks, indirect blocks filled with pointers to data blocks also reside in this part of the cylinder group.

Free space is allocated primarily in block sizes. Blocks can be either 4 Kbytes or 8 Kbytes if you write in block size chunks. Block size is set at file system creation.

Having a large block size has both benefits and costs. In big files, a large block size significantly reduces the number of disk accesses, thereby increasing file system throughput. The problem is that most HP-UX files are small; using a large block size for small files creates wasted space. To circumvent the wasted space problem, a block can be divided into either 1, 2, 4, or 8 Kbyte fragments.

Fragment size is bounded on the lower end by DEV_BSIZE (defined in **/usr/include/sys/param.h**) and on the upper end by the block size, and must be a multiple of DEV_BSIZE. Fragment size is specified at file system creation.

### Allocation of Disk Space

Free space is determined from a bit map associated with each cylinder group. The bit map contains one bit for each fragment. To determine if a block is available, consecutive fragments are examined. A piece of the bit map from a file system using 1024 byte fragments and 8192 byte blocks is shown in Figure 9-4.

| | | | | |
|---|---|---|---|---|
| Bit map | 00000000 | 00000011 | 11111100 | 11111111 |
| Fragment numbers | 0-7 | 8-15 | 16-23 | 24-31 |
| Block numbers | 0 | 1 | 2 | 3 |

**Figure 9-4. Example Free Fragment Bit Map in an 8k/1k System**

The free fragments in this example are fragment numbers 14 to 21 and 24 to 31, indicated by **1**s in the bit map. The allocated fragments are fragment numbers 0 to 13 and 22 to 23, indicated by **0**s in the bit map. Fragments in adjacent blocks cannot be used to create a full block; only 8 contiguous fragments starting on a block boundary can be used to allocate a full block. In this example, fragments 24 to 31 can be coalesced to form a full block, but fragments 14 to 21 cannot be. Also, if a partial block is allocated, the fragments must be consecutive and not cross a block boundary. For example, if 3 fragments are needed, fragments 16 to 18 can be allocated, but fragments 14 to 16 cannot be.

In an already existing file, each time additional data needs to be written to the file, the system checks to see if file size must increase. If file size must increase, one of three conditions exists:

1. There is enough space in the existing block or fragment. In this case, the new data is written into the already allocated space.
2. The file contains only whole blocks, and there is not enough room in the last block to hold the additional data. If more than a full block of data needs to be written, a new block is allocated and the first additional block of data is written there. This process is repeated until less than a block of new data needs to be written. When this happens, a block containing enough contiguous fragments is located and the new data is written there.
3. The file contains fragments, but not enough fragments to hold the new data. If enough adjacent free fragments (within the block) exist, then they are allocated to the file. If the size of the existing data in fragments, plus the new data, exceeds the size of a full block, a new block is allocated. Both the old and the new data are written to the new block following the process in condition 2 above. If the size of the old and new data is less than a full block, a block with enough contiguous fragments (or a full block) is located and allocated.

When a block or fragment is located, the address is recorded in the inode and the free block bit map is updated.

A certain percentage of free space is kept available in a file system. This percentage is specified at file system creation using the **-m** option of the **newfs** command or the **minfree** argument of the **mkfs** command. The default is 10%. You can alter the percentage of free space at any time using the **-m** option of the **tunefs** command. The reserved free space is inaccessible to the normal user; once this threshold has been met, only the superuser can continue to allocate blocks.

## Allocation Policies

Allocation is performed on a global level to determine placement of new directories and files, and on a local level to actually place the data in blocks.

The decision of which cylinder group to put a file or directory in is made at the global level. An attempt is made to put all files from a single directory in the same cylinder group. When a new directory is created it is put in the cylinder group that has the greatest number of free inodes and the smallest number of directories in it.

Global policy first specifies that when a file exceeds a size of "NDADDR*fs_bsize", the blocks are allocated from another cylinder group. In addition, at each MAXBPG (maximum number of data blocks per cylinder group), the allocation is again directed to another cylinder group. After another cylinder group is chosen for indirect block allocation, once the file size reaches MAXBPG, HFS allocates blocks from another cylinder group (MAXBPG is defined in **/usr/include/sys/fs.h**). This helps to enforce the grouping of all files within one directory into a single cylinder group by spreading the less common larger files over several cylinder groups.

The global allocation routines call local allocation routines with requests for specific data blocks. The global information, however, is not always up to date. It is the local allocation routines, therefore, that determine which blocks to allocate. Block(s) are allocated in the following order:

1. Allocate block asked for
2. Allocate a block on the same cylinder that is closest to the requested block
3. Allocate any block within the same cylinder group
4. Use a quadratic hash to find a new cylinder group; allocate a block somewhere in the new cylinder group
5. Brute force search to find an available block

## Updating the HFS File System

Every time a file is modified, the HP-UX operating system performs a series of file system updates. These updates are designed to ensure a consistent file system.

When a program does an operation that changes the file system, such as a **write**, the data to be written is copied into an in-core buffer, called the buffer cache. The physical disk update is handled asynchronously from the buffer write. The data, along with the inode information reflecting the change, is written to the disk sometime later unless the file was opened in the synchronous mode (see **open(2)**, **O_SYNCIO**). The process is allowed to continue even though the data was not yet written to disk. If the system is halted without writing the in-core information to disk, the file system on the disk is left in an **inconsistent** state.

The superblock, inodes, data blocks, and cylinder group information is updated as explained in the following sections.

### Superblock

The superblock of a mounted file system is written to disk whenever a **umount** command is issued, or when a **sync** command is issued and the file system has been modified. The root file system is mounted during boot, and cannot be unmounted.

### Inodes

An inode contains information specific to the file it describes. An inode is written to the file system when the file associated with the inode is closed, when a **sync** or **fsync** command is issued, when the file system is unmounted, or as soon as the file is written if **O_SYNCIO** is set for the file.

### Data Blocks

In-core blocks are written to the file system whenever they are modified and released by the operating system. More precisely, they are buffered or queued for eventual writing. Physical I/O is deferred until the operating system needs the buffer, a **sync** command is issued, an **fsync** is issued for the file, or **O_SYNCIO** is set for the file. If a file is opened with the **O_SYNCIO** flag set, all writes are immediate.

### Cylinder Group Information

The cylinder group information is updated whenever a **sync** command is executed or when the system needs a buffer and the cylinder group is written.

File system inconsistency can also occur if you execute **fsck** on a mounted file system, other than the root file system. If you perform a file system check on a mounted file system, information could be in the buffer, but not yet written to the file system. A subsequent flushing of the buffer-cache could overwrite the corrections which **fsck** had just made. **Do not fsck mounted file systems**, and always reboot without syncing the system after altering the root device with **fsck**.

## Maintaining Your File System

You invest a significant amount of time when installing HP-UX and creating file systems; it is important to maintain the file system to ensure system integrity for your users. Simple daily checks and procedures, and correcting problems before they become catastrophic, will save you from remaking the entire system. More importantly, you can safeguard valuable data.

Chapter 5 describes the proper procedures for maintaining your file system. **All** the file system maintenance tasks are important. Here is a reminder of what you need to do:

- Proper system shutdown.

**Every time you shut down the system**, you should follow the procedure given in Chapter 3. **Do not simply shut off the power!**

- File system consistency check (**fsck**).

As shipped, **fsck** runs automatically if an improper shutdown is detected at bootup. Whenever you suspect there is a problem with your file system, you should run **fsck** interactively. Read Appendix C: "Using the fsck Command" for a description.

- Sync the system.

**Sync** writes information in the system I/O buffers to disk. This helps prevent corruption of the file system. A program, called **syncer**, does automatic periodic syncing for you.

- Check and understand disk usage.

Unused and large files use space on your file system. You should remind users to remove or archive large and unused files monthly, or when you are running low on disk space. Follow the procedure outlined in Chapter 5.

- Back up your system

After you install the HP-UX operating system, back up the complete system. In case of a system crash, you can recover using the backup. Your HP Service Center can also provide system recovery assistance.

## Corruption of the File System

A file system can become corrupted in a number of ways. The most common ways are **improper shut down procedures** and **hardware failures**.

### Improper System Shut Down and Startup

File systems can become corrupted when proper shut down procedures are not observed, such as:

- Not using the **reboot** or **shutdown** command to halt the CPU
- Physically write-protecting a mounted file system
- Taking a disk off the system without syncing and unmounting the file system(s) on the disk.

File systems can become further corrupted if proper startup procedures are not observed, such as:

- Not checking a file system for inconsistencies
- Not repairing inconsistencies
- Allowing a corrupted file system to be further modified can be disastrous.

### Hardware Failure

Although your HP 9000 Series 800 computer and disks are highly reliable, any piece of hardware can fail at any time. As the System Administrator, you should follow the preventative maintenance procedures outlined in the installation guides and in this manual. This may avert serious problems. Failures can be as subtle as a bad block on a disk pack, or as blatant as a non-functional disk controller.

## Detection and Correction of Corruption

If a file system is unmounted, you cannot access any files on it. However, a quiescent and unmounted file system may be checked for structural integrity by executing **fsck**. The **fsck** command checks on data which is intrinsically redundant in a file system. The redundant data is either read from the file system or computed from known values. A quiescent state is important during the checking of a file system because of the multipass nature of the **fsck** program. The **init** run-level s (single-user mode) is the only safe state to check file systems.

When an inconsistency is discovered, **fsck** reports the inconsistency. Refer to Appendix C, "Using the fsck Command", for an explanation of the actions **fsck** takes in response to these inconsistencies with different options.

### Superblock Consistency

The summary information associated with the superblock can become inconsistent. This information is highly exposed to error because every change to blocks or inodes in the file system modifies the summary information.

The superblock and its associated parts are most often corrupted when the computer is halted and the last command involving output to the file system was not a **reboot** or **shutdown** command.

The superblock can be checked for inconsistencies involving:

- file-system size
- free-block count
- free inode count

If **fsck** detects corruption in the static parameters of the primary (default) superblock, **fsck** requests the System Administrator to specify the location of an alternate superblock. The alternate superblock addresses were listed during file system creation and you should have them recorded and kept available. You can always find an alternate superblock at the block number specified by "**(SBSIZE+BBSIZE)/DEV_BSIZE**". A typical location is block number 16. If this superblock is also corrupted, you must supply the address of another superblock. The addresses of all copies of the superblock were reported during file system creation.

### File System Size

The superblock is checked for inconsistencies involving file system size, number of inodes, free block count, and the free inode count. The file system size must be larger than the number of blocks used by the superblock and the number of blocks used by the list of inodes. The file system size and layout information are critical pieces of information to the **fsck** program. While there is no way to actually check these sizes, **fsck** can check that they are within reasonable bounds. All other checks of the file system depend on the correctness of these sizes.

### Free Block Checking

A check is made to see that all the blocks in the file system can be found.

The **fsck** command **checks that all the blocks marked as free in the free block map** are not claimed by any files. When all the blocks are accounted for, a check is made to see if the number of blocks in the free block map plus the number of blocks claimed by the inodes equals the total number of blocks in the file system.

If anything is wrong with the free block maps, **fsck** rebuilds them, excluding all blocks in the list of allocated blocks.

The summary information contains a count of the total number of free blocks within the file system. The **fsck** command compares this count to the number of blocks it found free within the file system. If they don't agree, **fsck** replaces the count in the summary information with the actual free-block count.

### Inode Checking

The summary information contains a count of the total number of free inodes within the file system. The **fsck** command compares this count to the number of inodes it found free within the file system. If they do not agree, **fsck** replaces the count in the summary information with the actual free inode count.

### Inodes

An individual inode is not as likely to be corrupted as the summary information. However, because of the great number of active inodes, some inodes may become corrupted.

The list of inodes is checked sequentially starting with inode 2 (inode 0 marks unused directory entries and inode 1 is reserved for future use) and going to the last inode in the file system. Each inode can be checked for the following inconsistencies:

- Format and type
- Link count
- Duplicate blocks
- Bad blocks
- Block count

#### Format and Type

Inodes may be one of the following types:

- Regular file
- Directory
- Block special
- Character special
- Network special
- Fifo
- Symbolic link

Inodes may be found in one of three states:

- Unallocated
- Allocated
- Neither unallocated nor allocated

This last state indicates an incorrectly formatted inode. An inode can get in this state if bad data is written into the inode list through, for example, a hardware failure. The only possible corrective action is for **fsck** to clear the defective inode.

#### Link Count

Contained in each inode is a count of the total number of directory entries linked to the inode.

The **fsck** command verifies the link count **stored** in each inode by traversing the total directory structure (starting from the root directory) and calculating an **actual** link count for each inode.

If the stored link count is nonzero and the calculated link count is zero, it means that no directory entry appears for the inode. The **fsck** command can link the disconnected file to the **/lost+found** directory. If the stored and calculated link counts are nonzero and unequal, a directory entry was added or removed without updating the inode. The **fsck** command can replace the stored link count by the calculated link count.

### Duplicate Blocks

Contained in each inode is a list, or for large files, pointers to lists (indirect blocks), of all the blocks claimed by the inode.

The **fsck** command compares each block number claimed by an inode to a list of already allocated blocks. If a block number is already claimed by another inode, the block number is added to a list of duplicate blocks. Otherwise, the list of allocated blocks is updated to include the block number. If there are any duplicate blocks, **fsck** makes a partial second pass of the inode list to find the other inode that claims this block.

This condition can occur on a file system that has blocks claimed by both the free-block list and by other parts of the file system.

### Bad Blocks

Contained in each inode is a list of pointers to all the blocks claimed by the inode.

The **fsck** command checks each block number claimed by an inode for a value outside the range of the file system (lower than that of the first data block, or greater than the last block in the file system). If the block number is outside this range, the block number is a bad block number. The **fsck** command prompts you to clear the inode.

### Inode Size

Each inode contains a 64-bit (8-byte) size field. This size indicates the address of the last byte of data in the inode. This size can be checked for inconsistencies, such as the number of blocks actually used not matching that indicated by the inode size.

A directory inode within the HP-UX file system has the mode word set to directory. In file systems not configured to support long filenames (with a 255 character limit), the directory size must be a multiple of 32. This is because a directory entry always contains 32 bytes of information. However, in files systems that allow long filenames, the size of directory entries may vary. The **fsck** command warns of directory misalignment. This is only a warning because not enough information can be gathered to correct the misalignment.

You can roughly check the consistency of the size field in an inode by comparing the number of blocks specified in the size field to the actual number of blocks claimed by the inode. The **fsck** command calculates the number of blocks that should be claimed by an inode by dividing the number of characters in the file by the number of characters per block and rounding up. The **fsck** command then counts actual direct and indirect blocks associated with the inode. If the actual number of blocks does not match the computed number of blocks, **fsck** warns of a possible file-size error. This is only a warning because HP-UX does not fill in blocks in sparse data files.

### Indirect Blocks

Indirect blocks are owned by an inode. Therefore, inconsistencies in indirect blocks directly affect the inode that owns it.

Inconsistencies that can be checked are:

- Blocks already claimed by another inode
- Block numbers outside the range of the file system

Detection and correction of the inconsistencies associated with indirect blocks follow the same scheme used for direct blocks, and are done iteratively.

### Data Blocks

The three types of data blocks are:

- Ordinary data blocks that contain the information stored in a file. The **fsck** command does not check the validity of the contents of an ordinary data block.
- Directory data blocks that contain directory entries.
- Symbolic link data blocks that contain pathnames used to interpret symbolic link information.

Each directory data block can be checked for inconsistencies involving:

- Directory inode numbers pointing to unallocated inodes
- Directory inode numbers greater than the number of inodes in the file system
- Incorrect directory inode numbers for "." and ".." (current and parent directories respectively)
- Directories which are disconnected from the file system hierarchy

To remove a directory with illegal characters, find out the inode number, then remove it as follows:

```
ls -i    (This command returns the inode_number)
find . -inum inode_number -exec rm {} \;
```

If a directory entry inode number points to an unallocated inode, **fsck** can remove that directory entry.

If a directory entry inode number points beyond the end of the inode list, **fsck** can remove that directory entry. This occurs if bad data is written into a directory data block.

The directory inode number entry for "." should be the first entry in the directory data block. Its value should be equal to the inode number for the directory data block.

The directory inode number entry for ".." should be the second entry in the directory data block. Its value should be equal to the inode number for the parent of the directory entry (or the inode number of 2 if the directory is the root directory).

If the directory inode numbers for "." and ".." are incorrect, **fsck** can replace them with the correct values.

The **fsck** command checks the general connectivity of the file system. If directories are not linked into the file system, **fsck** links the directory back into the file system in the **/lost+found** directory.

### Uncorrectable File System Corruption

The **fsck** command may not be able to proceed in certain instances, such as if all copies of the superblock are lost. The **fsdb** (file system debugger) command is provided for such situations. **fsdb** should only be used by an HP-UX file system expert, because it can easily destroy the entire file system. See the **fsdb(1M)** entry in the *HP-UX Reference* for details.

You should ensure that a lost+found directory exists for each file system and that it is truncated periodically. It is created by **newfs** or **mkfs** when the file system is created. It is normally a very large directory so that **fsck** can place lost files there. If the lost+found directory is deleted from a mounted file system, create a new one, write about 100 files into it, and then delete those files. By writing 100 files into the directory and deleting them, you are creating enough empty directory slots to hold a reasonable number of lost files. The **fsck** command can link files back into existing empty directory slots but cannot enlarge a directory when all entries are filled.

## File Format and Compatibility

The format of the mass storage media on which HP-UX files are stored is the High-Performance File System (HFS).

Use the following commands to transfer files from one HP-UX (or other UNIX) system to another:

- **dump** and **restore**
- **cpio**
- **fbackup**
- **frecover**
- **tar**
- **uucp**
- LIF utilities (HP-UX to HP-UX only). These include **lifcp, lifinit.**
- Mounted file system
- LAN

## File Protection

When each file in the file system is created, it is assigned a set of file protection bits, also called the **mode** of the file. The mode determines who can read the file, write to the file, or execute a program stored in the file. Read, write, and execute permissions for a file can be set for the file's owner, all members of the group (other than the file's owner), and all other system users. These three classes of users (user, group, and other) are mutually exclusive - no member of one class of users is included in any other class of users. When a file is created, it is associated with an owner and a group ID. These values specify which user owns the file and which group has special access capability.

The default mode of a file is initially determined by **umask**, or by parameters passed to **creat, mknod,** or **mkdir**, when the file is created. You can change the mode using the **chmod** command. The mode of the file is represented using the binary form of four octal digits as shown in Figure 9-5. The figure interprets only the three least significant digits. When the most significant digit is not specified, its value is assumed to be zero.

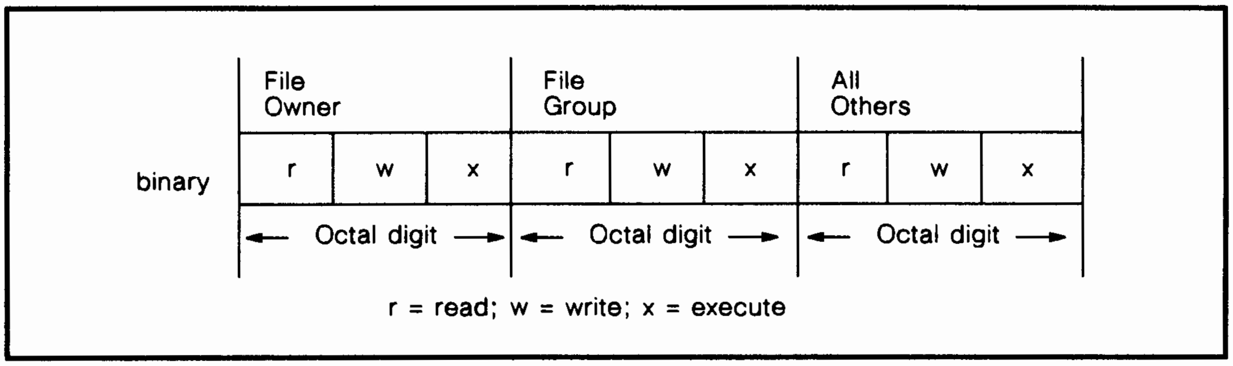


**Figure 9–5. File Protection Mode**

Each octal digit represents a 3-bit binary value: one bit specifies read permission, one bit specifies write permission, and one bit specifies execute permission. If the bit value is one, then permission is granted for the associated operation. Similarly, if the bit value is zero, permission is denied for the associated operation.

For example, assume a file has mode 754 (octal). Octal 754 is equivalent to 111 101 100 binary. From the diagram in Figure 9-6, you can see that this grants the owner of the file read, write, and execute permission. It grants read and execute permission to all users who are members of the file's group (except the file owner). That is, any user (except the file's owner) whose effective group ID is equal to the ID of the file's group, or whose group access list includes the file's group ID, may read and execute the file. It grants read permission to all other system users. The long listing command represents this mode as -rwxr-xr--.

|  | File Owner |  |  | File Group |  |  | All Others |  |  |
|---|---|---|---|---|---|---|---|---|---|
| binary | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
|  | ← | 7 | → | ← | 5 | → | ← | 4 | → |

**Figure 9-6. Sample File Protection Mode**

You can also use a **symbolic mode** to change permissions with **chmod**. To change protection using the symbolic mode, type in:

**chmod** *[who] operation permission filename [filename...]*

where:

| | |
|---|---|
| *who* | can be **u** (user), **g** (group), or **o** (other) |
| *operation* | can be + (add the following permission), - (remove the following permission), or = (assign **only** the following permission all other permissions are taken away) |
| *permission* | **r** (read), **w** (write), **x** (execute), **s** (set owner or group ID), or **t** (set sticky bit) |

*Filename* one or more files or directories whose permissions you want to change. For example, to deny write permission to others, enter:

```
chmod o-w filename
```

To make a file executable for everyone, enter:

```
chmod +x filename
```

## Protecting Directories

Directories, like all files in the HP-UX file system, have modes. The format of a directory mode is identical to the mode of an ordinary file; however, the read, write, and execute permissions are slightly different when applied to a directory. Read permission provides the ability to list the contents of a directory. Write permission provides the ability to add a file to the directory and remove a file from the directory. It does not allow a user to directly write the contents of the directory itself. This capability belongs to the HP-UX system only. Execute permission provides the ability to search a directory for a file. If execute permission is not set for a directory, the files below that directory in the file system hierarchy cannot be accessed even if you supply the correct pathname for the file.

## Setting Effective User and Group IDs

In the section discussing effective user and group IDs, you learned that a file can be protected so that when executed, the process's effective IDs are set equal to the file owner's IDs. This capability is specified through the most significant digit of the four octal file protection mode digits (previously discussed). This digit is represented by a 3-bit binary value. When its most significant bit is 1, the effective user ID of the process executing the file is set equal to the user ID of the file's owner. This bit is called the **set user ID bit**. Similarly, if the middle bit of the most significant octal digit is 1, then the effective group ID of the process executing the file is set equal to the group ID of the file's group. This bit is called the **set group ID bit**.

If the set group ID bit is on for an ordinary file, and the file does not have group execute permission, then the file is in enforced locking mode. See the section on file locking, or see **lockf(2)** in the *HP-UX Reference*.

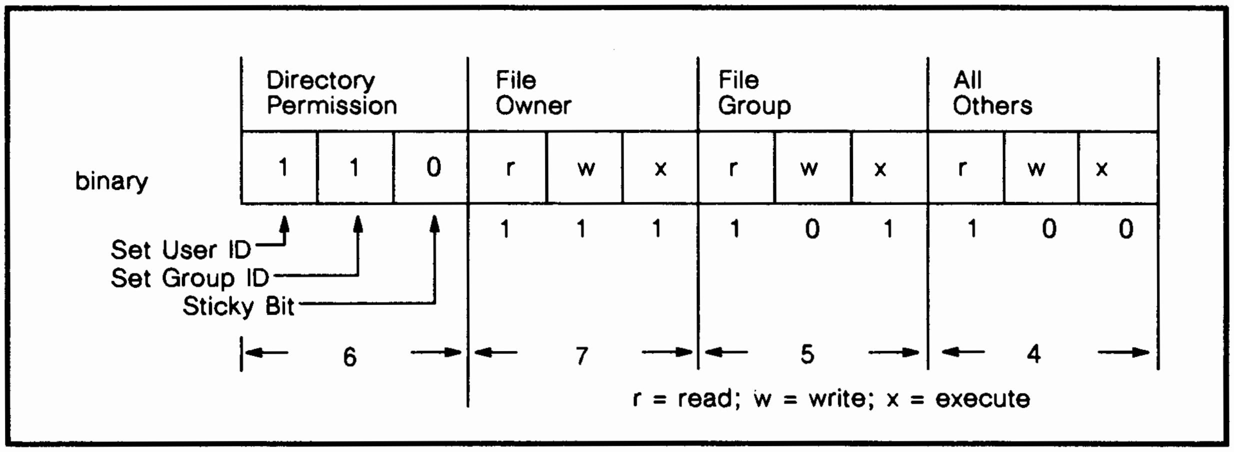


**Figure 9–7. File Protection**

Table 9–1. File Protection

| Octal Digit | Binary Form | Permission | Meaning |
|---|---|---|---|
| 6 | 1 | Set user ID | Effective user ID of the process executing this file is equal to the real user ID of the file owner. |
| | 1 | Set group ID | Effective group ID of the process executing this file is equal to the group ID of the file owner. |
| | 0 | Sticky Bit | Sticky bit is not set. Refer to the Sticky Bit section for explanation. |
| 7 | 1 | read | File owner can read the file. |
| | 1 | write | File owner can write to the file. |
| | 1 | execute | File owner can execute the file. |
| 5 | 1 | read | Members of same group can read the file. |
| | 0 | write | Members of group cannot write to the file. |
| | 1 | execute | Members of group can execute the file. |
| 4 | 1 | read | Any user can read the file. |
| | 0 | write | General users cannot write to the file. |
| | 0 | execute | General users cannot execute the file. |

## The Sticky Bit

The least significant bit of the upper octal digit is called the **sticky bit**. Only the superuser can set the sticky bit. Although it can be set for all programs, setting the sticky bit affects a program only if it is shared (refer to "Memory Management Concepts" later in this chapter). The following discussion assumes that all files marked sticky are also shared.

When the sticky bit of a program file is set and the program is executed, the code segments of the file are loaded into the swap area where they remain until you specify that they are to be removed. All users executing the file actually use the copy of the program that resides in the swap area. Setting the sticky bit reduces the amount of time needed for a number of users to access a program (since the code is transferred from the file system to the swap area only once). However, since the contents of the file remains in the swap area, the amount of swap space available for other users decreases.

Once a program is in the swap area (via the sticky bit), it can only be removed by changing the program mode so that the sticky bit is no longer set. When the file is executed again, the system recognizes the new mode and deletes the swap area copy of the file upon termination. The swap area copy is deleted by the system when the file system containing the executable program is unmounted.

## File Sharing and Locking

In a multiuser, multitasking environment such as HP-UX, it is often desirable to control interaction with files. Many applications share disk files, and the status of information contained in them could have serious implications to the user (such as lost or inaccurate information).

For example, you may be responsible for maintaining online technical reports for several projects that many different people need to access at the same time. The content of a particular report could significantly affect a company decision, so you need to control how the reports are accessed.

A potential problem arises if multiple processes can simultaneously access the same part of a file for reading while one or more processes can write to the same file. The problem is intensified when buffered I/O (for example, **libc** routines) is used to view or modify the file data.

### Advisory Locks

A solution to this problem of file sharing is called **file locking**. You can use the **lockf** and **fcntl** system calls to "lock" files. See **lockf(2)** and **fcntl(2)** for information on these calls. Advisory locks are placed on disk resources to inform (warn) other processes attempting to access these same resources that they are currently being accessed or potentially being modified. Advisory locks are only valuable for cooperating processes that are both aware of and use file locking.

In the example shown previously, the programs that access the online technical reports could use advisory locks. When George begins to work on the FubNibWitz report his program could call **lockf** or **fcntl** and set an advisory lock. A few minutes later when Sarah tries to access records in the FubNibWitz report, she would get an error message informing her that the report is busy. Her program could wait until George is done and then access the report, by virtue of doing a call to **lockf** or **fcntl**.

### Enforcement Mode

If advisory locks are used, as in the previous example, a shared file can still be overwritten. To ensure that this does not occur, use **enforcement mode**. Then, when a process attempts to read or write to a locked record in a file opened with O_NDELAY cleared, the process sleeps until the record is unlocked. An error is returned if the file is opened with O_NDELAY set.

You can only use enforcement mode on regular files. Enable enforcement mode by turning the set group ID bit on (**chmod g+s**) and turning the group execute bit off (**chmod g-x**). For example, following is a long listing of a file called FubNibWitz with its file access set to 644:

```
-rw-r--r--   1 George   LubHood          May 7 16:11 FubNibWitz
```

To turn enforcement mode on, turn the set group ID bit on (resulting in 02644). You can do this from the shell using the **chmod** command or from a program using the **chmod** system call. A long listing on the same file then shows:

```
-rw-r-Sr--   1 George   LubHood          May 7 16:11 FubNibWitz
```

**CAUTION**

**It is possible to cause a system deadlock in enforcement mode. By calling the** wait **or** pause **system calls immediately after locking a record, the locking process can hang one or more processes which attempt to access the locked record.**

### Locking Activities

As stated earlier, all file locking is controlled using the **lockf** and **fcntl** system calls. **lockf** (or **fcntl**) essentially controls the following activities:

1. Tests file accessibility by checking to see if another process has a lock on a specific file record.
2. Attempts to lock a file. If the record is already locked by another process, **lockf** puts the requesting process into a sleep state until the record is free again.
3. Tests file accessibility and locks the record if it is free, and returns immediately if it is not.
4. Unlocks a record previously locked by the requesting process. When the locking process either closes the locked file, or terminates, all locks placed by that process are removed. For more details on how specific locking activities work on HP-UX, see the **lockf(2)** entry of the *HP-UX Reference*.

## The File System Buffer Cache

The file system buffer cache is used for all file system I/O operations and for all other block I/O operations in the system (such as **exec, mount**, inode reading, and some device drivers).

The file system buffer cache contains one or more buffers that the system uses as a temporary holding place for code/data being transferred between the file system and the user's main memory. The size of each buffer and the number of buffers in the cache is determined when you power up your system, and is based upon the amount of available RAM (see the **nbuf** and **npages** entries in Appendix D). As the code and data are moved into the buffer cache, the system copies the information from the buffer cache into the user's main memory.

If a user requests information that is already in the buffer cache, the information is copied from the cache to the user's main memory, eliminating the I/O operation required to bring it in from the file system.

## The HP-UX File Structure

The file system of HP-UX is organized in a hierarchical structure. The form is similar to the root branches of a tree. At the top is a directory that is referred to as the root directory. It is represented as /. Under the root directory are certain standard directories and files created when you install the system, for example, **bin, dev, etc, lib, hp-ux, tmp, mnt,** and **usr**.

This section describes the basic purpose of the major directories and files in the HP-UX file structure. You will find this useful as you add files and modify your system in the future.

- **/bin** - contains frequently used commands, and those required to restore, recover, and/or repair the system.
- **/dev** - contains special device files used to communicate to peripherals. For more information, see **insf(1M)** and **mksf(1M)**.
- **/etc** - all system administrative commands and configuration files reside here.
- **/etc/conf** - contains object code and header files for driver generation and system reconfiguration.
- **/etc/newconfig** - new versions of configuration files and shell scripts are stored here following an update. You should keep these files intact here for future reference.
- **/lib** - frequently used object code libraries and related utilities are placed in this directory.
- **/hp-ux** - the HP-UX operating system (kernel).
- **/tmp** - a place for temporary files (those normally with short life span and which may be removed without notice).
- **/mnt** - user home directories are organized below this directory.
- **/usr** - less frequently used commands and other miscellaneous files are stored within this directory.
- **/usr/adm** - system administrative data files reside here.
- **/usr/bin** - less frequently used commands and those not required to boot, restore, recover, and/or repair the system reside here.
- **/usr/include** - high-level C language header files (shared definitions).
- **/usr/include/sys** - low-level (kernel-related) C language header files.
- **/usr/local/bin** - typically contains unsupported commands that are used locally (on this particular machine).
- **/usr/lib** - less frequently used object code libraries, related utilities, and miscellaneous data files reside here.

- **/usr/lib/uucp** – configuration files for **uucp** reside here.
- **/usr/mail** – where user mail boxes reside.
- **/usr/man** – all online documentation shipped with the system can be found here.
- **/usr/man/man1 ... man8, man1m** – contains the unformatted version of **man** pages.
- **/usr/man/cat1 ... cat8, cat1m** – contains man pages that were already processed and can be accessed faster than the unformatted versions.
- **/usr/spool/uucppublic** – used for free access of files to other systems via **uucp** or LAN.
- **/usr/spool** – spooled (queued) files for various programs.
- **/usr/spool/uucp** – queued work files, lock files, log files, status files, and other files for **uucp**.
- **/usr/spool/cron** – spooled jobs for **cron** and **at**.
- **/usr/spool/lp** – control and working files for the **lp** spooler reside here.
- **/usr/tmp** – an alternate place (to **/tmp**) for temporary files; this directory is usually used for storing many files or for large temporary files.
- **/usr/contrib** – contains any contributed files and commands (from user groups).
- **/usr/contrib/bin** – any contributed commands are placed here.
- **/usr/contrib/lib** – any contributed object libraries are placed here.
- **/usr/contrib/man** – the online documentation for any contributed files is placed in this directory.
- **/usr/news** – the system wide news is placed in this directory.
- **/usr/diag** – diagnostic tools reside in this directory.

## Memory Management

The following paragraphs describe memory management practices for your HP-UX system.

### Demand Paging

HP 9000 Series 800 computers running under the HP-UX operating system use a demand paged virtual memory management scheme. This scheme allows the logical address space of user processes to be larger than the physical memory. The process code and data are stored on disk and loaded into computer memory on demand in page increments. Programs often contain routines and code that are rarely accessed. For example, error handling routines might constitute 80 percent of some program code and yet may be rarely accessed. In the HP-UX system, when a program is executed, no memory is allocated for the unloaded pages until those pages are actually accessed. Then the page (or pages) is allocated memory space and the page is loaded from disk.

The demand paged virtual memory management scheme manages three types of resources: logical address space, physical memory and swap space. Various system parameters manage these resources. Each parameter has a predefined default that supports a broad range of user applications. You can change these parameters to make memory management more efficient for your particular requirements.

### Swapping and the Pageout Daemon

When the size of all processes and their associated data exceeds the system limit, certain processes and associated data are written to disk (swapped) to make room for other processes. A system process, the **pageout** daemon, monitors the amount of free memory and tries to keep it above a system threshold. If the system is heavily loaded and the demand for memory cannot be met, the swapping mechanism is enabled.

### Logical Address Space Management

When a process is executed, the HP-UX operating system allocates memory for its code and associated data. This space can be larger than the installed physical memory; it is called the logical address space. The logical address space is managed in segments. Each process that executes on the HP 9000 Series 800 computer system has at least three logical segments:

- code segment
- data segment
- stack segment

There are additional segments if shared memory is used.

The code segment (also known as the text segment) is write-protected and shared. Each segment handled by the system is divided into equal size pages. The page size is 2 Kb. Each page is loaded into memory on demand, that is, only when it is accessed. The user process logical address space is shown in Figure 9-8.

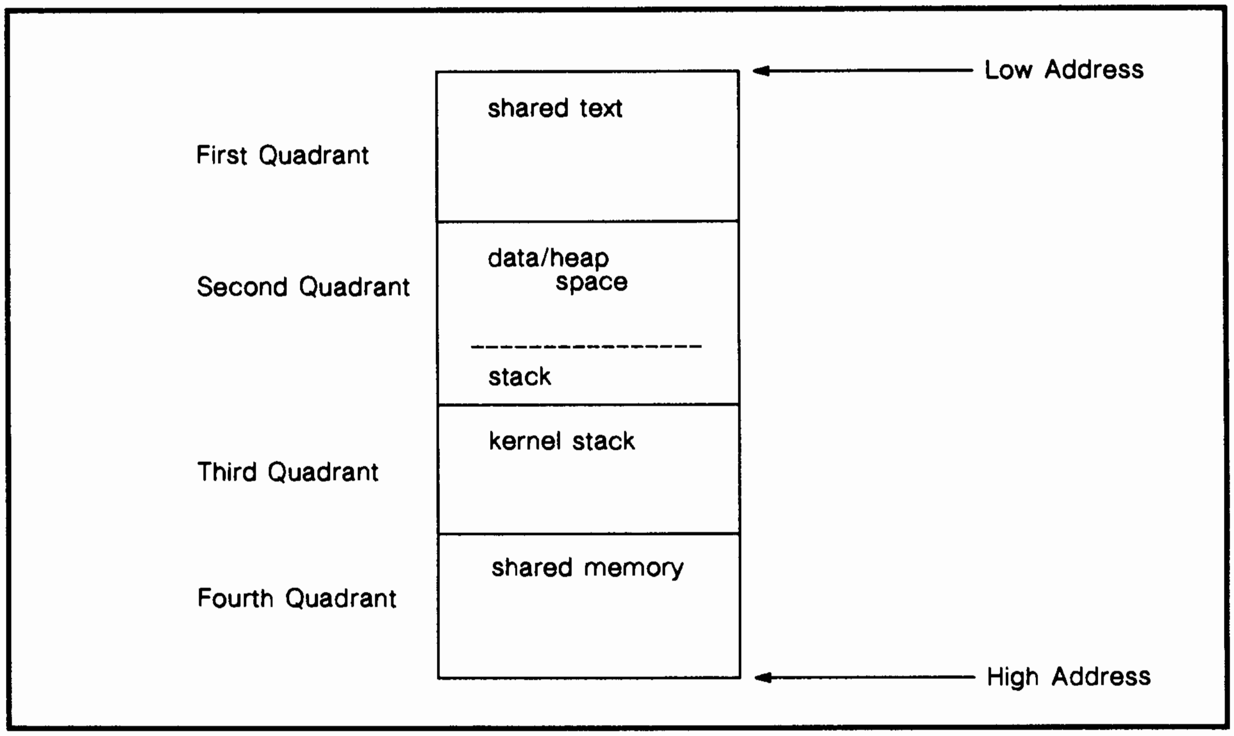


**Figure 9–8. User Process Logical Address Space**

The logical address space is addressed via 32-bit addresses. The address space is divided into four quadrants, each quadrant being one Gigabyte in size. The code (or text), data (with the dynamic heap area) and stack and shared memory segments occupy different quadrants. The code segment starts at the base of the first quadrant. The data segment starts at the base of the second quadrant with the user stack starting in the middle of the quadrant and growing toward the higher addresses.

Shared memory segments are attached to the process address space at an address chosen by the system in the fourth quadrant. Typically, **shmat(2)** is used with an address of zero. The system then returns a pointer to the shared memory segment, and the process is given access to that segment.

The data segment can be dynamically expanded into higher addresses in the second quadrant as required by a program's run-time logic. This can be done with the **brk(2)** and **sbrk(2)** system calls.

System parameters limit the size of the code, data, stack, and shared memory segments. These parameters have predefined defaults, but you can reconfigure them with **uxgen** to suit your system requirements. When these parameters are changed during **uxgen**, the maximum values for all processes are changed. Refer to

Appendix D of this manual for a detailed description of each of the system parameters. The default settings are listed below.

| | |
|---|---|
| ■ maxtsiz | Limits code segment size; default is 64 Mbytes. |
| ■ maxdsiz | Limits data segment size; default is 64 Mbytes. |
| ■ maxssiz | Limits stack segment size; default is 8 Mbytes. |
| ■ shmmax | Limits shared memory segment size; default is 64 Mbytes. |
| ■ shmseg | Limits the number of shared memory segments that can be attached; default is 12. |

## Physical Memory Utilization

The amount of memory required for your system to run effectively depends on the applications and the number of users and peripheral devices your system supports. As more users and peripheral devices are added to the system, you may need to add more memory to provide adequate performance.

When the system is powered up, it determines the amount of physical, available, and lockable memory. The system displays this information on the system console.

physical — is displayed on the system console in the form

```
memory "real mem = xxxxxx"
```

HP-UX reserves part of this memory for the operating system code and associated data structures; this part of memory cannot be divided into pages. The remaining physical memory is available for user processes.

available memory — is displayed on the system console in the form

```
"avail mem = xxxxxx"
```

The number of kernel device drivers and the size of various kernel data structures can be configured using **uxgen** to increase or decrease the user available memory. For example, a larger value of **nproc** means the user process kernel data structure must be larger. Changing configurable system parameters is discussed in Chapter 7 and the system parameters are described in Appendix D.

lockable memory — is displayed on the system console in the form

```
"lockable mem = xxxxx"
```

All or part of available memory can be locked by user processes using **plock** or **shmctl** calls.

Locked memory cannot be paged; if most of the available memory is locked, the system can become inoperable (in a deadlock situation). Some unlockable memory must be available to prevent this from occurring. The system parameter **unlockable_mem** reserves the amount of memory that cannot be locked. You may change this parameter with **uxgen**. If **unlockable_mem** is set to 0 or -1, a reasonable value is chosen by the system. This value is a function of **nproc**.

The available memory minus the memory locked by subsystems or user processes is the memory that is actually available for virtual memory system usage. When the **pageout** daemon process fails to keep up with the memory demand, swapping is turned on to select and swap some processes to disk to free up memory. Constant swapping can degrade system performance. If swapping of active processes occurs often, perhaps more physical memory should be installed. Monitoring the system activity using **ps** and **vmstat** can determine which processes are being swapped and how much overhead the **pageout** daemon is consuming.

## Swap Space Management

Swap space is a contiguous area on the secondary storage device, usually a disk drive, reserved for use by the HP-UX operating system. The system keeps a copy of all existing processes and shared memory objects there. Swap space is separate from the file system.

On HP 9000 Series 800 computers, more than one device can be used for swapping. For example, swap space may be present on several disk drives to expand the amount of swap space. Using multiple swap devices may also increase performance in some situations.

The space for the entire image of every existing segment is allocated to the swap space; therefore, swap space must be large enough to hold all segments of all existing processes. If there is insufficient swap space, the system either returns an error (such as ENOMEM for some system calls) or kills the user process. If you need more swap space, you can add more swap devices or choose a larger disk section for swapping.

The default settings of the swap space system parameters allow the image on the swap space for each of the code, data/heap, stack, and shared memory segments to be up to 64 Megabytes. If you need larger default settings for application programs, you can change them using **uxgen**, **adb**, and the information provided in Appendix D. The system parameters used are **dimmin**, **dmmax**, **dmshm**, and **dmtext.**

For example, if an application program requires a data segment of up to 120 Mb, make the following parameter value changes:

1. Increase the virtual segment limit of the data (**maxdsiz**). This parameter is changed in the input file to **uxgen**. Change **maxdsiz** to

   **maxdsiz = 0xf000**

2. Run **uxgen** to create a new **hp-ux** file. Before running the new kernel, you must change the kernel's swap block allocation sizes with the **adb** command.

For example:

```
adb -w hp-ux
        dmmin? W0d640d
        dmmax? W0d8192
```

Note that you need to modify **dmmin, dmmax, dmshm,** or **dmtext** only if you are increasing the virtual limits above 64 Megabytes. It is not necessary to use **adb** to decrease these parameter values.

Refer to the Appendix D for more information on the system parameters.

---

**NOTE**

**Larger values of** dmmin **and** dmmax **result in more fragmentation in the swap space. Do not reconfigure these values to be greater than the default settings unless it is necessary to support large application programs.**

---

The **exec** system call uses an area at the beginning of the swap space for a scratch area. While overlaying the old process image with the new process image, **exec** uses the scratch area to hold temporarily the arguments and environment variables. The size of the scratch area must be taken into consideration when reconfiguring swap space. Refer to Chapter 7, "System Reconfiguration," for details on how to compute the swap size.

## Shared Code

Often, several processes run the same program simultaneously (such as a text editor program). If such a program is not shared, each process running the program gets a copy of the program's code and data. If the processes share one copy of the code, the amount of memory and swap space required for each process space dramatically decreases.

The term "shared code" describes user code which is loaded into the text segment. When a process executes shared code, it is directed to the copy of the code in the text segment. If the shared code is not yet loaded (no other process is currently accessing the code), a text segment is first set up before the process begins execution. Only one copy of the code exists in memory regardless of the number of processes running the program.

The system knows how many processes are accessing shared code by maintaining a count (called the **use count**) of the number of processes accessing the code. When a shared program is loaded into the text segment, the use count for the program is set to one. When the process finishes executing the code, the use count is decremented.

For example, suppose that the text processor program **ed** is marked "shared". When a process first executes **ed**, its code segments are loaded into the user text area and its use count is set to one. Suppose that while the first process is executing **ed**, another process executes the **ed** program. Since the code already resides in main memory, no additional memory is allocated. The new process simply executes the copy of **ed**'s shared code; its use count is incremented from one to two. When the first process terminates the **ed** program, the system decrements the use count of **ed**'s shared code. Since the use count is not yet zero, the shared code remains in memory. Later, the second process terminates the **ed** program. The system decrements the use count and, finding it to be zero, releases **ed**'s shared code data structure, associated physical memory, and swap space.

The shared text "image" can be swapped in or out (between memory and disk) in the same manner as any other user process segments. The HP 9000 Series 800 computers support only shared code executables.

## Shared Code and the Sticky Bit

If the sticky bit is set for a file containing shared code, when the last process accessing the shared code terminates, the memory associated with the code is freed but the code still remains in the swap area.

Suppose that the file in which **ed** resides has its sticky bit set. If two different processes execute the **ed** program and then terminate, the same action occurs as previously described (under the "shared code" section) with one exception: when the use count drops to zero, the swap space allocated for the **ed** program is **not** released (and thus it is not freed for other uses).

To release shared code marked with the sticky bit, you (the superuser) have to reset the sticky bit. If you do this while the text is not being used, its swap space and associated data structures are released after execution of the next **exec** of the process.

When a file containing shared code has the sticky bit set, response time improves. This is because bringing the code in from the swap area is faster than bringing it in from the file system. The trade-off for this is that the code occupies space in the system's swap area at all times.

# Error Messages

## ISL Error Messages

The following ISL errors are shown on the hexadecimal LED displays on Model 840 and 850 computers and on the Access Port on Model 825 computers. Errors with a single digit in the error code are information messages. Other error codes (1x and 2x) are either fatal or nonfatal errors. Fatal errors cause the system to halt. The problem must be corrected and the system must be reset to recover.

| | |
|---|---|
| **Message** | **CE00** |
| **Cause** | ISL is executing. |
| **Message** | **CE01** |
| **Cause** | ISL is autobooting from the autoexecute file. |
| **Message** | **CE02** |
| **Cause** | Cannot find an autoexecute file. Autoboot aborted. ISL will be interactive. |
| **Message** | **CE03** |
| **Cause** | No console found, ISL can only autoboot. |
| **Message** | **CE05** |
| **Cause** | Directory of utilities is too big, ISL reads only 2 Kbytes. |

| | |
|---|---|
| **Message** | **CE06** |
| **Cause** | Autoexecute file is inconsistent. Autoboot aborted. ISL will be interactive. |

| | |
|---|---|
| **Message** | **CE07** |
| **Cause** | Utility file header inconsistent; object module values invalid. |
| **Action** | Rebuild the boot area with corrected utility or boot a different utility. |

| | |
|---|---|
| **Message** | **CE08** |
| **Cause** | Autoexecute file input string exceeds 2048 characters. Autoboot aborted. ISL will be interactive. |

| | |
|---|---|
| **Message** | **CE09** |
| **Cause** | ISL command or utility name exceeds 10 characters. |
| **Action** | Use ? for valid commands and utility names. |

| | |
|---|---|
| **Message** | **CE0F** |
| **Cause** | ISL has transferred control to the utility. |

| | |
|---|---|
| **Message** | **CE10** |
| **Cause** | Internal inconsistency: Volume label - FATAL. |
| **Action** | Check boot device path including media. |

| | |
|---|---|
| **Message** | **CE11** |
| **Cause** | Internal inconsistency: Directory - FATAL. |
| **Action** | Check boot device path including media. |

| | |
|---|---|
| **Message** | **CE12** |
| **Cause** | Error reading autoexecute file. ISL will be interactive. |
| **Message** | **CE13** |
| **Cause** | Error reading from console – FATAL. There is a problem in the system console hardware path. |
| **Message** | **CE14** |
| **Cause** | Error writing to console – FATAL. There is a problem in the system console hardware path. |
| **Message** | **CE15** |
| **Cause** | Not an ISL command or utility. |
| **Action** | Use ? for valid commands and utility names. |
| **Message** | **CE16** |
| **Cause** | Utility file header inconsistent: Invalid System ID. |
| **Action** | Rebuild boot area with corrected utility or boot a different utility. |
| **Message** | **CE17** |
| **Cause** | Error reading utility file header. |
| **Action** | Check boot device path including media. |
| **Message** | **CE18** |
| **Cause** | Utility file header inconsistent: Bad Magic number. |
| **Action** | Rebuild boot area with corrected utility or boot a different utility. |

|  |  |
|---|---|
| **Message** | **CE19** |
| **Cause** | Utility would overlay ISL in memory. |
| **Action** | Load utility at a higher memory address. Rebuild boot area. |

|  |  |
|---|---|
| **Message** | **CE1A** |
| **Cause** | Utility requires more memory than is configured. |
| **Action** | Increase memory on first memory controller. |

|  |  |
|---|---|
| **Message** | **CE1B** |
| **Cause** | Error reading utility into memory. |
| **Action** | Check boot device path including media. |

|  |  |
|---|---|
| **Message** | **CE1C** |
| **Cause** | Incorrect checksum: Reading utility into memory. |
| **Action** | Rebuild boot area with corrected utility or boot a different utility. |

|  |  |
|---|---|
| **Message** | **CE1D** |
| **Cause** | System console needed - FATAL. |

|  |  |
|---|---|
| **Message** | **CE1E** |
| **Cause** | Internal inconsistency: Invalid boot device class - FATAL. |

|  |  |
|---|---|
| **Message** | **CE21** |
| **Cause** | Destination memory address of utility is invalid. |
| **Action** | Rebuild boot area with corrected utility or boot a different utility. |

| | |
|---|---|
| **Message** | **CE22** |
| **Cause** | Internal inconsistency: **pdc_cache** entry. |
| **Action** | Reboot. |

| | |
|---|---|
| **Message** | **CE23** |
| **Cause** | Internal inconsistency: **iodc_entry_init** – FATAL. |
| **Action** | Reboot. |

| | |
|---|---|
| **Message** | **CE24** |
| **Cause** | Internal inconsistency: **iodc_entry_init** – console – FATAL. |
| **Action** | Reboot. |

| | |
|---|---|
| **Message** | **CE25** |
| **Cause** | Internal inconsistency: **iodc_entry_init** – boot device – FATAL. |
| **Action** | Reboot. |

| | |
|---|---|
| **Message** | **CE26** |
| **Cause** | Utility file header inconsistent: Bad **aux_id**. |
| **Action** | Rebuild boot area with corrected utility or boot a different utility. |

| | |
|---|---|
| **Message** | **CE27** |
| **Cause** | Bad utility file type. |
| **Action** | Use ? or ls(UTIL) for valid utilities on system. |

## Front Panel Codes

Series 800 HP-UX systems can display the following codes on the front panel:

**(HPUXBOOT)**

These codes describe the current state of hpuxboot.

| | |
|---|---|
| **Message** | **CB00** |
| **Cause** | Transfer-of-control initiated by the firmware (see also the D*** codes). |
| **Message** | **CEC0** |
| **Cause** | Hpuxboot has been loaded and initialization begun (realmain() has been entered). |
| **Message** | **CED0** |
| **Cause** | Hpuxboot has entered main(). |
| **Message** | **CED2** |
| **Cause** | Hpuxboot is about to configure the I/O system. |
| **Message** | **CED4** |
| **Cause** | Hpuxboot is about to mount the root file system. |
| **Message** | **CEDA** |
| **Cause** | Hpuxboot is about to list the contents of a directory. |
| **Message** | **CEDB** |
| **Cause** | Hpuxboot is about to load the kernel into memory. |

| | |
|---|---|
| **Message** | **CEDC** |
| **Cause** | Hpuxboot is about to start a copy operation. |

| | |
|---|---|
| **Message** | **CEDD** |
| **Cause** | Hpuxboot is about to stop (return to rdb). |

| | |
|---|---|
| **Message** | **CEDE** |
| **Cause** | Hpuxboot is about to return to ISL. |

| | |
|---|---|
| **Message** | **CEDF** |
| **Cause** | Hpuxboot is about to launch the kernel. |

## System Initialization

The following codes indicate the kernel is initializing the system.

| | |
|---|---|
| **Message** | **CEE0** |
| **Cause** | Kernel was loaded and initialization has begun (realmain() was entered). |

| | |
|---|---|
| **Message** | **CEF0** |
| **Cause** | Kernel has entered main(). |

| | |
|---|---|
| **Message** | **CEF2** |
| **Cause** | Kernel is about to configure the I/O system. |

| | |
|---|---|
| **Message** | **CEF4** |
| **Cause** | Kernel is about to mount the root file system. |

| | |
|---|---|
| **Message** | **CEF6** |
| **Cause** | Kernel is about to set up the page-out daemon. |

| | |
|---|---|
| **Message** | **CEF8** |
| **Cause** | Kernel is about to start the init process. |

## System Panic

The following codes are displayed if the kernel panics.

| | |
|---|---|
| **Message** | **B000** |
| **Cause** | Kernel panic. |

| | |
|---|---|
| **Message** | **B009** |
| **Cause** | Panic dump completed (disks not fully sync'ed) . |

| | |
|---|---|
| **Message** | **B00A** |
| **Cause** | Panic dump completed (disks fully sync'ed). |

## System Shutdown in Progress

When the system is shutdown (say, by using reboot or causing a transfer of control or high-priority machine check), the progress of the shutdown is monitored on the front-panel display. The first digit is a 'D', and the second indicates the progress of the shutdown.

---

**Message** **D000**

**Cause** Shutdown begun (boot() has been entered).

---

**Message** **D400**

**Cause** Shutdown in progress (returned from update(), about to wait for buffers to be flushed).

---

**Message** **D600**

**Cause** Shutdown in progress (busy-wait after update() has completed).

---

**Message** **D900**

**Cause** Shutdown completed (disks not fully sync'ed).

---

**Message** **DA00**

**Cause** Shutdown completed (disks fully sync'ed).

---

**Message** **D004**

**Cause** Transfer-of-control core dump begun.

---

**Message** **D904**

**Cause** TOC dump completed (disks not sync'ed).

---

**Message** **D010**

**Cause** High-priority machine-check core dump begun.

---

**Message** **D910**

**Cause** HPMC dump completed (disks not sync'ed).

---

## System Running

---

**Message** **F*FF**

**Cause** An 'F' in the first, third, and fourth digits implies the system is running normally. The second digit is updated every 5 seconds with the length of the run queue at that time (an instantaneous reading, NOT an average). Loads higher than 9 display as 'A'. This is designed to cover the range 0-100% in 10% increments.

---

## Booting Diagnostic Error Messages

Following are diagnostic error messages displayed on the system console. These messages are generated by **hpux**.

### General

| | |
|---|---|
| **Message** | **bad minor number in devicefile spec** |
| **Cause** | The minor number in the devicefile is illegal. |

| | |
|---|---|
| **Message** | **bad path in devicefile spec** |
| **Cause** | The hardware path in the devicefile is illegal. |

| | |
|---|---|
| **Message** | **command too complex for parsing** |
| **Cause** | The command line contains too many arguments. |

| | |
|---|---|
| **Message** | **no path in devicefile spec** |
| **Cause** | The devicefile specification does not contain a hardware path component and must. |

| | |
|---|---|
| **Message** | **panic (in hpuxboot): (display==number, flags==number) string** |
| **Cause** | A severe internal **hpux** error has occurred. Report to your nearest HP Field Representative. |

### Boot

| | |
|---|---|
| **Message** | **bad magic** |
| **Cause** | The specified object file does not have a legal magic number. |

**Message** **bad number in flags spec**

**Cause** The flags specification in the **-f** option is illegal.

---

**Message** **booting from raw character device**

**Cause** In booting from a raw device, the manager specified only has a character interface. This may cause problems if the block size is incorrect.

---

**Message** **isl not present, please hit system RESET button to continue**

**Cause** An unsuccessful boot operation has overlayed ISL in memory. It is impossible to return control to ISL without manual intervention.

---

**Message** **short read**

**Cause** The specified object file is internally inconsistent, it is not long enough.

---

**Message** **would overlay**

Loading the specified object file would overlay **hpux**.

---

## Copy

---

**Message** **cannot open destination device/file**

**Cause** The destination device or file could not be opened for writing.

---

**Message** **cannot open source device/file**

**Cause** The source device or file could not be opened for reading.

---

| | |
|---|---|
| **Message** | **fchmod failure (warning only)** |
| **Cause** | The access mode of the destination file could not be changed. |
| **Message** | **fchown failure (warning only)** |
| **Cause** | The owner and/or group of the destination file could not be changed. |
| **Message** | **fstat failure (warning only)** |
| **Cause** | One or more of the owner, group, or mode of the source file could not be determined. The default values of owner and group are 0 and 0. The default mode is 0777. |
| **Message** | **read failure** |
| **Cause** | An error was encountered reading from the source device or file. |
| **Message** | **umount failure on destination device** |
| **Cause** | The destination device could not be dismounted. Its file system may have been damaged as a result. The **fsck** command should be run before mounting the file system. |
| **Message** | **umount failure on source device** |
| **Cause** | The source device could not be dismounted. Since it was mounted read-only, the integrity of its file system is not at risk. |
| **Message** | **write failure** |
| **Cause** | An error was encountered writing to the destination device or file. |

## Configuration

---

**Message** **cannot add path, error number**

**Cause** An unknown error has occurred in adding the hardware path to the I/O tree. The internal error number is given. Contact your HP Field Representative.

---

**Message** **driver does not exist**

**Cause** The manager specified is not configured into **hpux**.

---

**Message** **driver is not a logical device manager**

**Cause** The manager name given is not that of a logical device manager and cannot be used for direct I/O operations.

---

**Message** **error rewinding device**

**Cause** An error was encountered attempting to rewind a device.

---

**Message** **error skipping file**

**Cause** An error was encountered attempting to forward-space a tape device.

---

**Message** **negative skip count**

**Cause** The skip count, if specified, must be greater than or equal to zero.

---

**Message** **no major number**

**Cause** The specified manager has no entry in the block of character device switch tables.

---

| | |
|---|---|
| **Message** | **path incompatible with another path** |
| **Cause** | Multiple incompatible hardware paths were specified. |

| | |
|---|---|
| **Message** | **path long** |
| **Cause** | The hardware path specified contains too many components for the specified manager. |

| | |
|---|---|
| **Message** | **path short** |
| **Cause** | The hardware path specified contains too few components for the specified manager. |

| | |
|---|---|
| **Message** | **table full** |
| **Cause** | Too many devices have been specified to **hpux**. |

## Booting Error Messages

The following messages may occur when you boot the system. The messages relate to the **dasetup** program which is run to determine the I/O configuration of the kernel. Some of these are *warning* messages because they do not prevent the system from being booting.

### .I/O Tree Errors

---

**Message** `/etc/dasetup: /hp-ux: iotree error: VMUNIX_OPEN`

**Cause** The system (getiotree) cannot locate **/hp-ux**.

**Action** If you have configured a new kernel, make sure that it is named **/hp-ux** and that it is readable by root.

---

**Message** `/etc/dasetup: /hp-ux: iotree error: KMEM_OPEN`

**Cause** The system (getiotree) tried to open **/dev/kmem** but couldn't.

**Action** Make sure that the file**/dev/kmem** exists and that it wasn't deleted accidentally.

---

**Message** `/etc/dasetup: /hp-ux: iotree error: KMEM_READ`

**Cause** The system (getiotree) tried to read **/dev/kmem** but couldn't.

**Action** Make sure that the file**/dev/kmem** exists and that it is a special file.

---

**Message** `/etc/dasetup: /hp-ux: iotree error: IOHEADER_NLIST`

**Cause** The system (nlist) could not locate the ioheader symbol. This indicates a serious kernel error.

**Action** Call your HP representative.

---

**Message** `/etc/dasetup: /hp-ux: iotree error: IOTREE_NLIST`

**Cause** The system (nlist) could not locate the iotree symbol in the symbol table. This indicates a serious kernel error.

**Action** Call your HP representative.

---

**Message** `/etc/dasetup: /hp-ux: iotree error: MGR_TABLE_NLIST`

**Cause** The system (nlist) could not locate the mgr_table symbol in the symbol table. This indicates a serious kernel error.

**Action** Call your HP representative.

---

**Message** `/etc/dasetup: /hp-ux: iotree error: IOTREE_NAMES_NLIST`

**Cause** The system (nlist) could not locate the iotree_names symbol in the symbol table. This indicates a serious kernel error.

**Action** Call your HP representative.

---

**Message** `/etc/dasetup: /hp-ux: iotree error: VERSION_WRONG`

**Cause** The system found that you have the wrong version of the I/O tree. This is because (1) the kernel is out of date with the version of dasetup that is running, or (2) the kernel you booted wasn't named **/hp-ux** (e.g., **ISL>hpux SYSBCKUP**).

**Action** Make sure that the kernel is named **/hp-ux** and that you didn't boot an alternate kernel. You can manually run **dasetup -d** which will prompt you for the kernel name.

---

**Message** `/etc/dasetup: /hp-ux: iotree error: NO_MEMORY`

**Cause** The system (getiotree) couldn't allocate enough memory for its internal data structure.

**Action** Call your HP representative.

---

## MUX Card Errors

The following errors are detected by the **/etc/download** program. In these messages, **/dev/muxN** indicates which MUX card is affected. For example, **/dev/mux1** means that the problem concerns MUX logical unit 1.

---

**Message** `/etc/dasetup: /dev/muxN: bad download calling sequence`

**Cause** This indicates that **/etc/download** was invoked incorrectly.

**Action** **/etc/dasetup** must invoke **/etc/download** correctly. You should not see this message unless there is a serious software problem. Call your HP representative.

---

**Message** `/etc/dasetup: /dev/muxN: EOF reading firmware file`

**Cause** This indicates that an error occurred while reading the firmware file (**/etc/UMUX.download**) that gets the MUX card to download; it could also mean that the end-of-file was encountered without reading an end-of-file firmware record. This error may occur for every MUX card configured on your system.

**Action** If you have other HP-UX systems up and running, try copying the file **/etc/UMUX.download** onto the system having the error. Make sure that you have the correct firmware for the version of the operating system you are trying to boot. Otherwise, call your HP representative.

---

**Message** `/etc/dasetup: /dev/muxN: bad record type in firmware file`

**Cause** This indicates that a firmware record with an unknown type was read from the firmware file. This error may occur for every MUX card configured on your system.

**Action** Call your HP representative.

---

**Message** `/etc/dasetup: /dev/muxN: bad checksum in firmware file`

**Cause** This indicates that the checksum information in the firmware file is incorrect; the firmware file is either corrupted or put together improperly.

**Action** Call your HP representative.

---

**Message** `/etc/dasetup: /dev/muxN: download verify failed`

**Cause** The MUX card was not able to verify the correctness of the firmware after the download was complete. This is caused by a hardware problem, a corrupt firmware file, or a firmware file that was put together improperly. If this message appears for only one of several MUX cards, that MUX may have a hardware problem.

**Action** Check the MUX mentioned in the message and make sure it is plugged in properly. Otherwise, call your HP representative.

---

**Message** `/etc/dasetup: /dev/muxN: incorrect hardware mounted`

**Cause** The I/O configuration in the **uxgen** input file (S800) does not match the machine hardware configuration. You may have used a nonstandard hardware system configuration.

**Action** Refer to the *Installation and Configuration Guide* for your computer to determine acceptable configurations. Otherwise, call your HP representative.

---

## MUX Card Driver Errors

The following errors are detected by the MUX card driver. They come from the set of errors described in **errno(2)** in the *HP-UX Reference*.

---

**Message** `/etc/dasetup: /dev/muxN: Not owner`

**Cause** This means that someone other than superuser is trying to run **dasetup**.

**Action** The message appears on the terminal of the person trying to run **dasetup**. If booting the system, make sure that you are logged on as superuser.

---

**Message** `/etc/dasetup: /dev/muxN: Mount device busy`

**Cause** The MUX card mentioned in the message is already being accesses by a download or diagnostics. It may occur after a powerfail when someone is trying to run diagnostics on the MUX card.

**Action** If running diagnostics on a MUX card, then the card is in use. Wait to reboot the system until the diagnostics are completed.

---

**Message** `/etc/dasetup: /dev/muxN: No such device or address`

**Cause** This may happen if some serious hardware failure occurs while downloading the MUX card.

**Action** Call your HP representative.

---

**Message** `/etc/dasetup: /dev/muxN: I/O Error`

**Cause** This is a warning message that often occurs during system boot. It happens if more I/O cards are configured into the S800 file than actually exist on the system. It can also happen if a minor hardware failure occurs during downloading. Often it means that either there is no MUX card plugged in or a powerfail occurred during downloading. If there was a powerfail, the card will be downloaded automatically after the powerfail.

**Action** Check to see if there is a MUX card where the system expects one. If there is no card there, ignore this warning. If there is a card there, make sure that it is plugged in tightly. Otherwise, this means that there is a hardware problem with the MUX card.

---

## Spooling Errors

The following error messages may be reported by the spooling system.

**NOTE:** "dest" represents a printer or class name

### Accept

**Message** `usage accept "dest"`

**Cause** If no arguments were given then this message is displayed

**Message** `accepting`

**Cause** This message is saved in the queue status file to indicate that the "dest" is accepting requests.

**Message** `destination "dest" non-existent`

**Cause** The specified "dest" is not accessible. The specified "dest" should have a directory to store spool requests in. This directory should be of the path **/usr/spool/lp/request/"dest"**. You need to be root or you must have read, write, and execute permission for the directory.

**Message** `destination "dest" has disappeared`

**Cause** The entry for "dest" cannot be found in the status file (qstatus).

**Message** `destination "dest" was already accepting requests`

**Cause** Message to inform you that the printer is already accepting requests

**Message** `destination "dest" now accepting requests`

**Cause** Message to confirm that the operation has been completed

**Message** `spool directory non-existent`

**Cause** An error occurred when trying to do a "chdir" to the spool directory.

---

**Message** `this command for use only by LP Administrators`

**Cause** access to this command is restricted to the users "lp" and "root".

---

## disable

---

**Message** `usage: disable [-c] [-r[reason]] printer.`

**Cause** If no arguments were given, then this message is displayed.

---

**Message** `unknown option "xx"`

**Cause** The option "xx" is not valid. Valid options are c and r.

---

**Message** `printer "dest" non-existant`

**Cause** The specified "dest" is not accessible. The specified "dest" should have a directory (**"/usr/spool/lp/request/"dest""**) to put the request in. The command disable must have read, write, and execute permission for the directory.

---

**Message** `no printers specified`

**Cause** Informative message to remind you to specify a printer.

---

**Message** `printer "dest" has disappeared!`

**Cause** The entry for "dest" cannot be found in the status file (qstatus)

---

**Message** `printer "dest" was already disabled`

**Cause** Messagae telling you that the printer was already disabled.

---

**Message** `request "ids" is cancelled`

**Cause** Message telling you that a request has been cancelled.

---

**Message** `reason unknown`

**Cause** This message is used for the reason the printer was disabled if a message is not supplied.

---

**Message** `printer "dest" now disabled`

**Cause** Message telling you the operation is completed.

---

**Message** `your printer request xxxx-yyyy was cancelled by zzzz.`

**Cause** Message mailed to you when your request is cancelled when the printer is disabled with the c option.

---

**Message** `spool directory non-existent`

**Cause** An error occurred when trying to do a "chdir" to the spool directory.

---

## enable

---

**Message** `usage:enable printer`

**Cause** If no arguments were given, then this message is displayed.

---

**Message** `enabled`

**Cause** Message telling you the printer is enabled. This message is placed in the printer status entry. This message is displayed as the reason for the printer working. This replaces the reason provided when disable is used to remove a printer from use.

---

**Message** `printer "dest" non-existent`

**Cause** The specified "dest" is not accessible. The specified "dest" should have a directory (**"/usr/spool/lp/request/"dest""**) to put the requests in. The command enable must have read, write, and execute permission for the directory.

---

**Message** `printer "dest" has disappeared!`

**Cause** The entry for "dest" cannot be found in the status file (qstatus).

---

**Message** `printer "dest" was already enabled`

**Cause** This message is to inform you that the printer was already enabled.

---

**Message** `printer "dest" now enabled`

**Cause** This message is to confirm that the operation has been completed.

---

**Message** `spool directory non-existent`

**Cause** An error occurred when trying to do a "chdir" to the spool directory.

---

## Line Printer Errors

---

**Message** `LPDEST destination "dest" illegal`

**Cause** The name “dest” in the environment variable LPDEST is greater than 14 characters.

---

**Message** `"xx" is a directory`

**Cause** The specified file “xx” is a directory. You are not allowed to print a directory.

---

**Message** `acceptance status of destination "xx" unknown`

**Cause** No entry for the specified destination “dest” could be found in the qstatus file.

---

**Message** `can't accept requests for destination "dest"`

**Cause** This message is to inform you that the destination “dest” is not accepting requests and why.

---

**Message** `can't access file "file"`

**Cause** An error occurred when performing a stat on the file “file”.

---

**Message** `can't access file "file"`

**Cause** Do not have read access for the file.

---

**Message** `can't create new sequence number file`

**Cause** Unable to create a new sequence number file (SEQLOCK) when unable to find an existing sequence number file.

---

| | |
|---|---|
| **Message** | `can't create request file "xx"` |
| **Cause** | An atempt to link the temp request file to the permanent request file "xx" failed. |

| | |
|---|---|
| **Message** | `can't lock sequence number file` |
| **Cause** | Unable to lock the sequence number file (SEQLOCK) |

| | |
|---|---|
| **Message** | `can't open default destination file` |
| **Cause** | The file that holds the system default printer name could not be opened. |

| | |
|---|---|
| **Message** | `can't open file "xx"` |
| **Cause** | Could not open file "xx" when trying to copy it to the spool directory. |

| | |
|---|---|
| **Message** | `can't read current directory` |
| **Cause** | When attempting to determine the full path to a file to spool, an error occurred in determining the current directory. |

| | |
|---|---|
| **Message** | `cannot create temp file "xx"` |
| **Cause** | Unable to create a temporary file when copying to the request directory. |

| | |
|---|---|
| **Message** | `default destination "dest" non-existent` |
| **Cause** | The specified "dest" is not accessible. The specified "dest" should have a directory to store spool requests in. This directory should be of the path **/usr/spool/lp/request/"dest"**. You need to be root or you must have read, write, and execute permission for the directory. |

| | |
|---|---|
| **Message** | `destination "dest" non-existent` |
| **Cause** | The specified "dest" is not accessible. The specified "dest" should have a directory to store spool requests in. This directory should be of the path **/usr/spool/lp/request/"dest"**. You need to be root or you must have read, write, and execute permission for the directory. |

**Message** `file "xx" is empty`

**Cause** The specified file is empty. An empty file may not be spooled.

---

**Message** `no system default destination`

**Cause** The system default destination printer hs not been specified.

---

**Message** `request id "xx" failed to enqueue`

**Cause** Could not queue the request.

---

**Message** `request id is "xxxx-yyyy"`

**Cause** Message telling you the request has been queued and what the request id is.

---

**Message** `request not accepted`

**Cause** Message informing you that your request to spool files/stdin could not be performed because you did not specify any files that were accepted and/or stdin was empty.

---

**Message** `spool directory non-existent`

**Cause** An error occurred when trying to do a "chdir" to the spool directory.

---

**Message** `standard input is empty`

**Cause** The file built from the standard input was empty. This will not be spooled.

---

**Message** `too many options for interface program`

**Cause** The option line accepts a maximum of 512 characters.

---

**Message** `unknown keyletter "?"`

**Cause** A keyletter “?” was specified that was not an alphabetic character.

---

**Message** `unknown keyletter`

**Cause** A keyletter “?” was specified that was not one of the expected keyletters. Valid keyletters are c, d, m, n, o, s, t, and w.

---

B

# System Accounting Files

This appendix describes the files processed by HP-UX System Accounting. The files are grouped by their directories.

## Files in the /usr/adm directory

| Filename | Contents |
|---|---|
| **dialog** | Output from the **delog** program. Diagnostic events reported from the I/O system. Read with the **decode** program. |
| **dtmp** | Output from the **dodisk** program. |
| **fee** | Output from the **chargefee** command (ASCII total accounting records). |
| **pacct** | The current active process accounting file. |
| **pacct*** | Process accounting files switched via **turnacct switch**. |

## Files in the /usr/adm/acct/nite directory

| Filename | Contents |
|---|---|
| **active** | Used by **runacct** to record progress. It contains warning and error messages. **activeMMDD** is the same as **active** after **runacct** detects an error. |
| **ctacct.MMDD** | Total accounting records created from connect session accounting. |
| **ctmp** | Output of **acctcon1**--connect session records. |
| **daycms** | ASCII daily command summary used by **prdaily**. |
| **daytacct** | Total accounting records for current day. |
| **disktacct** | Total accounting records created by the **dodisk** command. |
| **fd2log** | Diagnostic output from the execution of **runacct** (see **crontab** entry). |
| **lastdate** | The last day that **runacct** was executed, in **date +%m%d** format. (See **date(1)** for a description of **+%m%d** date format.) |
| **lock & lock1** | Used to control serial use of **runacct**. |

| | |
|---|---|
| **lineuse** | Terminal (tty) line usage report used by **prdaily**. |
| **log** | Diagnostics 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 the current state of **runacct**. |
| **tmpwtmp** | **wtmp** file, corrected by **wtmpfix**. |
| **wtmperror** | Error messages, if any, from **wtmpfix**. |
| **wtmperrorMMDD** | Same as **wtmperror** after **runacct** detects an error. |
| **wtmp.MMDD** | The previous day's **wtmp** file. |

## Files in the /usr/adm/acct/sum directory

| **File Name** | **Contents** |
|---|---|
| **cms** | Total command summary file for current month in internal summary format. |
| **cmsprev** | Command summary file without latest update. |
| **daycms** | Command summary file for previous day in internal summary format. |
| **loginlog** | Shows the last login date for each user. |
| **pacct.MMDD** | Concatenated version of all process accounting files for the date **MMDD**. This file is removed after reboot. |
| **rptMMDD** | Daily accounting report for date **MMDD**. |
| **tacct** | Cumulative total accounting file for current month. |
| **tacctprev** | Same as **tacct** without latest update. |
| **tacctMMDD** | Total accounting file for date **MMDD**. |
| **wtmp.MMDD** | Saved copy of wtmp file for **MMDD**. Removed after reboot. |

## Files in the /usr/adm/acct/fiscal directory

| **File Name** | **Contents** |
|---|---|
| **cmsMM** | Total command summary for month **MM** in internal summary format. |
| **fiscrptMM** | Report similar to **prdaily** for the month **MM**. |
| **tacctMM** | Total accounting file for the month **MM**. |

# Using the fsck Command

The file system consistency check (**/etc/fsck**) checks for and repairs inconsistencies in your file system.

You must have a thorough understanding of the file system before making any **fsck** decisions. Read "File System Implementation" in Chapter 9 before going on.

The **fsck** command should be performed:

- During bootup, if you did not have a clean shutdown (if you did not use **shutdown** or **reboot**).

  Run **fsck** before the system is taken to run-level 2. As shipped, your system should do this automatically if it detects an improper shutdown, via the **bcheckrc** entry in **/etc/inittab**. An improper shutdown is one where you did not use the **shutdown** command described in the "System Shutdown" section in Chapter 3.
- **Any** time you suspect problems with the HP-UX file system.

The **fsck** program, when run on the root file system, **must** use the block device (for example, **/dev/rdsk/c0d0s0**).

The **fsck** program can be run in several different modes.

**–p** Preening mode.

This option fixes many potential problems, but never removes data. When you preen the system, you are not running interactively. The **fsck** command determines what to do, and if it cannot deal with a situation, it terminates. For any inconsistencies preening mode fixes, it prints a message identifying the file system, and the corrective action taken. The preening option can fix the following inconsistencies:

- unreferenced inodes
- unreferenced pipes and fifos
- link counts in inodes too large
- missing blocks in the free list
- blocks in the free list also in files
- wrong counts in the superblock
- clean byte marked wrong

Other problems terminate **fsck –p** and prompt for manual execution of **fsck**.

**–P** Preening mode.

This option is used by **/etc/bcheckrc**. It operates in the same manner as the **–p** option except it ignores those file systems marked clean by commands such as **umount** and **reboot**.

**–y** Yes mode.

Using the **–y** option can be very dangerous. This option causes **fsck** to answer YES to all questions, which might remove data. **Do not use the –y option if you have important data on your file system** unless you have first used the **–n** option and understand the potential damage.

**–n** No mode

Using the **–n** option causes **fsck** to answer NO to all questions. This option never removes data, so it is safe. You can use the **–n** option anytime: in multiuser (though not recommended) or single-user mode, or in the background.

If you use **fsck** with the **–n** option in multiuser mode, you will probably come up with some inconsistencies due to file system action. However, you will not damage your system.

**default** Interactive mode.

The interactive mode allows you to choose whether to perform each action or not.

**–q** Quiet mode.

The **fsck** command prints only the messages that require a response.

The system should **always** be in a single-user state and quiescent (inactive and not being written on) with all file systems unmounted before executing the **fsck** command. The only exception is the root which is always mounted. Use **fsck** in run level **s**, after the **shutdown** command is executed. Running **fsck** when there is file system activity can cause loss of data.

The **fsck** command should be executed using a character special device file, not a block special device file, except for the root file system. Refer to the section "Adding and Removing Peripheral Devices" in Chapter 5 for a discussion on block and character devices, and naming conventions for device files.

Only the System Administrator should run **fsck**. If this check discovers an inconsistency, corrective action must be taken.

Before running **fsck**, make sure that a directory called **/lost+found** exists on the file system you plan to examine. One **/lost+found** should be created for each file system when you installed HP-UX, or when you ran **newfs** or **mkfs**. The **fsck** command uses this directory to place any problem files or directories that it finds. After you run **fsck**, examine the files placed in **/lost+found** and move them back where they belong or remove them. You should clear the **/lost+found** directory before executing **fsck** again.

To place these files, follow this procedure:

1. Mount the file system.

2. Change to the **lost+found** directory (cd **/lost+found**).

3. Find out what type of file it is (executable, text, etc.) and who owns it by typing:

   ```
   file *
   ll *
   ```

If the file is text, you can examine its contents by typing "more filename" where filename is the name of the file.

4. If the file is executable, you can try one of two things:
   a. If the file has an SCCS ID string, the **what** command will list it.
   b. If the file does not have an SCCS ID string, use the **strings** command to print the literal strings from the file. The strings (such as error message strings) might help identify the owner.
5. From this information, determine where the file belongs, or who it belongs to, and move the file to the correct directory.

## How fsck Handles Inconsistencies

The **fsck** command is a multi-pass file system check program. Each phase of the **fsck** program invokes a different file system pass. After the initial setup, **fsck** performs successive passes over each file system, checking blocks and sizes, path names, connectivity, reference counts, and the free block map (possibly rebuilding it), and doing some cleanup.

Refer to the beginning of this appendix for a description of the different modes in which you can run **fsck**.

When an inconsistency is detected while running interactively, **fsck** reports the error condition. If a response is required, **fsck** prints a prompt message and waits for a response. When preening, **fsck** will choose a response and note it on the screen. In this section, each error message and possible responses are presented.

The error conditions are organized by the phase of the **fsck** program in which they can occur. The error conditions that can occur in more than one phase are discussed in "Initialization Phase Errors" below.

### Initialization Phase Errors

During the initialization phase and before the file system check is performed, tables have to be set up and certain files opened. This section lists error conditions resulting from command line options, memory requests, opening of files, status of files, file system size checks, and creation of the scratch file. All of the initialization errors are fatal if you are preening. See the **fsck(1M)** entry in the *HP-UX Reference* for further information.

#### "C" option?

The character represented by **C** is not a legal option for **fsck**. Legal options are **-b**, **-y**, **-n**, **-q**, **-P**, and **-p**. The **fsck** command terminates on this error condition. See the **fsck(1M)** entry in the *HP-UX Reference* for further information.

**cannot alloc NNN bytes for "XXX"**

**XXX** is either **blockmap, freemap, statemap,** or **lncntp**. The **fsck** command's request for memory failed. This should never happen; **fsck** terminates on this error condition. Contact your local HP Sales and Service Office for assistance.

**Can't open checklist file:**

The default file system checklist file (**/etc/checklist**) cannot be opened for reading. The **fsck** command terminates on this error condition. Check for the existence of the file and the access modes of the file.

**Can't stat root**

The **fsck** command's request for statistics about the root directory (/) failed. This should never happen; **fsck** terminates on this error condition. Contact you local HP Sales and Service Office for assistance.

**Can't stat ...**
**Can't make sense out of ...**

The **fsck** command's request for statistics about the file system failed. When running manually, it ignores this file system and continues checking the next file system given. If this happens, check for the existence and the access modes of the file system.

**"file_system_name" is not a block or character device; OK?**

You have given **fsck** a regular file name by mistake. You should check the file type of the file system. Possible responses to the "OK" prompt are:

| | |
|---|---|
| *YES* | Ignore this error condition. |
| *NO* | Ignore this file system and continue checking the next file system given. |

**Can't open ...**

The file system listed cannot be opened for reading. When running manually, it ignores this file system and continues checking the next file system given. Check the access modes of the file system.

**"file_system_name": (NO WRITE)**

Either the **-n** flag was specified or **fsck**'s attempt to open the file system, "file_system_name", for writing failed. When running manually, all the diagnostics are printed, but no modifications are attempted to fix them.

**Other messages:**

**MAGIC NUMBER WRONG**

**NCG OUT OF RANGE**

**CPG OUT OF RANGE**

**NCYL DOES NOT GIVE WITH NCG*CPG**

**SIZE PREPOSTEROUSLY LARGE**

**TRASHED VALUES IN SUPER BLOCK**

will be followed by the message:

*file_system:* BAD SUPER BLOCK: *superblock_address*
USE -b OPTION OF FSCK TO SPECIFY LOCATION OF AN ALTERNATE superblock TO SUPPLY NEEDED INFORMATION; SEE fsck(1M).

The superblock is corrupted. An alternative superblock must be used. See the discussion on alternative superblocks under the "Superblock Consistency" section in this appendix.

**INTERNAL INCONSISTENCY:** *message*

An internal problem occurred in **fsck**. **"message"** indicates the problem. This should never happen. Contact your local HP Sales and Service Office for assistance.

**CANNOT SEEK: BLK** *bn* **(CONTINUE)?**

The **fsck** command's request for moving to the specified block number in the file system failed. This should never happen. Contact your local HP Sales and Service Office for further assistance. Possible responses to the "CONTINUE" prompt are:

*YES* Attempt to continue to run the file system check. Often, however, the problem will persist. This error condition will not allow a complete check of the file system. Run **fsck** a second time to recheck this file system.

*NO* Terminate the program.

**CANNOT READ: BLK ... (CONTINUE)?**

The **fsck** command's attempt to read a specified block number in the file system failed. This can happen when you interrupt **fsck** before it finishes. Contact your local HP Sales and Service Office for further assistance.

Possible responses to the "CONTINUE" prompt are:

*YES* Attempt to continue to run the file system check. Often, however, the problem will persist. This error condition will not allow a complete check of the file system. Run **fsck** a second time to recheck this file system.

*NO* Terminate the program.

**CANNOT WRITE: BLK ... (CONTINUE)?**

The **fsck** command's attempt to write a specified block number in the file system failed. The disk is probably physically write-protected. Remove write protection from the disk, and rerun **fsck**.

Possible responses to the "CONTINUE" prompt are:

*YES* Attempt to continue to run the file system check. Often, however, the problem will persist. This error condition prevents a complete check of the file system. Run **fsck** a second time to recheck this file system.

*NO* Terminate the program.

## Phase 1 Errors: Check Blocks and Sizes

This phase concerns itself with the inode list. This section lists error conditions resulting from checking inode types, setting up the zero-link-count table, examining inode block numbers for bad or duplicate blocks, checking inode size, checking block count, and checking inode format. All errors in phase 1 are fatal if you are preening the file system, except for INCORRECT BLOCK COUNT, BAD INDIRECT ADDRESS, and NON-ZERO READER/WRITER COUNT.

**CG ... : BAD MAGIC NUMBER**

The magic number of cylinder group is wrong. This usually indicates that the cylinder group maps have been destroyed. When running manually, the cylinder group is marked as needing to be reconstructed.

**NON-ZERO READER/WRITER COUNT(S) ON PIPE I=... (CORRECT)?**

The inode indicates that a process is reading or writing from or to the pipe. Possible responses to "CORRECT" prompt are:

*YES* Restart the number of readers and writers of this pipe to 0.

*NO* Ignore this error condition.

**BAD DIRECT ADDRESS, SHOULD BE ZERO: inode.didb[n]=... (CORRECT)?**

The inode contains a direct disk block address for regions beyond the allocated size of the file. During the preening process, these entries are zeroed.

Possible responses to the "CORRECT" prompt:

*YES* Zero the entry.

*NO* Ignore this error condition. Later attempts by the operating system to extend the file into this region may cause a system crash.

**UNKNOWN FILE TYPE I=... (CLEAR)?**

The mode word of the inode indicates that the inode is not a character special, block special, regular, network special, fifo, symbolic link, or directory inode.

Possible responses to the "CLEAR" prompt are:

*YES* Deallocate the inode by zeroing its contents. This always invokes the UNALLOCATED error condition in Phase 2 for each directory entry pointing to this inode.

*NO* Ignore this error condition.

**LINK COUNT TABLE OVERFLOW (CONTINUE)?**

An internal table for **fsck** containing allocated inodes with a link count of zero has no more room.

Possible responses to the "CONTINUE" prompt are:

*YES* Continue with the program. This error condition prevents a complete check of the file system. Run **fsck** a second time to recheck this file system. If another allocated inode with a zero link count is found, this error condition is repeated.

*NO* Terminate the program.

**"*block_number*" BAD I=...**

The inode represented by "I=..." contains the block number *block_number*. This block number is out of the range of the file system. This error condition may invoke the EXCESSIVE BAD BLKS error condition in phase 1 if this inode has too many block numbers outside the file system range. This error condition always invokes the BAD/DUP error condition in Phase 2 and Phase 4.

**EXCESSIVE BAD BLKS I=... (CONTINUE)?**

There are more than 10 blocks with a block number out of the range of the file system associated with the inode.

Possible responses to the "CONTINUE" prompt are:

*YES* Ignore the rest of the blocks in this inode and continue checking with the next inode in the file system. This error condition will not allow a complete check of the file system. Run **fsck** a second time to recheck this file system.

*NO* Terminate the program.

***block_number* DUP I=...**

The inode contains block number, *block_number,* which is already claimed by another inode. This error condition may invoke the EXCESSIVE DUP BLKS error condition in phase 1 if this inode has too many block numbers claimed by other inodes. This error condition will always invoke Phase 1b and the BAD/DUP error condition in Phase 2 and Phase 4.

**EXCESSIVE DUP BLKS I=... (CONTINUE)?**

There are more than 10 blocks claimed by other inodes.

Possible responses to the "CONTINUE" prompt are:

*YES* Ignore the rest of the blocks in this inode and continue checking with the next inode in the file system. This error condition prevents a complete check of the file system. Run **fsck** a second time to recheck this file system.

*NO* Terminate the program.

**DUP TABLE OVERFLOW (CONTINUE)?**

An internal table in **fsck** containing duplicate block numbers is full.

Possible responses to the "CONTINUE" prompt are:

*YES* Continue with the program. This error condition prevents a complete check of the file system. Run **fsck** a second time to recheck this file system. If another duplicate block is found, this error condition repeats.

*NO* Terminate the program.

**PARTIALLY ALLOCATED INODE I=... (CLEAR)?**

The bit map of the file system is inconsistent with inode status.

Possible responses to the "CLEAR" prompt are:

*YES* Deallocate the inode by zeroing its contents.

*NO* Terminate the program.

**INCORRECT BLOCK COUNT I=... (CORRECT)?**

The block count for the inode, *inode_number*, is *X* blocks, but should be *Y* blocks. When you are preening the count is corrected.

Possible responses to the "CORRECT" prompt are:

*YES* replace the block count of the inode with *Y*.

*NO* ignore this error condition.

**BAD INDIRECT ADDRESS: IND BLOCK n[m] = val I=... (CORRECT)?**

An indirect address block, allocated in the inode indicated by "I=...", contains block address for regions beyond the allocated size of the file. When you are preening, these entries are zeroed.

Possible responses to the "CORRECT" prompt are:

*YES* Zero the entry.

*NO* Ignore this error condition. Later attempts by the operating system to extend the file into this region may cause a system crash.

## Phase 1b: Rescan for More Dups

When a duplicate block is found in the file system, the file system is rescanned to find the inode which previously claimed that block. This section lists the error condition when the duplicate block is found.

#### *block_number* DUP I=...

The inode contains the block number, *block_number,* which is already claimed by another inode. This error condition will always invoke the BAD/DUP error condition in Phase 2. You can determine which inodes have overlapping blocks by examining this error condition and the DUP error condition in Phase 1.

## Phase 2: Check Path Names

This phase concerns itself with removing directory entries pointing to error conditioned inodes from Phase 1 and Phase 1b. This section lists error conditions resulting from root inode mode and status, directory inode pointers in range, and directory entries pointing to bad inodes. All errors in this phase are fatal if you are preening your file system.

#### ROOT INODE UNALLOCATED. TERMINATING

The root inode (inode number 2) has no allocated mode bits. This should never happen. The program will terminate. Contact your local HP Sales and Service Office for further assistance.

#### NAME TOO LONG

The path name shown is too long. This is usually indicative of loops in the file system name space. This can occur if the super user has made circular links to directories. The offending links must be removed.

#### ROOT INODE NOT DIRECTORY (FIX)?

The root inode (inode number 2) is not directory inode type.

Possible responses to the "FIX" prompt are:

*YES* Change the root inode's type to be a directory. If the root inode's data blocks are not directory blocks, many error conditions are produced.

*NO* Terminate the program.

#### DUPS/BAD IN ROOT INODE (CONTINUE)?

Phase 1 or Phase 1b found duplicate blocks or bad blocks in the root inode (inode number 2) for the file system.

Possible responses to the "CONTINUE" prompt are:

*YES* Ignore the DUPS/BAD error condition in the root inode and attempt to continue to run the file system check. If the root inode is not correct, this may result in a large number of other error conditions.

*NO* Terminate the program.

**I OUT OF RANGE I=... DIR=*NAME* (REMOVE)?**

*NAME* has an inode number *I*, which is greater than the end of the inode list.

Possible responses to the "REMOVE" prompt are:

| | |
|---|---|
| *YES* | The directory entry *NAME* is removed. |
| *NO* | Ignore this error condition. |

**UNALLOCATED I=... (REMOVE)?**

Two possible error messages begin like this. One is for directory entries and the other is for files. A directory has an entry for a directory or file, but the inode I for the directory or file is not allocated. The owner, mode, size, modify time, and directory or file name are printed.

Possible responses to the "REMOVE" prompt are:

| | |
|---|---|
| *YES* | The directory entry is removed. |
| *NO* | Ignore this error condition. |

**DUP/BAD I=... (REMOVE)?**

There are two possible error messages that start like this. One is for directory entries and one is for files. Phase 1 or Phase 1b found duplicate blocks or bad blocks associated with the directory or file having inode I. The owner, mode, size, modify time, and directory are printed.

Generally, the inode with the earliest modify time is incorrect, and should be cleared.

Possible responses to the "REMOVE" prompt are:

| | |
|---|---|
| *YES* | The directory entry is removed. |
| *NO* | Ignore this error condition. |

**ZERO LENGTH DIRECTORY I=... (REMOVE)?**

The directory entry's size is zero. The owner, mode, size, modify time, and directory name are printed.

Possible responses to the "REMOVE" prompt are:

| | |
|---|---|
| *YES* | the directory entry is removed. This will always invoke the BAD/DUP error condition in phase 4. |
| *NO* | ignore this error condition. |

**DIRECTORY TOO SHORT I=... (FIX)?**

The directory entry's size is less than the minimum size for a directory. The owner, mode, size, modify time, and directory name are printed.

Possible responses to the "FIX" prompt are:

| | |
|---|---|
| *YES* | increase the size of the directory to the minimum directory size. |
| *NO* | ignore this error condition. |

**DIRECTORY CORRUPTED I=... (FIX)?**

A directory entry has an inconsistent internal state. The owner, mode, size, modify time, and directory name are printed.

Possible responses to the "FIX" prompt are:

| | |
|---|---|
| *YES* | Throw away all entries up to the next directory boundary. This drastic action can throw away directory entries, and should be taken only after other recovery efforts have failed. |
| *NO* | Skip to the next directory boundary and resume reading, but do not modify the directory. |

**BAD INODE NUMBER FOR '.' I=... (FIX)?**

The directory entry doesn't have an inode number for '.' which is equal to the inode number. The owner, mode, size, modify time, and directory name are printed.

Possible responses to the "FIX" prompt are:

| | |
|---|---|
| *YES* | change the inode number for '.' to be equal to the inode number given after 'I='. |
| *NO* | leave the inode number for '.' unchanged |

**MISSING '.' I=... (FIX)?**

The directory doesn't have its first directory entry allocated. The owner, mode, size, modify time, and directory name are printed.

Possible responses to the FIX prompt are:

| | |
|---|---|
| *YES* | make an entry for '.' with inode number equal to the inode number given after 'I='. |
| *NO* | leave the directory unchanged. |

**MISSING '.' I=...**
**CANNOT FIX, FIRST ENTRY IN DIRECTORY CONTAINS ...**

The directory has, as its first entry, the file name given. The **fsck** command cannot resolve this problem. The file system should be mounted and the offending entry moved elsewhere. To do this, exit the **fsck** program, mount the file system (you can force a mount by using **mount** *file_system* **–f**, find the file name, and move it to a different directory. The file system should then be unmounted and **fsck** should be run. The owner, mode, size, modify time, and directory name are printed.

**MISSING '.' I=...**
**CANNOT FIX, INSUFFICIENT SPACE TO ADD '.'**

The directory does not have '.' as its first entry. The **fsck** command cannot resolve this problem. If this happens, contact your local HP Sales and Service office. The owner, mode, size, modify time, and directory name are printed.

**EXTRA '.' ENTRY I=... (FIX)?**

The directory has more than one entry for '.'. The owner, mode, size, modify time, and directory name are printed.

Possible responses to the "FIX" prompt are:

*YES* remove the extra entry for '.'.

*NO* leave the directory unchanged.

**BAD INODE NUMBER FOR '..' I=... (FIX)?**

The directory's inode number for '..' does not equal the parent of the inode number *I*. The owner, mode, size, modify time, and directory name are printed.

Possible responses to the "FIX" prompt are:

*YES* change the inode number for '..' to be equal to the parent of the inode given *I*.

*NO* leave the inode number for '..' unchanged.

**MISSING '..' I=... (FIX)?**

The directory doesn't have its second directory entry allocated.

Possible responses to the "FIX" prompt are:

*YES* make an entry for '..' with inode number equal to the parent of the inode number given *I*.

*NO* leave the directory unchanged.

**MISSING '..' I=...**
**CANNOT FIX, SECOND ENTRY IN DIRECTORY CONTAINS ...**

The directory has, as its second entry, the file name given. The **fsck** command cannot resolve this problem. The file system should be mounted and the offending entry moved elsewhere. To do this, exit the **fsck** program, mount the file system (you can force a mount by using **mount** *file_system* **-f**, find the file name, and move the file to a different directory. The file system should then be unmounted and **fsck** should be run again.

**MISSING '..' I=...**
**CANNOT FIX, INSUFFICIENT SPACE TO ADD '..'**

The directory does not have '..' as its second entry. The **fsck** command cannot resolve this problem. If this happens, contact your local HP Sales and Service office.

**EXTRA '..' ENTRY I=... (FIX)?**

The directory has more than one entry for '..'.

Possible responses to the "FIX" prompt are:

*YES* remove the extra entry for '..'.

*NO* leave the directory unchanged.

**UNUSED SPACE BETWEEN '.' AND '..' I=. . . (FIX)?**

There is enough empty space between '.' and '..' in this directory for a new entry to be allocated between the two entries. A new entry between '.' and '..' would violate the requirement that these two entries be the first two in a directory.This condition will typically occur when a file system has been incorrectly or incompletely converted to allow long filenames.

If you are preening, the directory is fixed.

Possible responses to the "FIX" prompt are:

*YES* copy the '..' next to the '.' to remove the gap between the two entries

*NO* leave the directory unchanged

## Phase 3: Check Connectivity

This phase concerns itself with the directory connectivity seen in Phase 2. This section lists error conditions resulting from unreferenced directories, and missing or full **/lost+found** directories.

**UNREF DIR I=... (RECONNECT)?**

The directory inode *I* was not connected to a directory entry when the file system was traversed. The owner, mode, size, and modify time of the directory inode are printed. If you are preening, the directory is reconnected if its size is non-zero, otherwise it is cleared.

Possible responses to the "RECONNECT" prompt are:

*YES* Reconnect the directory inode to the file system in the directory for lost files (**/lost+found**). This may invoke the "lost+found" error condition if there are problems connecting the directory inode to **/lost+found**. This may also invoke the "CONNECTED" error condition in phase 3 if the link was successful.

*NO* Ignore this error condition. This error will always invoke the "UNREF" error condition in Phase 4.

**SORRY, NO lost+found DIRECTORY**

There is no **/lost+found** directory in the root directory of the file system. The **fsck** command ignores the request to link a directory in **/lost+found**. This will always invoke the UNREF error condition in Phase 4. Check access modes of **/lost+found**. This error is fatal if you are preening the system.

**SORRY. NO SPACE IN lost+found DIRECTORY**

There is no space to add another entry to the **/lost+found** directory in the root directory of the file system. The **fsck** command ignores the request to link a directory in **/lost+found**. This will always invoke the "UNREF" error condition in Phase 4. Clean out unnecessary entries in **/lost+found** or make **/lost+found** larger and try again. This error is fatal if you are preening the system.

**DIR I=... PARENT WAS I=...**

This is an advisory message indicating a directory inode was successfully connected to the **/lost+found** directory. The parent inode of the directory inode is replaced by the inode number of the **/lost+found** directory.

## Phase 4: Check Reference Counts

This phase concerns itself with the link count information seen in Phase 2 and Phase 3. This section lists error conditions resulting from unreferenced files, missing or full **/lost+found** directories, incorrect link counts for files, directories, or special files, unreferenced files and directories, bad and duplicate blocks in files and directories, and incorrect total free-inode counts. All errors in this phase are correctable if you are preening your file system unless you run out of space in **/lost+found**.

**UNREF FILE I=... (RECONNECT)?**

The inode *I* was not connected to a directory entry when the file system was traversed. The owner, mode, size, and modify time of the inode are printed. If you are preening, the file is cleared if either its size, or its link count is zero. Otherwise it is reconnected.

Possible responses to the "RECONNECT" prompt are:

*YES* Reconnect the inode to the file system in the directory for lost files (/lost+found). This may invoke the "lost+found" error condition if there are problems connecting the inode to **/lost+found**.

*NO* Ignore this error condition. This will always invoke the "CLEAR" error condition in phase 4.

**(CLEAR)?**

The inode mentioned in the immediately previous error condition cannot be reconnected. If you are preening, this error cannot occur, since lack of space to reconnect files is a fatal error.

Possible responses to the "CLEAR" prompt are:

*YES* deallocate the inode mentioned in the previous error condition by zeroing its contents.

*NO* Ignore this error condition.

**SORRY. NO lost+found DIRECTORY**

There is no **/lost+found** directory in the root directory of the file system. The **fsck** command ignores the request to link a file in **/lost+found**. This will always invoke the "CLEAR" error condition in phase 4. Check access modes of **/lost+found**. If you are preening the file system, this error is fatal.

#### SORRY. NO SPACE IN lost+found DIRECTORY

There is no space to add another entry to the **/lost+found** directory in the root directory of the file system. The **fsck** command ignores the request to link a file in **/lost+found**. This always invokes the "CLEAR" error condition in phase 4. Check the size and contents of **/lost+found**. This error is fatal if you are preening the file system.

#### LINK COUNT... (ADJUST)?

The link count for the file, directory, or inode is one link count *COUNT*= but should be a different link count. The owner, mode, size, and modify time are printed. If you are preening, the link count is adjusted.

Possible responses to the "ADJUST" prompt are:

*YES* Replace the link count of the file in the inode with what the number should be.

*NO* Ignore this error condition.

#### UNREF ... (CLEAR)?

The file or directory with the inode number, *I*, was not connected to a directory entry when the file system was traversed. The owner, mode, size, and modify time of the inode are printed. If you are preening, the inode is cleared, since this is a file that was not connected because its size or link count was zero.

Possible responses to the "CLEAR" prompt are:

*YES* deallocate inode by zeroing its contents.

*NO* Ignore this error condition.

#### BAD/DUP ... (CLEAR)?

Phase 1 or Phase 1b found duplicate blocks or bad blocks associated with the file or directory inode given in *I*. The owner, mode, size, and modify time of the inode are printed. This error does not occur if you are preening, since it would have caused a fatal error earlier.

Possible responses to the "CLEAR" prompt are:

*YES* deallocate the inode by zeroing its contents.

*NO* Ignore this error condition.

Often deleting only one of the files containing DUPS will cure the problem. The **fsck** command should be rerun to confirm that the problem was fixed. A "NO" means that **fsck** must be rerun to finish cleaning up the file system.

#### FREE INODE COUNT WRONG IN SUPERBLK (FIX)?

The actual count of the free inodes does not match the count in the superblock of the file system. If you are preening, the count is fixed.

Possible responses to the "FIX" prompt are:

*YES* Replace the count in the superblock by the actual count.

*NO* Ignore this error condition.

## Phase 5: Check Cylinder Groups

This phase concerns itself with the free-block maps. This section lists error conditions resulting from allocated blocks in the free-block maps, free-blocks missing from free-block maps, and the total free-block count not matching the count contained in the summary information area.

#### CG ...: BAD MAGIC NUMBER

The magic number of the cylinder group is wrong. This usually indicates that the cylinder group maps have been destroyed. When running manually, the cylinder group is marked as needing to be reconstructed. If you are preening your system, this error is fatal.

#### EXCESSIVE BAD BLKS IN BIT MAPS (CONTINUE)?

You should never get this message. If you do, contact your local HP Sales and Service office.

#### SUMMARY INFORMATION *t* BAD

where *t* is one or more of:

(INODE FREE)

(BLOCK OFFSETS)

(FRAG SUMMARIES)

(SUPER BLOCK SUMMARIES)

The indicated summary information was found to be incorrect. This error condition will always invoke the "BAD CYLINDER GROUPS" condition in phase 6. If you are preening, the summary information is recomputed.

#### *x* BLK(S) MISSING

A number of blocks *x* that are unused by the file system were not found in the free-block maps. This error condition always invokes the "BAD CYLINDER GROUPS" condition in phase 6. If you are preening, the block maps are rebuilt.

#### FREE BLK COUNT WRONG IN SUPERBLOCK (FIX)?

The actual count of free blocks does not match the count in the superblock of the file system. If you are preening, the counts are fixed.

Possible responses to the "FIX" prompt are:

| | |
|---|---|
| *YES* | Replace the count in the superblock by the actual count. |
| *NO* | Ignore this error condition. |

**BAD CYLINDER GROUPS (FIX)?**

Phase 5 has found bad blocks in the free-block maps, duplicate blocks in the free-block maps, or blocks missing from the file system. If you are preening, the cylinder groups are reconstructed.

Possible responses to the "FIX" prompt are:

| | |
|---|---|
| *YES* | Replace the actual free-block maps with new free-block maps. |
| *NO* | Ignore this error condition. |

## Phase 6: Salvage Cylinder Groups

This phase concerns itself with reconstructing the free-block maps. No error messages are produced.

## Cleanup

Once a file system has been checked, a few cleanup functions are performed. This section lists advisory messages about the file system and modification status of the file system.

***f*** **files,** ***b*** **used,** ***r*** **free (*y* frags,** ***z*** **blocks)**

This message indicates that the file system just checked has a total of **f** files, using **b** fragment-sized blocks, with **r** fragment-sized blocks available (free) for use. The numbers in parenthesis divides the free count into **y** free fragments and **z** free full sized blocks.

No action is required on your part.

******* REBOOT HP-UX; DO NOT SYNC (USE reboot -n) *******

This message indicates that the root file system has been modified by **fsck**. If HP-UX is not rebooted immediately, the work done by **fsck** may be undone by the in-core (memory) copies of tables HP-UX keeps. If you are preening, **fsck** exits with a code of 4. The **bcheckrc** script interprets an exit code of 4 by executing the **reboot** command.

******* FILE SYSTEM WAS MODIFIED *******

This message indicates that the current file system was modified using **fsck.** If this file system is mounted, **fsck** exits and you should reboot your system. If the system is not rebooted immediately, the changes made by **fsck** may be undone by the memory tables of HP-UX.

---

**NOTE**

**If you are preening, you will not get this message. If you execute fsck -p outside the bcheckrc program, you must check the return code to see if the system needs to be rebooted.**

---

# System Parameters

You can use these tunable parameters with the **uxgen** command to customize the HP-UX kernel. The **uxgen** command is described in Chapter 7.

**NOTE**

**You can damage the operation of HP-UX by changing these parameters. BE CAREFUL, and make sure you know all the implications of using the parameters before you configure your system. Never set system parameters outside the indicated range; this can cause data loss, impaired performance, and system crashes. Keep in mind that these parameters interact and that they should be changed in a balanced way.**

Parameter headings are described below.

| | |
|---|---|
| **Name**: | The name of this parameter when it is used to configure a new system using **uxgen**. |
| **Range**: | The maximum possible range of this parameter. Due to interactions with other parameters, sometimes this range cannot be attained. |
| **Default**: | If no value is assigned at configuration time, this is the value that will be given. This may be specified as a formula which depends on other system parameters. If these other system parameters change, the default value changes correspondingly. |
| **Use**: | A brief description of how the parameter is used by the system. |
| **Space Utilization**: | A formula indicating how dependent space is allocated. Not every parameter has this field even though space may be allocated with this parameter. Usually space is not an important factor for most systems. However, if this space is a critical issue for your system space, the space difference can be calculated by using the **size(1)** command on two system files (created with and without your changes). The space change will be reflected in the BSS space. |
| **Dependencies**: | The interaction of the parameter with other system parameters, or how changing it affects system performance, or a formula showing relationships with other parameters. |

## acctresume

**Name** **acctresume** – resume accounting due to disk usage

**Range** –100 -> 101

**Default** 4

**Use**

The system automatically disables process accounting when the available space on the file system where the accounting file resides falls below a certain threshold. The threshold is described under **acctsuspend**. The system also automatically re-enables process accounting when sufficient space becomes available. The parameter, **acctresume**, is the threshold (percentage of free space) which the system must have to re-enable process accounting. This percentage is added to minimum free percentage (minfree) for the file system.

A value of zero re-enables accounting when the free space reaches minfree. A value less than zero allows process accounting to use the space which is reserved for superuser use. A value greater than 100 prevents process accounting from ever being re-enabled due to space becoming available.

When accounting is re-enabled in this way, the message

```
    Accounting resumed
```

is printed on the system console.

Parameter **acctresume** is only relevant to systems which turn on process accounting.

**Dependencies (interactions with other system values)**

acctsuspend < acctresume

## acctsuspend

**Name** **acctsuspend** – suspend accounting due to disk usage

**Range** –100 -> 100

**Default** 2

**Use**

The system automatically disables process accounting when the available space on the file system where the accounting file resides falls below a certain threshold. The parameter, **acctsuspend** (specified as a percentage of free space), is the threshold. This percentage is added to minimum free percentage (minfree) for the file system.

A value of zero disables accounting when the free space falls below **minfree**. A value less than zero allows process accounting to use the space which is reserved for superuser use. If the sum of acctsuspend and minfree is less than zero, process accounting can never be disabled for this reason.

When accounting is disabled in this way, the message

```
Accounting suspended
```

is printed on the system console.

Parameter **acctsuspend** is only relevant to systems that use process accounting.

**Dependencies ( interactions with other system values )**

acctsuspend < acctresume

## bufpages

| | |
|---|---|
| **Name** | **bufpages**--number of buffer pages |
| **Range** | 0, 64 |
| **Default** | 0 (configured dynamically) |

If at boot time, **bufpages = 0**, two pages are allocated for every buffer header defined by **nbuf**. If **nbuf** is also zero, then 10 percent of available memory is used.

**Use**

Parameter **bufpages** defines the number of pages in the file system buffer cache. Each page is allocated 2048 bytes of memory.

These buffers are used for all file system I/O operations, plus all other block I/O operations in the system (exec, mount, inode reading and some device drivers.)

---

**NOTE**

**If you set nbuf to a number less than 64 or greater than the maximum supported by the system, the number will be increased or decreased as appropriate, and a message is printed at boot time.**

---

**Dependencies ( interactions with other system values )**

This variable may override the value specified for nbuf.

The maximum memory allocated to the buffer pool will be limited based on the memory allocated to the system for other purposes. Thus modifying parameters that affect system memory may affect the maximum memory allocatable to the buffer pool.

## dst

| | |
|---|---|
| **Name** | **dst**--daylight savings time |
| **Range** | 0 -> 5 |
| **Default** | 1 |

**Use**

This parameter defines the daylight savings time correction.

(from file **/usr/include/sys/time.h**)

```
#define DST_NONE        0       /* not on dst */
#define DST_USA         1       /* USA style dst */
#define DST_AUST        2       /* Australian style dst */
#define DST_WET         3       /* Western European dst */
#define DST_MET         4       /* Middle European dst */
#define DST_EET         5       /* Eastern European dst */
```

**Dependencies (interactions with other system values)**

It is used with time zone.

## iomemsize

| | |
|---|---|
| **Name** | **iomemsize** - amount of memory |
| **Range** | None. |
| **Default** | "(32+4* NUM_IOTREE_RECS)*NBPG" |

**Use**

This value is entered in bytes. Parameter **iomemsize** defines the maximum number of bytes that are available to the I/O system.

The default value is 32 pages plus four pages per I/O device (NUM_IOTREE_RECS). The default value should be large enough to accommodate most systems. If **iomemsize** is too small, the system will panic with the message:

```
panic: io_get_pages: out of memory
```

**Dependencies (interactions with other system values)**

None.

## itebuflines

**Name** **itebuflines** - ITE buffer lines

**Range** 60 -> 999

**Default** 100

**Use**

Parameter **itebuflines** defines the number of lines of emulated terminal screen memory on each Internal Terminal Emulator (ITE) port configured into the system. This is also known as the scrolling area; it is the area currently on- and off-screen.

For each configured graphics interface in the system, the graphics driver consumes `2*line_length*itebuflines` bytes of data. For example, a 98720 graphics display has an ITE line length of 128 characters. Setting **itebuflines** to 100 causes `2*128*100 =25600` bytes of kernel space to be used for each graphics terminal's screen memory.

**Dependencies (interactions with other system values)**

None.

## maxdsiz

**Name** **maxdsiz** - maximum data size

**Range** 256 Kbytes -> 640 Mbytes

If **maxdsiz** is increased beyond the default, refer to Table D-1 at the end of this appendix.

**Default** 32768 (64 Mbytes)

**Use**

This value is entered in pages (2048 bytes/page).

Parameter **maxdsiz** defines the maximum size of the data segment of an executing process.

The default value is large enough for the data used by most processes. The **maxdsiz** parameter should be increased only if you have one or more processes that use large amounts of data.

Each time the system loads a process, or an executing process attempts to expand its data segment, the system checks the size of the process's data segment.

If the **maxdsiz** is exceeded, the process will be terminated or returned with an error.

**Dependencies (interactions with other system values)**

Changes in **maxdsiz** should be coordinated with changes to **maxssiz, maxtsiz,** and the amount of system swap space.

## maxuprc

**Name** **maxuprc** - maximum number of user processes

**Range** 3 -> (nproc - 4)

**Default** 25 processes

**Use**

Parameter **maxuprc** defines (for each user) the maximum number of simultaneous processes. A user is identified by the UID (user ID) number, not by the number of login instances. Each user will need at least one process for the shell, plus an adequate number to be able to do useful work (25 is usually more than enough).

The superuser is exempt from this limit.

Pipelines need at least one simultaneous process for each side of a '|'. Some commands, such as **cc**, **fc**, and **pc**, use more than one process per invocation.

If a user tries to start a new process from a shell and the total number of processes for the user is larger than **maxuprc**, this message will be printed to their tty:

```
    no more processes
```

If a user is doing a fork ( ) system call to create a new process and the total number of processes for the user exceeds **maxuprc**, fork ( ) will return -1 and set the **errno** to **EAGAIN**.

**Dependencies (interactions with other system values)**

If **maxuprc** is set to a value greater than or equal to **nproc** (maximum number of processes in the system) then **maxuprc** is no longer a limit, and the system can be hoarded by a single user.

## maxusers

**Name** **maxusers** - limiter for system resource allocation

**Range** 0 -> 128

**Default** 32

**Use**

Parameter **maxusers** limits system resource allocation. By itself, **maxusers** does not determine the size of any structures in the system. The default value of other global system parameters depend on **maxusers**. If you tune the parameters that use **maxusers**, then the effect of **maxusers** on kernel size is proportionately smaller.

It is not used as a limit for the number of users in the system, therefore its name (inherited from System V) is confusing.

**Dependencies (interactions with other system values)**

The default values of **nproc, ncallout, ninode, nfile,** and **ntext** depend on **maxusers**.

## msgmap

| | |
|---|---|
| **Name** | **msgmap** - message map |
| **Range** | 3 -> memory limited |
| **Default** | 100 |

**Use**

Each set of messages allocated per identifier occupies 1 or more contiguous slots in the msg array. As messages are allocated and deallocated the msg array may become fragmented.

Parameter **msgmap** dimensions the resource map used to allocate the buffer space for messages. This map shows the free holes in the msg array. An entry in the map is used to point to each set of contiguous unallocated slots, and it consists of a pointer to the set, plus the size of the set.

If message usage is heavy, and a request for a message set cannot be accommodated, the message:

```
DANGER: mfree map overflow
```

appears. If you get the error, make a new kernel with a larger msgmap.

There is less fragmentation of the msg array if all message identifiers have the same number of messages. Parameter **msgmap** can then be smaller.

**Dependencies (interactions with other system values)**

msgmap <= (msgtql+2)

If **msgmap** is greater than **msgtql+2,** then part of the space allocated for **msgmap** will not be used.

## msgmax

| | |
|---|---|
| **Name** | **msgmax** - message maximum size |
| **Range** | 0 -> memory limited |
| **Default** | 8192 bytes |

**Use**

Parameter **msgmax** limits the size, in bytes, of a single message.

It should be increased only if you have applications which require larger messages. Its main value is to keep malicious or poorly written programs from using all the message buffer space.

A **msgsnd** system call which attempts to send a message larger than **msgmax** bytes returns an error.

**Dependencies (interactions with other system values)**

msgmax <= msgmnb

## msgmnb

| | |
|---|---|
| **Name** | **msgmnb** – maximum number of bytes on message queue |
| **Range** | 0 -> memory limited |
| **Default** | 16,384 bytes |

**Use**

Parameter **msgmnb** is the maximum total size, in bytes, of all messages that can be queued on a message queue at the same time.

A **msgsnd** system call which attempts to exceed this limit returns either:

- an EAGAIN error if IPC_NOWAIT is set.
- an EINVAL error if IPC_NOWAIT is not set.

**Dependencies (interactions with other system values)**

msgmnb >= msgmax
msgmnb <= (msgssz * msgseg)

## msgmni

| | |
|---|---|
| **Name** | **msgmni** – number of message queue identifiers |
| **Range** | 1 -> memory limited |
| **Default** | 50 |

**Use**

Parameter **msgmni** dimensions the array of message queue identifiers.

A message queue identifier is needed for each message queue in the system.

An attempt to allocate a new message queue with the **msgget** system call when **msgmni** message queues already exist returns a ENOSPC error.

If a message queue is not deallocated, it remains on the system even after the process(es) using it have died. Deallocate message queues using the **ipcrm(1)** program.

All users of messages should deallocate them when through with them, as running against the **msgmni** limit usually means that users have not freed them up.

**Dependencies (interactions with other system values)** None.

## msgseg

| | |
|---|---|
| **Name** | **msgseg** – message segments |
| **Range** | 1 -> memory limited |
| **Default** | 1024 |

**Use**

Parameter **msgseg,** together with **msgssz,** determines the size of the buffer available for queueing messages.

Parameter **msgssz** determines the size, in bytes, of the units in which messages are allocated space. When a message is allocated, its size is rounded up to the nearest multiple of **msgssz.**

Parameter **msgseg** is the number of these units available.

In most cases the product of **msgseg * msgssz** is of most interest since it determines the total amount of space available for messages. Different **msgseg:msgssz** ratios which yield the same product will just cause this space to be fragmented differently for the same usage.

**Space Utilization** msgseg * msgssz bytes

**Dependencies (interactions with other system values)**

(msgseg * msgssz) >= msgmnb
(msgseg * msgssz) >= msgmax

## msgssz

| | |
|---|---|
| **Name** | **msgssz** – message segment size |
| **Range** | 1 -> memory limited |
| **Default** | 8 bytes |

**Use**

Parameter **msgssz,** together with **msgseg,** determines the size of the buffer available for queuing messages.

Parameter **msgssz** determines the size, in bytes, of the units in which messages are allocated space. When a message is allocated, its size is rounded up to the nearest multiple of **msgssz.**

Parameter **msgseg** is the number of these units available.

In most cases, the product of **msgseg * msgssz** is of most interest since it determines the total amount of space available for messages. Different **msgseg:msgssz** ratios which yield the same product will just cause this space to be fragmented differently for the same usage.

**Space Utilization** msgseg * msgssz bytes

**Dependencies (interactions with other system values)**

(msgseg * msgssz) >= msgmnb
(msgseg * msgssz) >= msgmax

## msgtql

| | |
|---|---|
| **Name** | **msgtql** – number of message headers |
| **Range** | 1 -> memory limited |
| **Default** | 40 |

**Use**

Parameter **msgtql** dimensions an array of message headers. A message header is used for each message queued in the system.

A **msgsnd** system call which attempts to exceed this limit either:

- blocks waiting for a free header or,
- returns EAGAIN error

depending on whether the IPC_NOWAIT flag is set with the call.

**Dependencies (interactions with other system values)**

msgmap <= msgtql + 2

If **msgmap** is greater than **msgtql + 2**, then some allocated space will be wasted.

## nbuf

| | |
|---|---|
| **Name** | **nbuf** – number of buffer headers |
| **Range** | 0, 16 -> memory limited |
| **Default** | 0 (configured dynamically) |

If at boot time **nbuf** is = 0, then one buffer header is allocated for every two pages of buffer memory defined by the **bufpages** parameter. If **bufpages** is also zero, 10% of available memory is used.

**Use**

Parameter **nbuf** defines the number of file system buffer-cache buffer headers. Each buffer is allocated 4096 bytes of memory unless overridden by a conflicting value for **bufpages**.

These buffers are used for all file system I/O operations, plus all other block I/O operations in the system (**exec, mount**, inode reading, some device drivers, etc).

While **nbuf** is available for compatibility with previous releases, it is recommended that the size of the buffer pool be configured with the **bufpages** parameter.

**NOTE**

**If you set nbuf to a number less than 16, greater than the maximum supported by the system, or to a value that is inconsistent with the value of bufpages, the number will be increased or decreased as appropriate, and a message printed at boot time.**

**Dependencies (interactions with other system values)**

**Bufpages** controls the actual memory allocated to the buffer pool. If both **bufpages** and **nbuf** are set and the values conflict so that it is impossible to configure a system using both of them, **bufpages** overrides.

## ncallout

| | |
|---|---|
| **Name** | **ncallout** - number of timeouts |
| **Range** | 6 -> memory limited |
| **Default** | (64+NPROC) |

**Use**

Parameter **ncallout** is the maximum number of timeouts that can be scheduled by the kernel at any one time. Timeouts are used by:

- **alarm** (system call)
- **setitimer** (system call)
- **select** (system call)
- Drivers
- **uucp** processes
- Process scheduling

When the system runs out of timeouts, it prints the following fatal error to the console:

```
panic: timeout table overflow
```

**Dependencies (interactions with other system values)**

The larger **nproc** is, the larger **ncallout** should be. A rough guideline of 1 callout per process should be used unless you have processes that use many of the callouts.

## netisr_priority

**Name** **netisr_priority** - a real-time process priority for networking

**Range** 0 -> 127

**Default** 100

**Use**

The **netisr_priority** parameter specifies the real-time process priority of the **netisr** process. **Netisr** is a daemon process that runs on systems with networking capabilities. It is responsible for processing all networking input packets. Refer to the *HP-UX Real-Time Programming Manual* for more information.

**Dependencies (interactions with other system values).**

None.

## netmeminit

**Name** **netmeminit** - number of bytes of memory to be preallocated at system initialization time

**Range** 0 -> 1536 * NETCLBYTES (3000 kbytes)

**Default** 0

**Use**

The **netmeminit** parameter specifies the number of bytes of memory to be preallocated for networking at system initialization time. There is no delay associated with networking getting physical memory from the system. For more information on setting this parameter, refer to the *LAN, NS, ARPA Services/9000 Series 800 Node Manager's Guide*.

**Dependencies (interactions with other system values)**

netmeminit < netmemmax

## netmemmax

**Name** **netmemmax** - maximum number of bytes of physical memory networking will ever use

**Range** 32 * NETCLBYTES -> 1536 * NETCLYBYTES (128 kbytes--> 3000 kbytes)

**Default** (512 * NETCBYTES) (1000 kbytes)

**Use**

The **netmemmax** parameter specifies the maximum number of bytes of memory that will ever be used for networking. You can, under the open reservation system, reserve more memory than this value. However, this is the upper limit of the amount of allocated memory. For more information on setting this parameter, refer to the *LAN,NS,ARPA Services/9000 Series 800 Node Manager's Guide.*

**Dependencies (interactions with other system values)**

None

## netmemthresh

| | |
|---|---|
| **Name** | **netmemtresh** - number of bytes at which the networking subsystem switches from a full memory reservation scheme to an open memory reservation scheme. |
| **Range** | 0 - netmemmax +1, -1 |
| **Default** | 0 |

**Use**

The **netmemthresh** parameter specifies the threshold at which the system will change from full reservations of memory to open reservations. For more information on setting this parameter and networking memory reservation schemes, refer to the *LAN,NS,ARPA Services/9000 Series 800 Node Manager's Guide.*

**Dependencies (interactions with other system values)**

netmemthresh < netmemmax

## nfile

| | |
|---|---|
| **Name** | **nfile** - number of files |
| **Range** | 14 -> memory-limited |
| **Default** | (16 * (NPROC+16 + MAXUSERS)/10+32+2 * NETSLOP) |

**Use**

Parameter **nfile** defines the maximum number of open files at any one time in the system.

It is the number of slots in the file descriptor table. Be generous with this number since the cost is low, and not having enough slots would cut down on the amount of work that can be done simultaneously in the system.

**Dependencies (interactions with other system values)**

| | |
|---|---|
| **Processes** | At least 3 file descriptors per process (stdin, stdout, stderr). |
| **Pipes** | 2 per pipe (1 per side). |

## nflocks

| | |
|---|---|
| **Name** | **nflocks** – number of file locks |
| **Range** | 2 -> 2000 |
| **Default** | 200 |

**Use**

Parameter **nflocks** gives the possible number of file/record locks in the system. When choosing this number, note that one file may have several locks and databases may need an exceptionally large number of locks (if they use **lockf** at all).

**Dependencies** None.

## ninode

| | |
|---|---|
| **Name** | **ninode** – number of inodes |
| **Range** | 14 -> memory-limited |
| **Default** | ((NPROC+16+MAXUSERS)+32) |

**Use**

Parameter **ninode** defines the maximum number of open inodes which can be in-core.

It is the number of slots in the inode table. The inode table is used as a cache memory. For efficiency reasons, the last **ninode** (number of) open inodes is kept in main memory. The table is hashed.

**Dependencies (interactions with other system values)**

Each unique open file has an open inode associated with it. Therefore, the larger the number of unique open files, the larger **ninode** should be.

The default value of **ninode** depends on **nproc** and **maxusers**.

## nproc

| | |
|---|---|
| **Name** | **nproc** – number of processes |
| **Range** | 10 -> 1024 |
| **Default** | (20+8 * MAXUSERS ) |

**Use**

Parameter **nproc** specifies the maximum total number of processes that can exist simultaneously in the system.

There are at least four system overhead processes at all times, and one entry is always reserved for the superuser.

When the total number of processes in the system is larger than **nproc**, the following messages are printed:

At the console:

```
proc: table is full
```

If a user tries to start a new process from a shell, the following message prints on their terminal:

```
no more processes
```

If a user is doing a fork( ) system call to create a new process, fork( ) will return -1 and set the **errno** to **EAGAIN**.

**Dependencies (interactions with other system values)**

The default values of **ninode, nfile** and **ncallout** depend on **nproc.**

maxuprc <= (nproc - 4)

## npty

| | |
|---|---|
| **Name** | **npty** - number of pseudo-teletypes |
| **Range** | 1 -> memory limited |
| **Default** | 60 |

**Use**

Parameter **npty** limits the number of the following structures that can be used by the pseudo-teletype driver:

```
struct tty          pt_tty[npty];
struct tty          *pt_line[npty];
struct pty_info     pty_info[npty];
```

**Dependencies (interactions with other system values)** None.

## nstlbe

| | |
|---|---|
| **Name** | **nstlbe** - number of software TLB entries requested |
| **Range** | 256 (MINSTLBE) - memory limited |
| **Default** | 0 (all other Models) |

**Use**

The overhead of a Translation Lookaside Buffer (TLB) miss may adversely affect the performance of Precision Architecture implementations that use small and/or single-set TLBs. The software TLB is an intermediate data structure that provides buffering between the hardware TLB and the software page directory. The two-level mechanism that uses the Software TLB can decrease the overall TLB miss ratio and can also significantly reduce the number of cycles necessary to resolve a TLB miss. However, the software TLB is no substitute for a larger hardware TLB. The software TLB is two-way set associative and equally split, in its organization, between instruction translations and data translations.

Parameter **nstlbe** enables or disables use the two-level software TLB mechanism by specifying the number of software TLB entries requested. A zero value disables this software TLB entirely. A nonzero value specifies the number of software TLB entries requested. Each entry contains an address translation, the associated protection and a tag, and occupies 16 bytes of physical memory. The system adjusts any nonzero **nstlbe** request to meet the following constraints:

- The minimum number of software TLB entries (**MINSTLBE**) that will be allocated is 256.
- At least twice as many software TLB entries as there are physical hardware TLB entries, including set associativity, will be allocated to make the software TLB effective.
- The number of software TLB entries allocated must be a power of two.

The actual number of software TLB entries allocated is displayed on the console with the kernel configuration messages. For example,

```
using 1024 software TLB entries occupying 16384 bytes of memory
```

The software TLB is always disabled on multi-processor configurations because broadcast purge of the software TLB is not implemented.

**Dependencies (interactions with other system values)**

None.

## ntext

| | |
|---|---|
| **Name** | **ntext** - number of texts |
| **Range** | 10 -> 1044 |
| **Default** | (24+MAXUSERS+NETSLOP) |

**Use**

Parameter **ntext** defines the maximum number of shared text (that is, code) descriptors (data structures describing the text) that can be active at any one time. Note that this does not limit the number of processes sharing the same shared text program.

Attempting to start a new process that may require a new text descriptor (when no more text descriptors are available) generates a "text table is full" message on the console and the new process is killed.

**Dependencies (interactions with other system values)**

If there are no more text descriptors when starting the execution of a process (at load time), the console displays a message "text table is full", and the process is killed.

## semaem

| | |
|---|---|
| **Name** | **semaem** – "adjust on exit" maximum value for semaphores |
| **Range** | 0 -> min (semvmx, 32767) |
| **Default** | 16,384 |

**Use**

An undo is a special, optional, flag in a semaphore operation which causes that operation to be undone if the process which invoked it dies.

Parameter **semaem** is the maximum value by which a semaphore can be undone.

This value is cumulative per process, so if one process has more than one undo operation on a semaphore, the value of each undo operation is added up in the variable **semadj**. **semadj** is the number by which the semaphore will be incremented or decremented if the process dies.

Read the manual page for **semop(2)** for more detailed information on semaphore undos.

Any **semop** calls which attempt to set |**semadj**| > **semaem** result in an ERANGE error.

**Dependencies (interactions with other system values)**

semaem <= semvmx

## semmap

| | |
|---|---|
| **Name** | **semmap** – semaphore map |
| **Range** | 4 -> memory limited |
| **Default** | ((SEMMNI+1)/2+2) |

**Use**

Each set of semaphores allocated per identifier occupies 1 or more contiguous slots in the sem array. As semaphores are allocated and deallocated the sem array may become fragmented.

Parameter **semmap** dimensions the resource map which shows the free holes in the sem array. An entry in this map is used to point to each set of contiguous unallocated slots, and it consists of a pointer to the set, plus the size of the set.

If semaphore usage is heavy and a request for a semaphore set cannot be accommodated, the following message appears:

```
danger: mfree map overflow
```

You should then configure a new kernel with a larger **semmap**.

Fragmentation of the sem array is reduced if in usage of the system all semaphore identifiers have the same number of semaphores. Parameter **semmap** can then be somewhat smaller.

Four is the lower limit: 1 slot is overhead for the map and the second slot is always needed at system initialization to indicate that the **sem** array is completely free.

**Dependencies (interactions with other system values)**

(semmap-2) = maximum # of contiguous unallocated pieces of the sem array.
semmap<=(semmni+1)/2+2
If semmap is greater, then some allocated space will be wasted.

## semmni

| | |
|---|---|
| **Name** | **semmni** – number of semaphore identifiers |
| **Range** | 2 -> memory limited |
| **Default** | 10 |

**Use**

Parameter **semmni** defines the number of sets (identifiers) of semaphores available to the users.

When the system runs out of semaphore sets, the **semget** system call will return a ENOSPC error message.

**Dependencies (interactions with other system values)**

semmni <= semmns
semmns <= (semmni * semmsl)
semmap <= (semmni+2)

## semmns

| | |
|---|---|
| **Name** | **semmns** – total number of semaphores in system |
| **Range** | 2 -> memory limited |
| **Default** | 60 |

**Use**

Parameter **semmns** defines the total number of semaphores available to the users of the system.

When the system does not have enough contiguous semaphores in the sem array to satisfy a **semget** request, the call returns a ENOSPC error. This error may occur even though there may be enough free semaphores, but they are not contiguous.

**Dependencies (interactions with other system values)**

semmni <= semmns
semmns <= (semmni * semmsl)

## semmnu

**Name** **semmnu** – number of semaphore **undo** structures

**Range** 1 -> (nproc – 4)

**Default** 30

**Use**

An **undo** is a special, optional, flag in a semaphore operation which causes that operation to be undone if the process which invoked it terminates.

Parameter **semmnu** is the number of processes which can have undo's pending on a given semaphore. It determines the size of the sem_undo structure.

Read the manual page for **semop(2)** for a more detailed explanation of **undo.**

You should increase **semume** if the user gets ENOSPC errors on **semop** calls using the SEM_UNDO flag.

**Dependencies (interactions with other system values)**

- **semmnu** determines the size of the structure sem_undo, which in turn contains the substructure dimensioned by **semume.**
- There is no point in having **semmnu** > (**nproc**-4) since it is the largest number of processes in the system which could use semaphores simultaneously.

## semume

**Name** **semume** – semaphore **undo** entries per process

**Range** 1 -> semmns

**Default** 10

**Use**

An **undo** is an optional flag in a semaphore operation which causes that operation to be undone if the process which invoked it dies.

Parameter **semume** limits the number of semaphores that each process can have undos pending on.

Read the manual page for **semop(2)** for a more detailed explanation of **undo**.

When the user gets EINVAL errors on **semop** calls with the SEM_UNDO flag, then you should increase the value of **semume**.

**Dependencies (interactions with other system values)**

semume <= semmns

Parameter **semume** is the size of the substructure undo, which is part of the sem_undo structure. The size of sem_undo is determined by **semume**.

## semvmx

| | |
|---|---|
| **Name** | **semvmx** – semaphore maximum value |
| **Range** | 1 -> 65,535 |
| **Default** | 32,767 |

**Use**

Parameter **semvmx** is the maximum value that a semaphore is allowed to reach. This limit is imposed by the largest number that can be stored in a 16-bit unsigned integer (65,535).

A **semop** system call which tries to increment a semaphore value greater than **semvmx** will result in an ERANGE error. If **semvmx** > 65535, then semaphore values can overflow and these errors will not be caught.

**Dependencies (interactions with other system values)**

semaem <= semvmx

## shmmax

| | |
|---|---|
| **Name** | **shmmax** – shared memory maximum |
| **Range** | 2 Kbytes -> 767 Mbytes |

If **shmmax** is increased beyond the default, refer to Table D-1 at the end of this appendix.

| | |
|---|---|
| **Default** | 0x04000000 (64 Mbytes) |

**Use**

Parameter **shmmax** defines the maximum shared memory segment size in bytes.

**Dependencies (interactions with other system values)**

shmmin <= shmmax

## shmmni

| | |
|---|---|
| **Name** | **shmmni** – shared memory maximum number of identifiers |
| **Range** | 3 -> 1024 |
| **Default** | 100 identifiers |

**Use**

Parameter **shmmni** defines the maximum number of shared memory segments systemwide. Make it large enough to hold as many shared memory segments as will be used simultaneously.

**Dependencies (interactions with other system values)** None.

## shmseg

|  |  |
|---|---|
| **Name** | **shmseg** – shared memory segments |
| **Range** | 1 -> shmmni |
| **Default** | 12 |

**Use**

Parameter **shmseg** defines the maximum number of shared memory segments that can be attached to a process at any given time.

**Dependencies (interactions with other system values)** None.

## timeslice

|  |  |
|---|---|
| **Name** | **timeslice** – scheduling timeslice interval |
| **Range** | -1 -> 2147483647 |
| **Default** | HZ/10 |

**Use**

The timeslice interval is the amount of time one process is allowed to run before the CPU is given to the next process at the same priority.

The value of timeslice is specified in units of (10 millisecond) clock ticks. There are two special values:

|  |  |
|---|---|
| **0** | Use the system default value (currently 10 ticks, or 100 milliseconds) |
| **–1** | Disable round-robin scheduling completely |

**Impact on System**

This parameter will cause a process to check for pending signals when its timeslice expires. This guarantees that a process which does not make any system calls (including a runaway process in an infinite loop) can be terminated. Thus setting timeslice to a very large value, or to -1, can prevent such processes from getting signals.

You only need to change this parameter on systems dedicated to applications with specific realtime needs.

No memory allocation relates to this parameter. Some CPU time is spent at each timeslice interval, but this time has not been precisely measured.

**Dependencies ( interactions with other system values )** None.

## timezone

**Name** **timezone** - minutes west of the Greenwich meridian

**Range** 0 -> 1440

**Default** 420

**Use**

The timezone parameter indicates the minutes west of the Greenwich meridian:

```
struct  timezone tz = { TIMEZONE, DST };
struct timezone {
int tz_minuteswest; /* minutes west of Greenwich */
int tz_dsttime;     /* type of dst correction */
};
#define DST_NONE      0      /* not on dst */
#define DST_USA       1      /* USA style dst */
#define DST_AUST      2      /* Australian style dst */
#define DST_WET       3      /* Western European dst */
#define DST_MET       4      /* Middle European dst */
#define DST_EET       5      /* Eastern European dst */
```

**Dependencies (interactions with other system values)**

It is used with **dst** (daylight savings time). This should be made consistent with the TZ environment variable (see **environ**(MISC) and **login**(1) in the HP-UX Reference).

## unlockable_mem

**Name** **unlockable_mem** - unlockable memory

**Range** 0 -> (the available memory indicated at power-up)

**Use**

The parameter **unlockable_mem** defines the minimum amount of memory that will always be available for virtual memory and/or system overhead.

It limits the amount of memory that can be locked (lockable memory) to **unlockable_mem** (the available memory indicated at power up).

If **unlockable_mem** is greater than available memory, the system sets **unlockable_mem** to available memory.

Lockable memory is used for:

- Process images and overhead locked with **plock(2)**
- Shared memory segments locked with the SHM_LOC command of the **shmctl(2)** system call
- Miscellaneous dynamic kernel data structures used by the shared memory system and some drivers.

Any call that needs lockable memory may fail if the value is too small. Note that lockable memory limits the amount of memory that can be locked, but that this memory is available for virtual memory except when it is locked.

**Dependencies (interactions with other system values)**

unlockable_mem <= physical memory

## Virtual Memory Configuration

If you change **maxdsiz** or **maxssiz**, you must also change **dmmin** and **dmmax.** If you change **shmmax**, you must also change **dmshm.** If you change **maxtsize**, you must also change **dmtext**. Theses parameters interact and an inconsistent value might make your virtual memory system unworkable.

Also, as process size increases, the total amount of non-pageable physical memory needed increases. This is because page tables are non-pageable. Tables D–1, D–2, and D–3 contain the virtual memory parameters (with the values in decimal and hexidecimal).

**Table D–1. Virtual Memory Parameter Table**

| if maxdsiz or maxssiz is (in bytes): | dmmin should be: (in decimal) Values in 1 KB disk blocks | (in hex) | dmmax should be: (in decimal) Values in 1 KB disk blocks | (in hex) |
|---|---|---|---|---|
| 0 MB < X < 64MB | 32 | 20 | 4096 | 1000 |
| 64MB <= X < 128MB | 64 | 40 | 8192 | 2000 |
| 128MB <= X < 256MB | 128 | 80 | 16384 | 4000 |
| 256MB <= X < 512MB | 256 | 100 | 32768 | 8000 |
| 512MB <= X < 1 GigB | 512 | 200 | 65536 | 10,000 |

**Table D–2. shmmax and dmshm**

| if shmmax is (in bytes): | dmshm should be: (in decimal) Values in 1 KB disk blocks | (in hex) |
|---|---|---|
| 0 MB < X < 64MB | 2048 | 800 |
| 64MB <= X < 128MB | 4096 | 1000 |
| 128MB <= X < 256MB | 8192 | 2000 |
| 256MB <= X < 512MB | 16384 | 4000 |
| 512MB <= X < 1 GigB | 32768 | 8000 |

**Table D-3. maxtsiz and dmtext**

| if maxtsiz is (in bytes): | dmtext should be: (in decimal) Values in 1 KB disk blocks | (in hex) |
|---|---|---|
| 0 MB < X < 64MB | 2048 | 800 |
| 64MB <= X < 128MB | 4096 | 1000 |
| 128MB <= X < 256MB | 8192 | 2000 |
| 256MB <= X < 512MB | 16384 | 4000 |
| 512MB <= X < 1 GigB | 32768 | 8000 |

# Index

# Index (Continued)

# Index (Continued)

# Index (Continued)

# Index (Continued)

# Index (Continued)

# Index (Continued)

# Index (Continued)

# Index (Continued)

# Index (Continued)

# Index (Continued)

# Index (Continued)