HP-UX System Administrator Manual

for the HP 9000 Series 500

Manual Reorder No. 97089-90049

This software and documentation is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California.

Hewlett-Packard Company

3404 East Harmony Road, Fort Collins, Colorado 80525

Printing History

New editions of this manual will incorporate all material updated since the previous edition. Update packages may be issued between editions and contain replacement and additional pages to be merged into the manual by the user. Each updated page will be indicated by a revision date at the bottom of the page. A vertical bar in the margin indicates the changes on each page. Note that pages which are rearranged due to changes on a previous page are not considered revised.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.

April 1985...Edition 1

Table of Contents

Chapter 3: Concepts

Chapter 4: System Boot and Login

Getting Started 1

Welcome

This manual is written for you, the Series 500 HP-UX system administrator. Although some familiarity with computers is assumed, this manual will serve people with varying levels of expertise. If you are already a UNIX[1] expert, you will find much here that is familiar but may well encounter something new. The HP-UX Operating System is composed primarily of Bell Laboratories' System V.2 UNIX. However, Hewlett-Packard has incorporated its own extensions as well as features from the University of California at Berkeley Unix 4.1 and 4.2 BSD (Berkeley Systems Distribution) systems and from Bell's System V UNIX.

Who is the system administrator? The system administrator is the person responsible for installing the HP-UX Operating System software, 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 become an HP-UX "guru", the local expert to whom other HP-UX users go for help.

What's In This Manual?

This manual is a guide designed to help you fulfill your duties as system administrator. The following is an overview of the chapters in this manual. The manual also contains two appendices: one describes how to use the file system integrity check command; the other contains a list of features and commands that have been added at each system revision or update.

Chapter 1: Getting Started

This chapter provides an overview of the *System Administrator Manual*, explains the conventions the manual uses, mentions other manuals which will aid you in administrative tasks, points out differences between single-user and multi-user systems, and discusses the system administrator's responsibilities.

[1] UNIX is a trademark of AT&T Bell Laboratories.

Chapter 2: Installing HP-UX

This chapter defines terminology used throughout the manual, provides step-by-step instructions for installing the HP-UX Operating System software and explains what to do after the system has been successfully installed.

Chapter 3: Concepts

Certain concepts used and implemented in HP-UX must be understood by the system administrator. This chapter discusses such concepts as processes, IDs, the super-user, block and character input/output, the file system and its use of mass storage devices, compatibility issues, file protection, and memory management.

Chapter 4: System Boot and Login

Many wonderful and mysterious things happen between the time power is applied to the computer running HP-UX and the time a user has logged in (gained access) to the system. This chapter examines what happens and offers guidelines for modifying various files that affect this process. It also describes other configuration files and suggests how to use them to customize your system.

Chapter 5: The System Administrator's Toolbox

Arranged alphabetically by task, this chapter contains instructions for accomplishing tasks the system administrator generally performs.

Chapter 6: System Accounting

As system administrator you may want to periodically evaluate how well your Series 500 HP-UX system is operating, as well as how many resources those logging onto your system are using. This chapter discusses the various accounting features available on HP-UX, how to install them and how to produce various useful reports.

Conventions Used in this Manual

Naming Conventions

The following naming conventions are used throughout this manual.

- Italics indicate files and HP-UX commands, system calls, subroutines, etc. found in the *HP-UX Reference* manual as well as titles of manuals. Italics are also used for symbolic items either typed by the user or displayed by the system as discussed below. Examples include */etc/profile*, *date(1)*, *getpid(2)* and *tty(4)*. The parenthetic number shown for commands, system calls and other items found in the *HP-UX Reference* is a convention used in that manual.
- The first time a file is mentioned, the complete pathname is given; subsequent references to that file contain only the final file name unless there is some chance of ambiguity. For example, */etc/profile* is used instead of *profile* to avoid possible confusion with a user's personal *.profile* file.
- Boldface is used when a word is first defined (as **flebnee**) and for general emphasis (**do not touch**).
- Computer font indicates a literal either typed by the user or displayed by the system. Keys are shown capitalized and enclosed in an oval. A typical example is:

  `fsck /dev/7912` [RETURN]

  Note that when a command and/or file name is part of a literal, it is shown in computer font and not italics. However, if the command or file name is symbolic (but not literal), it is shown in italics as the following example illustrates.

  `fsck` *device_file_name* [RETURN]

  In this case you would type in your own *device_file_name*.
- 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* manual. Some of these entries will be under an associated heading. For example, the *chgrp(1)* entry is under the *chown(1)* heading. If you cannot find an entry where you expect it to be, use the *HP-UX Reference* manual's permuted index.

Keyboard Conventions

While installing the HP-UX system, you need to know about two keys: BACK SPACE and RETURN. The BACK SPACE key will let you fix typing mistakes made when entering information. Press BACK SPACE to "back up" over a mis-typed letter.

Most information typed at the system console must be followed by the RETURN key. All Hewlett-Packard terminals supported by HP-UX have a RETURN key.

Using Other HP-UX Manuals

Besides this manual, three other manuals will aid you in your system administrator tasks:

- The *Installation Guide* for your specific Series 500 computer contains instructions for installing the computer hardware, interface cards, and peripherals. The guide supplies all the hardware-specific information needed to set up the HP-UX system.
- The *HP-UX Reference* contains the syntactic and semantic details of all commands and application programs, system calls, subroutines, special files, file formats, miscellaneous facilities, and system maintenance procedures available on the Series 500 HP-UX Operating System. Use this manual when looking for complete specifications of a given command, special file, etc.
- The six volume set of *HP-UX Concepts and Tutorials* contains information on a broad range of HP-UX topics and tools. Several sections may be of particular interest to you: *Serial Network Communications, The Bourne Shell, The Model 520 Console,* and *vi.*

Single-user vs. Multi-user Systems

The Series 500 HP-UX Operating System is supplied as either a single-user or multi-user system (with a 16, 32 or 64 user license). The difference between the two types of systems is implied by their names — a single-user system can only be used by one person at a time; a multi-user system, by many at a time.

If you have a single-user system, many of the multi-user topics covered in this manual will still benefit you. Consider a discussion on how to set up and configure the LP Spooler (used to control line printer spooling). While this topic may be more critical in multi-user systems (where the demand on system resources is generally higher), using the LP Spooler on a single-user system can increase the performance and flexibility of that system.

If you have a single-user system and a procedure or task seems irrelevant, you probably can ignore it. Don't reject multi-user topics too quickly though — they often contain information useful for single-user systems.

The Administrator's Responsibilities

This section contains a brief discussion of the system administrator's responsibilities and tells you where to find related information.

Installing and Testing the Hardware

As system administrator, you should make sure that your computer is installed and operating properly by using the instructions and tests in the installation guide supplied with your computer. The computer hardware must function properly before HP-UX is installed.

Installing the HP-UX Operating System

The HP-UX Operating System is supplied on a ¼-inch cartridge tape and installed using a Command Set '80 (CS/80) hard disc drive. As system administrator, you are responsible for installing HP-UX. Instructions for accomplishing this task are provided in chapter 2, "Installing HP-UX".

Evaluating Users' Needs

Once HP-UX is installed, you should analyze the intended uses of the system. Knowledge of the number of users, the characteristics of each user, the system resources and peripherals required by each user, and the data/programs that must be shared by various user groups, will help you set up HP-UX for optimum performance. This also applies to single-user systems.

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 500 000 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.

Configuring HP-UX

How the operating system uses computer resources depends on certain values and configurations that you control. Configuring the system influences its efficiency and response time. Once familiar with the system, you can use the instructions in chapter 5, "The System Administrator's Toolbox", to alter the system configuration.

Allowing Users Access to the System

Once HP-UX is installed, you are responsible for modifying the operating system to allow access by other users. This involves providing each user a user name, a password, and a portion of the file system for his use. Instructions for adding users and assigning passwords are contained in chapter 5, "The System Administrator's Toolbox".

Adding and Moving Peripheral Devices

Another of your responsibilities is to add/move peripherals (printers, terminals, mass storage devices, etc.) to the HP-UX system as they are required. A list of peripherals supported by the Series 500 HP-UX system can be found in the *HP 9000 Series 500 Configuration Information and Order Guide* supplied with your system. Directions for installing the peripherals can be found in chapter 5, "The System Administrator's Toolbox".

Monitoring File System Use and Growth

As HP-UX is used, more and more files are added to the file system. If unused files are not removed, the amount of space required to store the files eventually exceeds available space. One of your responsibilities is to monitor the size of the file system and identify unused files. Unused files should be archived (if needed in the future) and then removed from the file system. Also, you should watch for files that continually increase in size. Consult the file's owner to ensure that the file is needed and to see if its size can be reduced. Instructions for monitoring the use and growth of the file system are supplied in chapter 5, "The System Administrator's Toolbox".

Updating the HP-UX System

You will receive software updates by purchasing HP support services that provide periodic updates. These updates modify existing capabilities and add new capabilities, ensuring that your system contains the latest version of the software.

As system administrator, you are responsible for installing each software update. You should update the manuals to include the documentation changes provided with each update and keep a log showing when the update was installed. Notify all system users of the changes caused by the update. Because each update depends on changes made by the previous update, it is imperative that each update be installed when it arrives. Instructions for installing updates are in chapter 5, "The System Administrator's Toolbox" and the "System Software History" appendix contains a list of features and commands that have been added at each system revision or update.

Backing Up and Restoring the System

The HP-UX Operating System, programming languages and applications software represent a large investment of time and money. Files can be unintentionally removed and each access to the system provides an opportunity for error. A critical error can cause additional errors in the file system and, when the system becomes sufficiently corrupt, file system errors increase rapidly.

Loss of the system can also occur through unwelcome circumstances (such as spilled coffee, smoke contamination, dust or fire) by damaging a mass storage device, its media, and/or the data it contains.

As system administrator, you should make a **backup** - a copy of the HP-UX Operating System, file system and programming languages. Depending on your system usage, consider backing up the system on a daily basis. Generally, base the frequency of your system backups on the answer to the question "How much data can I afford to lose?".

If your system is destroyed, you can recover by restoring the latest version of your 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 a system backup is the **only** way to recover a deleted or destroyed file. Instructions for using the supplied *backup* command and the CS/80 Tape Backup utility are given in chapter 5, "The System Administrator's Toolbox".

Detecting/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. With each update to the file system it is possible that one or more of the updates could fail (for example, because of abnormal system shutdown or abnormal program termination). When an update fails, the file system can become corrupt.

HP-UX provides the *fsck(1M)* command — a program that checks the integrity of a file system and (optionally) repairs that system. You should check the file system's integrity on a daily basis as well as each time HP-UX is booted. 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. Also, see the *fsck(1M)* entry in the *HP-UX Reference* manual.

Assisting Other Users

Since you carry the title "System Administrator", users may come to you for help with the system. You should plan on allocating a portion of your time for consulting and problem solving.

If you have purchased certain 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 "Back-up" Administrator

At least one other person should be trained as the system administrator to handle your responsibilities in the event of your absence.

To ease your job as system administrator and the job of the "back-up" system administrator, you should automate as many of your tasks as possible. Chapter 4 ("System Boot and Login") and chapter 5 ("The System Administrator's Toolbox") show you how to create programs that automatically perform tasks at specific times. By scheduling programs using system routines, you can automatically back up the system or initiate communications between your system and another HP-UX system (using the *uucp* utilities).

User Survey

Name ______________________ Location ______________________

Phone ______________________

Location where you will be using the system. ______________________

____

User Category

☐	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?

- ☐ Thermal printer
- ☐ Impact printer
- ☐ Graphics terminal
- ☐ Laser printer
- ☐ Plotter
- ☐ Removable mass storage device
- ☐ Other ______________________________

____

____

Are there other users with whom you want to share programs or data? ☐ yes ☐ no

If so, list them. ____

____

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

If so, how much must be "on-line" (accessible at all times)?

____

____

What long term data storage does your application require? ____________________

____

____

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

____

Installing HP-UX 2

This chapter provides a step-by-step procedure for installing the HP-UX Operating System on HP 9000 Series 500 computers, models 520, 530, 540 and 550. All references to HP-UX in this manual apply equally to the Series 500 HP-UX Operating System on all the products listed above unless specific variations are noted.

Useful Definitions

The following terminology is used throughout this procedure as well as in later chapters. Please consider it mandatory reading.

CS/80

A family of mass storage devices that communicate with a computer via the CS/80 (Command Set '80) or SS/80 (Sub Set '80) command set. A complete list of CS/80 and SS/80 devices supported on HP-UX can be found in the *HP 9000 Series 500 Configuration Information and Order Guide* supplied with your system. Note that **HP-UX can only be installed on one of the following peripherals containing an HP 88140 cartridge tape drive**:

- HP 7911P 28.1 Mbyte Disc/Tape Drive.
- HP 7912P 65.6 Mbyte Disc/Tape Drive.
- HP 7914P 132.1 Mbyte Disc/Tape Drive.
- HP 7914TD 132.1 Mbyte Disc/Tape/9-Track Tape Drive.
- HP 7942A 23.8 Mbyte Disc/Tape Drive.
- HP 9144A Cartridge Tape drive.
- HP 7946A 55.5 Mbyte Disc/Tape Drive.

Note also that the following CS/80 devices are supported on HP-UX:

- HP 7907A 20.5 Mbyte Fixed and 20.5 Mbyte Removable Disc Drive.
- HP 7933H 404 Mbyte Winchester Disc Drive

- HP 7935H 404 Mbyte Winchester Disc Drive with removable media.
- HP 7941A 23.8 Mbyte Disc Drive.
- HP 7945A 55.5 Mbyte Disc Drive.
- HP 9130K — the built-in 0.25 Mbyte flexible disc drive of the HP 9000 Model 520 Computer.
- HP 97093A — the built-in 10 Mbyte Winchester disc drive of the HP 9000 Model 520 Computer.

HP 88140 cartridge tape drive
The built-in cartridge tape drive of a CS/80 or SS/80 disc drive.

address
In the context of peripheral devices, a set of values which specify the location of an I/O device to the computer. The address is composed of up to four elements: select code, bus address, unit number and volume number.

Asynchronous Interface (ASI)
With the exception of an ITE as found on a Model 520 or on a Model 550 with the HP 98700 display device, every terminal on your system must communicate with the host via either an ASI or MUX card. The ASI and MUX cards communicate in an RS-232C compatible protocol, and support the standard 25 pin connector.

block
The fundamental unit of information Series 500 HP-UX uses for access and storage allocation on a mass storage medium (such as a CS/80 disc). A block is usually 1024 bytes.

boot or boot-up
The process of loading, initializing and running an operating system.

boot area
Each Structured Directory Format (SDF) volume has one boot area consisting of zero or more contiguous logical blocks. This area contains the kernel, device drivers and some optional products if installed, and is read into the memory of your Series 500 computer by the boot ROM upon power up. The boot area is completely outside the file area, and cannot be used for data storage. Its size is determined when the volume is initialized. To change the size of a boot area you must re-initialize the volume.

boot ROM

A program residing in ROM (Read Only Memory) that executes each time the computer is powered-up. The function of the boot ROM is to run tests on the computer's hardware, find all devices accessible through the computer and then load either a specified operating system or the first operating system found according to a specific search algorithm. The bootstrap program uses the boot ROM's mass storage drivers to load and pass control to the kernel. When the kernel gains control, it completes the job of bringing up the HP-UX operating system. Details of the boot ROM's search algorithm are given in chapter 4 ("System Boot and Login").

Depending on your boot ROM version, the boot ROM displays may differ slightly from those shown in this manual; any differences between boot ROM versions are noted in this manual when the topic in question is discussed. The boot ROM identifies its version when power is applied to the computer.

bus address

Part of an address used for devices, especially devices on an HP-IB (HP Interface Bus); a number determined by the switch setting on a peripheral which allows the computer to distinguish between two devices connected to the same interface. A bus address is sometimes called a "device address", and no two devices on the same HP-IB can have the same bus address.

destination device

The mass storage device on which HP-UX is to be installed. The destination device must be a CS/80 device other than the HP 9000 Model 520's built-in flexible disc drive or its built-in Winchester disc drive. The built-in cartridge tape drive of a CS/80 device should not be used as a destination device when installing the system.

Internal Terminal Emulator (ITE)

The "device driver" code contained in the HP-UX kernel and associated with the built-in keyboard and display on the Series 500 Model 520, or with the HP 98700 display device configuration on the Model 550.

kernel

The core of the HP-UX operating system. The kernel is the compiled code responsible for managing the computer's resources; it performs such functions as allocating memory and scheduling programs for execution. The kernel resides in RAM (Random Access Memory) whenever HP-UX is running.

login
The process of a user gaining access to HP-UX. This process consists of entering a valid user name and its associated password (if one exists).

multi-user state
A state of HP-UX when terminals in addition to the system console allow communication between the system and its users. The multi-user state (not to be confused with a multi-user system) is usually state 2; see Chapter 4 for details.

MUX
An interface card which communicates via the backplane to the I/O processor of your Series 500. MUX is an abbreviation for Asynchronous Multiplexer. There are two flavors of MUX cards available for the Series 500: the HP 27130A 8-channel MUX card, and the HP 27140A 6-channel modem MUX which supports modem protocol. Each channel is an RS-232C port which is normally associated with a */dev/ttyXX* file.

root volume
The mass storage volume upon which the root (i.e. /) directory resides. A disc may be marked as the root volume but not have a boot area. See the description of the boot ROM's search algorithm in Chapter 4 "System Boot and Login".

select code
Part of an address used for devices; a number determined by which slot in the I/O bus a particular interface card is inserted. Each interface card is in turn connected to a peripheral. Multiple peripherals connected to the same interface card share the same select code.

single-user state
A state of HP-UX when the system console provides the only communication mechanism between the system and its users.

source device
The mass storage device from which HP-UX is installed. The source device must be a CS/80 cartridge tape drive.

special file
Often called a device file, this is a file associated with an I/O device. Special files are read and written just like ordinary files, but requests to read or write result in activation of the associated device. These files should reside in the */dev* directory.

system console

A keyboard and display (or terminal) given a unique status by HP-UX and associated with the special device file */dev/console*. All boot ROM error messages (messages sent prior to loading HP-UX), HP-UX system error messages, and certain system status messages are sent to the system console. Under certain conditions (for example, the single-user state), the system console provides the only mechanism for communicating with HP-UX.

NOTE

The conventions for the system console as described here are **significantly different** than in versions earlier than 5.0.

HP-UX assigns the system console function according to a prioritized search sequence when the HP-UX kernel gains control during the boot-up process. On the Series 500, the search priority is:

1. On the Model 520, the keyboard and display associated with the ITE.
2. The lowest numbered HP 98700 display device with a monitor and keyboard on a Model 550.
3. A keyboard and display (terminal) associated with an HP 27140A 6-channel modem MUX, HP 27130B 8-channel MUX card, or an HP 27128A ASI card at minor number `0x000000` (the first port on the first slot).

If none of the above conditions is met, no system console exists. HP-UX does not tolerate this, and you cannot use HP-UX without a system console.

Following the installation or update of your HP-UX system to version 5.0, you should have a special device file called */dev/systty*. This is the device which is usually used for booting HP-UX, and will usually be linked to */dev/console*.

Another special device file, */dev/syscon*, is normally linked with */dev/systty*. If the super-user issues the *init* command from a device other than */dev/systty*, init will link */dev/syscon* to the originating device. Whenever the system is rebooted the original location of */dev/syscon* will be restored.

NOTE

To install the HP-UX system it is necessary to communicate with the boot ROM and the Series 500 HP-UX Installation Utilities. Hence, you must have a **supported** console as described in order to install HP-UX on your Series 500 computer.

For further information on the system console, see the *HP-UX Concepts and Tutorials* document "HP-UX and the HP 9000 Model 520 As System Console". For a complete list of Hewlett-Packard terminals supported by HP-UX, see the *HP 9000 Series 500 Configuration Information and Order Guide* supplied with your system.

unit number

Part of an address used for devices; a number whose meaning is software- and device-dependent but which is often used to specify a particular disc drive in a device with a multi-drive controller. When referring to single-controller CS/80 integrated disc/tape drive, a unit is used to distinguish between disc and cartridge tape drives.

volume number

Part of an address used for devices; a number whose meaning is software- and device-dependent but which is often used to specify a particular volume on a multi-volume disc drive.

Installation Overview/Checklist

For your convenience, here is a checklist of the major steps involved in the installation of the HP-UX Operating System. Use this list (or a copy of it) as you follow the instructions in this chapter and check off each item as you complete it.

- ☐ Install and test the hardware
- ☐ Confirm the part numbers on the HP-UX distribution media
- ☐ Check the CS/80 switch settings
- ☐ Confirm proper operation of the computer's boot ROM
- ☐ Load the first of the "Series 500 HP-UX Installation" tapes
- ☐ Follow and respond to the installation utility menus
- ☐ Remove the second installation tape, storing both tapes in a safe place
- ☐ Reboot from your newly installed HP-UX system
- ☐ Log in as the super-user (`root`)
- ☐ Follow the guidelines in the "After Installing HP-UX" section

Before Installing HP-UX

Before installing the HP-UX software, the hardware must be installed and tested. This includes the computer, the CS/80 disc drive (and its switch settings), and all interface and memory cards. The installation manuals that come with each system component explain how to install that component; a few critical items are reviewed here.

This section also contains information about the media on which your system is supplied.

System Distribution Media

The system is supplied on two HP 88140SC (DC-150) ¼-inch cartridge tapes for installation from the cartridge tape drive. Check the product numbers on the cartridge tape labels. One **must** be labeled "Series 500 HP-UX Installation, Tape 1" (contains the HP-UX Operating System). The other tape **must** be labeled "Series 500 HP-UX Installation, Tape 2". No other versions of HP-UX are available at this time for the Series 500 computers listed at the beginning of this chapter.

Verify the Hardware

With the computer, the CS/80 disc and any other devices attached to the system turned off, check the CS/80 switch settings and connections. Switch settings are explained in the Installation guide supplied with your computer. Your CS/80 device must be connected via either a high speed HP-IB interface card (HP 27110A), or the internal medium speed HP-IB interface card of the Model 550. The CS/80 device should be connected to the HP-IB interface card with an HP-IB cable. The cable connects from the end of the HP-IB card on the rear of the Series 500 computer to the port on the rear of the CS/80 device. The HP 7908 has only a single HP-IB port; use the lower port on the HP 7911, HP 7912, and HP 7914.

If you are using an internal cartridge tape drive in your CS/80 disc, and you have purchased the dual controller option for that device, the upper port connects to the tape drive. In this case you will either need a second HP-IB interface card to connect to the tape drive, or you will need a short HP-IB cable to jump from the lower port to the upper port. Should you jump the ports in this manner, you will need to make sure the bus address switch settings on the rear of the CS/80 device are different. If the bus address switch settings are the same, your Series 500 computer will be unable to determine which device it is talking to, and you won't be able to continue installing HP-UX.

CAUTION

DO NOT ATTEMPT TO UNPACK AND CONNECT A CS/80 DISC DRIVE (OTHER THAN THE HP-7908P) YOURSELF. THE CS/80 DISC DRIVES ARE PACKED TO PREVENT DAMAGE DURING SHIPMENT. TO PREVENT DAMAGE TO THE DEVICE, AN HP CUSTOMER ENGINEER MUST UNPACK, INSTALL AND TEST THE DEVICE.

Next, confirm that the bus address switch on the rear of the CS/80 root device (the device on which HP-UX is to be installed) and all other devices on it's select code are set to zero (0). Instructions for setting the switch are in the CS/80 Installation Manual. (The CS/80 Installation Manual refers to the bus address as "HP-IB device address".)

NOTE

Models 530, 540 and 550 must have an HP **supported** terminal installed to function as a system console before continuing with the installation procedure.

The system console must next be connected. If you are connecting the system console via an HP 27128A ASI Card (Asynchronous Serial Interface Card), the card **must** be installed such that its select code is zero (0). Refer to the computer's installation manual to learn how to install the interface card and determine its select code. Before installing the card, locate the set of eight switches on the ASI Card. **Set the eight switches on the ASI Card such that switches 1 through 7 are open and switch 8 is closed.** This configures the card in the following way:

Switch	Value	Meaning
1	open	Not used
2	open	Indicates a direct connection
3	open	No parity is used
4	open	Character length is 8 bits
5	open	
6	open	Switches 5-8 set baud rate to 9600
7	open	
8	closed	

Alternately, an HP 27130A Asynchronous 8-channel MUX card or an HP 27140A 6-channel modem MUX card may be used when connecting to the system console. It must be installed such that its select code is zero (0). The terminal's interface cable must be connected to channel zero (0) of the Multiplex Interface.

Next the terminal must be configured so that it can "talk" to HP-UX. The manual supplied with the terminal describes how to use the functon keys to configure the terminal. First, press the appropriate keys or switches for terminal configuration; select the hard-wired default values. If necessary, alter the appropriate fields such that the terminal's configuration parameters have the indicated values:

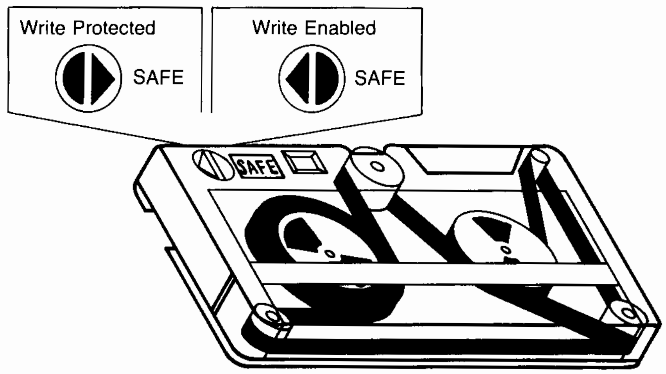
LocalEcho:	OFF	Bits/Character:	8
CapsLock:	OFF	Parity:	none
InhHndShk(G):	YES	RecvPace:	Xon/Xoff
Inh DC2(H):	YES	Full Duplex	
Baud:	9600	EnqAck:	YES

After the system is installed and running, you may change the configuration parameters to suit your own needs.

Now, verify that the computer, the CS/80 disc drive, the HP-IB interface card and console are installed and operating properly by following the instructions supplied in the computer's Installation Guide.

Load the First "Installaton" Tape

Now power up the CS/80 disc drive and remove the tape labeled "Series 500 HP-UX Installation, Tape 1" from its case. Locate the write-protect mechanism (labeled "SAFE") on the top, rear, left-hand corner of the cartridge tape as shown in the diagram below. **The arrow on the protect screw should point away from the word SAFE**. If it does not, use a coin or screwdriver to turn the protect screw such that the arrow points away from the word SAFE. In order for the installation process to be successful, your sytem **must** be able to write to the tape.



The CS/80 Cartridge Tape SAFE Mechanism

Holding the tape with the SAFE label in the rear left hand corner, insert the utilities tape into the tape drive door and push until it clicks into place. Only the BUSY indicator should now be lit. The tape drive will begin a cartridge tape conditioning sequence that takes approximately two minutes.

Begin the Installation

Turn the power to your Series 500 computer on. When the boot ROM assumes control of your system, it will attempt to boot an operating system. Since the loader finds removable media first, the installation program on the first installation tape will be immediately loaded into memory if your machine is properly configured. You will see a series of messages **similar** to the following on your console:

```
Loader Rev B
Testing Memory...
Searching for System...
Series 500 HP-UX Installation Utility


Load done.
```

NOTE

If any other message is displayed, refer to the appendix entitled "System Loader Messages" in the computer's installation manual. If no message is displayed, verify the operaton of the computer and its peripherals by performing the tests described in the computer's installation manual.

It will take a couple of minutes for the installation program to be loaded. Note that the menus from later versions of the installation utility may be somewhat different from those shown below. The procedure **will not** be any different. Once loaded, a menu will appear to ask you whether you wish to continue the installation proceedure:

```
*******************************************************
       HP-UX INSTALLATION UTILITY -- MAIN MENU
*******************************************************

   Choice          Description

    "b":           BEGIN Installation Process
    "e":           EXIT  Installation Process

Please enter choice >>
```

Installing Series 500 HP-UX will result in the complete initialization of the specified disc root volume. If you have anything you wish to keep stored on the destination CS/80 device, or if you wish to change your hardware configuration, type in `e` followed by a `RETURN`. Otherwise enter `b` and `RETURN`.

```
**************************************************************
          HP-UX INSTALLATION UTILITY -- DEVICE MENU
**************************************************************
Source Device                      Destination Device

Major Number               1       Major Number              1
Select Code                5       Select Code               5
Bus Address                0       Bus Address               0
Unit Number                1       Unit Number               0
Volume Number              0       Volume Number             0
Device File         /dev/mtc       Device File         /dev/hd


 Choice            Description

[Return]:          CONTINUE Installation Process
  "e":             EXIT Installation Process
  "s":             Change the SOURCE Device
  "d":             Change the DESTINATION Device

Please enter choice >>
```

Once you have begun the installation process a second screen menu will appear, showing you the default source and destination addresses. You may do any one of four things at this point.

1. You may decide to continue with the installation process, using the device parameters as shown. However it is highly unlikely that the parameters shown will exactly match your current machine configuration, so double check once more. If you are sure, type `RETURN` to continue (be careful not to type in several returns).

2. You may exit the installation process by typing `e` and `RETURN`. This will end the program and rewind the first installation tape.

3. You may change the address for the source device by typing [s] and [RETURN]. If you wish to change a value, type in the new value followed by a [RETURN]; otherwise the old value will be retained. Once you decide to change the address for the source device, you will be prompted for each of the following values.

   a. The source major number. This will usually be 1.

   b. The source select code. The range for the select code is 0 through 23, inclusive.

   c. The source bus address. The range for the bus address is 0 through 7, inclusive.

   d. The source unit number. The range for the source unit number is 0 or 1.

   e. The source volume number.

4. You may change the address for the destination device by typing [d] and [RETURN]. If you wish to change a value, type in the new value followed by a [RETURN]; otherwise the old value will be retained. Once you decide to change the address for the destination device, you will be prompted for each of the values discussed above in step 3.

When you see that the addresses for the source and destination devices are correct as shown on the device menu, press [RETURN] to continue installing HP-UX on your Series 500.

```
******************************************************
     HP-UX INSTALLATION UTILITY -- BOOT AREA MENU
******************************************************


        DISC INTERLEAVE FACTOR:             1
        DISC BOOT AREA SIZE:          1100000
        DISC BLOCK SIZE:                 1024


    Choice         Description

    [Return]:      CONTINUE Installation Process
     "e":          EXIT Installation Process
     "i":          Change Disc INTERLEAVE Factor
     "a":          Change Disc Boot AREA Size
     "b":          Change Disc BLOCK Size

    Please enter choice >>
```

Now that you have determined the source and destination device addresses, you will next get the boot area menu. This menu displays the optimal interleaving factor, boot area size and optimal block factor for the root disc (destination device). At this point you have five options:

1. You may decide that the values shown are correct, and continue the installation process by pressing RETURN.
2. You may exit the installation process by typing e and RETURN. This will end the program and rewind the first installation tape.
3. You may want to use a different interleaving factor than the one shown — type i and RETURN to change the value. When prompted type in the new value followed by a RETURN; if you do not type in a value the default will remain unchanged.
4. You may decide to use a different boot area size for your system. The boot area is that part of the disc which contains the kernel, device drivers and optional products such as LAN code. The kernel requires about about 600 kbytes, with LAN requiring an additional 300 kbytes. In order to install certain drivers, software updates and optional products you will want to leave additional space in your boot area. The only way to make your boot area larger is to re-initialize your disc and re-install HP-UX, so the recommended boot area size is 1100000 bytes (1.1 Mbytes).

   To change the boot area size, type a and RETURN. You will be prompted for a new value — if you wish to change the size, type in the new value followed by a RETURN. Otherwise just press RETURN.
5. You may want to change the block size for the destination from that shown. This is the smallest unit in bytes by which you can segment your disc. Type b and RETURN to change the value shown. Enter the new value followed by a RETURN to enter the new value.

```
**************************************************
     HP-UX INSTALLATION UTILITY -- INSTALL MENU
**************************************************

                 DESTINATION DEVICE

            Major  Number            1
            Select Code              5
            Bus    Address           0
            Unit   Number            0
            Volume Number            0
            Device File       /dev/ext


  Choice         Description

   "b":          BEGIN Initialization and Installation
   "e":          EXIT Installation Process

  Please enter choice >>
```

By successfully completing all of the proceeding steps you should be at this final menu. The destination device for the installation procedure should be shown along with the proper address and device file. At this point you may decide to abort the installation process by typing [e] and [RETURN], or to begin the initialization of the root disc and the installation of HP-UX by pressing [b] and [RETURN].

If you decided to begin the initialization/installation process, the following prompt will appear at the bottom of your screen:

```
WARNING !!!!!! Completion of Install will RE-INITIALIZE Destination Device!
Do you want to proceede?  >>
```

The **only** way to install HP-UX is to type in `yes` and [RETURN]. Any other response will abort the installation process. If you think that any of the parameters you entered on previous menus are wrong, or if you do not wish to erase any information on the destination disc, type `no` and [RETURN].

CAUTION

DO NOT SWITCH OFF POWER TO THE COMPUTER OR CS/80 DEVICE DURING DISC INITIALIZATION. TERMINATING THE INITIALIZATION PROCESS IN THIS FASHION MAY SERIOUSLY CORRUPT THE DISC MEDIUM.

The following table shows the typical initialization times (it varies with different interleave factors) for the CS/80 discs supported by Series 500 HP-UX:

Initialization Time for CS/80 Discs

CS/80 Disc	Size	Initialization Time
HP 7908P	16.6 Mb	9 minutes
HP 7911P	28.1 Mb	4 minutes
HP 7912P	65.6 Mb	10 minutes
HP 7914P/TD	132.1 Mb	14 minutes
HP 7933/7935	404 Mb	49 minutes

Following disc initialization, the HP-UX installation facility will install a full kernel without optional segments (e.g. un-used device drivers) and a subset of commands onto the destination disc. Once this procedure is done the source tape drive will rewind and release the tape labelled "Series 500 HP-UX Installation, Tape 1". When you are prompted, remove the first tape and replace it with the tape labelled "Series 500 HP-UX Installation, Tape 2".

After the second tape is loaded, a menu will be displayed that shows all of the products on the tape and will ask you which products should be installed. **You must install the core product.** This process should take about 25 minutes, and once completed will rewind the second tape and release it. Remove the second tape, and place both tapes in a safe place away from harsh environments. By leaving a data cartridge near a large motor or other EMF emitting device, you will risk loosing the integrity of your data.

NOTE

Now that your system is installed, you have the ability to modify and customize the system. However, if you have purchased software support services from Hewlett-Packard, only limited changes can be made without voiding your agreement. Consult your support plan or HP System Engineer regarding intended changes.

You may make certain modifications suggested in this manual without concern for voiding your agreement. Specifically, all the files listed in the "Summary" (of configuration files) section at the end of Chapter 4 and in the "Controlling Disc Use" section of Chapter 5 may be modified within reason.

After Installing HP-UX

Setting Minimum Protections

Some of the system's protections are not set up when you first install your HP-UX system. The most important of these is that the user `root` has no password. This means that anyone with access to the system console (or any system terminal for that matter) can login as `root` (also called the "super-user"). The super-user can execute critical system commands not accessible by regular users. By definition, the super-user is potentially dangerous to the system.

NOTE

Do not execute any commands — except those specified in this section — while logged in as the super-user (the user with the user name, `root`) until you are very familiar with the system. Otherwise, you may inadvertently damage the operating system. While getting familiar with the system, log in with the user name **`guest`**.

Setting the Password for Root

To protect your system, log in as the user `root` and assign a password to the root user by typing:

`passwd` [RETURN]

The system will prompt you for a password. Enter at least 7 characters and/or digits of your choosing and press [RETURN]. Note that control characters like those generated by [BACK SPACE] are accepted but sometimes difficult to remember. The password you enter is not displayed on the console. The system then prompts you to re-enter 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 will be prompted to enter a password twice again. The user `root` will now have to enter that password to login to the system.

Write down the password you assigned (in a secure place); if it is lost or forgotten, no one can log in as the super-user. If the super-user password is lost, the system will have to be re-installed (which will destroy any existing files on the system). If this happens, call your local HP sales and service office for assistance.

An Unattended System Console

One other protection item deserves mention: depending on the perceived need for security at your installation, use discretion about leaving a system console unattended while logged in as the `root` user as this defeats the password protection. Remember, any user logged in with the name `root` can execute **any** HP-UX command — a situation possibly hazardous to the integrity of your system.

Setting the System Clock

Check to see if the system clock is set correctly. Enter:

`date` RETURN

The time displayed is in the format of a twenty-four hour clock (for example, 2:00 pm is 14:00). If the time displayed is not correct, you need to first set the correct time zone and then set the system clock.

As shipped to you, the system is set up to run in the Mountain Time Zone. To temporarily change the time zone to your time zone, type two entries of the following form:

`TZ=XXXHYYY` RETURN

`export TZ` RETURN

where `XXX` and `YYY` are three letter representations of the standard and daylight time zones for your area and `H` represents the difference between current local time and Greenwich Mean Time, in hours. The `export TZ` line will remain the same regardless of the time zone. For example, in Denver, Colorado you would enter the following:

`TZ=MST7MDT` RETURN

`export TZ` RETURN

where MST stands for Mountain Standard Time and MDT stands for Mountain Daylight Time. Here are some other examples:

- In the Eastern time zone, use: `TZ=EST5EDT`
- In the Central time zone, use: `TZ=CST6CDT`
- In the Pacific time zone, use: `TZ=PST8PDT`

Until you change the TZ variable in the file */etc/profile* (described in the section "Setting the System Clock" in the "System Administrator's Toolbox" chapter), you will need to set the time zone as described here each time the computer is powered up.

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} [RETURN]
```

`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. For example, 84 specifies 1984 as the current year. This parameter may be omitted if the year is already correct.

When *date* is executed it echoes the time and date on your screen. If the time and date are not correct, repeat the above procedure. Note that you must be the super-user to change the date.

Log out by holding the [CTRL] key depressed as you press the [D] key. Alternatively, typing

```
exit [RETURN]
```

will also log you off the system. A few seconds after you log off, the `login:` prompt will re-appear.

Moving On

Now that your system is successfully installed and a few preliminary tasks are done, here are a few guidelines for learning more about HP-UX.

As shipped to you, the HP-UX system is set up to allow any of several users to log in. (This is true even for single-user systems where only one user can be on the system at a given time.) Two of the user names are of immediate interest: `root` and `guest`. Until you are familiar with HP-UX and understand its operation, you should avoid logging in as `root` for the reasons previously discussed.

HP-UX is a large and somewhat complex operating system, so time invested in learning the system can be substantial. This time, however, is well spent; the more you know about the system's features, the less work you will do in the long run as the system administrator.

You should familiarize yourself with the tutorial supplied with your HP-UX system: *Introducing the UNIX System*. Try the examples on your system as you read. **While gaining familiarity with the system, log in as the user `guest`**. The user name `guest` is shipped to you without a password. Thus, the procedure for logging in as `guest` is the same one you used to log in as the user `root` earlier in this chapter.

It is assumed that you have already read chapter 1 ("Getting Started") of this manual and the terminology at the beginning of this chapter. The following reading is also necessary.

If you already know the HP-UX (or UNIX) system, the next step is to explore the contents of the *Introducing the UNIX System* tutorial. Minimally, you should at least know what is contained in this tutorial so you can refer to it when needed.

If you are not familiar with either HP-UX or UNIX, the next step is to read closely *Introducing the UNIX System*, trying the examples as you read.

Finally, regardless of your expertise with HP-UX or UNIX, read chapter 3 ("Concepts") and chapter 4 ("System Boot and Login") in this manual. These chapters will give you a good understanding of how the system operates.

It is also a good idea to read the introduction to the *HP-UX Reference* manual and to familiarize yourself with using the "permuted index" section of that manual. As mentioned in the "Conventions Used in this Manual" section of chapter 1, the permuted index is useful for finding entries that are listed under similar, related entries.

Once you are comfortable with the material listed above you will have to finish configuring your system. Until you have done so, **you should be the only one using the system as the appropriate protections and system configuration are not in place.**

After completing the recommended reading, log in to the system as the super-user (log in with the user name `root`). Then perform the following operations:

- Add peripheral devices such as terminals, printers, plotters and disc drives to the system as well as the special (device) files required to support them. See the "Adding/Moving Peripheral Devices" section of the "System Administrator's Toolbox" chapter.
- Add other users to the system. See the "Adding/Removing Users" section of the "System Administrator's Toolbox" chapter.
- Modify */etc/motd*, */etc/profile*, and */etc/rc* to fit your installation's requirements. See the "System Boot and Login" chapter for guidelines.

After you have completed these configuration tasks, it is strongly recommended that you make a backup of the entire system. There are two major ways to accomplish this and both are explained in the "Backing Up and Restoring the File System" section of the "System Administrator's Toolbox" chapter.

Notes

Concepts 3

This section discusses several essential concepts needed to manage an HP-UX system. It is not necessary to initially understand all of these concepts in depth, however you should at least be familiar with the terms.

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 associated with a unique integer value (called the process ID) which is used to identify the process.

Process Creation (Parent and Child Processes)

A process consists of a single executing program at any given time. However, a process can create another process to:

- concurrently execute another program.
- execute another program 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.

Using fork

When a child 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 is 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 non-zero 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 his 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. And because each process has its own copy of the test data, each can modify the data without affecting the other process.

Using exec

One modification which often follows the *fork* system call, is to *exec* another program. *exec* is a system call which overlays a separate code segment on top of already existing process code. In this manner a parent process is able to create a new process using *fork*, and subsequently execute an entirely different program via *exec*.

As an example, let's suppose we are writing a text editor. We would like to let the user of our editor pause and list directories on the system — say before choosing a file to edit. One way of doing this would be to *fork* a different process, and then immediately *exec* the program *ls*. Let's look next at the *vfork* system call for a more efficient way of doing this.

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 alternatate 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 is simply going to call *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:

- the program that is executing in the process successfully completes.
- the process intentionally terminates itself by calling the *exit* or *_exit* system call.
- the process receives from any process a signal for which the default action is taken.

When a process "dies" (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 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. Each active member of the process group is identified by the process ID of the process group leader. 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 ensure that when an affiliated process group leader terminates, all members of its process group also terminate.

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 character (DEL), QUIT (CTRL /), the BREAK key or a modem hangup 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. Three of these are usually opened automatically when a process is created: **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**, and an **effective group ID**.

A **real user ID** is a integer value which identifies the owner of the process. Similarly, a **real group ID** is an integer value which identifies the group to which the user belongs. The real group ID is a unique integer identifer that is shared by all members of a group. It is used to enable members of the same group to share files without allowing access to these files by non-group members. The real user ID and real group ID are specified by the file */etc/passwd* and are assigned to the user at login.

Effective user and group IDs allow the process executing a program to appear to be the program's 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 program can be protected such that when executed, the process's effective IDs are set equal to the real IDs of the program's owner. The new effective ID values remain in effect until:

- the process terminates
- the effective IDs are reset by an "overlaying" process (a process is "overlaid" via the *exec* system call)
- 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 program in a limited manner. When the effective user ID is zero, the user is allowed to execute system calls as the super-user (described in the following section).

For example, suppose that the dean of a university keeps all of his student's records in a file on the system. He wishes to enable a professor to modify a student's record only for that professor's class (an English professor shouldn't be allowed to modify a student's grade in physics). The dean first protects the file containing the student's records such that only he may read or write to it. He then writes a program which receives the modifications requested by a user, checks to see that the user is allowed to make such changes, and then modifies the record if allowed. Finally, the dean protects the program such that the effective IDs of the user are set equal to the dean's real IDs when the program is executed. Then when the program accesses the student record file, the system allows the program to read from or write to the file because it believes that the dean is accessing the file (the effective user and group IDs are that of the dean).

The Super-User

The term super-user describes those system user whose effective user ID equals 0. Users with effective user ID equal to 0 are provided with special capabilities by HP-UX (hence the name "super-user"). Many commands and system calls can only be accessed by a super-user. Other commands and system calls provide additional features that can only be accessed by a super-user. A super-user is granted the ability to:

- execute any command in the system, as long as any execute permission permission bit is set in the command file's mode.
- override any protections placed on user files (except those created and protected by other systems, such as the BASIC Language System).
- modify any system configuration files, add (and remove) users to the system.
- other system functions.

In short, a super-user has the ability to modify the system in any fashion he desires (for better or worse). Thus, super-user status should only be granted to those users who have a thorough understanding of the system and have a need for super-user capability. As system administrator, you must have super-user capability.

Some super-user commands and some system calls (those used heavily by the system administrator) require the user's name to be "root" and his real user ID to be zero. You should maintain a super-user on the system whose user name is "root" and whose real user ID is zero. (This user is often referred to as "the root user".) Log in as this user when acting as system administrator. To prevent other users from accessing super-user capabilities, assign a password to "root". Only you and the "back up" system administrator(s) should know this password.

By convention, the super-user is a member of the group "other".

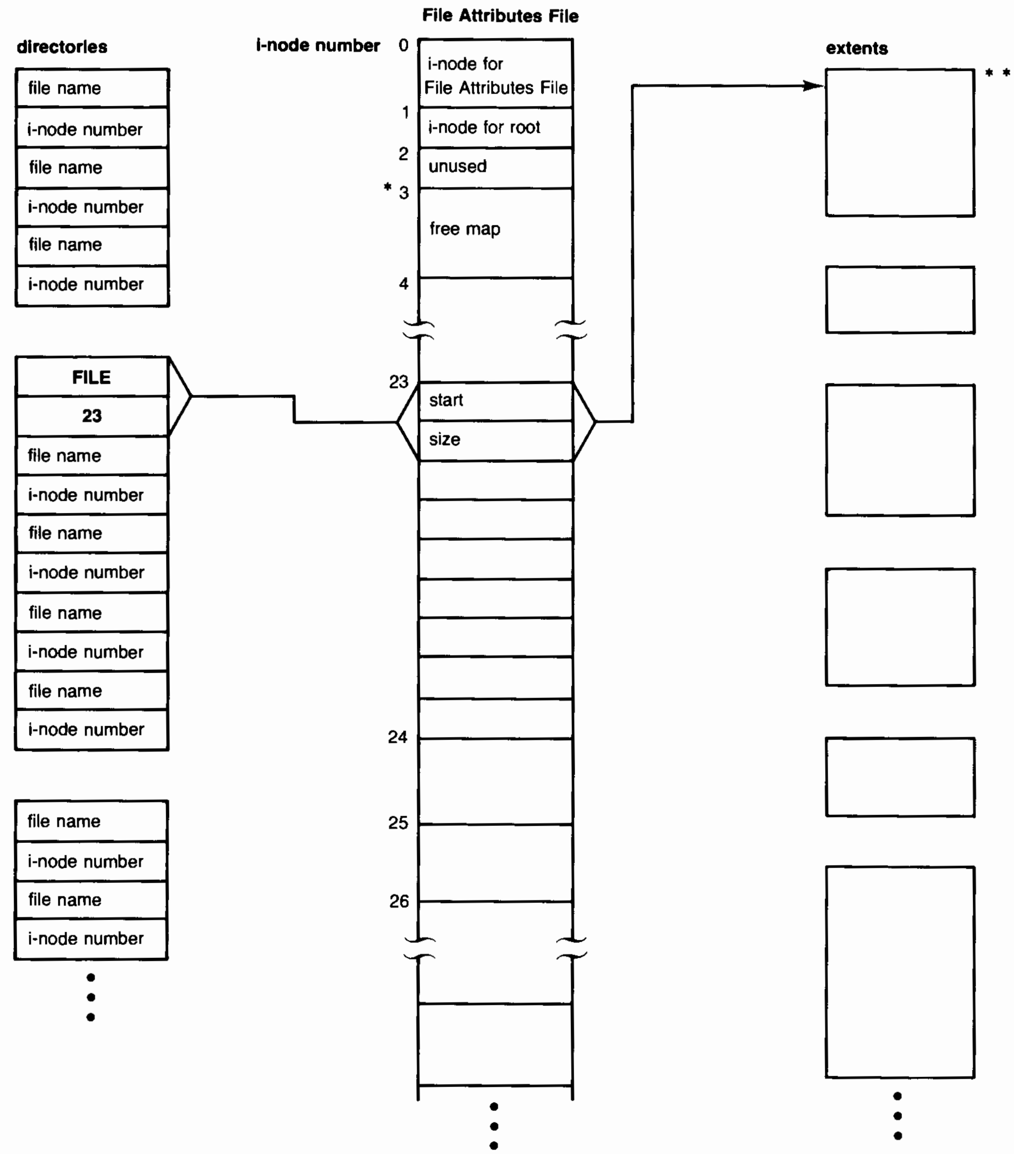
File System Implementation

The supplied tutorial text for the operating system discusses the structure of the file system and introduces some basic concepts and terms. This section expands on those concepts and introduces new concepts which are unique to HP-UX. This information should aid you when verifying, maintaining and repairing the HP-UX file system(s).

The files of the HP-UX file system are stored on a formatted mass storage medium, usually a disc. A file is specified by a path name — a series of directory names separated by / characters and terminated with the file name. This method is easy to use and remember but difficult for the computer and disc to manage efficiently. The actual method in which files are stored is explained by the text and diagrams that follow.

The file system implementation is a concept best explained by example. Suppose that you have a file named FILE which contains a list of mailing addresses. When FILE is created, the system places its name in a directory (another HP-UX file). It then associates a unique integer value (in this case, 23) with the file name. This integer value (called an i-node value or number) points to a structure called an **i-node**. In turn, the i-node contains a pointer to one or more groups of contiguous disc blocks, called **extents**, in which FILE's data is actually stored. A pointer in the i-node consists of the starting address of the file's extent(s) and the size of the extent(s). This is shown in the illustration that follows. All i-nodes for a file system are kept in a file called the **File Attributes File**. Each disc has a single File Attributes File which describes the contents of the disc.

File System Implementation
File with a Single Extent



* The free map may actually occupy more than one i-node.

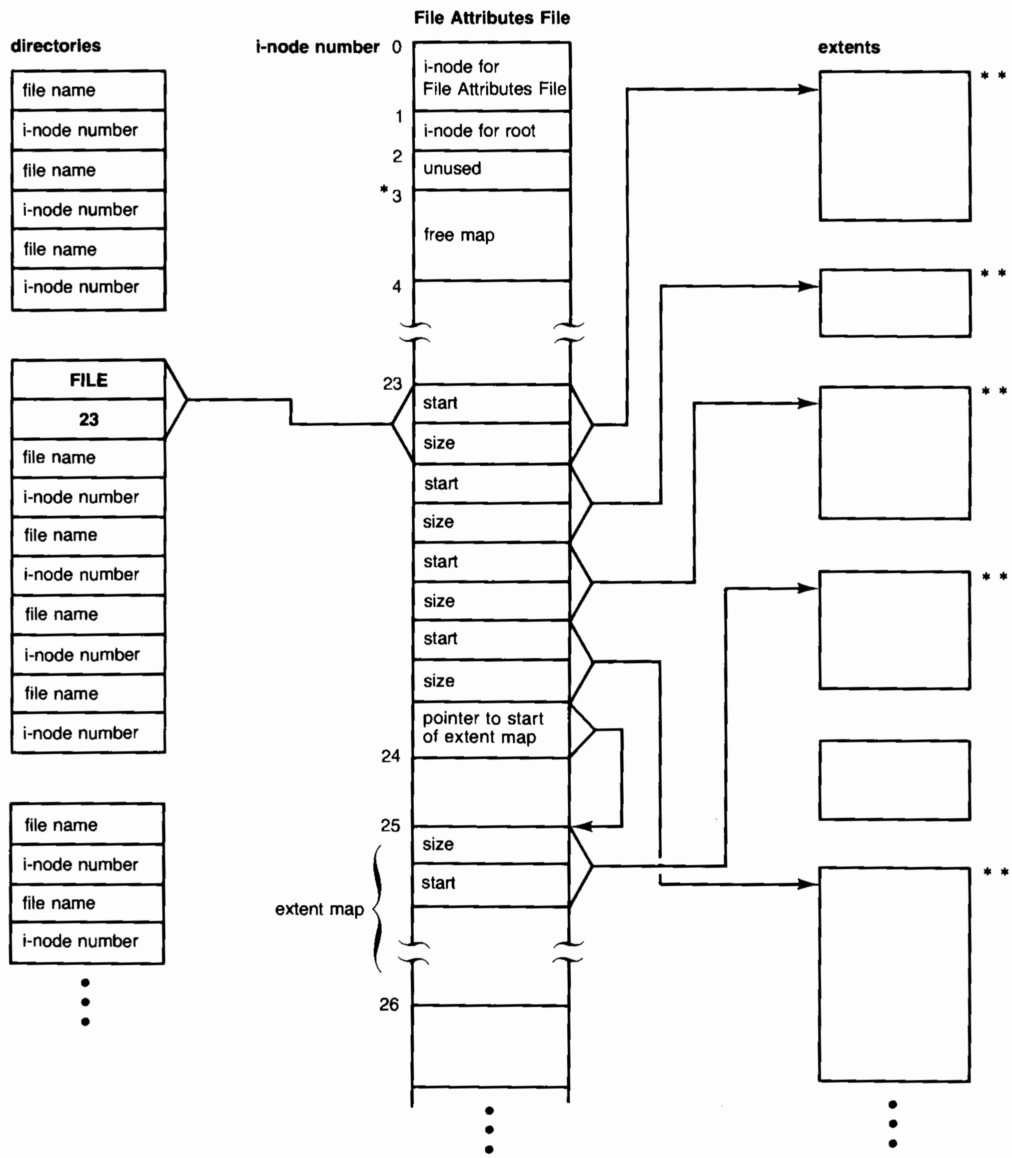
* * Actual contents of FILE.

From the diagram, you can see that besides an i-node for each file stored on the disc, the File Attributes File contains an area called the **free map**. The free map is an array of binary values, one for each block on the disc. If a bit value is 0, the corresponding disc block is currenty being used for storage. If a bit value is 1, then the corresponding disc block is available for storage.

When a file is created, its contents are placed in one large extent, if possible. If the file is later modified and enlarged, the system attempts to place the added information in blocks that are contiguous with the existing extent. If there are not enough free blocks that are contiguous to the existing extent, the system places the added information in a single new extent (if possible); it then adds to the file's i-node, a pointer to the additional extent. The i-node is capable of holding four pointers to the extents which comprise the file. If the system finds it necessary to create more than four extents for a file, an extent map is automaticaly added (chained) to the file's i-node. An extent map is capable of holding an additional 13 pointers; there is no limit to the number of extent maps which may be added to an i-node. You should note, however, that as the number of extents claimed by a file increases, the amount of time required to access the information also increases.

The following diagram shows the structure of the file, FILE (from the previous example) after it has been modified several times. When FILE was modified, there were not enough free blocks contiguous with the existing extent in which to place the added information. Thus, additional extents were created to hold the new information. When the number of extents claimed by FILE surpassed four, the system added an extent map to FILE's i-node.

File System Implementation
File with Multiple Extents



* The free map may actually occupy more than one i-node.

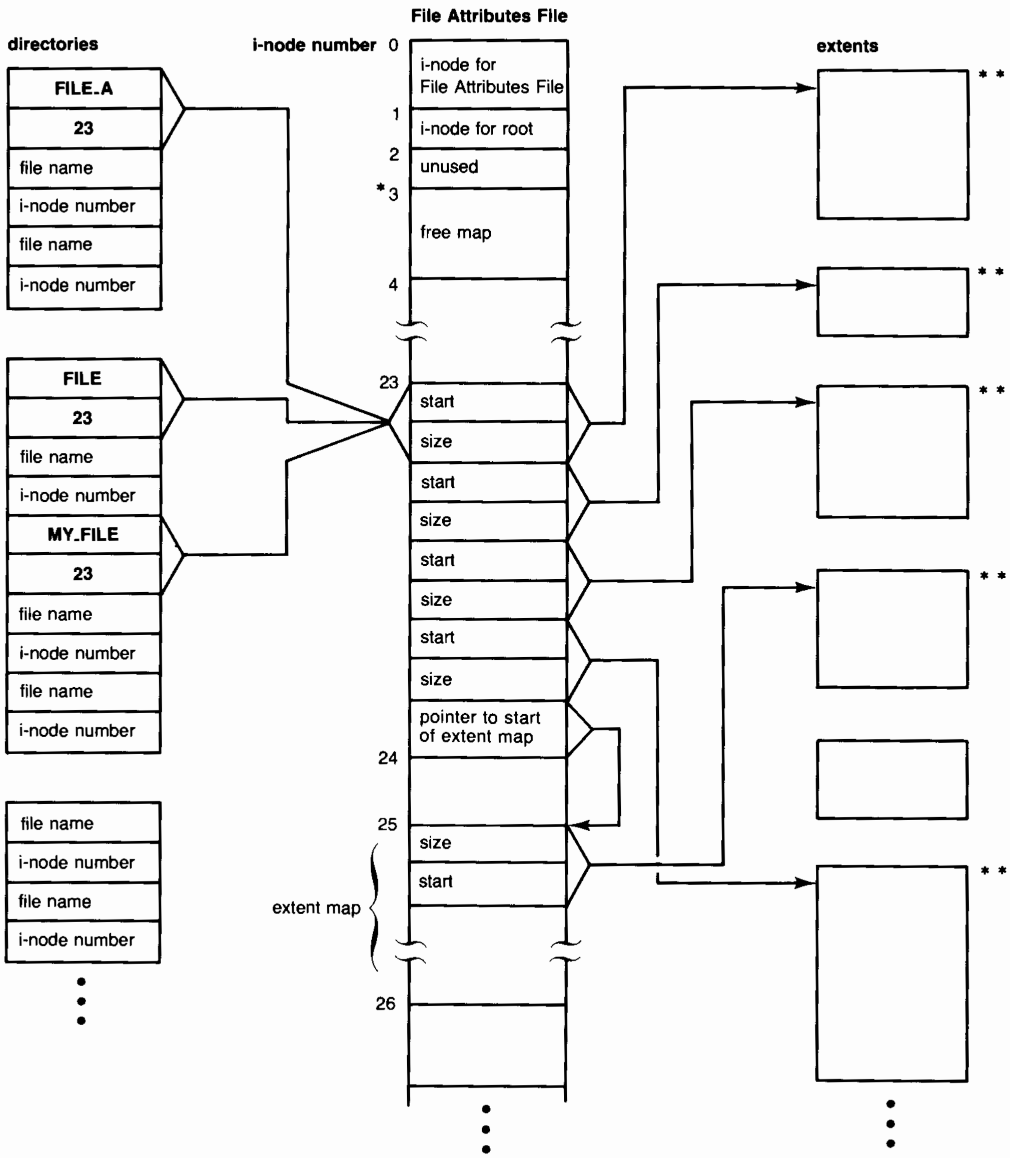
* * Actual contents of FILE.

In HP-UX, it is not unusual to have a single file on a disc which has several file names; the file is said to have multiple links. A file has multiple links when the file's i-node is pointed to by two or more directory entries. Each directory entry has its own name for the file but all share the same i-node value and thus, the same extents containing the file's data.

For example, suppose that three users all wish to have the file FILE (from the previous examples) in their login directory. Each user creates a link to FILE (via the *link* or *ln* command) from his login directory. From the viewpoint of each of these users, the file resides in their login directory.

A link is removed when a user removes the file from his directory (for example, with the *rm* command). When all links to an i-node have been removed, the disc blocks forming the file's extents are marked "free" in the free map. The file's i-node and extent maps (if it has extent maps) are marked as unused.

File System Implementation
File with Multiple Links



* The free map may actually occupy more than one i-node.

* * Actual contents of FILE.

The Volume Header

Each disc or tape medium available to your HP-UX system is called a volume. Each volume has a **volume header** located at block 0 of the volume. The volume header:

- identifies the volume format (always Structured Directory Format [SDF] for HP-UX files).
- specifies the size in bytes of each disc block.
- lists the maximum number of disc blocks available on the disc.
- lists the starting block number of the File Attributes File.
- lists the size and the starting location of the boot area.
- contains other system information (as described in *fs(5)* in the HP-UX Reference manual).

File Format and Compatibility

The format of the mass storage media on which HP-UX files are stored is Structured Directory Format (SDF). This is not necessarily the format for other operating systems patterned after the UNIX* operating systems.

If you are using both HP-UX and the BASIC Language System on your computer (supported on the Model 520 only), you should consider the following compatibility issues:

- BASIC recognizes many different file types (BDAT, BIN, ASCII, DATA and BCD).
- To BASIC, all HP-UX files are BDAT files.
- When BASIC files are copied with HP-UX commands, the file type of the copy is always BDAT (regardless of the file's original type).
- If a file or directory is used by both BASIC and HP-UX, be careful in assigning protections. A file protected in BASIC may not be accessible from HP-UX. Files with protection (access restrictions) in HP-UX may not be accessible from BASIC.

This last issue needs to be examined in more detail. When an HP-UX file is accessed from BASIC, the file's HP-UX mode corresponds to the file protection of a BASIC file. The HP-UX file's read and write permissions for the class of users "other system users" (not the file's owner nor the file's group) are interpreted as the read and write protections of a BASIC file. The file's execute permission (as specified by its mode in HP-UX) is ignored when the file is accessed from BASIC.

---

[1] UNIX is a Trademark of AT&T Bell Laboratories, Inc.

The read, write and execute permissions of an HP-UX directory affect the manner in which the directory (and the files it contains) can be accessed from BASIC. The interpretations of the different protections for directories are shown in the following table:

HP-UX Directory Protection	BASIC Directory Protection
r - (read permission) allows the user to list the contents of the directory. (read permission)	R - (read permission) allows the user to list the contents of the directory with the BASIC CAT statement.
w - (write permission) allows the user to add and remove files from the directory.	W - (write permission) allows the user to create or delete entries from the directory.
x - (execute permission) allows the user to search the directory. This permission must be set in order to access a file below the directory in the file system hierarchy.	No equivalent BASIC protection.

For an HP-UX file to be accessed from BASIC, all directories in the file's path must have both the read and execute permissions enabled for "others". When an HP-UX file or directory is examined with the BASIC CAT statement, it is not possible to determine whether or not its HP-UX execute permission is enabled.

Compatibility Tips

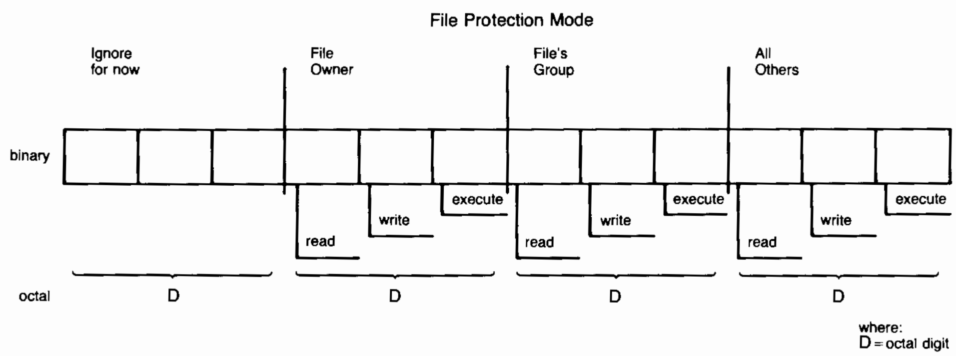
When creating files for use by both the BASIC Language System and HP-UX, assign no file protection to the file or to the directories containing the file. This ensures that both systems can access the file.

Use BASIC to back up volumes containing both BASIC Language System files and HP-UX files. This preserves the file types of all of the files.

File Protection

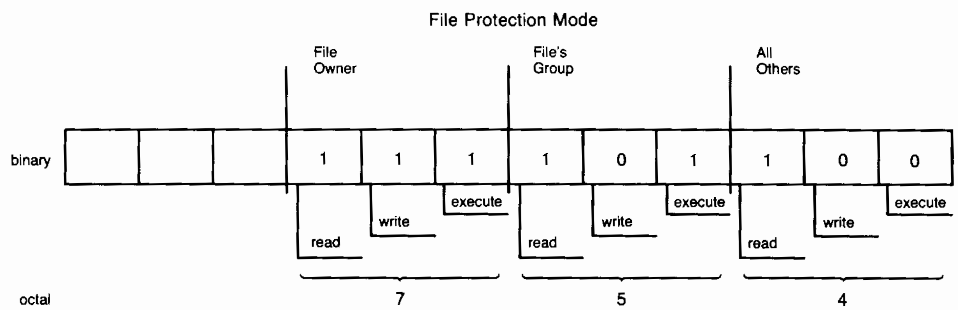
When each file in the file system is created, it is assigned a set of file protection bits, called the mode of the file. The mode determines which users may read from the file, write to the file, or execute the program stored in the file. Read, write, and execute permissions for a file can be set for the file's owner, all members of the file's group (other than the file's owner) and all other system users. The three permissions are mutually exclusive - no member of one permission group is included in any other permission group. 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 the *umask* command when the file is created. The mode may be changed with the *chmod* command. The mode of the file is represented as the binary form of four actal digits as shown in the following diagram. The initial discussion deals with only the three least significant digits. When the most significant digit is not specified, its value is assumed to be zero (0).



Each octal digit represents a three-bit binary value: one bit specifies read permission, one bit speciffies 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 a mode, 754 (octal). Octal 754 is equivalent to 111 101 100 binary. From the following diagram, 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 may read and execute the file. It grants read permission to all other system users. The *ls(1)* command represents this mode as `rwxr-xr--`.



Protecting Directories

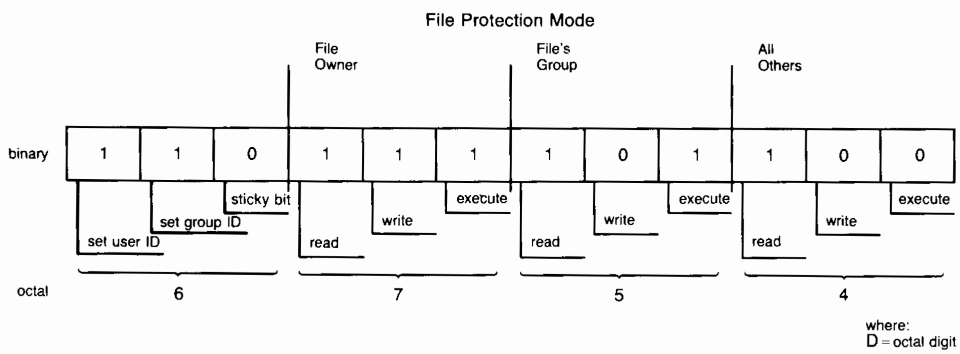
Directories, like all files in the HP-UX file system, have a mode. The directory's mode is identical to the mode of an ordinary file. The read, write, and execute permissions have a slightly different meaning when applied to a directory.

- 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 modify the contents of the directory itself. This capability is given 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 path name for the file.

Setting Effective User and Group IDs

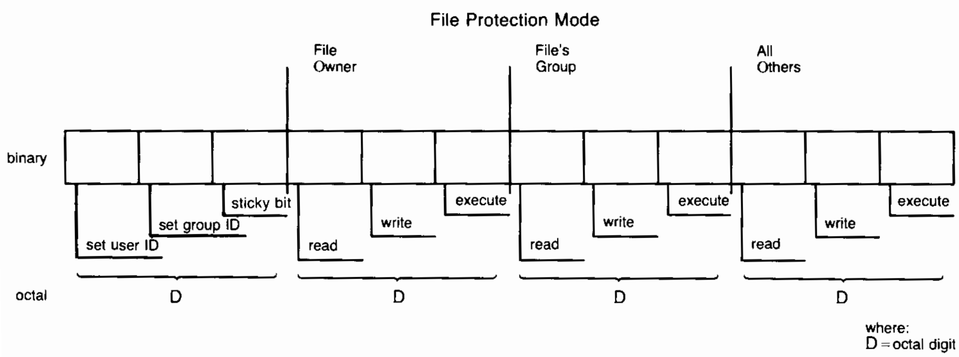
In the section discussing effective user and group IDs, you found that a file can be protected such that when executed, the process's effective IDs are set equal to the file owner's IDs. This capability is specifed through the most significant digit of the four octal file protection mode digits (previously discussed). This digit is represented by a three-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** (suid). Similarly. if the middle bit of the most significant octal digit is 1, then the effective group ID of the process executing the ffile is set equal to the group ID of the file's group. This bit is called the **set group ID bit** (sgid).

If the sgid bit is on for an ordinary file, and the file does not have group execute permission, then the file is in enforcement locking mode. See the section which follows on file locking, or *lock(2)* in the *HP-UX Reference.*



For example, suppose that the file mode is 6754. Octal 6754 is equivalent to 110 111 101 100 binary. The meaning of the mode is shown in the following table and diagram:

Octal Digit	Binary Form	Permission	Meaning
6	1	set user ID	Effective user ID of the process executing this file is set equal to the real user ID of the file's owner.
	1	set group ID	Effective group ID of the process executing this file is set equal to the group ID of the file's group.
	0	sticky bit	The sticky bit is discussed in the section that follows.
7	1	read	File owner may read the file.
	1	write	File owner may write to the file.
	1	execute	File owner may execute the file.
5	1	read	Members of the file's group (other than the file's owner) may read the file.
	0	write	Members of the file's group (other than the file's owner) cannot write to the file.
	1	execute	Members of the file's group (other than the file's owner) may execute the file.
4	1	read	Any other user may read the contents of the file.
	0	write	Other users cannot write to the file.
	0	execute	Other users cannot execute the program contained in the file.



The Sticky Bit

Although the sticky bit can be set for all programs, **setting the sticky bit affects a program only if it is shared** (refer to Memory Management Concepts discussed later in this chapter). The following discussion assumes that all files marked sticky are also shared.

The least significant bit of the upper octal digit is called the **sticky bit**. 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 system partition (discussed with the memory management concepts later in this chapter), 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 system partition. 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 disc only once). However, since the contents of the file remains in memory, the amount of memory available for other users decreases.

Once a program is in the system partition (via the sticky bit), it can only be removed by changing the file's mode such that the sticky bit is no longer set. When the file is executed again, the system recognizes the new mode and deletes the memory-resident copy of the file.

Only the super-user can set the sticky bit.

File Sharing and Locking

In a multi-user, multi-tasking environment such as HP-UX it is often desirable to control interaction with files. Many applications share disc files, and the status of information contained in them could have serious implications to the user (such as lost or inaccurate information).

For example, imagine we are responsible for maintaining on-line technical reports for a myriad of projects, and we have many different people who must have simultaneous access to these reports. The content of a given report at a given time could significantly affect a company decision, and so we want a way to control how records are accessed.

One potential problem could arise if one person (let's call him George) adds to or modifies information in a report while someone else (Sarah) is working on it. Sarah is unaware of changes that George has just made in the report. And once she is done, Sarah overwrites the information George added. The result is that we have lost **all** of George's information, and when Sarah added data she was unaware of information which could have been pertinent.

Advisory Locks

A solution to this common problem of file sharing is called **file locking**. On your Series 500, file locking is done with the *lockf(2)* system call, and it handles two modes of file integrity. **Advisory locks** are placed on disc resources to inform (warn) other processes desiring to access these same resources that they are currently being accessed or potentially being modified. Advisory locks are only valuable for cooperating processes which are both aware of and use file locking.

In our example, the programs used to access the on-line technical reports could use advisory locks. When George begins to work on the FubNibWitz project his program could call *lockf* 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*.

Locking Activities

As stated earlier, all file locking is controlled with the *lockf* system call. There are essentially four activities which *lockf* controls:

- Testing file accessibility by checking to see if another process is present on a specific file record.
- Attempting to lock a file. If the record is already locked by another process, *lockf* will put the requesting process into a sleep state until the record is free again.
- Testing file accessibility and locking the record if it is free, and returning immediately if it is not.
- Unlocking 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)* section of the *HP-UX Reference* manual.

Enforcement Mode

Even if we use advisory locks in our example, Sarah would still be able to overwrite the FubNibWitz. She needs some way to insure that no records are written until George is done with the report. HP-UX does this with **enforcement mode**. When a process attempts to read or write to a locked record in a file opened in enforcement mode, the process will sleep until the record is unlocked.

Enforcement mode is enabled by turning the set-group-id bit on (ORing the file mode with octal 2000) and clearing the group-execute bit (ANDing the file mode with octal 7767). For example if we opened a file which normally has its file access mode set to 755, an *ll* of the file would look something like:

```
-rwxr-xr-x   1 George   LubHood        0 May  7 16:11 FubNibWitz
```

To turn enforcement mode on, we would turn the set-group-id bit on (resulting in 2755) and then clear the group-execute bit (resulting in 2745). This could be done from the shell with the *chmod* command, or from a program with the *chmod(2)* system call. A long listing would show:

```
-rwxr-sr-x   1 George   LubHood        0 May  7 16:11 FubNibWitz
```

By now using enforcement mode, George could prevent Sarah from overwriting his changes, and Sarah would have the data which George went to all the trouble of adding to begin with.

WARNING

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 could hang one or more processes which attempt to access the locked record.

When attempting to access a command which is locked under enforcement mode, the shell will go into a sleep state until the command is released. This provides a means for one script to control execution of another, separate script. Care should be exercised when doing this, because as just noted a system deadlock is possible.

Magnetic Tape

Since computers are sometimes used to process massive amounts of data, there must be a way to store large files on-line. Applications such as atmospheric studies, which minute by minute record megabytes of information and then sort it out, require cheap media to store data on. Even with the advent of larger and larger capacity hard disc drives, they are still too small and far too expensive for such purposes.

Perhaps the closest to an industry standard for mass media, 9-track (½ inch) magnetic tape serves as a low cost, high volume media to store information. And beyond this, magtape is also the most interchangeable media between different hardware and operating systems.

Hewlett-Packard also manufactures a series of ¼-inch data cartridge tapes which are used for the installation and updates of HP-UX on the Series 500. These data cartridges, model HP 88140, have most of the benefits of 9-track magtape but are cheaper and easier to handle. See chapter 2, *Installing HP-UX*, for a list of CS/80 devices which have a built in data cartridge tape drive.

With the 5.0 release of HP-UX on the Series 500, there are new drivers intended to optimize I/O throughput to the HP 7970, HP 7974 and HP 7978 magtape drives. This discussion is provided here to help you with a task you may often encounter if you own an HP 797x series magnetic tape drive — reading tapes written by other systems.

Magtape Definitions

Here are some common terms and concepts used in the discussion of magnetic tape. Consider them required reading if you use magnetic tape.

coding

Tape is recorded in several different ways. Older systems use **Non Return to Zero Immediate** (NRZI) coding, and record with a tape density of either 200, 556 or 800 bpi. Newer tapes use **Phase Encoding** (PE) and record at 1600 bpi, or they use **Group Coded Recording** (GCR) and record at 6250 bpi. There may be other forms of coding as well, but these are the most common. The HP 7970, HP 7974 and HP 7978 magtape drives all support a density of 1600 bpi; the HP 7978 also supports 6250 bpi. For more specifications see the *HP 9000 Series 500 Configuration Information and Order Guide.*

Obviously the higher the density, the more information can be stored on a tape. An HP 7974 storing information on a 2400 foot tape at 800 bpi can only store up to 22 Mbytes, while an HP 7978 storing at 6250 bpi can write up to 140 Mbytes of data to a tape at a rate of 16 Mbytes per minute.

bpi

The most common measure of tape density, bpi is an abbreviation for bits per inch.

cyclic redundancy check

When writing a tape, a number of frames are written by the drive in a single transaction. This collection of frames is called a **record**. Part of the record, but invisible to the user, is a **cyclic redundancy check** (CRC) recorded as some additional frames on the tape. There is a very short blank section between the true record and the CRC. Following the CRC is a nominal ½-inch gap of unrecorded tape, known as the **inter-record gap** or IRG. The next record follows the gap. When the tape drive reads the tape if either the frame parity or the CRC is incorrect, an error is generated by the drive. Newer formats (1600 bpi and above) generate a preamble and postamble to help synchronise the read logic.

end of tape

At the end of a tape is a double file mark which is known as the **end of tape** (EOT). Reading or writing past the EOT causes a non-fatal error to be reported. The transaction is completed and the data is either read or written. Depending upon the policies in the driver, it is often permitted to write some trailer information past the EOT point, but no new real data. The magtape drive hardware enforces nothing.

Note that the distance between the EOT detector and the read/write head may vary between different model tape drives. So, one drive may return an EOT warning associated with the 1000th record on the tape, while another drive may return an EOT warning with the 999th or the 1001st record. For small records this variation is large; for large records this variation is small. This is why the EOT indication is a warning instead of a hard error.

file marks

A **file mark** is a "controlled" error, and is recognized by the drive and reported as a boolean condition during reading. It is not possible to write a tape mark as ordinary data; it requires a special order to the drive.

load point

About 10 feet from the beginning of a tape, a short piece of silver tape is stuck to the media. This is called the **load point**. It is placed on one edge of the tape on the non-recorded side. A similar piece of tape is placed about 25 feet from the end of the reel and is known as the **physical end of tape** (EOT). Both marks are placed by the manufacturer. These are generically called the "reflector spots."

When you load a tape the drive searches forward until the load point is found and placed under the sensor. The first write is then treated specially: several feet of tape are skipped and then a special burst of data is written to the tape (which is invisible to the user). This is the **identify burst**. Data is recorded after the identify burst in the usual way. The first read expects the identify burst, and quietly skips over it. Some smart drives, such as the HP 7978, can determine the tape density from the identify burst (1600 and up).

magnetic tape (magtape)

Magnetic tape is a media similar to an everyday home cassette tape, used to store digital information. All standard magtape is ½-inch wide, and comes in three sizes: 600, 1200 and 2400 foot reels (for a rule of thumb, a 2400 foot reel is about 1 foot in diameter). The size of the the reels, hubs, tape width and other mechanical properties are all specified by ANSI standard.

operations

Several operations that a tape drive can be expected to perform are to read and write to the media, rewind to the load point, forward or back space one record, and forward or back space to the next file mark. A variation on the theme of rewind is to **unload** where the tape is rewound and taken off line. Some tape drives actually rewind the tape out of the threading path; others simply set an interlock that requires manual intervention to release.

records

A series of frames written to the media is known as a **record**. The physical record size is variable, and ranges up to a practical limit of about 32 to 65 Kbytes. Beyond this limit, errors become a severe problem, especially with write/read re-tries. The maximum record size for the HP 7970 and HP 7971 is 32 Kbytes. For the HP 7974 and HP 7978 the maximum record size is 16Kbytes.

tape density

The measure of the amount of information which can be stored in a given area of tape is known as **tape density**. **Bits per inch** (bpi), a common measure of tape density, is the number of bits per track, recorded per inch on the tape. For 9-track tape 8 data bits and one parity bit are written across the width of the tape simultaneously. Thus for 9-track tape, bpi is synonymous with **characters per inch** (cpi). One of these characters is sometimes called a **frame**.

tracks

When digital information is written to a tape, it is written in a series of tracks (a lot like an 8-track car stereo). Most magtape today is written in a 9-track format. Older systems often wrote only 6 tracks plus a parity bit, resulting in 7 tracks. This 7-track format is now nearly obsolete, but occasionally you'll get one. It is possible to record data as if a 7-track tape were 9-track by writing 3 bytes into 4 frames ($3{\times}8 \approx 6{\times}4$). Some hardware supports this. Low density tape (800 bpi and below) can be written in either even or odd parity format, but odd parity is strongly preferred.

write/read re-tries

Tape is not treated as gently as discs are, and is somewhat more prone to error. One of these errors is known as a **write/read re-try**. When your tape drive is reading from a tape and it detects an error (it could not read for some reason), the normal procedure is to backspace the tape over the error and re-read it a maximum of 10 times. Some software also will give a sequence of orders which has the effect of wiping the tape head off with a piece of tape, and then re-reading. An error message is reported to the user only after the driver gives up. Many more tape errors are caused by dirty tape heads than by real recording errors, so you should periodically clean your tape drive as outlined in its service manual.

Tape drives do some form of read-after-write, and if the data is not properly recorded an error will be detected. The normal procedure is to backspace and re-try writing the record once, and if that fails, to backspace, write a **long gap** and try again on a new piece of tape. A **long gap** is several inches of erased tape. That's why we said an IRG is "nominally" ½ inch long.

write ring

On the back of the reel there is a removable soft plastic "teething ring" (and it is useful as such if you have kids) called the **write ring**. Every magtape drive has a sensor mechanism to detect the presence of this ring. When a ring is present the tape can be written to by the host, and cannot be written when absent (it is **write protected**). Most sensors detect the ring once and then retract, but sometimes there is raised lettering in the channel where the ring goes and it can knock the sensor enough to fool it into thinking there is a ring. Watch out if you hear a tape drive going "swish-swish" once per turn of the reel; it may be on its way to giving you a free write ring. Normally once a tape is written the ring is removed and left out indefinitely except when being rewritten.

Preventive Maintenance

There are several maintence procedures for tape. A tape can be completely erased (degaussed), or the beginning of the tape can be discarded and a new load point put on (stripped). There is also a tape cleaning and certifying machine that will knock off any loose oxide and check that the tape will record properly over its full length (certified). This usually makes any data on the tape unusable. Commercial shops certify their tapes fairly often, and discard them if they get too short or fail to certify. It is also an excellent idea to clean the tape head and guides of your drive periodically as they tend to accumulate loose oxide and other crud.

Tape Streaming

The HP 7970 transfers data to and from your Series 500 directly through the drive's read/write head. This means that your Series 500 must be prepared to transfer data at the exact moment that the drive is ready. If your Series 500 is not ready, then the drive must **reposition** the tape, which involves stopping it, rewinding it, and bringing it back up to speed again. The HP 7970 can reposition the tape fairly quickly.

Unlike the HP 7970, the HP 7974 and HP 7978 are **streaming** magnetic tape drives. Streaming increases the rate at which a tape drive can write data onto tape. Before a tape drive can write data onto a tape, the drive read/write head must be positioned at the proper place on the tape, and the tape must move across the head at the proper speed. After writing a record on the tape, if a streaming drive has already received the data for the next record from the computer, it can continue to move the tape across the head without slowing down to write the next record.

If the drive has not received the data for the next record after writing a record on the tape, then the drive must reposition the tape. This involves stopping the forward motion of the tape, rewinding the tape to some point preceeding the beginning of the next record to be written, stopping the tape, and waiting for your Series 500 to send the data for the next record. **The average data transfer rate is much higher when the drive streams than when it repositions**, especially for the HP 7978. The HP 7974 supports a **start-stop** mode which repositions the tape very quickly. The HP 7978 does not support start-stop mode. Both drives are much faster than the HP 7970 when they stream. When they do not receive data fast enough to stream, the HP 7974's performance is similar to the HP 7970; the HP 7978 is much slower.

Immediate Report

To help your Series 500 to send data fast enough to permit the drive to stream, the HP 7974 and HP 7978 support **immediate report**. Ordinarily the actions of your computer and the drive are serialized. Your Series 500 sends data to the drive. Then the drive writes the data to the tape. After the data is written, the drive returns status information to the host indicating whether the write succeeded or failed. When immediate report is enabled the drive returns status before it writes the data to the tape.

This is accomplished because the drive buffers the data it receives from your Series 500 in high speed memory which is built into the drive. The transfer rate between the host and this buffer memory is much faster than the transfer rate would be if the drive transferred the data directly to the tape. Because the drive returns status to your Series 500 very quickly the host's and the drive's activities overlap, so the average transfer rate to the drive has a much better chance of being fast enough to permit the drive to stream.

Version 4.02 Drivers

When immediate report is enabled, the drive may not inform your Series 500 that the end of tape mark has been detected until after several records have been written across and beyond the mark. The Series 500's HP 7970 driver handles the end of tape mark in a way which requires that no records be written across or beyond the mark. For the 4.02 revision of the Series 500 HP-UX kernel the HP 7974 and HP 7978 driver is fully compatible with the HP 7970 driver. The 4.02 HP 7974 and HP 7978 driver does not support immediate report. Consequently the HP 7974 and HP 7978 drives don't stream.

Version 5.0 Drivers

For the 5.0 revision of the Series 500 HP-UX kernel, the HP 7974 and HP 7978 driver supports immediate report. The default is to enable it. For single-reel magtape archives, the only consequence of this change is that the drive streams much more when it writes, and so it writes much faster. For multiple-reel magtape archives, the consequence of this change is that the 4.02 HP 7970 driver cannot cross from the end of the first tape to the beginning of the next (in particular, *cpio(1)* can't handle the new end of tape semantic). The 5.0 HP 7970 driver handles the new semantic correctly. Even with immediate report enabled the HP 7974 and HP 7978 tape drives typically don't stream continuously because the programs running on the your Series 500 don't collect their data from the disc fast enough to supply it to the tape drive fast enough.

Backward Compatibility

The 5.0 version HP 7970, HP 7974 and HP 7978 drivers support a non-default 4.02 compatibility mode which the user may select by setting the eighth least significant bit in the device file minor unit number (i.e. 0x000080). In this mode these drivers can read and write tapes with 4.02 end-of-tape semantics. In this mode immediate report is disabled, so data transfer rates for the HP 7974 and HP 7978 drop to their 4.02 values.

Memory Management

Your computer is equipped with a large but finite amount of Random Access Memory (RAM). It is important that you understand the manner in which this valuable system resource is used since you, in part, control its allocation through system configuration.

Memory Allocation

Before you consider the manner in which memory is allocated, you must understand some fundamental memory concepts. The basic concepts of memory allocation are determined by the computer's hardware architecture, not by the operating system. Thus, some terms and concepts that explain the computer's memory management are presented first. Then HP-UX's methods of memory allocation and management are presented.

Your Series 500 computer is a 32-bit computer. With 32 bits, an absolute pointer can address 4 294 967 296 different bytes of memory. Three of the 32 bits are used by the system for purposes other than addressing. This leaves the system with 29 bits with which to address absolute physical memory. Thus, the system can address 536 870 912 bytes or 2^{29} bytes (the range of addresses is called the **absolute address space**).

This limit of 2^{29} bytes on the absolute physical address space is inconsequential since it is not possible to physically fit 500 MBytes of real memory in the machine! Each user, however, can address 31 bits of logical address space through pointers translated by segment tables. Each user in a multi-user system has her own private 30 bit virtual address space plus simultaneous access to the system's 30 bit virtual address space shared by all users on the system.

Mapping RAM to Absolute Address Space

Although the computer can address the entire absolute address space, it is equipped with a much smaller amount of Random Access Memory (RAM) hardware. This amount depends on the number and type of RAM boards that are installed. There are three types of RAM boards available for the Series 500:

- 256 Kbyte RAM (HP97040A)
- 512 Kbyte RAM (HP97047A)
- 1 Mbyte RAM (HP97046A)

Each of these boards contains a **memory controller** that provides the means to **map** (assign) the physical memory into the absolute address space. The memory is mapped contiguously from absolute memory address zero. In nearly all instances, the 1 Mbyte RAM is mapped first, starting at absolute memory address zero, and then the 256 Kbyte RAM and 512 Kbyte RAM are mapped.

The 1 Mbyte RAM is somewhat slower than the 256 Kbyte RAM and the 512 Kbyte RAM; therefore, it is **interleaved** to improve its performance. The interleave can be one-way, two-way, four-way, or eight-way; the higher the interleave, the better the performance.

The interleave of the memory supplied by 1 Mbyte RAM boards is determined by grouping the memory into one, two, four, and eight Mbytes, where each group has the associated interleave. For example:

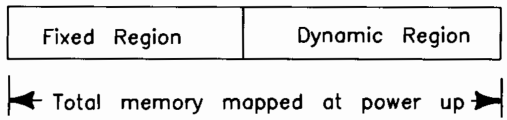
- If you have four 1 Mbyte RAM boards, the four Mbytes of memory have a four-way interleave.
- If you have three 1 Mbyte RAM boards, two Mbytes of memory have a two- way interleave, while the third has a one-way interleave.
- If you ony have one 1 Mbyte board, the one Mbyte of memory has a one-way interleave.

In some configurations, adding an extra 1 Mbyte RAM to a system can result in a performance degradation. It is important that the memory have the highest interleave possible in order to maximize performance. A Series 500 with four 1 Mbyte RAM boards has four-way interleaved memory throughout the absolute address space. If another 1 Mbyte RAM is added, four Mbytes of memory have a four-way interleave and one Mbyte has a one-way interleave. That one Mbyte of memory having a one-way interleave may cause a net loss in system performance.

Note that since the memory supplied by 256 Kbyte RAM and 512 Kbyte RAM boards is not interleaved, it has a set performance level that is not changed by how the boards are combined in a system.

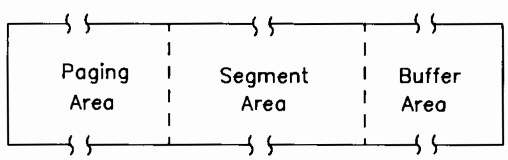
Dividing the Absolute Address Space

The absolute address space is divided into two regions: the **fixed region** and the **dynamic region**.



The fixed region contains space for the hardware-defined locations, the system segment table, system data structures, and system code. The size of this region is determined at power-up and it is accessible only to the operating system.

The dynamic region is split into three areas: the **paging area**, the **segment area**, and the **buffer area**.



The paging area is used for paged segments (see "Paged External Data Segments" later in this chapter) and is managed by the virtual memory system. The segment area contains all other segments and is the principle source for free space. The buffer area contains buffers used by the system for objects that cannot move. The boundaries between these three areas are dynamic and change as the system load shifts between activities associated with each area.

Partitions

The system provides 512 **partitions**, one of which is reserved for the system. A partition is a logical section of memory composed of:

- a **process**
- an address space consisting of a collection of memory segments managed by a **segment table**.

Each partition has one segment table that points to segments in the segment area that are used by the associated process. A user process accesses memory segments through its segment table. The segment table is managed by HP-UX; it is not directly accessible to the user process. Although one user process cannot directly access another user process' memory, it is possible for one process to force the suspension of another process, in which case the system reassigns the partition's memory.

The partition reserved by the system is called the **system partition**. It is used for code and data segments which may be shared between different user processes. This is possible since the system segment table is accessible to all other partitions.

Each user process uses:

- one or more **code segments** (CS) - segments containing the code that the process is executing.
- a **stack segment** (SS) - a segment that contains procedural parameters, the return addresses for procedures calls, and local variables for procedures.

  The stack segment is used for variables which are allocated when a procedure is called and are de-allocated when the procedure returns. The size of this segment grows as needed by the executing program. The stack segment only shrinks during an exec, though the logical stack size grows and shrinks dynamically. The stack is limited to a maximum size which is determined by the system configuration.
- a **global data segment** (GDS) - a segment that holds the program's global data (for example, statically allocated data).

  The global data statement is used for variables which are allocated when a program is loaded. These variables remain in memory for the duration of the program and are de-allocated when the program terminates. A GDS can be removed to disc in response to memory demand.

- one or more **external data segments** (EDS)

  External data segments are general purpose segments. They are used mainly for data. Because of attributes discussed later in this chapter, they can allow access to extremely large quantities of data.

Stack and global data are "fast access" segments. The system has mechanisms which allow the data they contain to be read and written faster than external data segments. Any virtual segment is subject to removal in response to memory demands of higher priority processes.

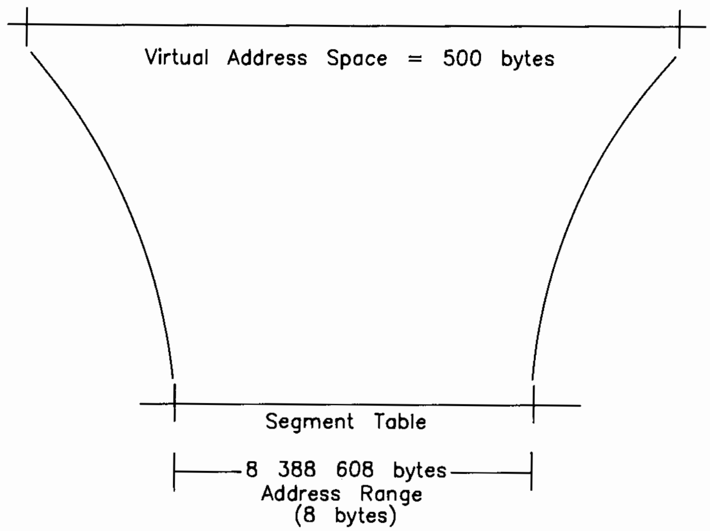
Virtual Memory

When a program executes, it typically performs one function (such as totalling an employee's hours) and passes its output on to another function within the same program (such as a wage calculating program or a paycheck program). Although the entire program is loaded into memory, only one such function is used at any one time. The remaining portions of the program may sit idly by, possibly occupying memory.

As you learned with the memory allocation concept, memory is mapped to a partition to hold the process' code and data segments. Without virtual memory, if there is not enough memory available for a process' segments, the process cannot execute; it must wait until sufficient memory is free to be mapped to its partition. When large programs or programs that access large amounts of non-virtual data are run, they must wait until few processes are running so that they may acquire the memory needed for their partition. Then, once running, other processes must wait for the large process to terminate before they can run (even though the large program may only be using a few segments and a small amount of memory at any given time). This also implies that no process can exceed the total memory size of the computer (since there is no way for sufficient memory to be mapped to the process' partition).

To avoid this restriction, HP-UX has implemented a memory management attribute, called **virtual memory**. Segments composing an executable file are marked virtual or non-virtual with the */usr/bin/chatr* command. The segments of the file are marked as virtual segments by default when the compiled objects are linked with the *ld* command. When you execute virtual code or access virtual data, the system places those segments that are currently needed into the available memory within the process' partition. If there are more segments in the object than memory available, the additional virtual segments are copied to files on the virtual memory device. You specify which device is the virtual memory device when you configure the system.

When the executing code attempts to access a segment that is not present in memory, the system attempts to map additional memory to the process' partition. If the additional memory is not available, the system attempts to free memory that contains virtual segments no longer needed. These segments are "swapped out" (copied out) to files on the virtual memory device and the needed segments are "swapped in" to the newly freed memory. Because of the way that virtual memory is implemented, the "virtual address space" is much larger than the absolute address space of the process' partition.



In determining which segments to swap out, HP-UX uses the following criteria. No segments are swapped out until additional memory is needed. When memory is needed (to swap in a virtual segment being accessed), the system looks for any virtual segment that has been resident longer than a set length of time without being accessed. Segments meeting that criteria are "swapped out" until enough memory is free to load the needed segment. The set length of time referred to previously is specified with the system configuration. If no segment has been in memory without being accessed for the set length of time, the system attempts to recover memory through a variety of tactics, including suspending lower priority processes and using their memory and attempting to shrink the page frame pool. If the request still cannot be granted, the requesting process can be suspended until the system determines that enough memory is available to resume that process.

Since only the segment currently needed is loaded at any one time, your programs and data may actually be larger than the physical memory within the computer. The maximum size of a virtual program is about 500 Mbytes. Similarly, the maximum size of virtual programs that utilize the system partition (such as shared code segments) is about 500 Mbytes.

Virtual Memory: Benefits vs. Cost

Making segments virtual dramatically reduces the amount of memory required by each user partition. However, the costs of virtual memory are extensive. Virtual code segments often occupy twice their original space on disc: one non-virtual copy (the original copy) and one copy of the file in the virtual address space. Additionally, each time the computer swaps a segment in or out, time is required to copy the program to/from disc. Thus, the program runs more slowly. As more users utilize memory, the computer must do more and more swapping.

Shared Code

Often, several processes want to run the same program simultaneously (such as a text editor program). If the program is not shared, then each process running the program has a copy of the program's code and data. If the processes share one copy of the code, the amount of memory required for each process' partition dramatically decreases.

The term **shared code** describes user code which is loaded into the system partition. When a process executes shared code, it is directed to the copy of the code in the system partition. If the shared code is not yet loaded (no other process is currently accessing the code), the code is first loaded into the system partition before the process begins execution. Only one copy of the code exists in memory regardless of the number of processing running the program. When all processes accessing the shared code terminate, the memory in the system partition in which the shared code currently resides becomes available to satisfy requests for memory (unless the sticky bit was set on the file from which the code was loaded). The memory is not actually freed until a process requires it.

The system knows how many users 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 system partition, the use count for the program is set to one. When the process finishes executng 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 system partition 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 the system partition, no additional memory is allocated. The new process simply executes the copy of *ed*'s code that resides in the system partition; its use count is incremented from one to two. The first process now finishes editing and 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. Finally, the second process finishes editing and terminates the *ed* program. The system decrements the use count of *ed*'s shared code segment and, finding its value to be zero, makes *ed*'s shared code segments available for use by other programs.

See *ld(1)* and *chatr(1)* for information about making programs shareable or non-shareable.

Shared Code and the Sticky Bit

Setting the sticky bit on a file containing shared code increments the shared code's use count by two instead of one when the shared code is loaded into the system partition. This keeps the use count from ever returning to zero, thus keeping the shared code segment in memory whether it is being accessed or not.

For example, suppose that the text processor program *ed* is marked "shared". Also suppose that the sticky bit for the file in which *ed* is stored is set. When a process first executes *ed*, the shared code is loaded into the system partition and its use count is set to two. Suppose that while the first process is executing, another process executes *ed*. The new process executes the copy of *ed*'s code that resides in the system partition; the use count of the shared code is incremented from two to three. The first process now finishes editing and terminates its execution of the *ed* program. The use count of the shared code decrements from three to two as the second process continues to edit. When the second process finishes editing, it terminates execution of the *ed* program. The use count of the shared segment decrements from two to one. Note that even though no process is accessing the shared code, its use count does not decrement to zero. The shared code remains in memory; the memory is not freed for other uses.

To return the use count of a "shared", "sticky" program to zero, the super-user must set the mode bits of the file containing the program such that the sticky bit is no longer set. When the program is again executed, the system recognizes that the sticky bit is no longer set and decrements the use count.

Shared Code: Benefits vs. Cost

Shared code significantly reduces the amount of memory required for process partitions when multiple processes are executing the same program. There is no cost associated with sharing code unless the file containing the shared code has its sticky bit set, or the system segment table overflows due to an excessive number of shared objects. In this case, the code occupies memory in the system partition whether the code is being used or not. Consider making such code virtual as well as shared and "sticky". Since the number of shared code segments in the system is limited by the number of entries in the shared system segment table, overuse of sharing can result in system segment table overflow errors.

Demand Load

Programs often contain routines and code which are rarely accessed. For example, error handling routines can comprise 80% or more of some program code and yet may be rarely accessed. As mentioned previously, when a program is loaded, it is copied in its entirety into main memory. If the unused code segments are a significant portion of the program, the memory allocated for that code is wasted.

With HP-UX, it is possible to mark code segments, data segments, and data pages (discussed later) as **demand loadable**. When a demand loadable segment is requested to be loaded, no segments are actually loaded. Instead, a bit is set specifying that the segment is demand-loadable; the system is "fooled" into believing that the segment is actually present in memory even though it is not. No memory is allocated for the non-loaded segments until those segments are actually accessed. Only when the program attempts to access a demand-loadable segment is memory actually allocated and the segment loaded from disc.

To make code demand-loadable, mark the segments that you want demand-loadable when the compiled code is linked (via the *-z* option of the *ld* command). Alternately, you can use the */usr/bin/chatr* command to mark an existing executable file as demand-loadable. You can change a demand-loadable file such that it is no longer demand-loadable by using the *chatr* command.

Demand Load: Benefits vs. Cost

Making code segments demand-loadable reduces the amount of memory allocated for a user's partition. However, the program may actually take longer to execute than a program that is not demand-loadable if many of the demand-loadable segments are accessed.

Note that memory blocks are not allocated until the segment is actually accessed. Thus, if there is insufficient memory for the segment, an error will occur while the program is running; the program fails. If the demand-loadable attribute is not used, the lack of memory is detected when the program is loaded (because memory is allocated when a program is loaded).

Paged External Data Segments

A limitation of virtual memory as discussed so far is that all of an external data segment used by a program must be resident in memory when that segment is accessed. Suppose that you have written a program to analyze 30 Mbytes of data. (For example, finite element analysis programs and circuit analysis progams both use extremely large quantities of data.) That means that when your program runs, all 30 Mbytes of data need to be present in a process' partition, if the data was allocated in one unpaged EDS. However, your system does not have enough memory to hold the data. Even if the system had 30 Mbytes of memory to map into the partition, your program would be using so much memory that other users would not be able to do anything. Typically, you only need to access a small portion of that data at any one time. The remainder of the data simply sits in memory, occupying system resources. To solve this problem, HP-UX allows virtual exernal data segments to be **paged**.

A **paged segment** is a segment that is divided into parts of a fixed size called **pages** (1024 bytes is the default). The segment table, instead of storing the size and location of the entire segment (as is the case with un-paged virtual segments), stores a pointer to a page table. A **page table** is an ordered table indicating the address of each page in memory for the segment. If the page is currently on the disc, then the page table stores its disc location. The virtual memory system then moves virtual pages in and out of memory as needed, just as the virtual memory system moved entire segments in and out of memory in the previous discussion.

Not all pages of a segment have to be in memory at any given time. Thus, only a portion of a segment must be in memory at any given time, not the entire segment. This allows the segment to be larger than the size of memory, and still be accessed by the process; only the page being currently accessed must be resident in memory. Additionally, since each page is uniform in size, the virtual memory system can easily decide where to place the pages in memory. If one page is removed, then any other page can fit into its place.

The memory used for the pages of all the paged segments is allocated from a kind of memory "parking lot", called the **page frame pool**. When an executing program needs access to a paged external data segment, the system allocates the appropriate number of pages in the page frame pool and loads only that data into the pages. The memory pages are returned to a "free list" in the system's page frame pool when the process terminates.

When a new page is needed (a non-memory resident page is accessed) and there is not enough memory in the page frame pool, a page is swapped out to the virtual memory device. When a page is needed and is not resident, it is swapped in from the virtual memory device.

All paged segments in the system compete for the use of pages in the page frame pool. If no paged segments are active, then memory allocated to the page frame pool can be used for segments (for example, mapped to a process' partition) until the memory is needed by a paged segment. The maximum size of the page frame pool (and the size of each page) is determined by the system configuration. Page sizes are powers of two in the range 512 to 8 192. Programs that access pages in sequential patterns usually benefit from larger page sizes, while those that access pages randomly benefit from smaller page sizes.

Pages that are currently in the page frame pool and are entered into the page table for a particular segment are known as **working set pages**. The working set pages are linked together in a per segment list by the virtual memory system. The system provides a method to guarantee that a certain number of pages (called the **guaranteed working set**) are always available for a specific segment.

All pages that are not in the working sets of allocated paged segments fall into three categories:

- **dirty pages** — pages that were removed from a working set and must be written back to the disc before they can be re-assigned to another working set.
- **used pages** — pages that were removed from a working set and were not modified since being read into memory (and thus do not have to be written back to the disc).
- **free pages** — pages that are available for use and have no valid data in them. All pages start out as free pages. If a segment is de-allocated, then all the pages associated with it are placed in the free list.

All pages removed from some segment's working set are placed in either the list of used pages or the list of dirty pages, depending on whether they were modified. The dirty and used lists are maintained for the instance when a page is removed from a working set, is not written back to disc yet, and then is requested again. The dirty or used page can be restored to a working set much faster than if it had to be read back from the disc.

External data segments are automatically specified to be paged when the data objects are lined via the *ld* command. Additionally, you may specify that data is to be paged with the Extended Memory System (via the *ems* system call). You can use the */usr/bin/chatr* command to specify that an existing paged external data segment is not to be paged.

Page Clustering

The system recognizes three types of paged access:

- sequential
- random
- normal

Sequential accessing occurs when a paged segment is accessed linearly in either direction. Random accessing occurs when there is no pattern or locality associated with the data accesses. Normal accessing has a high degree of locality with occasional occurrences of sequential and random accessing. The performance of sequential accessing can be improved by taking advantage of page clustering.

You can force page clustering by specifying the referencing pattern of an area of process memory as sequential with *memadvise(2)*. Once page clustering is enabled for a paged object, one or more seqential pages will be read into memory whenever a page fault is detected. This reduces the number of page faults that occurs during sequential accessing. You also specify the size of the page cluster with *memadvise*. See the entry for *memadvise(2)* in the *HP-UX Reference* for more information.

With normal accessing (the system default), the system automatically enables page clustering when it detects a series of sequential page accesses. No clustering is ever attempted when a paged segment is marked as random.

Paging Benefits vs. Costs

The chief benefit of paging is that it allows a process to use quantities of data that exceed the physical memory limits of the computer. The maximum amount of memory allocated to the page pool is specified by the system configuration. Memory allocated to the page pool cannot be allocated to process partition(s).

A primary cost of paging data is the time required to read the file each time a page is read from disc. An additional cost of paging data is the time required to access the page. Each time a page is accessed, the process must first access the partition segment table, then access the page table (an external data segment) and then access the page (and possibly wait for it to be swapped in, if it is not already memory resident); access to paged external data segments is slower than accessing external data segments resident in the process' partition.

Combining Memory Management Tools

The memory management attributes previously discussed can be used in any combination, with some restrictions. These restrictions are:

- code segments cannot be paged.
- setting a file's sticky bit affects only the code segments in shared programs.
- global data segments cannot be paged or demand loaded.
- stack segments are always virtual and cannot be paged or demand loaded.

Wise system adminstration calls for a mixture of the memory management attributes previously discussed. Most program files should be shared, virtual, paged, demand loadable or some combination of these. Your system is shipped with a proper mixture for the "average system". The system configuration, as shipped, is set up for the same "average system". The average system is based on a Series 500 computer equipped with 1.5 Mbytes of memory with an environment of the following four concurrently active tasks:

- compilation of a FORTRAN program.
- random and sequential read and write access to elements in a 2 Mbyte virtual array.
- execution of a computationally intensive program.
- invocation of interactive commands (such as editing or listing the contents of a file).

By changing the mixture, altering the system configuration and observing the change in system response, you can "tune" the system for the optimum response for your application mix.

The Buffer Cache

Program code and the data which it uses must be transferred from disc into physical memory before it can be executed. The manner in which code is transferred depends on the attributes of the code and the manner in which the code is executed. All code executed via the *exec* system call is transferred into the process' partition via a buffer called the file system buffer cache. Additionally, most data that is read from or written to disc is transferred through the file system buffer cache.

The file system buffer cache is a collection of one or more buffers which the system uses as a temporary holding place for code/data being transferred between the disc and physical memory. The size of each buffer and the number of buffers in the cache is specified when the system is configured. As the code and data are moved into the buffer cache, the system copies the information from the buffer cache into the user's partition.

The buffer cache can be thought of as a long tube (with sections of uniform size) that is shared by all processes. When code and data are loaded into the system, the system reads a portion of the file, equal in size to one buffer and places it in the tube. The system repeats this procedure until the entire program is transferred. When another program is loaded, it is placed into the tube behind the previously loaded program. This continues until there is not enough space in the file system buffer cache to load the next program. At that time, the last recently accessed program is removed from the cache until there is enough room to load the new program.

If a user requests a program that is already in the buffer cache, the program is copied from the cache to the user's partition, eliminating the intermediate step of copying the file from disc to the buffer. The copy of the program in the buffer cache is then moved to the beginning of the cache, indiciating that it is the most recently accessed program (and thus placing it last in the queue to be removed from the cache).

The primary advantage of the file system buffer cache is the speed with which large numbers of sequential reads are made to a data file. This advantage is largely due to an attribute of the file system buffer cache, called the **read ahead level**.

Read Ahead Level

The file system buffer cache has an attribute called the **read ahead level.** It is applicable primarily to users who are reading data from the disc with system calls. The read ahead level is the number of "buffers-full" of contiguous data the system should attempt to copy per disc access. For example, assume the read ahead level is four and a user is attempting to read 256 bytes of data from a file (via a system call). When the file is accessed, you would expect only the requested number of bytes to be copied into the buffer cache and then into the user's partition. However, because the read ahead level is four, the system attempts to copy enough data to fill four cache buffers. Only the requested 256 bytes are copied from the cache into the user's partition. The remaining data stays in the cache in anticipation of the user's next data read request. Since the system has to access the disc (a time consuming process) to copy the requested 256 bytes, it might as well copy some additional data (up to a maximum specified by the read ahead level) the user is likely to need.

Data accessed in this manner must be contiguous and accessed sequentially. For example, assume a user is attempting to access 256 bytes of data from a file consisting of two extents (the first extent contains 2 Kbytes, the second contains 3 Kbytes). Also assume that each cache buffer is 1 Kbyte in size and that the read ahead level is four. When the user reads the first extent to access the requested 256 bytes, the system examines the read ahead level and attempts to copy 4 Kbytes (4 x 1K/buffer). However, after copying the first 2 Kbytes of data into the cache, it encounters the end of the extent and terminates the copy. Thus, only 2 Kbytes of data were copied into the buffer cache.

The File System Buffer Cache: Benefits vs. Cost

The primary benefit of the buffer cache is the increased speed with which large numbers of sequential or repeated data accesses are made. When the first of such transfers is made, the system recognizes that a sequential access has begun and attempts to read enough additional data to fill the read ahead level. Then when additional sequential accesses are made, the data is already present in memory, in the file system buffer cache. Thus less time is required to access the data after sequential access is detected.

Transferring a program from the buffer cache to a process' partition is much faster than transferring a program from disc, through the buffer cache, to a process' partition. Thus by increasing the size of the buffer cache, more program and file data can be held in memory and the apparent system response time decreases. However, memory used by the system cache is unavailable for use in user partitions and thus decreases that precious system resource. When the file system buffer cache exceeds a certain size, system peformance begins to decrease since less memory is available for other system functions.

A major factor in determining the ideal size of the file system buffer cache is the amount of memory in the system. For a system with 2.0 — 2.5 Mbytes of RAM, performance begins to degrade when the cache is larger than 50 - 100 Kbytes. However, for a system with 4.0 - 8.0 Mbytes performance may not begin to degrade until the cache exceeds 200 - 800 Kbytes. The ideal size for the buffer cache also depends on such things as the amount of sequential data access performed and the speed of the disc drive. By default, the system chooses a reasonable size buffer cache based on the available memory in the system. The system administrator can alter the default size if she desires.

As a rule of thumb, consider that as the size of the buffer cache increases, the probability of accessing a segment via the cache increases and the probabilityof accessing a segment via the disc decreases. However, as the size of the cache increases, the amount of physical memory available decreases (thus decreasing the performance of other processes).

Device I/O

The supplied tutorial text discusses re-directing program output to a file and program input from a file. HP-UX treats I/O to a device in the same fashion as I/O to a file. In fact, before your computer can "talk" to a device, a file (called a special file) must be created. This file defines the location of the device and the manner in which the computer and the device must communicate. Device files are created with the *mknod* or *mkdev* commands and are usually stored in the */dev* directory. To communicate with a device, simply redirect input from, or output to, the device file. The computer then uses the information contained in the special file to manage all transfer of data between it and the device.

Device Classes

All I/O devices can be classified as either block or character devices. Block devices are devices which transmit and receive data in blocks (typically 1 block = 1024 bytes, but the size of a block, is specified when initialized). Typically, block devices are disc mass storage devices. However, a disc's built-in cartridge tape drive (available with HP7908, HP7911, HP7912 or HP7914 disc drives) and other magnetic tape drives are occasionally used as block devices. Character devices include any device which is not a block device, including printers, plotters, terminals, magnetic tape drives, and paper tape punches/readers. Disc mass storage devices are occasionally treated as character devices, even though they normally operate as block devices.

Drivers

The *mknod* command creates a device file from a driver number, an address (defined next) and a driver. A driver is compiled code (supplied with your system) which defines the protocol and handshaking that allow an I/O device and the computer to communicate. For a list of the drivers available with HP-UX, use the *lsdev* command.

Address

An address is a set of values which specify the location of an I/O device to the computer. The address is composed of up to four hexadecimal fields; the select code field, the bus address field, the unit field, and the volume field.

An address is specified in a packed field that has the form:

`0xScAdUV`

`0x` is a two character field specifying that the following field is a packed hexadecimal field.

`Sc` is a two digit, hexadecimal value specifying the select code.

`Ad` is a two digit, hexadecimal value specifying the bus address or port number.

`U` is a single digit, hexadecimal value specifying the unit number.

`V` is a single digit, hexadecimal value specifying the volume number.

The select code is determined by the I/O slot in which the device's interface card is installed; refer to your computer's installation manual. (See the section describing the installation of interface cards for more information).

The bus address allows the computer to distinguish between two devices connected via the same HP-IB interface. Refer to the manual supplied with the peripheral to determine if the device has an address, and the method in which that address may be changed.

The unit number is used to specify a specific disc drive in a system with a multi-disc controller.

The volume number specifies a volume on a multi-volume drive.

Many drivers assign different meanings to the second, third, and fourth field of the address. For a precise definition of these fields, refer to the discussion entitled "Adding/Moving Peripheral Devices" in this manual.

System Boot and Login **4**

From the time you switch on power to the computer until you have successfully logged in, many tasks are automatically performed by the system. These tasks include: testing the computer hardware, loading and initializing the operating system, communicating messages to the user(s), and running scheduled routines. To manage your HP-UX system effectively, you must understand which tasks are performed at which times.

This chapter provides you with a step-by-step description of the computer's activities from power-up through successful completion of the *login* routine. Throughout this tour, you will learn about features of HP-UX that can ease your role as system administrator. Later material in this chapter shows you how to modify these features for your specific needs.

The terminology defined at the beginning of Chapter 2 ("Installing HP-UX") is used throughout this chapter; you may want to review those terms before continuing with this material.

System Boot

As system administrator, you must understand how the boot ROM determines which operating system to load. 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.

What the Boot ROM Does

The boot ROM is a general purpose, albeit complex, piece of software residing in Read Only Memory that was specifically developed to support a wide variety of present and future Hewlett-Packard operating systems. Because different operating systems use different aspects of the boot ROM and, because you are primarily concerned with how HP-UX works with the boot ROM, the following description of the boot ROM's operation focuses on its use with HP-UX.

When power is first applied to the computer, the code contained in the boot ROM executes and performs a series of hardware tests. During this phase of the boot ROM's operation, the boot ROM looks for all interface cards (such as the HP 27110B HP-IB Interface), internal peripherals (such as the internal HP-IB of the Model 550 and the internal keyboard and CRT in the Model 520) and performs certain tests on them; refer to the Installation Guide supplied with your computer for a description of these tests and a guide for interpreting their results.

The boot ROM also assigns a console for its output by using a search sequence similar to the one that HP-UX uses to find a system console. The HP-UX and boot ROM search sequence is described under the **system console** entry in Chapter 2.

Your boot ROM then begins polling all mass storage devices connected to the computer to ascertain which operating system to boot, and loads the first "loadable" operating system it finds. When booting HP-UX, it accomplishes this last task by loading, and passing control to the HP-UX kernel which is the core program of the HP-UX operating system.

The following shows a typical display at this stage of the boot ROM's operation with an Internal Terminal Emulator (ITE) as the system console, on a Model 550 using an HP 98700 display system.

```
Loader Rev 4.1
HP-UX Model 550 Release 5.0 (97079C)


Load done.
```

Once the `load done` message appears on an HP 98700 display system or on a Model 520, the screen will clear and the following copyright messages appear. The screen will not clear on any other console configuration (for example, an HP 2392a terminal).

```
(c) Copyright 1983, 1984 Hewlett-Packard Company, All rights reserved.
(c) Copyright 1979 The Regents of the Univer. of Colo., a body corporate.
(c) Copyright 1979, 1980, 1983 The Regents of the Univer. of Cal.
(c) Copyright 1980, 1984 AT&T Technologies.  All Rights Reserved.
```

The display varies depending on the version of the boot ROM, and the label placed on the boot area.

On each CS/80 and SS/80 device connected to your Series 500 is a boot area defined when that device was initialized. For more information on this, see the boot area definition in chapter 2. Operating systems residing in the boot area are marked either as "loadable" or "not loadable" with the */etc/osmark* command. If you have not modified your system since you installed it, you will only have one operating system (HP-UX) residing in one boot area.

NOTE

If you are counting on the order of the operating systems found by the boot ROM's search sequence (described below), make sure the mass storage device containing the operating system has completed its power-up cycle and is ready for use before powering up the computer; otherwise, the order in which the operating systems are found may be different.

NOTE

If the boot ROM can neither locate nor successfully load an operating system, one or more boot ROM error messages are issued to the system console. These errors are explained in the Installation Guide supplied with the computer. If no messages at all are displayed, suspect a hardware problem in your system. Use the procedures described in the installation manual to verify that a hardware problem exists before calling your HP Customer Engineer for service.

The Boot ROM's Search Sequence

When the boot ROM searches for an operating system to load, it scans all CS/80 and SS/80 devices connected to your Series 500 which were successfully found in the test phase. HP-UX must both be marked as an operating system, and as loadable code. The boot ROM was designed to give removable media first priority (to allow for things like operating system updates), and so any media loaded in a ¼-inch data cartridge tape drive, removable disc pack, micro-floppy and so on, will be searched for an operating system. Following this, all fixed media are also searched. Lower device addresses (minor numbers) are searched before higher addresses.

NOTE

The removable disc pack of an HP 7935 drive is treated as removable media by the Rev A boot ROM; Rev B, Rev 4.0, and later version boot ROMs treat this as fixed media.

According to the above priorities, external devices are searched in order of select code; a system at select code 4 would be found before a system at select code 5. Also, multiple units at the same select code and bus address are searched before moving to the next ascending select code or bus address.

To summarize, the search levels in ascending order are:

1. device class (removable media, internal discs, external discs)
2. select code
3. bus address (also called device address)
4. unit number

The Root of the File System

Once the HP-UX Operating System has been found and loaded successfully, HP-UX searches for the root of the file system. The **root of the file system** is the portion of the file system that forms the base of the file system hierarchy (i.e., the portion of the file system on which other volumes are mounted). The file system root contains the files required for HP-UX to properly run.

The root of the file system can exist on any CS/80 or SS/80 mass storage medium supported by Series 500 HP-UX, provided that it has been marked as a root device, and has at least 16 Mbytes of storage (though this minimum size leaves little storage space for significant development work). To mark a device as the root volume, see the *rootmark(1M)* command in the *HP-UX Reference.* If you have made no modifications to your HP-UX system since you installed it, you probably won't have to worry about *rootmark*ing or *osmark*ing a disc. Note however, that HP-UX may only be initially installed on one of the CS/80 or SS/80 devices listed in Chapter 2 of this manual, or better yet, see the *HP 9000 Series 500 Configuration Information and Order Guide.*

The Init Process

HP-UX then sets up its first process, */etc/init*, which becomes process one (1) and has no parent. When *init* is executed at this time, it comes up in the "initdefault" state as defined in */etc/inittab.* See the *init(1M)* entry in the *HP-UX Reference* for more information. As shipped, your system will boot in state s, which is a true single-user state.

The process *init* then reads the configuration file */etc/inittab* as follows:

- entries marked as `bootwait` are executed before the system goes into any state other than the single-user state `s` (they are executed only once per re-boot). As shipped to you, this should include */etc/bcheckrc*, which will do an *fsck* of your file system, and */etc/brc* which will remove the mount table */etc/mnttab* if it exists.
- entries marked as `wait` are executed every time the system goes into those run states specified in the "rstate" field of */etc/inittab.* Included among these should be commands linking */dev/syscon* to */dev/systty*, as well as the execution of the script */etc/rc.*
- entries marked as `powerfail` will be run, if the system encounters any powerfails.
- entries marked as `off` are ignored, and are often used for comments.

Each line in the file */etc/inittab* 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 two character identification code, `rstate` are the run states to which this entry applies, `action` tells *init* what todo with the entry, and `process` is an HP-UX command to execute.

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

```
08:2345:respawn:/etc/getty tty08 H
```

The action `respawn` tells *init* to continuously re-create a *getty* process at the console in states two thru five. There are several other *init* actions possible; for a complete list, see *init(1M)* in the *HP-UX Reference.* Leaving the rstate field null (as shown below) will cause execution in all states:

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

This is highly recommended for the system console.

NOTE

The documentation which follows describes 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 paragraphs which follow may no longer apply. If you make changes to these files: save a copy of the original file, document the changes and keep a copy of that documentation with this manual.

The /etc/rc File

After reading */etc/inittab* and signalling processes as required, a line in */etc/inittab* invokes */etc/rc*. This script uses three arguments returned by doing a `who -r` which are: the current state, the number of times this state has been entered previously and the prior state. At power-up, the values of these arguments are s, 0, and 0 respectively, if you have not modified your system. The following describes the actions that take place whenever */etc/rc* is invoked. This is followed by a description of the contents of *rc*.

You should examine the */etc/rc* shell script. The comments within the script describe its actions and indicate the appropriate places to modify it. For example, the script includes a comment which states

```
Mount volumes:
```

Add *mount(1)* commands at this location to automatically mount different volumes under the HP-UX file system.

Upon starting, the */etc/rc* script sets the environment variables PATH (the default search path that the system uses to find commands) and TZ (for time zone). */etc/rc* next exports the TZ variable (using the *export* command). Exporting TZ causes *rc* (and any child process of *rc*) to override the default time zone (`EST5EDT`); for more information, see the *ctime(3C)* entry in the *HP-UX Reference* manual.

Next, */etc/rc* sets up and initializes the system console by using the *stty* command to set such attributes as the baud rate, communications protocol and tab settings.

The Single-user State

Unlike versions earlier than HP-UX 5.0, your Series 500 HP-UX system has a true single-user state. If the `initdefault` entry in */etc/inittab* is s (as it is shipped), then immediately you will get a Bourne shell at the console, logged in as root. `bootwait` and `wait` entries are not executed. In this state no *getty*'s are issued, nor are any other actions taken by the script */etc/rc*.

The Multi-user State

As shipped to you, your Series 500 HP-UX system supports state 2 as the default multi-user state. To enter state 2, enter `init 2`[Return] at the console. *init* will now take actions as defined in every entry of */etc/inittab* which has "2" listed in it's run state.

/etc/rc in State 2

If your system is in state 2 (as it is after you type `init 2`), and this is *rc*'s first time in state 2, the shell script sets the system node name (used in *uucp* facilities) and mounts any volumes listed in the *mount* command. Next, various devices are mounted and the system mount table, */etc/setmnt*, is created and initialized.

The */etc/rc* shell script next executes the */etc/cron* program. *cron* executes commands at specific dates and times, according to the instructions submitted by the *crontab* command. These instructions are kept in the directory */usr/spool/cron/crontabs*. To run one or more commands on a scheduled basis, execute the *crontab* command while logged on as the proper user. All commands executed by *crontab*, run as if they were invoked by the user logging into that account.

/etc/rc then performs miscellaneous "housekeeping" chores, such as preserving editor files (if they exist), starting up the "lp daemon" (which prints any files waiting in the printer spooler), and removing temporary files that are no longer needed by the system.

As its final tasks in state 2, */etc/rc* saves various logging files (by renaming them) and prints *revision(1)* information about the HP-UX operating system software.

Returning Control to the /etc/init Process

Once */etc/rc* has finished its state 2 execution, control returns to */etc/init* which now executes the commands from the command field of all state 2 entries in */etc/inittab*. Typically, */etc/inittab*'s state 2 command field entries consist of */etc/getty* commands, one for each terminal on which users are to log in. This sets up, on each terminal, the process that runs the login program and eventually runs the shell program once someone successfully logs in.

Execution of /etc/getty

As shipped to you, */etc/getty* is invoked only for the system console in state 2. You need to add a "2" entry to */etc/inittab*'s rstate field for each terminal connected to the system. Instructions for doing this are in "The System Administrator's Toolbox" chapter of this manual.

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.

When that process is terminated (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 flagged as continuous in the *inittab* entry. If the process is continually "respawned", *init* again invokes the command in the command field of the appropriate *inittab* entry as described above. If the process is not flagged as continuous, it is not restarted.

NOTE

Do **not** add */etc/getty* entries to */etc/inittab* for terminals not present. If you do, *getty* will send an error message to the console, wait 20 seconds, and then exit. On earlier versions of HP-UX, a *getty* to a non-existent terminal would consume large amounts of SPU time. This is no longer the case.

Login

The tutorials supplied with your system describe how to log in (gain access to the system). This section describes the function of the operating system during that process.

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

   If you are using an HP 27128A ASI card or an HP 27140A modem MUX card as your interface to a modem (or a terminal emulator such as an HP Portable Computer), the system will hang up after a third failure.

3. *login* 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) whatever command is present in the command field of your */etc/passwd* entry. Typically, this is the shell, */bin/sh.* However, any command may be placed in the command field of */etc/passwd.* If no command is present in the command field, */bin/sh* is executed by default.

6. Next, assuming a shell was generated in step 5, the shell executes the shell script */etc/profile* for the Bourne shell (*/bin/sh*), or */etc/csh.login* for the Berkley C shell (*/bin/csh*). As shipped to you, these scripts define and export the environment variables PATH, TZ, and TERM. Since */etc/profile* and */etc/csh.login* execute for each user as he logs in and since the super-user (root) owns these scripts, you (as system administrator) can modify */etc/profile* and */etc/csh.login* to change and export each user's default settings for the environment variables.

7. */etc/profile* and */etc/csh.login* also execute the *stty* command to set a terminal's characteristics. In addition, they define the path for the MAIL environment variable and they perform the following tasks:

   - displays the message-of-the-day (contained in */etc/motd*).
   - uses *mail -e* to detect if any mail is present; if there is any mail, the message `You have mail.` is displayed.
   - executes `news -n` which displays the names of all new files added to */usr/news* since the last *login.*

8. Finally, the shell executes the script *.profile* for the Bourne shell and *.cshrc* for the C shell, if they exist in your home (login) directory. Typically, a *.profile* or *.cshrc* file is created by the system administrator for each user or is created by the user to customize 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. *.login* is executed following the execution of *.cshrc.*

9. Now that you have successfully logged in, the shell waits for your first command.

Changes You May Want to Make

You should now have an understanding of how the operating system is loaded and initialized and the procedures used to log in on the system. This is a good time to consider how you can modify the boot and login procedures to customize your HP-UX system. This section is designed to help you in this process.

The discussions of system boot and login presented earlier in this chapter have been condensed in the following presentation to provide a "functional tour" through these same procedures. In this tour, the boot and login procedures are reduced to simple statements of function. Suggestions are made as to the type, scope, method and advisability of the modifications that can be made. **Note that changes to the files */etc/inittab*, */etc/rc*, and */etc/passwd* should not be major ones; the system may not boot if these files are radically modified**. The procedures for making the suggested modifications are either detailed here or in Chapter 5 ("The System Administrator's Toolbox").

The computer is powered up and the first operating system found is loaded.

The boot ROM cannot be changed. However, by understanding the search algorithm the boot ROM uses to find an operating system, you can load any supported system that you want.

The first process is brought up and run; the shell script */etc/rc* is executed in state s to run some commands.

You can place any command in */etc/rc* that you wish executed while still in the single-user state (i.e., before anyone can log in except on the system console). Simply edit the file using one of the editors supplied with your system. Environment variables set and exported at this time are inherited by other processes started by */etc/rc*.

You may wish to change */etc/rc* to include one or more *mount* commands (to mount other file systems) before users log in to the system. */etc/rc* is also a good place to monitor and/or clean up logging files.

NOTE

It is no longer necessary than you run *fsck* **every** time the system is powered up, since this is done by */etc/bcheckrc* at power up. See the "Using the FSCK Command" appendix in this manual and the *fsck(1M)* entry in the *HP-UX Reference* for details.

The system administrator takes the system into state 2 causing certain commands from */etc/rc* to be executed.

State 2 (the multi-user state) is entered by typing:

```
init 2 [Return]
```

The */etc/rc* file executes different commands when entering state 2 than when entering state s. One command of interest is the *cron* command which runs programs at a scheduled time.

If *cron* is executed by */etc/rc*, you may want to add entries using the *crontab* command to automatically and periodically perform certain procedures. For example, you might add entries for *cron* to:

- back up the system.
- periodically execute the *sync* command which writes the information in the system's I/O buffers to the disc. This helps prevent corruption of the file system.
- call other HP-UX systems for mail and other *uucp* transactions.
- execute system accounting commands (see chapter 6).

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

Insure "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.

A user attempts to log in to HP-UX.

To keep track of all bad login attempts, you must create a log file called */etc/btmp*. Once this file exists, the system uses it to log unsuccessful login attempts. A summary of these attempts may be viewed using *lastb(1)*.

The system also keeps track of all logins and logouts in a log file called */etc/wtmp*, if created. A variety of login and logout information may be accessed with the *last(1)* command.

You can restrict 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 can contain more than one entry but only a single entry per line. If you do not explicitly 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(1)* to become the super-user on any device; see the *login(1)* entry in the *HP-UX Reference*. Here is a typical */etc/securetty* file with two entries:

```
console
tty05
```

A user successfully logs in; after some initialization, login changes the user's current working directory to his login directory.

To set the user's login directory, edit the file */etc/passwd* (see *passwd(5)* in the *HP-UX Reference* manual).

Login executes a command from the user's entry in */etc/passwd*.

Typically, the command invokes a shell for the user. However, you may 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 login directory. You might add an entry to */etc/passwd* of the form:

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

The name `john` is the user's login name and the values `135` and `12` are his user ID and group ID respectively. The entry `run testx only` is a comment in */etc/passwd*. The entry `/users/john` is the user's login directory. The last entry 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 in the more common case of a user executing a shell: when the shell is terminated (with a CTRL-D or `exit` command given by the user), the user is logged off.

The command field of */etc/passwd* is also useful for enabling a user 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 causes a list of all system users (all users currently logged in including the user `who`) to be displayed on the terminal. The user is logged in only for the duration of the *who* program and is logged off when the program terminates.

As shipped to you, */etc/passwd* has entries for the users `who` (used as described above), `date` (which executes the *date* command providing handy access to the time and date), and `sync` (which executes the *sync* command). The *sync* command writes all the information contained in the system's I/O buffers (in RAM) to the disc. Executing *sync* fairly often helps ensure the integrity of the file system.

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

/etc/profile (for the bourne shell) or */etc/csh.login* (for the Berkley C shell) allows you to force execution of one or more commands for each user when she logs in, every time that she logs in. This is ideal for communicating regularly with each system user and for forcing the execution of commands that each should execute at login. For example, */etc/profile* and */etc/csh.login* (as shipped to you) assume that */etc/motd* contains one or more messages and send the contents of that file to each user's terminal at login (via the *cat* command; the output appears on the user's standard output). 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* when you want to alter their function. New commands can be added to these scripts; old commands can be removed or modified. The */etc/profile* and */etc/csh.login* shipped to you are commented and, with the foregoing documentation, should be fairly easy to follow. Changes made to these scripts do not go into effect until the script is executed (either at login or explicitly).

The shell scripts *.profile* (for the bourne shell), *.cshrc* and *.login* (for the C shell) are automatically executed, if they exist in the user's login directory.

Each user may have a personal version of *.profile* or *.cshrc* in his login directory; it is this personal copy that is executed once */etc/profile* or */etc/csh.login* have finished. *.profile* and *.login* are used to execute those commands that each user wants executed at login, as well as each time a new shell is spawned. For example, a user may want prompts from the shell that are different from the default prompts. To change these, the user simply adds `PS1="new prompt"` and `PS2="new secondary prompt"` to his these scripts, in the Bourne shell, and `set prompt = "[\!]:  "` in the C shell. Remember, in the Bourne shell environment variables must be exported to have an effect upon subsequent processes. In the C shell you must use `setenv`.

Following the execution of *.cshrc* in 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 path name) when you want to alter its function. New commands can be added to the script; old commands can 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(1)* 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 in order to operate properly. See the *stty(1)* command in the *HP-UX Reference* and "The vi Editor" article in the *HP-UX Concepts and Tutorials* manual.

Changes made to the *.profile* script do not go into effect until the script is executed. By typing

```
. .profile              # In the Bourne shell
source .chsrc           # In the C shell
```

in your current shell, *.profile* or *.cshrc* will execute in the current environment.

Changing the System's State

As previously mentioned, the system "comes up" executing the commands from the command field of state `s` entries in */etc/inittab*. However, once the system has come up, you can use the *init* command to take the system to state 2 (**assuming you have modified your script to do this as outlined previously**).

For each login terminal, you need to add "respawn" *getty* entries to state 2. Entries not explicitly defined will automatically be terminated when entering that state. For example, if you have not added a *getty* "respawn" for state 3 for the console, entering state 3 will cause the console to die! Don't do this!! When you take the system into its single-user state, notify users in advance. Entries in */etc/inittab* look like:

```
03:236:respawn:/etc/getty tty03 H
```

`03` is the two-character "process ID" field; `236` are states 2, 3 and 6 (there should be at least one state entry for each of these entries), `respawn` tells *init* to continuously "respawn" the entry; and the colons (:) are mandatory separators and should be used as shown. See the *inittab(5)* entry in the *HP-UX Reference* for furtherdetails.

You can create new states if you wish. To create new states, make entries in */etc/inittab* that define how you want the system to operate in its new state. For example, identify the state entry by a state number (in the range s, 1 to 6), identify the command you want executed for each state entry and list any flags that are to be considered. Once */etc/inittab* contains all of the entries you want for the new state, be certain to warn all users to log off before you change states. Changing states while users are logged on can terminate their login processes in the middle of execution unless specifically defined in the new state.

Once all the users that need to be off the system are off, flush (clear) the system I/O buffers to the file system by entering:

```
sync [Return]
sync [Return]
```

The second *sync* forces execution of the first. Next, change states by making an entry of the form where:

```
init new_state [Return]
```

new_state is the number of the new state

Depending on how the new state is defined, even the foregoing may not be necessary. For instance, you can move freely between states as long as entering the new state does not kill user or system processes that may have begun in the previous state. Watch for side effects however. Consider the case where a user logs off, you then change states (from say, state 2 to state 4) and when the user attempts to log in, he cannot because an */etc/getty* entry does not exist for him in state 4 the way you have defined it.

When the system enters the new state, 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 state number. And of course some files, such as */etc/rc*, have no entries for states other than states **s** and 2. The */etc/rc* script may be customized to include more states. The system boot and login processes occur more or less as previously described.

Summary

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 configuration files. The following list summarizes the files that you may want to alter and identifies the types of changes you may want to make. Use these suggestions in conjunction with those listed 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. Commented versions of files discussed here that do not have a specific path name (because they are generally user files in a user's home directory) 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, you will not have an */etc/newconfig* directory.

NOTE

Changes made to the files */etc/inittab*, */etc/rc*, and */etc/passwd* should not be too extreme; the system may not boot if these files are radically modified. You can minimize this risk by checking the files */etc/passwd* and */etc/group* with the *pwck(1M)* and *grpck(1M)* commands.

/etc/inittab

Contains one state 2 entry for each terminal to be accessed as a login terminal at power up. Generally this will be only the system console. Additionally, this file contains entries for the different states (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 state change. It will take different actions for different states. This script typically contains commands to:

- run *expreserve* to preserve editor files (see the *ex(1)* entry).
- mount file systems via the *mount* command.
- start the *cron* program running.
- clean up any logging 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 on 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 that are printed to each user when he logs in. If you have a message that you want to communicate with every user (such as a message specifying a new system update), write the message in this file by using your favorite text editor. As each user logs in, the message will be printed (assuming that the scripts */etc/profile* and */etc/csh.login* are not modified to remove the command that prints */etc/motd*).

/usr/news

This is a directory owned by the user `root`. It is shipped as an empty directory and can be used by you to communicate with users on the system. You can also change the directory permissions to allow any users to put messages here. Place any message you want in a file contained in this directory. If there is a `news` command in either the file */etc/profile*, in *$HOME/.profile* or *$HOME/.login* (the user's personal version of profile), the file you placed in the */usr/news* directory will be printed when the user logs in. Depending on the options used with *news*, a user only receives the message once. See the *news(1)* entry in the *HP-UX Reference* for details.

/etc/profile or */etc/csh.login*

These shell scripts are automatically executed for each user, each time she successfully log into the shell. 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*) since it contains information that each user should see before beginning her work. This is accomplished by placing the statement:

```
cat /etc/motd
```

in the */etc/profile* or */etc/csh.login* shell scripts (see the *cat(1)* entry in the *HP-UX Reference*). 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 *.profile* shell script.

/etc/wtmp

A text file used by the system to keep a history of logins, logouts, and date changes. Created automatically by the system, the contents of this file are accessed with the *last(1)* command.

/etc/btmp

A text file which—if it exists—is used by the system to keep track of bad login attempts. You must explicitly create this file to use this feature. The contents of this file are accessed with the *lastb(1)* command.

/etc/securetty

A text file which—if it exists—specifies which tty files the `root` user can login on. You must explicitly create this file and place the tty special file names in it 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 the user's environment and that perform functions the user wants executed each time he logs in.

For example, a user's *.profile* or *.login* script may 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—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 like arrow keys can be used with the *ex(1)* family of HP-UX editors (*vi*, *ex*, etc.). The file *.exrc* must exist in the user's home directory ($HOME) to use these features. The editor searches for *$HOME/.exrc* and, if it exists, uses the definitions to create extra editor features.

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* or *$HOME/.profile*. The *.exrc* file serves a function similar to EXINIT. See the appendix to "The vi Editor" article in the *HP-UX Concepts and Tutorials* manual for further details.

/etc/terminfo

This subsystem identifies terminal capabilities for programs such as the *vi* text editor. It defines terminal attributes for all Series 500 models and HP supported terminals. It also contains terminal attributes for terminals **not** supported by Series 500 HP-UX; these are provided for your convenience, but Hewlett-Packard does not support their use.

/etc/checklist

This text file contains a list of device files to identify the default devices that the file system integrity check program (*fsck*) is to examine. When no device file specification is supplied with the *fsck* command, *fsck* performs its checks on the devices listed in */etc/checklist*. This file is also used by the system accounting *diskusg* command.

The file */etc/checklist* is shipped with a single special (device) file name: */dev/hd*. This file corresponds to the CS/80 disc or SS/80 on which you installed the system and which functions as the root device (contains the root of the file system).

/etc/catman

Executing the *catman(1M)* command expands the *nroff* versions of manual pages (used by the *man(1)* command) into their "processed" form. Subsequent accesses via *man* use the processed manual page, significantly improving response time. The price for the improved speed is disc space — the expanded files will use slightly more storage space than the originals. This roughly doubles the disc usage for manual pages because the original files remain intact. By default, running *catman* causes manual pages in all the */usr/man/man*`X` directories (where `X` is 0 through 9, `l`, `n`, or `p`) to be processed and stored in the corresponding */usr/man/cat*`X` directories.

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

```
cd /usr/man
for num in 1 2 3 4 5 6 7 8 l n p
do
  mkdir cat$num
done
```

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

- 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.
- Create selected sections of the processed manual pages by executing `/etc/catman` *sections* (where *sections* is one or more logical sections in the *HP-UX Reference* such as 1).
- Do not execute `/etc/catman` at all. Because the */usr/man/cat* directories now exist, the first time *man(1)* 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 disc space but do not want to use any more than is necessary. With this "build-as-you-go" alternative, the system only fills the *cat* directories as manual pages get accessed by *man(1)*.

Notes

The System Administrator's Toolbox

5

Organized alphabetically by task or procedure, this chapter is designed to guide you through your designated tasks as the system administrator. Each major heading in the chapter identifies one or more administrative tasks. The procedure supplies at least one method for achieving the stated task. The following topics are covered:

- Adding/Moving Peripheral Devices
- Adding/Removing Users
- Backing up and Restoring the File System
- Booting HP-UX
- Changing a Password
- Changing and Creating System States
- Communicating with System Users
- Configuring HP-UX
- Configuring the Terminal Emulator
- Controlling Disc Use
- Creating File Systems
- Creating Groups/Changing Group Membership
- Media Utilities
- Mounting and Unmounting Volumes
- Setting the System Clock
- Setting Up the LP Spooler
- Shutting Down the System
- Updating HP-UX and Installing Optional Products
- Modifying the Boot Area

It is a good idea to make a copy of any of the "configuration" files shipped with your system before you change them. The "Summary" section at the end of Chapter 4 contains a list and a brief description of these files. Other critical files should also be copied before altering them. Any files considered to be critical are mentioned in the procedure that discusses their alteration. You should also log the permission mask, owner and group id's of any files you backup, so that a clean recovery can be insured. Following this precaution provides you with a backup copy of the original version of the file.

Adding/Moving Peripheral Devices

The HP-UX operating system requires the existence of special (device) files in order to perform I/O to peripheral devices, such as disc drives, printers, tape drives, and terminals. Special files and an associated topic, block and character I/O, are discussed in the "Concepts" chapter of this manual. This section introduces you to the tools necessary to set up peripherals and their associated special files. The section also discusses terminal hardware configuration and the HP-UX modifications required for communicating with terminals.

During the system installation process, a number of special files are created on your CS/80 disc. To allow you the greatest flexibility and to avoid cluttering your disc, only a partial set of the available special files are created. If you have a fairly simple single- or multi-user system, the supplied set of special files may be all that you need. If, however, your system is a large or complex one (with many peripheral devices), you will need to create new special files to communicate with those devices. You may also want to create new special files or change existing ones if you decide to add or remove particular peripherals at some later date, or if you find it necessary to tune your system by changing the locations (interfaces or bus addresses) of peripherals.

The default set installed with your system contains, among others, special (device) files for the system console, a variety of mass storage devices including the CS/80 disc/tape on which your system was installed, and a printer. It also contains special files that are used and required by HP-UX to communicate with some pseudo-devices; that is, hardware not considered to be peripherals but that requires special files for system communication (such as `/dev/tty`). These latter files must not be removed from the system under any circumstances, if HP-UX is to operate properly.

To see all of the currently installed special files on your system, log in on the system and type:

```
ls -R /dev Return
```

The system will respond by listing all the special files under the */dev* directory. The listing should include all of the special files shown in the following table. The table also includes brief explanations of the devices associated with these files and other useful information.

While you are on the system, you may want to get a listing of a file that will be useful later in this section. If you have a printer already hooked up to your system, and have created the file */dev/lp*, type:

```
pr /etc/mkdev > /dev/lp & Return
```

which tells HP-UX to print the contents of the file */etc/mkdev*, send the output to the printer, and to accomplish this as a "background" process. Notice that the system's prompt returned almost immediately. It also echoed a number which is the background process number associated with the process you just started. This way, if something goes wrong (such as the special file not mapping properly to the printer), you still have access to the system and a way to deal with the situation. You can stop this background process by typing:

`kill -9` *pid* `Return`

where *pid* is the process id number that was echoed before the system's prompt returned. If you no longer have access to this number, use the *ps(1)* command to find it, and then execute the *kill(1)* command as illustrated above.

Default Special (Device) Files

Special File	c/b	Major	Minor	Device Description
console	c	31[1]	0xScAd00	System message port
syscon	c	31[1]	0xScAd00	System console (linked to console)
systty	c	31[1]	0xScAd00	System tty (linked to console)
tty	c	20	0x010000	Process group control terminal
null	c	15	0x000000	Null file ("bit bucket")
hd	b	1	0x050000	Root (CS/80) disc drive
rhd	c	1	0x050000	Root (CS/80) disc drive

Before delving into the software aspects of adding/moving peripheral devices, be sure your peripheral hardware is set up correctly. The guidelines given in the Installation Guide supplied with your computer will help you configure your hardware. Make sure that you gracefully shut down your system before you change any of the address switches on your peripherals. To shut down the system, type:

`/etc/shutdown` Return

For details on *shutdown*, see the section "Shutting Down the System" in this chapter.

Overview of the Task

There are several basic steps required to add or move peripheral devices to your system. Here is an overview of the tasks you will need to accomplish; they are explained in detail later:

1. Using the guidelines covered in the Installation Guide supplied with 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. Connect the peripheral device. If the peripheral device requires an interface card, set the appropriate switches on the card and install the card in the computer. **Never install an interface card while the computer is powered up**. Then set any required switches on the device and connect it to the computer (or interface card). If you ever change the switch settings on an HP-IB device, be sure to power cycle the device before attempting to address it.

3. Ascertain whether the peripheral device will be addressed as a block or character device, or both (disc drives will require both modes of access). Block and character I/O are discussed in the "Concepts" chapter in this manual, and examples are provided later in this section.

4. Next, determine if the special (device) file necessary to communicate with the peripheral device already exists on your HP-UX system. Some special files were shipped with your system and are shown in the table above. Default files reside in the */dev* directory and follow the naming conventions explained in the *intro(4)* entry in the *HP-UX Reference* manual.

---

[1] Use major number 31 for consoles on HP 27128 or HP 27130 interface cards, and major number 29 for all others.

5. If the appropriate special file does not exist for the device in question, you will have to create one. There are two ways to create special files — using the *mknod* command to create a particular device file or editing, then executing the *mkdev* shell script to create one or more device files. Your choice of which method to use depends on how many device files you need to create and how experienced you are at the process of creating them. Use the *mkdev* script or the tables given later in this section to get the parameters (major and minor numbers) needed by *mknod* to create the device files.

Determining the Peripheral's Location

If the peripheral you wish to add/move is an HP-IB device, determine the select code and bus address where the device will reside. The Installation Guide supplied with your computer lists several guidelines to help you identify an appropriate location for your HP-IB peripheral. The guidelines for interface selection are reviewed here; you should still consult your Installation Guide to determine all available HP-IB bus addresses for devices on these interfaces.

- The system root device (CS/80 disc) is usually located at select code 5 on a high speed HP 27110A or HP 27110B interface card.
- The system printer (if present) should be on a medium or low speed HP-IB interface, separate from the system root device.
- A 9-track tape (if present) should be placed on a low speed HP-IB.
- Avoid putting flexible disc drives on the same interface as the root device.
- Discs other than the root device should be placed on a separate HP-IB bus when possible.
- Graphics devices should be placed on low speed HP-IB interfaces when possible.
- If you have an HP 98288A Display Station Buffer (DSB) board with a model 550, it must be installed in main processor board slots 4, 5 or 6 by your HP Customer Engineer. This is usually bundled with the HP 98700H display station system.

Connecting the Peripheral

Connect the peripheral to your computer at the location you have just determined. The computer's Installation Guide provides instructions for installing an interface card and identifying its select code. The manual supplied with the peripheral details the procedure for connecting it to the computer and setting its address if it has one.

Terminal hardware configuration is covered in the computer's Installation Guide. You must create the associated special files for the terminal as well as follow the instructions at the end of this section to set up the software aspects of terminal configuration.

CAUTION

DO NOT ATTEMPT TO UNPACK AND CONNECT A CS/80 DISC DRIVE (OTHER THAN THE HP 7908) YOURSELF. THE CS/80 DISC DRIVES ARE PACKED TO PREVENT DAMAGE DURING SHIPMENT. TO PREVENT DAMAGE TO THE DEVICE, AN HP CUSTOMER ENGINEER MUST UNPACK, INSTALL AND TEST THE DEVICE.

Block versus Character Special Files

Determine whether you should create a **block special file** or a **character special file**. **Block special files** are used for communicating with disc mass storage devices that are to be used for mountable file systems. Block special files are also used for mounting a file system from a supported cartridge tape drive. However, mounting a file system from a cartridge tape drive is **NOT** a recommended procedure due to excessive wear on both the tape drive and the tape medium.

Character special files are used for communicating with terminals, printers, plotters, digitizers, magnetic tape drives, and, on occasion, disc mass storage devices. Communicating with a mass storage device (such as a disc) with a character special file causes the system to treat the disc like a magnetic tape drive (see the *disc(4)* entry in the *HP-UX Reference*) and the "Concepts" chapter of this manual. If you are going to use the *tar(1)*, *tcio(1)*, *cpio(1)* or*dd(1)* commands to write to tape, you will need a character special file.

In most cases, discs should have both block and character special file entries. All other devices should have one or the other.

Creating Special Files

If, by examining the entries in the *dev* directory, you have determined that the appropriate special file for the peripheral does not already exist, you must create one. Special files are created with the *mkdev* script or, alternatively, with the *mknod* command. *mkdev* is a shell script that uses the *mknod* command to create one or more special files; it allows you to create special files for every allowable address and device combination. It is also a source of information for parameters that you need to execute the *mknod* command. Those parameters are also given in tables later in this section. Before using *mkdev*, you must customize it (by editing the file) to select the special files to be created and to adjust their associated parameters.

The remainder of this subsection explains how to use the *mknod* command and the *mkdev* script to create special (device) files. While reading the remainder of this material, you may find it handy to have available a copy of */etc/mkdev*.

To create a special file with the *mknod* command, you need to login as `root` and make an entry of the form:

`/etc/mknod` *path_name file_type major minor* [Return]

where *path_name* is the pathname of the special file to be created. You should select a file name for the special file which easily identifies the associated peripheral. The entry *intro(4)* in the *HP-UX Reference* manual describes a naming convention for special files. Using this naming convention makes your system easier to support and maintain. Special files are kept in the directory */dev* for ease of housekeeping. Additionally, many commands expect to find special files in */dev* and will fail if the required special file is not there.

file_type is a single character `b`, `c`, `n` or `p`; `b` specifies that the file is a block special file, `c` specifies that the file is a character special file, `n` specifies that the file is a network special file and `p` specifies that the file is a named pipe. See *mknod(1M)* for making network special files.

major is the number of the driver used to communicate with the peripheral. A table of all major numbers is provided later in this section.

minor is a value specifying the minor device with its actual address on the I/O bus. The minor number is made up of the select code, bus address, unit and volume numbers.

Be aware that the *mkdev* command automatically handles other aspects of the creation process besides executing *mknod* commands. These aspects include setting up the correct access modes on special files.

If you choose to use the *mknod* command (instead of modifying, then executing the *mkdev* script), determine the needed major parameters from the tables given later in this section. Follow the guidelines in the "Setting Appropriate Permission Masks" section to insure that correct access modes are set on the special files you create.

Using the mkdev Script

Before using *mkdev*, first make a copy of the file */etc/mkdev* for archival purposes by typing:

```
cp /etc/mkdev /etc/mkdev.old [Return]
```

Now that you have saved an untouched version of the command (the file `/etc/mkdev.old`), you can use any of the HP-UX text editors to edit the "working version" of the script (the file `/etc/mkdev`). Customize it for your needs by following the instructions provided below and by using the detailed comments contained in the script. The commented script itself is a good source of information. Be certain the changes you make do not defeat the script.

NOTE

Before using the *mkdev* command, make a copy of the original file, then edit the "working version" of *mkdev* to customize it for your installation.

You must edit and customize the */etc/mkdev* file before using it.

After you have modified the file, you can execute the script by logging in as `root` and typing:

```
/etc/mkdev [Return]
```

Alternatively, you can save the informative and diagnostic messages produced by the script for later examination. This is accomplished by redirecting a copy of the output to a file, as well as the screen. To redirect output to a log file, type:

`/etc/mkdev | tee` *log_file* `Return`

where *log_file* is the path name of the file where you wish to receive diagnostics.

After you have executed the *mkdev* script, you can examine the results by typing:

`ls -lR` */special_file_directory* `Return`

where *special_file_directory* is the name of the directory used by the script to produce its special files. Unless you modified that portion of the script, this directory should be */dev* — the same directory in which the default special files were installed. Note that it is not recommended that you change this.

This "long" (-l option) listing gives you information about all of the parameters that were used to create the special file. For example, you should see an entry similar to the following:

```
crw--w--w-   1 root     other     31 0x000000 May 20 09:30 console
```

The first character in the entry tells you whether the special file is a character (`c`) or block (`b`) device and the next series of characters represent the file's access permissions. The major and minor numbers are the two numbers contained in the size field, in this case `31` and `0x000000` respectively (see *ls(1)*).

If you make a mistake, you can always delete the special files you wish to change and re-create them by editing and executing the *mkdev* script again. You should be aware, however, that deleting some of the default special files contained in */dev* will cause severe problems because the system needs these files to operate properly. For example, you will not be able to access your root device if you delete its special file.

Editing the mkdev Script

The *mkdev* script contains a series of *mknod* "templates" for creating every allowable device/address combination. They are called templates because they are models that must be modified to reflect the actual parameters you want associated with the peripheral's special file.

Because the script contains a lot of information, the first time you edit it be prepared to spend some time with the script and read through its comments. Do not expect to absorb all of the details; only try to familiarize yourself with its organization and general approach. Do not let its size and density overwhelm you; subsequent modifications become easier.

The script is organized into device classes. Separate sections and templates exist for: miscellaneous devices, terminals, CS/80 mass storage devices, non-CS/80 mass storage devices, magnetic (9-track) tape devices, printers, general HP-IB devices (plotters, digitizers, graphics printers, etc.), and CRT graphics devices.

Begin with reading the first few pages in the script. These pages describe the overall structure of the script in detail and tell you how to modify it. Then determine what device class a peripheral belongs in and read the corresponding paragraphs both below and in the script.

Before modifying the script, **be sure that you have created an archive copy** as described earlier. In general, modifying the *mkdev* script consists of the following steps:

1. Comment out the lines that read:

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

   by adding a comment sign (the # character) in front of each line. These lines indicate your intent to run a modified script. The script will not create any special files if you do not eliminate these lines.

2. Find the template or device class that corresponds to the peripheral.
3. Read the script's instructions pertaining to that device class.
4. Decide what name to use for the special file associated with the device.
5. Copy the *mknod* template if indicated by the instructions.
6. Modify the *mknod* template to correspond to the name you chose. Use the *intro(4)* naming convention or one of your own.

7. Fill in any of the template's missing parameters (or placeholders for parameters) where the instructions indicate; this includes the major and minor numbers for some device classes.

8. Remove the comment sign in front of the modified template so the line will be executed when you run the script.

Using mknod with the Supplied Tables

The following sections are structured by "device class" and contain the information and tables you need to create special (device) files using *mknod*. These device class sections are sequenced just as in the *mkdev* script and a fair bit of this material duplicates information from *mkdev*; it is provided here for your convenience.

Each line in the tables (supplied in each device class section) corresponds to either one or two special files; two if the "raw" device entry is given on the same line as the regular entry. These tables use the following representation:

The **Device** column represents either the literal name of a special file or a symbolic name (as discussed in the "Naming Conventions" section below); if not obvious from context, the section specifies whether the name is literal or symbolic.

The **C/B** column specifies Character or Blocked access and the **Major** and **Minor** (or **Maj** and **Min**) columns specify the major and minor parameters to be used with *mknod*.

The **RawDev** column (if present) represents the "raw" special device file for the device in question and should have the same name as that in the **Device** column with an added prefix of `r`. The **Notes** column contains commentary for your information only and should **never** be entered as part of a *mknod* command. Under no circumstances should you change any table entries flagged as: `Mandatory (do not change)`.

The rest of the "Adding/Moving Peripheral Devices" section contains:

- A table of all major numbers for each supported device class.
- Guidelines for setting "permission masks" to insure appropriate ownership of and access to special files.
- Guidelines on naming conventions.
- Specifics on Miscellaneous Devices.
- Specifics on Terminals and Modems.
- Specifics on Consoles.
- Specifics on Pseudo Terminals.
- Specifics on CS/80 Hard Discs and Cartridge Tapes.
- Specifics on Magnetic Tape Devices.
- Specifics on Printers.
- Specifics on Plotters and Digitizers.
- Terminal Configuration Information.
- Special Considerations for Terminals.

Here are the major numbers for each supported device class. The specific tables in each of the following sections also contain the applicable major numbers.

Major Numbers for Device Classes

Major #	Device Class
1	All CS/80 type mass storage devices.
6[2]	8-inch HP 9895 flexible disc drives.
8[2]	1/4-inch HP 8290X flexible disc drives.
9[2]	HP 9885M/S flexible disc drives.
10[2]	Memory disc configured by *sdfinit(1M)*.
11[2]	HP 7970 magnetic tape drive.
12	HP-IB devices, except CS/80 mass storage, CIPER protocol printers, or laser printers.
14[2]	CIPER protocol printers.
15	Null device.
18[2]	HP 27112A GP-IO interface card.
19[2]	Raw transfer to ASI or 8-channel MUX interface cards (see 31).
20	*/dev/tty*
22	Formatted output for HP 2631 type printers.
26	Raw format for the Model 520's built-in thermal printer.
28[2]	Models 520B and 520C graphics displays.
29[3]	Model 520 built-in console, HP 98700 console, 6-channel modem MUX, and the slave side of a pty.
31[2]	Normal ASI and 8-channel MUX interface cards.
32[2]	Model 520 standard color display and the HP 97062 color output interface.
33[2]	SRM (Shared Resource Manager) interface.
34[2]	HP 2285 EtherNet interface.
35[2]	HP 2680 and HP 2688 laser printers (not HP 2686).
36[2]	HP 7974 and HP 7978 magnetic tape drivers.
37[2]	Model 550 internal HP-IB.
38[2]	HP 27125 EtherNet interface.
39[2]	HP 27125 IEEE 802 interface.
40[2]	HP 2285 IEEE 802 interface (see 34).
41[2]	HP 98700 raw 8042 HP-HIL.
42[2]	HP 98700 HP-HIL devices.
43[2]	ITE (internal keyboard emulator) keyboard.
45[2]	Master side of a pty (see 29).

[2] This is an optionally installed driver. See the section on installing optional products which follows in this chapter.

[3] This driver accesses optionally installed system segments—see the section on installing optional products in this chapter.

Setting Appropriate Permission Masks

In each section on device-class-specific information, a permission mask is given. If you use the *mknod* command with the tables supplied below, you must insure that permission masks and a few other items (automatically handled by the *mkdev* script) are properly set up. **Before creating any special files with the *mknod* command**, use the instructions below to:

1. log in as the user `root`,
2. move to the */dev* directory,
3. establish an alias for *mknod*,
4. and set the appropriate protection mask.

After logging in as the `root` user, accomplish the other **mandatory** operations by executing the following lines (each followed by `Return`):

```
cd /dev
mknod=/etc/mknod
defumask=111
restrictedumask=166
```

Each device class section has one or more lines such as:

```
umask $defumask
or
umask $restrictedumask
```

In each case, execute the line as shown above or the equivalent:

```
umask 111
or
umask 166
```

This **must** be done before executing one or more *mknod* commands so the special file(s) will have correct access permissions.

Naming Conventions

In the tables that follow, you will see names such as "fd9121.0"; these need to reflect the appropriate names as explained in the *mkdev* script and in *intro(4)* (see the *HP-UX Reference*). Typically, these entries should be replaced with names based on the model number of the device in question. Consider an HP 9121 flexible disc drive with two flexible disc drives.

```
fd9121.0     rfd9121.0
fd9121.1     rfd9121.1
```

where `fd` signifies a flexible disc, the `.0` suffix represents the left drive, the `.1` suffix represents the right drive, and the `r` prefix (in the entries on the right side above) represents "raw" access to the left and right drives respectively.

Miscellaneous Devices

The special files associated with the miscellaneous device class are precisely those default special files that the system needs in order to run properly. Each HP-UX installation must have the special files */dev/null*, */dev/console*, and */dev/tty*. The special file */dev/null* is a null file (a "bit bucket") used by many HP-UX commands. The special file */dev/console* identifies the system console and the special file */dev/tty* is a synonym for the control terminal associated with a process group.

These miscellaneous special (device) files are copied to your system when HP-UX is installed. They should not be changed or modified. If one or more of these files is accidentally deleted or otherwise destroyed, you can recreate it by editing the *mkdev* script and removing the comment sign (the `#` character) from in front of the corresponding entry. Alternatively, recreate it with the `mknod(1M)` command using the character/blocked designation, major, and minor numbers given below.

The following entries require that `umask $defumask` (or its equivalent, `umask 111`) be executed before using *mknod.*

Device	C/B	Major	Minor	Notes
console	c	31	0x000000	Mandatory (do not change)
tty	c	20	0x010000	Mandatory (do not change)
null	c	15	0x000000	Mandatory (do not change)

For example, to recreate the special file for the `null` device, log in as the user **root** and type the following lines (each followed by `Return`):

```
cd /dev
umask 111
/etc/mknod  null  c 15 0x000000
```

There needs to be a */dev/systty* (which is linked to */dev/console*), and a */dev/syscon* (which is linked to some terminal—usually the console). This is explained in *init(1M)*.

Memory Volumes

Creating a special file using driver number 10 enables you to treat a portion of your computer's RAM as if it were a disc device. Once a memory "disc" has been created, it may be treated just as any disc device. It can be mounted, for instance, or used to contain a boot area. It can also be used to test procedures which access a disc frequently, enabling you to avoid disc delays during testing. One obvious difference is, since a memory volume is made up of RAM, it cannot survive a system power-down.

Treating RAM as a Disc Device

Description	Suggested Pathname	C/B	Major	Minor
Raw Memory Volume	/dev/rmdU	c	10	0x0000U0
Block Memory Volume	/dev/mdU	b	10	0x0000U0

When creating a special file for a memory "disc", the select code, bus address, and volume number parameters are ignored. The hexadecimal **unit** number is used to create multiple memory "discs" (a maximum of 16 "discs" are allowed). For example, the command:

```
mknod /dev/rmdc c 10 0x0000c0
```

creates a character special file called */dev/rmdc*, which can now be used to communicate with a portion of your computer's RAM as if it were a disc. The amount of RAM allocated to a particular memory volume is set by the *interleave* parameter of *sdfinit(1M)*. Note that memory allocated to a memory volume is **not** virtual, and thus is unavailable for any other use.

To de-allocate RAM from a memory volume and return it to the system, re-initialize the memory volume (using *sdfinit(1M)* with a *blocksize*, *bootsize*, and *interleave* of zero), as in:

```
sdfinit /dev/rmdc 0 0 0
```

Terminals and Modems

Communication ports — user terminals as well as modems — need to be identified by one or more special (device) files, depending on the intended use of the port. Special `ttyd` files are required for ports that receive incoming signals ("dial in" modems); `tty` files are required for terminals (hard-wired ports). Ports that transmit signals ("dial out") require `cua` and `cul` special files.

The following entries require that `umask $defumask` (or its equivalent, `umask 111`) be executed before using *mknod*. The general template for ports is:

Device	C/B	Major	Minor	Notes
ttyxx	c	M	0xScAd0V	Sc, Ad and V are explained below
ttydxx	c	M	0xScAd0V	Sc, Ad and V are explained below
cuaxx	c	M	0xScAd0V	Sc, Ad and V are explained below
culxx	c	M	0xScAd0V	Sc, Ad and V are explained below

where `xx` is a two digit line identifier, `M` is the driver number (29 for the HP 27140, and 31 for the HP 27128 or HP 27130), `Sc` is the select code of the interface being used, and `Ad` is the port address for each port. If you are using a single port interface such as the HP 27128A ASI card, the port address will be `00` (so the minor number would be `0xSc0000` in our first example above).

ASI and MUX Volume Numbers

The volume number in the special device file minor field, has special meaning with ASI and MUX cards. `v` should always be a zero (0) for the HP 27130 8-channel MUX. To properly configure your system to talk to modems and data links via ASI and 6-port modem MUX cards, refer to the following table:

Card Type	Bit Order	Clear (0)	Set (1)
ASI	3	Ignored	Ignored
	2	Ignored	Ignored
	1	Simple Protocol (United States)	CCITT Protocol (Europe)
	0	Dial-in (Direct Connect)	Dial-out
6-Port Modem	3	Ignored	Ignored
MUX	2	Ignored	Direct Connect (overrides bit 0)
	1	Simple Protocol (United States)	CCITT Protocol (Europe)
	0	Dial-in	Dial-out

Make sure your interface card is properly configured as outlined in the Installation manual which came with it. For multi-port interfaces such as the HP 27130B eight-channel MUX card or the HP 27140A six-channel modem MUX card, the port address is the RS-232 port number your terminal is plugged into. With, for example, `tty09` plugged into the second port of an HP 27130B interface at select code 6, the correct *mknod* would look like:

```
mknod tty09 c 31 0x060100
```

HP-UX associates the system console port with the special (device) file */dev/console*. This may or may not be an ITE; see the **Internal Terminal Emulator** and **system console** definitions in Chapter 2.

An example configuration might be:

Port #	tty File	cua File	cul File
1	/dev/console	no entry	no entry
2	/dev/ttyd00	/dev/cua00	/dev/cul00
3	/dev/tty01	no entry	no entry
4	/dev/tty02	no entry	no entry
.	.	.	.
.	.	.	.
.	.	.	.
N	/dev/ttyxx	no entry	no entry

where *xx* is the two digit tty/cua/cul number described above. An example of creating the `tty`, `ttyd`, `cua`, and `cul` special files for a port is provided in the "Creating Special Files for Ports" section below. Note that you do not need a tty, cua or cul file if you are not connecting a modem device.

Creating Special Files for Ports

As with other peripheral devices, you can either edit the appropriate section of the *mkdev* script or type *mknod(1M)* commands directly from the keyboard to create the appropriate special files.

Assume that you want to create special files for a modem at select code 2 (using an HP27128A ASI card), and associate it with */dev/tty04*. Because the modem will be used as a dial-in and dial-out port, use the following forms of *mknod(1M)* **after** logging in as the user `root` and setting up the appropriate protection mask; follow each line with `Return`.

```
cd /dev
umask 111      # or its equivalent,   umask $defumask
mknod=/etc/mknod
$mknod  ttyd04  c  31 0x020000
$mknod  cua04   c  31 0x020001
$mknod  cul04   c  31 0x020001
```

Notice that the minor number for the `cua` and `cul` special files, ends with a `1`. There are now three special (device) files associated with the dial-in and dial-out modem at select code 2.

NOTE

A single-user HP-UX Series 500 system can have a maximum of 2 ports: a system console and 1 additional port which is used with *uucp*.

A multi-user HP-UX Series 500 system can have a maximum of 16, 32 or 64 ports—depending upon which license you have purchased. (This is the maximum number of ports that may be used by *login*). For some systems (and for certain applications), using this maximum number of ports may be counter-productive in terms of performance, disc space and available memory.

Consoles

To successfully boot and use HP-UX, you must have a supported terminal device as the system console. This list includes the built in ITE of a Model 520, the HP 98700 display system on a Model 550, or any of a myrid of terminals such as the HP 2623. The console **always** has the special device file */dev/console*. HP-UX will also link two additional files to */dev/console*—they are */dev/syscon* and */dev/systty*. For more information on the selection of the console and how HP-UX selects the console, see Chapter 4 (System Boot and Login) of this manual.

There are four basic configurations possible to "talk" to your console; they are:

- the built-in terminal (ITE) of the Model 520, which **always** has a major number of `29`, and a minor number of `0x000000`.
- any ASI or 6-channel modem MUX card; the major number will be `31`, and the minor number must be `0x000000`. If the boot ROM does not find a supported console at select code 0, port 0, unit 0 and volume 0, the boot procedure will fail.
- a 6-port modem MUX, which has a major number of `29`, and a minor number of `0x000004`.
- an HP 98700 display device on a Model 550. Up to three HP 98288A Display Station Buffer (DSB) cards may be placed on the stack (by your Customer Engineer), from slots 4 thru 7. The lowest ordered DSB will become your system console, and would have a major number of `29` and a minor number of `0xffAd00` where `ff` indicates the device is located in the stack, and `Ad` is the slot. Slot 4 would become `0xff0000`, while slot 7 would become `0xff0300`.

- if you have both a terminal at `0x000000` and an HP 98700 display device, the display device will be selected as the console.

Make sure the appropriate driver is installed in your boot area, before attempting to reboot your system configured to talk to a different console.

Pseudo Terminals

Often applications need some form of software support which enables an application program to pretend it has a terminal. In HP-UX 5.0, this facility has been added with a new facility called a **pseudo terminal**. A pseudo terminal is a pair of character devices: a **master** device and a **slave** device. The slave device provides processes (in this case, user applications) an interface identical to that described in *termio(4)* of the *HP-UX Reference* manual.

The difference between an HP-UX pseudo terminal and the interface described in *termio*, is that the latter always have a hardware device of some sort behind them—like an HP 2623 terminal. A slave device, on the other hand, has another process manipulating it through the master half of the pseudo terminal. Anything written on the master device is given to the slave device as input, and anything written on the slave device is presented as input on the master device.

Device	C/B	Major	Minor	Notes
ptyXX	c	45	0xfeYY00	Master side of pseudo terminal
ttyXX	c	29	0xfeYY00	Slave side of pseudo terminal

According to HP-UX naming conventions, the master side special device file should be called */dev/ptym/ptyXX*, and the slave side */dev/pty/ttyXX*, where *XX* is an identifying letter from p to w, and a hexadecimal digit. *YY* is a unique hexadecimal value used to identify the relationship between master and slave (a lot like a cattle brand). As an example, */dev/ptym/ptyp0* (master) and */dev/pty/ttyp0* (slave) would be the lowest numbered pseudo terminal pair; */dev/ptym/ptywf* and */dev/pty/ttywf* would be the highest ordered pair.

As an example:

```
mknod /dev/ptym/ptyp0 c 45 0xf37700
mknod /dev/pty/ttyp0  c 29 0xf37700
```

would create a master and slave pair called *ptyp0* and *ttyp0*.

Note that all pseudo terminal devices are located in the directories */dev/pty* (slaves), and */dev/ptym* (masters)—it is recommended that you do not change these naming conventions. For more information on pseudo terminals, see both the *termio(4)* and *pty(4)* sections of the *HP-UX Reference* manual.

CS/80 and SS/80 Hard Discs and Cartridge Tapes

A special file must be created for every CS/80 disc drive and cartridge tape drive with which your system needs to communicate. Because it is sometimes difficult to predict whether or not your disc drives (and cartridge tapes) will be used as blocked devices or as character devices, it is common practice to create two special files for each device: a block special file and a character or "raw" special file. The address-dependent minor number and the major number (driver) are the same for both block and character entries.

The minor numbers in this device class have special meaning. For each possible address where CS/80 devices may be located, there are several minor numbers. One minor number corresponds to an entire CS/80 disc and/or separate controller cartridge tape.

There are templates that correspond to each minor number at each possible address, including a template for the root device (*/dev/hd*) that was installed with your system. Your main task is to determine which template you need based on which disc you have and which controller option the peripheral has (shared or separate). Further details are provided in the *mkdev* script and later in this section.

It is recommended that all CS/80 entries have restricted access permission (i.e., execute `umask $restrictedumask` or its equivalent, `umask 166`) before using *mknod.* This gives read/write permission to the owner (`root`) only and is intended to prevent the mounting of unauthorized media on your system. If you want to change the permission to give read/write permission to everybody, then execute `umask $defumask` (or its equivalent, `umask 111`) before using *mknod* or change the permission later using *chmod(1)*.

This section contains duplicate entry templates for tape cartridges and discs. These entry templates correspond to physical locations where either device is supported: you must eliminate the unused entries (either the `hd79xx` and `rhd79xx` **or** the `mt79xx` and `rmt79xx` entries) as shown below. The tables in this section use the following representation:

- **hd79xx** and **rhd79xx** represent one of the following CS/80 discs: HP 7908, HP 7911, HP 7912, HP 7914, HP 7933, or HP 7935. You should replace the `xx` in the entries with the disc's model number.
- **mt79xx** and **rmt79xx** represent a dual (i.e., separate) controller option cartridge tape. If you are creating device files for this tape, do not use the `hd79xx`, `rhd79xx`, `hd79xxct` or `rhd79xxct` entries.

Because most of the following tables are templates, you must replace the supplied device names with an appropriate name that contains the model number of your CS/80 disc; see the naming conventions in the *intro(4)* entry of the *HP-UX Reference*. Do not change the names of any entries marked `Mandatory (do not change)`.

CS/80 "root" Disc and Cartridge Tape

These are the default special (device) files created during the installation process and necessary for the proper operation of the HP-UX file system. You should only replace these special files if they have somehow been destroyed or damaged. The names given in the **Device** column are literal; do not change or remove these entries under any circumstances.

Device	C/B	Major	Minor	Notes
hd	b	1	0x050000	Mandatory (but changeable)
rhd	c	1	0x050000	Mandatory (but changeable)
mt	b	1	0x050010	Mandatory (but changeable)
rct	c	1	0x050010	Mandatory (but changeable)

A complete listing of supported CS/80 and SS/80 devices follows. As you add peripherals to your system, such as a second HP 7914P disc drive, you will want to look up the major and minor number configurations for that device.

Supported Mass Storage Devices

Product Number	Description	Suggested Pathname	Major	Minor
HP 7908P/R	16.5 Mbyte CS/80 Disc Drive	/dev/hd7908p	1	0xScAd00
HP 7911P/R	28.1 Mbyte CS/80 Disc Drive	/dev/hd7911p	1	0xScAd00
HP 7912P/R	65.6 Mbyte CS/80 Disc Drive	/dev/hd7912p	1	0xScAd00
HP 7914P/R	132.1 Mbyte CS/80 Disc Drive	/dev/hd7914p	1	0xScAd00
HP 7914TD	132.1 Mbyte CS/80 Disc Drive	/dev/hd7914t	1	0xScAd00
HP 7933H	404 Mbyte CS/80 Disc Drive	/dev/hd7933h	1	0xScAd00
HP 7935H	404 Mbyte CS/80 Disc Drive	/dev/hd7935h	1	0xScAd00
HP 7941A	23.8 Mbyte CS/80 Disc Drive	/dev/hd7941a	1	0xScAd00
HP 7945A	55.5 Mbyte CS/80 Disc Drive	/dev/hd7945a	1	0xScAd00
HP 7946A	55.5 Mbyte CS/80 Disc Drive and cartridge tape	/dev/hd7946a	1	0xScAd00
HP 97093A	Model 520's built in 10 Mbyte CS/80 Disc Drive	/dev/hd97093a	1	0x070000

Supported Mass Storage Devices continued

Product Number	Description	Suggested Pathname	Major	Minor
HP 9122S/D	Single (or dual) 630 kbyte micro Disc	/dev/fd9122s	1	0xScAd00
HP 9125S	Single 270 kbyte 5¼ inch Floppy	/dev/fd9125s	1	0xScAd00
HP 9130K	Model 520's built in 256 Kbyte CS/80 Flexible Disc Drive	/dev/hd9130k	1	0x070010
HP 9133H	22.3 Mbyte CS/80 Disc, 630 kbyte micro Disc	/dev/hd9133h	1	0xScAd00
HP 9134H	22.3 Mbyte CS/80 Disc	/dev/hd9134h	1	0xScAd00
HP 9144A	Stand alone CS/80 cartridge tape	/dev/ct9144a	1	0xScAd00
HP 82901M	5¼" 540 Kbyte Flexible Disc Drive (dual, master)	/dev/fd82901m	8	0xScAdU0
HP 82902M	5¼" 270 Kbyte Flexible Disc Drive (single, master)	/dev/fd82902m	8	0xScAdU0
HP 9895A	8" 2.4 Mbyte Flexible Disc Drive (dual, master)	/dev/fd9895a	6	0xScAdU0
HP 9885M	8" 0.5 Mbyte Flexible Disc Drive (single, master)	/dev/fd9885m	9	0xSc00U0
HP 9885S	8" 0.5 Mbyte Flexible Disc Drive (single, slave)	/dev/fd9885s	9	0xSc00U0
HP 88140L	67 Mbyte built-in CS/80 cartridge tape drive of the: • HP 7911P/R • HP 7912P/R • HP 7914P/R disc drives	/dev/fd88140l	1	0xScAd10[4]
HP 88140S	16.7 Mbyte built-in CS/80 cartridge tape drive of the HP 7908P/R disc drive	/dev/fd88140s	1	0xScAd10[4]
HP 7970E	9-track Tape Drive (1600 bpi)	/dev/mt7970e	11	0xScAd0V[5]
HP 7971A	9-track Tape Drive (1600 bpi)	/dev/mt7971a	11	0xScAd0V[5]
HP 7974A	9-track Tape Drive (800 or 1600 bpi)	/dev/mt7974a	36	0xScAdUV[5]
HP 7978A	9-track Tape Drive (1600 or 6250 bpi)	/dev/mt7978a	36	0xScAdUV[5]

[4] The minor number shown is for a single controller drive. For a dual controller drive, use **0xScAd00**.
[5] See the "Magtape Volume and Unit Numbers" section which follows.

Magtape Volume and Unit Numbers

For magnetic tape drives, the unit and volume numbers have special meaning. To help clarify our discussion, mt refers to magnetic tape; ct refers to cartridge magnetic tape. If you ordered a dual controller option for a disc/tape combination such as the HP 7914CT, the unit number should be 0 (and the select code will be different, since the tape drive is on a separate HP-IB). In all other cases the unit number should be 1. Your Series 500 distinguishes which device is which with either different unit numbers or different select codes.

The unit number field of the minor number has special meaning when creating special files for the HP 7970E, HP 7971A, HP 7974A or HP 7978A magnetic tape drives. The single hexadecimal unit number represents a four bit binary value. Setting and clearing the bits of this binary value affects the manner in which the tape drive operates, as indicated in the following table:

Bit Order	When Clear (0)	When Set (1)
7	Industry Standard mode	Old compatibility mode
6	Immediate Report on (ignored by HP 7970/7971)	Immediate Report off (ignored by HP 7970/7971)
5	Reserved (ignored)	Reserved (ignored)
4	Tape density of 1600 bpi (ignored by HP 7970/7971)	Tape density of 800 bpi on HP 7974 6250 bpi on HP 7978 (ignored by HP 7970/7971)

The volume number field of the minor number also has special meaning when creating special files for magnetic tape drives. The single hexadecimal volume number represents a four bit binary value. Setting and clearing the bits of this binary value affects the manner in which the tape drive operates, as indicated in the following table:

Bit Order	When Clear (0)	When Set (1)
3	Ignored	Ignored
2	Rewind on close	No rewind on close
1	System V file compatibility mode	Berkley file compatibility mode
0	Ignored	Ignored

For more information on the use of magnetic tape, see Chapter 3 (Concepts) of this manual, or the *mt(4)* section of the *HP-UX Reference* manual.

Printers

This section provides the *mknod* information needed to create special (device) files for the printer device class. 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.

Supported Printing Devices

Product Number	Description	Suggested Pathname	Major	Minor
HP 2225A	150 cps Thinkjet(TM) printer	/dev/lp2225a	12[6]	0xScAd0V[7]
HP 2563A	300 lines per minute dot-matrix impact printer	/dev/lp2563a	14	0xScAdUV[7/8]
HP 2608S	400 lines per minute dot-matrix impact printer	/dev/lp2608s	14	0xScAdUV[7/8]
HP 2631B	180 cps dot-matrix impact printer	/dev/lp2631b	12[6]	0xScAd0V[7]
HP 2631g	180 cps dot-matrix impact printer	/dev/lp2631g	12[6]	0xScAd0V[7]
HP 2932A	120 cps dot-matrix impact printer	/dev/lp2932a	12[6]	0xScAd0V[7]
HP 2933A	120 cps dot-matrix impact printer	/dev/lp2933a	12[6]	0xScAd0V[7]
HP 2934A	120 cps dot-matrix impact printer	/dev/lp2934a	12[6]	0xScAd0V[7]
HP 97090A	Model 520's built-in thermal printer	/dev/lp97090a	26[9]	0x06000V[7]
HP 2680A	45 page per minute laser printer	/dev/lp2680a	35	0xScAdUV[7]
HP 2688A	12 page per minute laser printer	/dev/lp2688a	35	0xScAdUV[7]

6 When connected via HP-IB, use driver 12 for raw mode and driver 22 for cooked mode.
7 See the "Volume Numbers for Printer Special Files" section which follows.
8 For raw output the volume number should be 0; for cooked output use 1.
9 When connected via HP-IB, use driver 26 for raw mode and driver 22 for cooked mode.

Volume Numbers for Printer Special Files

The volume number field of the minor number has special meaning when creating special files for printers. The single digit hexadecimal value is made up of four bits, two of which control the wrap-around and character-per-line characteristics of the printer. The table below shows and describes the bits involved (this is applicable only to major (driver) 22:

Bit Order	When Clear (0)	When Set (1)
3	Ignored	Ignored
2	Ignored	Ignored
1	Disable wrap-around	Enable wrap-around
0	132 character line length	80 character

Using Printers as Spooled Devices

There is one additional decision you need to make. Printers can be accessed, through the line printer spooler (see *lp(1)*), as spooled devices; files are kept in a spool directory until the device is ready to process them. If your printer is set up as spooled device, you can direct output to it at any time, whether it is busy or not. To set up your printer as a spooled device, create the printer's special file as explained here, then follow the instructions in the "Setting Up the LP Spooler" section of this chapter to change the special file's ownership, group, and permission mode. That section also explains commands which control the LP (line printer) Spooler.

NOTE

If you have a system printer, you should always name its corresponding special file */dev/lp* because some commands use this special file as a default. You can create an individual special file for your favorite printer and give it the pathname */dev/lp* (there is one created for you during system installation). Alternatively, you can take an existing special file for your favorite printer and create a link to it from the pathname */dev/lp*.

To create special files for a printer, first determine the printer's location (i.e., interface and bus address). Use the guidelines given earlier in this section and in the Installation Guide supplied with your computer to select the most appropriate location. Finally, create the special file for the printer using the *mknod* command. You must assign a **unique** special file name to each entry you create; see the *intro(4)* entry in the *HP-UX Reference* for a suggested naming convention. Also, execute `umask $defumask` (or its equivalent, `umask 111`) before using *mknod*.

Plotters and Digitizers

This section provides the *mknod* information needed to create special (device) files for the plotter and digitizer device classes.

To create a special file for a device in this class, first determine the peripheral's location (i.e., interface and bus address). Use the guidelines given earlier in this section and in the Installation Guide supplied with your computer to choose an appropriate location. Then use the following table to find the major number that corresponds to that specific device. Finally, create the special file for the printer using the *mknod* command. Remember to execute `umask $defumask` (or its equivalent, `umask 111`) before using *mknod*.

Supported Graphic Output Devices

Product Number	Description	Suggested Pathname	Major	Minor
HP 7470A	A-size, single pen plotter	/dev/plt7470a	12	0xScAd00
HP 7475A	B-size, 6 pen plotter	/dev/plt7475a	12	0xScAd00
HP 7550A	B-size, 8 pen plotter	/dev/plt7550a	12	0xScAd00
HP 7580B	D-size, 8 pen plotter	/dev/plt7580a	12	0xScAd00
HP 7585B	E-size, 8 pen plotter	/dev/plt7585b	12	0xScAd00
HP 7586B	E-size, 8 pen, roll feed plotter	/dev/plt7586b	12	0xScAd00
HP 9872C	B-size, 8 pen plotter	/dev/plt9872c	12	0xScAd00
HP 9872T	B-size, 8 pen plotter	/dev/plt9872t	12	0xScAd00
HP 98700	Graphics display system	/dev/plt98700	15	0xffAd00
HP 98710	Graphics accelerator	/def/plt98710	15	0xAd0000
HP 98760A	Standard color display on the Model 520	/dev/plt98760a	32	0x010000
HP 98770B	High performance display on the Model 520	/dev/plt98770b	28	0x010000
HP 98780B	Monocromatic display on the Model 520	/dev/plt98780b	28	0x010000
HP 97062A	Color output interface on the Model 520	/dev/plt97062a	32	0xSc0000

Supported Graphic Input Devices

Product Number	Description	Suggested Pathname	Major	Minor
HP 7908P/R	16.5 Mbyte CS/80 Disc Drive	/dev/hd7908p	1	0xScAd00
HP 9111A	Data Tablet	/dev/dig9111g	12	0xScAd00

Useful Naming Conventions

You must assign a **unique** special file name to each entry you create. The *intro(4)* entry in the *HP-UX Reference* explains special file naming conventions. Generally, use `rlp` followed by the product number for raw printers, `plt` followed by the product number for plotters, and `dig` followed by the product number for digitizers. If more than one device with the same product number is present, be certain not to duplicate their special file names. For example, to differentiate between two HP 9872 plotters, name the first one `plt9872` and the second `plt9872.1`.

Terminal Configuration Information

A complete list of the terminals supported by Series 500 HP-UX is provided in the *HP 9000 Series 500 Configuration Information and Order Guide* supplied with your system. The Installation Guide supplied with your computer discusses the hardware aspects of hooking up a terminal to your system. This section offers the software configuration information you need.

After dealing with the hardware hookups, terminals must be configured so they can "talk" to HP-UX. Series 500 HP-UX requires that terminals be configured to have the characteristics listed below. If a particular configuration option is not available on your HP terminal, then the option is already properly chosen (as a default value) by the terminal.

Terminal Parameters

The manual supplied with the terminal describes how to use the function keys to configure the terminal. Generally, you will press a key that chooses the "terminal configuration" option and alter the appropriate fields by answering prompts from the terminal's configuration program. Configure the terminal with the following values:

```
Tab=Spaces        NO
RETURN Def        CR
RETURN=ENTER      NO
LocalEcho         OFF
CapsLock          OFF
Start Col         1
ASCII 8 Bits      YES
XmitFnctn(A)      NO
SPOW(B)           NO
InhEolWrp(C)      NO
InhHndShk(G)      YES
Inh DC2(H)        YES
```

Datacomm Parameters

For datacomm purposes, the following configuration parameters should also be set. The method for setting these values is similar to that outlined above; details are provided in the manual supplied with the terminal. You must set your terminal to match those characteristics specified in the *getty* for the device.

```
Parity          NONE
DataBits        8
Clk             INT
StopBits        1
EnqAck          NO
TR(CD)          HI
Chk Parity      NO
RecvPace        Xon/Xoff
SRRXmit         NO
RR(CF)Recv      NO
XmitPace        Xon/Xoff
SRRInvert       NO
CS(CB)Xmit      NO
```

Except in the case of using the terminal as a system console (discussed below), you may use any **baud rate** that the terminal will 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.

If you are using the terminal as the system console, set the terminal's baud rate at 9600 to match the HP-UX expectation for the system console. After the system is installed and running, you may change some of the configuration parameters to suit your own needs. For example, changing the HP-UX expectation for a particular baud rate is done by modifying one of the parameters to the *getty* command located in the */etc/inittab* file (and associated with the terminal in question). Further information is provided in the section that follows ("Special Considerations for Terminals"), in the *getty(1M)*, *gettydef(5)*, and *inittab(5)* entries in the *HP-UX Reference*, and in the "System Boot and Login" chapter of this manual.

Special Considerations for Terminals

When a terminal is added to the system, you must perform the steps described in the preceding section and add entries to the */etc/ttytype* and */etc/inittab* files. This allows a user to login from the terminal. Add entries to these files as described below.

The */etc/ttytype* entries have the form:

model_number location

where *model_number* is the the product number of the terminal or computer (as defined in */etc/terminfo*) and *location* is the special (device) file associated with the terminal/computer and contained in the */dev* directory. Placing comments on each line (preceded by the # character) will help you remember which terminal belongs to each user.

Here is a sample */etc/ttytype*:

```
9020    console     # Frodo's (administrator) system console
2622    tty00       # Bilbo's terminal
2622    tty01       # Gandalf's terminal
2623    tty02       # Strider's terminal
dialup  tty03       # Greybeard's dialup modem
```

Most */etc/inittab* entries have the form:

id:rstate:action:/etc/getty *-txxx special_file_name N X*

The first three fields (shown as *id*, *rstate*, and *action*) are discussed both in *init(1M)* in the *HP-UX Reference* manual and in this manual's "System Boot and Login" chapter. The normal values for these fields are: **id** = unique two character string, **rstate** = 2, and **action** = **respawn** (for continuous).

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. 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 must contain the */etc/getty* command; it is immediately followed by two, optionally, three parameters. The first parameter, *-t xxx*, is the time-out option for use with modems. *special_file_name* is the file name (*tty04*) — not the complete path name (*/dev/tty04*) — of the terminal's or modem's character special file. The named file must reside in the */dev* directory. The second parameter to *getty*, represented by *N*, specifies a speed indicator for *getty*; a value of H is common for "hardwired" (terminal) lines, a value of 3 is common for dial-up (modem) lines. The third and optional parameter, represented by *X*, specifies a delay time. If not specified, it defaults to zero (an appropriate value for terminals); a non-zero value is relevant only for dial-up modem connections. For more information, see the *getty(1M)* and *gettydef(5)* entries in the *HP-UX Reference* manual.

On a multi-user 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/tty04* the */etc/inittab* entries would be:

```
04:2:respawn:/etc/getty tty04 H
```

Note that the `id` field `04` corresponds to the last two digits of the special file (`tty04`) for the terminal on which *getty* is invoked. This convention is often used with "continuous" *getty* processes that get killed in the single-user state but is **not** required syntax: any two-character string will suffice if used consistently.

Refer to the "System Boot and Login" chapter in this manual and to the *getty(1M)*, *gettydef(5)*, and *inittab(5)* entries in the *HP-UX Reference* for further details.

Dealing with a Catatonic Terminal

If, for whatever reason, a terminal will not respond (or does not appear to respond) to your commands, two solutions are available. The first is to simply log off the system (using *exit* or [CTRL]-[D]) and then login again. Generally this will clear up any problems.

Another solution is to type the following **exactly** as shown — blanks are significant. You may not see anything echoed on the screen. If you have a hardwired terminal on an HP 27128 ASI card, this action may log you off of HP-UX.

```
[CTRL]-[J] stty sane erase "^H" kill "^U" echo[CTRL]-[J]
```

This sets the "erase" character to [CTRL]-[H] and the "kill" character to [CTRL]-[U]. When the screen and keyboard response returns, type:

```
tset [Return]
```

Your terminal should now exhibit proper behavior. As another option to this procedure, you may also do a hard reset of your terminal, and then execute the *tabs(1)* command.

Adding/Removing Users

The material in this section covers only the software or configuration aspects of adding or removing a user to/from the HP-UX system. If the user will have his own terminal, you need to install the terminal and do some associated configuration before the user can log in to the system; see the "Adding/Moving Peripheral Devices" section of this chapter.

Each user 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 do a few other things. A complete description of the */etc/passwd* file can be found in the *passwd(5)* entry in the *HP-UX Reference* manual.

Two approaches for adding users to the system are offered here. Both approaches require the aforementioned 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 presented. If you expect to add a few users to the system, it will probably be worth your time to type in the shell script listed below to ease the task of adding users. If you have a single-user system or expect to add only one or two users to your system, the step-by-step process will probably be your best choice. Both approaches accomplish the same task; the "automation" of adding users is the only functional difference between the two.

Creating the /etc/passwd Entry

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

If this is the first time you are following this procedure, make a copy of the original */etc/passwd* that was shipped with your system before continuing; this useful precaution takes only a moment. To copy the file, type:

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

where `/etc/passwd.old` will be your unmodified (original) copy of the file.

Next, using the text editor of your choice (such as *ed*, *vi*, or *ex*), edit the file */etc/passwd.* Add a line to the file describing the new user. The new line must have the form:

user_name::user_id:group_id:comment:login_directory:command

The colon character (:) is used to delimit the various fields in the entry.

user_name is the user's login name, consisting of 1 to 8 lowercase letters, and at least one number and/or special character, and any other characters you desire.

:: represents an empty password field. Passwords and the *passwd* command are discussed later in this section.

user_id is the real user ID — a **unique** integer value that the system uses to identify the user. If the real user ID is 0, then that user has super-user capabilities. As the system was shipped to you, the real user ID 0 is associated with the user `root`. By convention, the values 1 through 99 are reserved for system use. Therefore, pick any unused number greater than 99 for this field.

Note: There should be only one entry per real user ID; the user whose real user ID is 0 should be named root.

group_id is the real group ID — an integer value shared by all members of the same group. This entry corresponds with the group entry in */etc/group*; see the "Creating Groups/Changing Group Membership" section in this chapter for details.

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 his location or phone number.

login_directory is the absolute path name of the user's login directory. This becomes the user's working directory when he logs in. 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's login directory is usually a subdirectory of the */users* directory and has the same name as the user's login name. For example, a user whose last name is Young might have login name `young` and home directory `/users/young`.

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 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. Once the program terminates, the user is logged out.

Once you are satisfied with the contents of */etc/passwd*, write the modified file to the disc and terminate the editing session.

The "Makeuser" Script

This section contains 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 the super-user `root`.
- Use one of the text editors to create the */etc/makeuser* file by typing in the listing below. The name */etc/makeuser* is only a suggestion.
- Change the mode of the file by typing:

  `chmod 744 /etc/makeuser` [Return]

  This gives you read, write and execute permission on 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` *user_name* [Return]

  where *user_name* is the new user's username from the */etc/passwd* entry.

Here is the "makeuser" shell script:

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

USERS=/users
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 decided 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* file.

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

   `mkdir /users/`*user_name* [Return]

   where *user_name* is the new user's name and the entire path (`/users/`*user_name*) matches the `login_directory` field of the user's */etc/passwd* entry.

2. Create *.profile* and *.exrc* files for the user. If the file *.profile* exists in a user's login directory, the shell attempts to execute that file at the end of the login process. As discussed in Chapter 4, this file typically contains shell commands and environment variable definitions which customize the user's environment and/or automatically run one or more programs. If the file *.exrc* (also discussed in Chapter 4) exists in the user's login directory, it is used to map terminal characteristics and key definitions for some of the HP-UX text editors.

3. Because you (the user `root`) created the new user's directory and copied the *.profile* and *.exrc* files into his directory, you own both his directory and his files. To change the ownership to the user, type:

   `chown` *user_name* `/users/`*user_name* [Return]

   (where *user_name* is the user and `/users/`*user_name* is his login directory) to change the directory ownership and type:

   `chown` *user_name* `/users/`*user_name*`/.[a-z]*` [Return]

   (where *user_name* is the user) to change the ownership of the files. The specification `/users/`*user_name*`/.[a-z]*` matches all the files in the user's directory that begin with a period and are followed by a lower-case letter and then anything else.

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

   `ls -ld /users/`*user_name* [Return]

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

   `ls -la /users/`*user_name* [Return]

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

5. If you are using the group access features available on HP-UX, see the "Creating Groups/Changing Group Membership" section in this chapter.

Whether you used the "makeuser" method or the step-by-step method, the new user is now installed on the system. A few optional considerations still bear examination.

Some Optional Items

If you are using HP-UX's group access capability, you may want to add the 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 the *chgrp* command can be used to change the group ID associated with the user's files. A user may only belong to one group at any given time. For details on these operations, refer to the "Creating Groups/Changing Group Membership" section in this chapter and the *chgrp(1)* and *group(5)* entries in the *HP-UX Reference.*

Depending on the needs at 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 0755 which provides read, write, and execute (search) permission for the file's owner while providing only read and execute (search) permission for all others.

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

Setting the New User's Password

The new user does not have a password at this point but may log in without one. Depending on the security needs at your installation and your own inclination, you can:

- Ask the user to create a password for himself.
- Create a password for the user and tell him what it is.
- Force the user to create a password for himself the first time he logs in to the system.

The procedures for the last two choices are supplied below.

Creating a Password for the User

To set a password for the user, first become super-user. Then type:

`passwd` *user_name* [Return]

and respond to the system's prompt for a new password; see the *passwd(1)* entry in the *HP-UX Reference* for details. This will set the new user's password to the password you typed.

Forcing the User to Create a Password

If you neither want to create a password for the user nor leave it up to him to create one, you can set a parameter that forces him to create a password the first time he logs in to the system. To accomplish this requires that, in the user's */etc/passwd* entry, the password field's optional "aging" field contains two periods. The optional aging field is separated from the password field with a comma (see the *passwd(5)* entry for details). Thus a typical entry for a user without a password might be:

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

Removing a User from the System

To remove a user from the system, delete his entries from the */etc/passwd* and */etc/group* files. Then remove the user's */user* directory and his other directories and files to release the disc space for other users. The easiest way to remove the user's files is to type:

rm -r /users/*user_name* [Return]

Note that this **removes all** the user's files and directories.

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 the "System Boot and Login" chapter in this manual and *inittab(5)* in the *HP-UX Reference* for details.

Suspending a User from the System

If, for whatever administrative reason, you wish to keep a user off the system for a time, you can temporarily revoke his login privileges by modifying his line in */etc/passwd*. If his password is replaced with an asterisk (*), he cannot use the system until you delete that asterisk (and then give him a new password). The following example illustrates this:

```
atilla:*:101:5:Atilla the Hun:/users/atilla:/bin/sh
```

Backing up and Restoring the File System

One of your ongoing responsibilities as system administrator is to make periodic backups (copies) of the file system. As mentioned at the end of the "Installing HP-UX" chapter, one of your first tasks (after installing and configuring the system) is to make a backup of the entire HP-UX system. This section contains instructions for accomplishing this task as well as guidelines for backing up the system as part of a regular maintenance program. Backing up the system is an important and useful investment of your time. The backup process provides a means of recovering lost data in the event of a hardware failure, a "crash" of the system, or accidental removal/corruption of a file by a user. Recovery procedures are also detailed in this section.

To minimize the chance of loss, backups should be stored at a different location from the main HP-UX system. "Data safes", specially designed air-tight, water-proof containers for mass storage media, are available from many computer accessory manufacturers. In the event that a file or the entire file system is lost or destroyed, you can recover by restoring the latest version of your system backup.

Backup Strategies and Trade-offs

The method, frequency and extent of the backup operation depends on the amount of use the system receives and the amount of data you feel you can afford to lose. Different strategies exist for dealing with the method and frequency of backups. The ideal method of backup would be to back up the entire system several times a day, keeping each backup permanently. Then, in case one or more files are lost (regardless of when they are discovered missing), you could restore the system to a point just prior to when the loss occurred.

Unfortunately, this ideal borders on the ludicrous due to the time and expense this would incur. Complete backups (as compared with partial ones) take significant time and the media costs involved cannot be ignored. On the other hand, losing days of work because backups were not made is clearly unacceptable. Do not despair; a reasonable compromise exists between these extremes.

Daily Archive Backups

One backup strategy is to make complete backups of the file system on a daily basis. A complete backup is often called an **archive backup**. Using this method, restoring the file system simply consists of restoring the most recent backup 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 minimal.

The HP-UX system supports the ability to make incremental backups as well as archive (complete) backups of the file system. An **incremental backup** contains only files that have changed since the last backup. Files that have been changed since a given time and date are easily identified with the *find* command; see *find(1)* in the *HP-UX Reference*. Incremental backups almost always require less time and, depending on the size of your disc, less backup media than archive backups.

Mixing Archive and Incremental Backups

The ability to use archive and incremental backups leads to two useful backup strategies. Both of these strategies are cost-effective and time-efficient without sacrificing the integrity of the HP-UX file system. Both methods involve making archive (complete) backups of the file system once per week and supplementing this with daily incremental backups. The difference between the two methods rests on how the incremental backup is defined. The first method defines the incremental backup as containing only files that have changed *since the last complete (archive) file system backup.* The second method defines the incremental backup as containing only files that have changed *since the last incremental backup.*

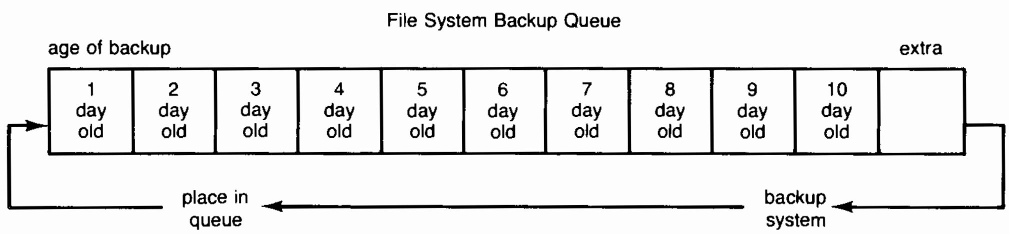
The first method of incremental backup (where files changed since the last archive backup are copied) generally takes longer than the second method (where files changed since the last incremental backup are copied). When restoring the file system from the backups, the first method is quicker than the second one. Conversely, the second method of incremental backup saves time during the backup process but usually takes longer when restoring the file system. This is because the second method builds on each of the previous incremental backups and several tapes are often necessary for full system restoration.

Suppose, for example, that you make a complete backup of the file system on Monday and make the first type of incremental backups on Tuesday and Wednesday. Each incremental backup contains only those files that have changed since Monday. Further assume that, on Thursday, the file system is destroyed. The file system may be reconstructed by first restoring Monday's archival backup of the file system and then restoring the files from Wednesday's (first method) incremental backup. If you had used the second method of incremental backup, restoration of the system would require restoring Monday's archival backup, then Tuesday's incremental backup (second method) and finally, restoring Wednesday's incremental backup.

Choosing a Backup Method

As the system administrator, you should weigh the advantages and disadvantages of these different backup strategies and decide what method and frequency of backup that makes the most sense for your installation. However, you should pick one method of backup and stay with it to lessen the chance of confusion.

Hewlett-Packard recommends that you make daily backups of the file system using whatever method you prefer. **Making an archive (complete) backup of the system either once a week or once every two weeks is strongly recommended**. Store the backup at least two weeks (10 working days). At the end of two weeks, re-cycle the oldest backup media by using it to make another backup. This is facilitated by keeping a queue of backup media where the medium needed for the backup is taken from the rear of the queue and, after the backup is made, the medium is added to the front of the queue. This limits the expense of backup media while providing protection against loss of data during the previous two weeks. Additionally, once every month you should take one of the complete file system backups created above and place it in a permanent archive.



File System Backup Queue

CS/80 Cartridge Tape Drives

HP-UX provides you with the capability of directly accessing the tape just as you would any other mass storage device. The CS/80 or SS/80 tape drive on which your system was installed is associated with two special (device) files which were assigned to it during the installation process. These files are */dev/ct*, used for "regular" access and */dev/rct*, used for "raw" access. Although the tape may be directly accessed or addressed, **using the tape as a mounted file system is NOT recommended**. If the tape is used as a mounted file system, overall performance of the file system will suffer due to the serial characteristics of the tape. More importantly, because the tape drive was not designed for constant file system type access, using the tape as a mounted file system will put tremendous wear and stress on the tape drive unit.

However, the tape is well suited for transporting files and data to other Series 500 HP-UX systems, for complete backup/retrieval and for selective backup/retrieval. When using the tape for selective retrieval of files, remember that a streaming tape drive has inherently slower seek times than a hard disc; it may take some time to find the files you are trying to restore.

Two lengths of tape are available for the following CS/80 disc/tape drives supported by Series 500 HP-UX: HP 7908, HP 7911, HP 7912, HP 7914, and HP 9144. The 150 ft. tape (HP 88140S) holds 17 Mbytes and the 600 ft. tape (HP 88140L) holds 67 Mbytes. Both tapes are often referred to as "DC600" tapes. DC600 indicates tape width **not** tape length and does not signify a 600 ft. tape. Tape lengths are designated by the "S" (for Short) and "L" (for Long) in the part numbers shown above.

Backing Up the System

The HP-UX system can be backed up by:

- Using the media utilities supplied with your system.
- Using the *backup(1M)* command supplied with your system.
- Interactively using the *cpio(1)* or *tar(1)* commands to copy selected files to a backup volume.
- Writing, then running your own backup program or shell script using *cpio(1)*, *tcio(1)*, *tar(1)*, or *cp(1)*. The *backup(1M)* shell script/command can serve as a model.

The following documentation describes how to use the *backup* command and modify the command for your own needs. The *backup(1M)* entry in the *HP-UX Reference* provides further information on this command. To use the other backup methods listed above, see the "Media Utilities" section in this chapter or the appropriate entries in the *HP-UX Reference* for details.

Customizing the Backup Script

The HP-UX *backup* command is in the form of a shell script. This script or command provides both incremental backup (using no options; the default) and complete/archive backup (using the `-archive` option). Once you have chosen a backup method based on the discussion earlier in this section, you will need to modify the */etc/backup* shell script to suit the particular needs of your installation.

As */etc/backup* is distributed, it will backup all files onto a CS/80 cartridge tape. If this is not what you desire, you must modify */etc/backup*.

NOTE

Before using the *backup* command, make a copy of the original file, then edit the "working version" of *backup* to customize it for your installation. Instructions for performing these tasks are provided below.

Before modifying */etc/backup* make a copy of it by typing:

```
cp /etc/backup /etc/backup.old [Return]
```

Now that you have saved the original version of the command (the file `/etc/backup.old`), you can use any of the HP-UX text editors to edit the "working version" of the script (the file */etc/backup*) and customize it for your needs. Be certain that the changes you make do not defeat the script. The commented script itself is a good source of information. Also, see the *backup(1M)* entry in the *HP-UX Reference* and use the following guidelines to modify the *backup* script:

- Modify the script's variable assignments as needed. Note that the `backupdirs` variable begins the backup starting at the directory / (in other words, the entire file system). If you want to back up just the user files, assign the directory */users* to the `backupdirs` variable.

- The `backuplog` variable assigns the standard error device file to which errors and other useful information is written during the backup process. Consider re-assigning this variable either to a printer special (device) file or to the system console's special file (both special files are supplied with your system as */dev/lp* and */dev/console* respectively). Sending *backup*'s messages to an external device rather than to a file allows you to monitor the backup process.

- The `outdev` variable signifies the character special (device) file (in raw mode) associated with the backup device. On the Series 500 HP-UX system, the backup device commonly will be the built-in cartridge tape drive of the CS/80 disc/tape device on which your system was installed. As the system was shipped to you, the character special file associated with the "raw" cartridge tape drive is */dev/rct.*

- If you are using the incremental backup method that copies only the files that have changed since the last incremental backup, you need to add the variable `incr` to the *backup* script. The variable should be assigned to the file */etc/incrdate.* (The suggested names are arbitrary, but reasonable.) Look for all the occurrences of the `archive` variable in the script and emulate that code for the `incr` variable. This includes such items as modifying the section that parses arguments and adding a *touch* command to update the */etc/incrdate*file. When you are finished, remember to create the file */etc/incrdate* on your file system so these modifications to the *backup* script will run properly.

- If */etc/backup* is such that it performs an *fsck*, the *backup* process should occur only when HP-UX is in the single-user state (state s) and when the file system is inactive (not being accessed). You may want to remove the comment signs in front of the commands in the `Warn all users` section of the script. The commands in this section will warn all users currently on the system of the impending backup and then take the system into the single-user state (state s). In order to use the *fsck* option in */etc/backup*, you must have the optionally installed memory volume driver residing in your boot area.

- As an alternative to the preceding item, the `Warn all users` section of the script could contain the *shutdown(1M)* command. This command gives users enough time to complete their tasks before safely taking the system into the single-user state. See the "Shutting Down the System" section in this chapter and the *shutdown(1M)* entry in the *HP-UX Reference* for further information.
- If you want to make the backup data go to a device other than the CS/80 cartridge (such as a "raw" 9-track tape), the script must be modified in two places. First, change the *outdev* variable to point to the special file associated with the device (e.g., */dev/rmt8* for a raw 9-track tape drive). Also, change the script such that output is **not** piped through *tcio*:
  - use *tcio cpio* for a cartridge tape.
  - use *cpio* for 9-track magnetic tape.
  - use *tar* for a quick and dirty backup to either cartridge or 9-track magnetic tape.

Once you are satisfied with the changes you have made to the script, save the new version as */etc/backup* and exit the text editor. As you learn more about the HP-UX system or the backup needs of your installation change, you can always re-modify the file. Be certain that both the original and working versions of this file have appropriate access permissions; only the super-user should have write or execute permission on this file.

A Standard Backup Procedure

The following is a step-by-step method for using the *backup* command for both archive and incremental backup of the file system. Steps 1 through 3, 6, and 7 should be followed for both archive and incremental backups.

1. Log in as the super-user `root`. If you are not the super-user, you will have problems copying files that you do not own or have permission to access. When restoring files, the ownership of the files is properly restored when the backup is restored.
2. The HP-UX system should be in the single-user state (state s) and the file system should be quiescent (inactive) when the *backup* script is run. This will assure that a complete "snapshot" has been taken. Therefore, run the *shutdown(1M)* command by following the procedure in the "Shutting Down the System" section of this chapter. The *shutdown(1M)* command warns all users that the system is going to be shut down, forces the contents of the file system's I/O buffers to be written to the disc (with the *sync(1)* command) and takes the system into the single-user state. See *shutdown(1M)* in the *HP-UX Reference* for further details.

3. Next, run the file system check program (*fsck(1M)*) to insure the integrity of the HP-UX file system. See Appendix A, "Using the FSCK Command" in this manual and *fsck(1M)* in the *HP-UX Reference* for instructions on using the command.

4. Now create a complete (archive) file system backup by putting a tape in the backup drive, and then typing:

   `/etc/backup -archive` Return

   This copies the entire file system to the backup device's special (device) file assigned to the script's **`outdev`** parameter. The file */etc/archivedate* is "touched" with the *touch(1)* command, effectively storing the time and date of this backup. Make sure that you are in the correct directory (for a full system archive, you should **`cd /`**).

   Unless modified, the *backup* script writes (logs) the following information to the file */etc/backuplog*: the start and finish times of the backup, the number of blocks copied, and any error messages that may have occurred during the backup. Information and messages written to this log file are appended to the existing file.

   If at any time, the *backup* command runs out of room on the backup medium, it prompts you to insert a new medium.

   Next, go to step 6.

5. To create an incremental backup, follow steps 1 through 3 above. After completing those steps, type:

   `/etc/backup` Return

   backup determines the date of the last archive by examining */etc/archivedate* (or the date of the last incremental backup by checking */etc/incrdate* if modified as discussed) and backs up all files that have been modified since that time and date.

   During an incremental backup, the *backup* script logs the same type of information to the */etc/backuplog* file (or elsewhere if modified) as before.

   Again, if the *backup* command runs out of room on the backup medium, it prompts you to insert a new medium.

   If you choose not to archive the file system either weekly or bi-weekly as recommended and are performing incremental backups, the information in the file */etc/backuplog* (elsewhere if modified) will help you determine when a new archive is necessary. Examine the sizes (the number of blocks copied) of recent incremental backups to estimate when a new archive will be needed.

6. When the backup has finished, the backup medium may be removed. If you are using the cartridge tape drive of a CS/80 mass storage device as the backup device, be certain that the cartridge tape is logically "unloaded" **before physically removing it from the tape drive**; see the "Special Considerations" material later in this section. Label the backup medium with the date and the type of backup (archive or incremental) before storing it in a secure place.

7. One minor task remains: examine the information and messages sent to the standard error device file associated with *backup*'s `backuplog` variable (the file */etc/backuplog* if unmodified) to determine if any errors occurred during the backup process.

Depending on the methods you have chosen, continue making/using incremental backups:

- until one or two weeks have passed since the last archive backup (if you are maintaining an archival schedule).
- until their size becomes unwieldy (for example, larger than one tape).
- until you otherwise feel it is necessary to create a new archive backup.

Restoring the System

A number of things should be done before attempting to restore a backup of the HP-UX system. If the system is still functioning, log in as the super-user `root` and run the *shutdown* and *fsck* commands as referenced in steps 2 and 3 under the heading "A Standard Backup Procedure" above. In many cases, the *fsck* command can repair even serious problems in the file system. Try that first.

If you are trying to recover one or more specific files (but not the entire system) from a backup, you should still be logged in as the super-user `root` and run the *shutdown* and *fsck* commands before restoring the files. This helps insure the integrity of the file system. To recover specific files, read the next few paragraphs (up to the "If Disaster Strikes" heading) and then follow the guidelines and examples under the *find(1)*, *cpio(1)* and *tcio(1)* entries in the *HP-UX Reference* manual.

Finally, before restoring a backup, you must know how the backup was made. When you customized the *backup* shell script, you specified how backups would be created: with only the *cpio* command or with the output of the *cpio* command piped to the *tcio* command.

If the backup was created with only the ***cpio*** **command**, restore the backup using the *cpio* command and its `-i` (copy in), `-c`, and `-B` options. Include the `-x` option if you want to restore special files and include the `-v` (verbose) option for additional information during the restoration process.

If the backup was created using both ***cpio*** **and** ***tcio***, restore the file system by piping the output of `tcio -i` (where the `-i` option is for copy in) to `cpio -ic`. Again, include the `-x` option with *cpio* if you want to restore special files and include the `-v` (verbose) option with *cpio* for additional information during the restoration process.

To restore the file system (you **must** be logged in as the super-user to restore an entire file system, but not just to restore a few files), place the backup medium in the mass storage device. If you are using a CS/80 cartridge tape as the backup medium, **wait for the cartridge tape drive's conditioning sequence to complete** before continuing with this process. Once the medium is ready, enter one of the following command forms. If your files were stored relative to a directory (which is the default), you must reside in the correct parent directory before restoring files. Select the appropriate form of the command based on the information supplied above.

`cpio -icxvB` *special_file*

or

`tcio -i` *raw_special_file* `| cpio -icxv`

Note that *special_file* (or *raw_special_file*) is the name of the character special file (or "raw" mode character special file) associated with the backup device.

If Disaster Strikes

If the entire file system is destroyed or if the system is in such poor shape that the *cpio* command will not function properly, then one of two options is available, depending on the type of archive backups you have made. If the CS/80 Tape Backup Utility was used to make an "image" backup of the file system, the system can be restored directly from the backup by using the same utility.

If you have been backing up the system using the *cpio* and/or *tcio* commands and the file system is in tragic shape, the system must be re-installed from the original HP-UX distribution medium before the file system can be restored. Follow the instructions in the "Installing HP-UX" chapter of this manual and install the system using the distribution tape labeled "Series 500 HP-UX Installation, Tape 1".

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 even begin restoring files.

Once you have re-installed the original system and it is operating properly, use the forms of the *cpio* and *tcio* commands given above to copy (restore) the most recent archive and incremental backup(s) from the backup device to the system's root device. Then perform an *fsck* to verify that the file system is correct.

NOTE

If you are recovering from a total loss of the file system, be careful when restoring the most recent backups. Because the backup is a copy of the system, it may contain the same error that caused the system to "crash" in the first place.

Special Considerations

One of the characteristics of incremental backups is that they depend heavily on the system clock. Both the current time and date as well as the time and date associated with the file being used as a reference point for the backup (such as */etc/archivedate* or */etc/incrdate*) have to be reasonably accurate to insure useful incremental backups. Always reset the clock (using the *date(1)* command) if the system has been powered down for any reason or if a check of the clock shows any appreciable amount of inaccuracy. See the "Setting the System Clock" section later in this chapter for further information.

Dealing with Backup Media

If the end of the medium is reached during the backup process, a message is sent to the standard error device by the *tcio* command. When this occurs, change the backup medium and type:

special_file_name `Return`

where *special_file_name* is the special (device) file name associated with the backup device. The backup process will then continue.

The CS/80 Cartridge Tape Drive as a Backup Device

If you are using a CS/80 cartridge tape drive for the backup device and the end of the backup medium is reached before the backup operation is complete, the cartridge tape will be logically "unloaded" by the system. (Cartridge tapes can be unloaded manually by pressing the tape drive's UNLOAD button.) While unloading a tape, the cartridge tape drive's BUSY indicator will be lit. **Always wait for the CS/80 cartridge tape drive's BUSY indicator to go out before physically removing a cartridge tape from the drive.**

To finish the backup process, remove the tape from the cartridge tape drive and insert another cartridge tape into the drive. The BUSY indicator will remain lit while the tape is conditioned for use. **After** the BUSY light goes out, type:

special_file_name `Return`

where *special_file_name* is the special file name associated with the CS/80 cartridge tape drive being used as a backup device. The backup process will then continue.

Using Cron for Incremental Backups

Incremental backups can be performed automatically by using the *crontab* command to schedule *backup* (where it gets executed by the *cron* "clock daemon"); see the *cron(1)* and *crontab(1)* entries in the *HP-UX Reference*. Placing the following entry in the file */usr/lib/cron.root*, and executing the command `crontab /usr/lib/cron.root` causes an incremental backup to be performed (at 11:55 pm) every weeknight:

```
55 23 * * 1-5 /etc/backup
```

The *fsck* command should only be run interactively because it may not handle certain situations without human intervention. Also, before performing even incremental backups, it is a good idea to first execute *fsck* to either confirm or restore the file system's integrity.

However, you can run *fsck* at night by adding the `-fsck` parameter to the *backup* command, placed in a *crontab* file. As in our previous example, the entry would now look like:

```
55 23 * * 1-5 /etc/backup -fsck
```

Despite these cautions, you should be able to do automatic incremental backups (using *cron*) **IF** you run *fsck* as either the first task of the morning, the last task of the evening or both. If you use this method of performing incremental backups, you should also:

1. Assign the *backup* script's error output device to a special file associated with a printer.
2. Modify the *backup* script to include the *shutdown* command (to do cleanup and take the system into the single-user state) **before** the actual backup is performed. Also, add the command `init 2` (to return the system to its normal state) **after** the backup is performed.
3. Every morning:
   a. Examine the information and messages listed to the printer during the previous night's backup process.
   b. Remove, label and store the backup medium.
   c. Run the *fsck* command.
4. Every evening:
   a. Be certain that the printer associated with the *backup* script's error output device is on-line.
   b. Install a blank backup medium.
   c. Run the *fsck* command.

System Restoration and Shared Files

In general, the system cannot write to a busy file. A file (pure Series 500 executable code) is "busy" if the file is open or is marked as shared and being executed with *exec*. (Shared files are discussed in the "Concepts" chapter.) Thus, files being used during a system restoration — particularly */bin/cpio* — should not be shared files. If they are marked as shared, they may not be recovered from the backup.

Booting HP-UX

Booting HP-UX (bringing up the system) is discussed in detail in Chapter 4 of this manual. This section provides some reminders and a few additional suggestions to supplement that material.

- If for any reason, the boot ROM is unable to find and load an operating system, informational and/or error messages are displayed on the system console. These messages are described in the "Boot ROM Error Messages" section of the installation guide supplied with your HP-UX system.

- Remember that the mass storage device containing the HP-UX system must be powered up and have achieved a "ready" state **before** powering up your Series 500 computer. If the disc drive has not completed its power-up sequence (which may require several minutes on some disc drives), the boot ROM will not be able to access the disc and load the system.

- **The *fsck* command should be executed EVERY time the system is powered up.** The file system consistency check program, (*fsck*) is vital to the maintenance of your file system. If the file system becomes corrupt (whatever the cause), continuing to use the corrupted file system invites certain disaster. **The file system must be quiescent (inactive)** for *fsck* to work properly. Therefore, manually execute this command **before** entering the multi-user state and **before** allowing other users to access the system. Refer to Appendix A ("Using the FSCK Command") in this manual for details on checking the file system.

- One other caution concerning *fsck*: **you should not use the *crontab* command and the *cron* routine to run this command**. Always execute *fsck* interactively because, even when run with the `-n` option (which effectively tells *fsck* "do not make any changes, just examine the file system and report any errors"), *fsck* may encounter situations that it cannot handle. Take the time to examine the file system on a regular basis. It will pay off.

On single-user systems or small multi-user systems, it may be useful to allow any user to power up the system. If this applies to your system, write a short document that describes the procedure for booting HP-UX (and changing system states if it applies) and distribute the document to all users. Knowing the specific details of your system — the hardware, configuration files, system states, and needs at your installation — should enable you to write a streamlined procedure for your users. This can ease your administration tasks and provide system users with more flexibility.

Changing a Password

The "Adding/Removing Users" section in this chapter discusses creating passwords for users. This section discusses how either a user or the system administrator may change passwords.

Any regular user on the system may change his own (but no one else's) password by typing:

`passwd` [Return]

The *passwd(1)* 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 7 characters and/or digits of your choosing followed by at least one numeric and/or special character, and [Return]. Actually, fewer characters may be used but they must be entered three times before the system will accept them (this is only true if you are super-user). Note that control characters like those generated by [Backspace] are accepted but sometimes difficult to remember. The password is not echoed on the screen (for security purposes). The command then prompts you to re-enter 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 will be prompted to enter it twice again. It takes approximately 15 seconds for the system to install the password.

Users will occasionally create passwords for themselves which they cannot remember. Once a user has forgotten his password, he cannot log in to the system and will probably come to you, the system administrator, for help. Because only the encrypted form of the password exists in */etc/passwd*, even you cannot determine the user's password, hence you must assign the user a new password.

To change a user's password, become super-user and type:

`passwd` *user_name* [Return]

where *user_name* is the user's login name. You will be prompted for the user's new password. Only the super user may use this method for changing a password.

To protect the security of the system, */etc/passwd* should be owned by `root` (the super-user) and no one should have write permission to the file. Not even the super-user. If you, as the super-user, want to modify */etc/passwd*, temporarily change the permission using *chmod(1)*, modify */etc/passwd*, then change the permission of */etc/passwd* back. Actually, non-super users should not be allowed to write to **any** of the files contained in the */etc* directory.

If you change the password for the user `root`, you may want to write down the password and keep it in a secure place. If this password is lost or forgotten, no one can log in as the super-user and a complete re-installation of the system may be required to recover (from the lost super-user password). **Note that a complete re-installation of the system will destroy all the files on the disc.** If this ever occurs, either re-install your system (as described in Chapter 2 of this manual) or call your HP System Engineer for assistance.

Changing and Creating System States

One of the tools you need as the system administrator is the ability to move properly from one system state to another. Also, you may find it useful to create new system states 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 state of the system and creating new system states.

As discussed in the "System Boot and Login" chapter in this manual, the system administrator (or anyone with the `root` user capabilities) may change the system's state by executing the *init(1M)* command. Also, anyone having write permission to the file */etc/inittab* can create new states or re-define existing states. Even if this user lacks the capability to invoke *init* to enter a modified state, his ability to re-define existing states in */etc/inittab* could wreak havoc upon your system. Make sure that the aforementioned permissions are correct.

If you purchased a single-user system, the system was shipped to you such that state s uses only the system console and state 2 uses both the system console and a single *uucp* modem (for data communication with other HP-UX and/or UNIX systems). You can create other states on this system but you cannot add any other users. Adding *getty* entries to state 2 on a single-user system has no effect other than to waste a large amount of processor time. If yours is a single-user system, note that much of the following material applies primarily to multi-user systems. It should be clear from the context which of the discussed features apply to single-user systems.

Changing the System's State

Many of the system maintenance tasks you perform as system administrator require the system to be in the single-user state. When the HP-UX system is first booted/powered up, it comes up in the single-user state (state s). The single-user state may also be entered by moving there (with the `init` command) from another state. These recommended entries are described in more detail in the "System Boot and Login" chapter and in the "Adding/Removing Users" section of this chapter.

Entering the single-user state insures that the only access to the system is through the system console by the user `root`. At this point, the only processes running on the system will be the shell on the system console, and processes that you invoke. The result of all this is that commands requiring an inactive system can now be run. One such command is *fsck* whose multiple passes and use of redundant information in the file system require a static system.

The following is a general procedure for changing the system from one state to another. You must be logged in as the super-user to change the system's state.

1. If any users are on the system, ask them to log off before you change states. Changing to another state while users are logged on will kill (terminate) their processes if the state you are moving to does not contain explicit rstate entries in */etc/inittab* for their *getty*. Use the *write(1)* or *wall(1)* 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 the differences in the new and old states are minor enough (for example, the addition of a printer), it may not be necessary to ask users to log off.

2. After users are off the system as necessary, force the system to write the contents of its I/O buffers to the file system by typing:

   `sync` `Return`

   It is a good idea to repeat the *sync(1)* command a second time. This is recommended because, after *sync(1)* executes the file system primitive *sync(2)* and returns, the writing of the I/O buffers has been scheduled but may not have yet taken place.

3. Next, change to the desired state by typing:

   `/etc/init` *new_state* [Return]

   where *new_state* is the number of the state you wish to enter. It is assumed that the state you are entering contains appropriate */etc/inittab* entries to kill any outstanding user processes, if those processes should not be present in the new state.

The *shutdown(1M)* command interactively performs a set of tasks similar to — but more extensive than — those above and then enters the single-user state. The *shutdown* command also allows you to specify how much time will elapse between notifying users of the impending shutdown (with *wall*) and the time all user processes on the system will be terminated. See the *shutdown(1M)* entry in the *HP-UX Reference* or the "Shutting Down the System" section in this chapter for a more complete description of *shutdown*.

When taking the system from the multi-user state (state 2) to the single-user state (state s), **use the *shutdown* command instead of `init s` to change the system's state**. The *shutdown* command performs certain operations not otherwise performed.

Also, be aware that a *cron* process is initiated only the first time the system enters state 2.

Creating New System States

You can create new states if you find it useful but **do not re-define state 2**. It is acceptable to make certain, suggested modifications to state 2 (such as the addition of more *getty* entries to */etc/inittab* previously mentioned). Before creating a new state, make a copy of the original */etc/inittab* file (using the *cp(1)* command) and save the original version of the file under a different name (such as `/etc/orig_inittab`). If anything goes wrong, you will still have a relatively untouched version of the file.

To create new states, use one of the HP-UX text editors to make entries in */etc/inittab*. These entries will define how you want the system to operate in its new state. Each one line entry in */etc/inittab* should contain:

- a two-character id used to identify a process or process group;
- a list of run states to which each entry applies;
- an action to be performed, such as `respawn`;
- the command that will be executed when that state is entered.

Refer to *init(1M)* and *inittab(5)* in the *HP-UX Reference* for a more complete description of *inittab*'s state entries. Once */etc/inittab* contains all of the entries you want for the new state, 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's State" section above **before you change states**.

In a few cases, such as when a newly-created state closely matches an existing state (i.e., the differences between the two are trivial), you can move freely between states as long as entering the new state does not kill (user or system) processes that may have begun in the previous state. If the new state 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 states (from say, state 2 to state 4) and when the user attempts to log back in, he cannot because an */etc/getty* entry does not exist for him in your state 4 definition.

Whenever the system enters the newly-defined or created state, its actions are similar to those described in the "System Boot" section (of the "System Boot and Login" chapter in this manual) except that the commands executed are those identified by the new state number. And of course some files, such as */etc/rc*, have no entries for states other than state 2. The system boot and login processes occur more or less as described (in that same chapter).

Example /etc/inittab

The following is an example */etc/inittab* for a system that contains a system console and six terminals. State s is single-user state. State 2 is a multi-user state, with a *getty* on every terminal. State 3 is a test state, with a *getty* on both the system console and the system administrator's terminal (*/dev/tty01*) and "kill" entries for the other terminals. This state could be used by a system administrator who preferred to work from his own terminal rather than from the system console.

```
is:s :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 H
01:23:respawn:/etc/getty tty01   H
02:2 :respawn:/etc/getty tty02   H
03:2 :respawn:/etc/getty tty03   H
04:2 :respawn:/etc/getty tty04   3
05:2 :respawn:/etc/getty tty05   M
```

Communicating with System Users

The */usr/news* directory and the *news(1)* command provide a way to get brief announcements to the system users. The files */etc/profile* and */etc/csh.login*, unless modified, notifies each user at login that news exists if */usr/news* contains entries.

More pressing items (such as announcing an upcoming archival backup) can be entered in the message-of-the-day file, */etc/motd.* Unless modified, */etc/profile* and */etc/csh.login* print the contents of */etc/motd* on a user's display during login. Keep these messages short enough to easily fit on the user's screen.

Longer messages or even major documents are best sent with the *mail(1)* command. Any user can send a message to any other user with *mail.* When activated to send a message, the *mail* command takes the message, flags it with the recipient's user name and stores it in a file. The next time the recipient executes *mail*, the command informs him that he has mail. Most users keep some form of the *mail* command in their *.profile* file so it will automatically be executed during login. This is done by default when */etc/profile* and/or */etc/csh.login* execute.

To write to users who are already logged in, use the *write(1)* or *wall(1)* commands. Note that if a user has executed the *mesg(1)* command with the `-n` (no) option, write permission to that user's terminal is denied and the *write* command will not work. In this case, use the *wall* command.

When the *wall* (write all) command is run by the super-user, any 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 super-user, 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 he is editing a file.

Configuring HP-UX

Series 500 HP-UX allows you to configure or customize various attributes of the system. Many of these items are contained in the configuration files discussed in the "System Boot and Login" chapter of this manual and are not repeated here. Follow the suggestions in that chapter and suggestions made throughout the procedures in this chapter to modify the relevant files and shell scripts for the needs of your installation.

The file system **should** be quiescent, in a single-user state (state s), and have an */etc/fsck* command performed before any of the following operations are attempted.

HP-UX allows certain operating system related attributes to be configured. These attributes determine the manner in which the system allocates memory, schedules processes, and performs file I/O— and thus, the performance of your system. The attributes that you can specify (configure) are:

- **virtual memory device**—the mass storage device that is used by virtual memory. The default is (0 0 0 0 0), which specifies that the default is the root device.
- **cache buffer size**—the size (in bytes) of an individual buffer in the file system buffer cache. The size of the cache buffer is optimized if it corresponds to the block size on your disc volume. The default is 1024 bytes.
- **number of cache buffers**—the number of buffers that form the file system buffer cache. The default is 0, telling the system to dynamically compute the value.
- **read ahead level**—the number of "buffers full" of information that is read into the buffer cache, when a sequential data access is detected. The default is 0, telling the system to dynamically compute the value.
- **interactive time**—the amount of CPU time a process can consume after it has achieved a high priority by accepting input from a keyboard, and still be favored as an interactive process. If the process receives no further input for a time equal to this parameter's value, then the process is no longer favored as being interactive. The interactive time is measured in "ticks", where one tick equals 10 milliseconds. 300 ticks is the default value.
- **swap time**—the criterion used by the virtual memory system when determining which segment(s) to swap out of memory. When a memory block is needed by the virtual memory system, it examines all virtual segments in memory. Any virtual segment that has been in memory longer than the "swap time" is eligible to be swapped out. If no virtual segment has been in memory longer than the "swap time", the system swaps out the least recently accessed virtual segment. The swap time is measured in ticks, with 0 telling the system to dynamically compute the default.

- **page size**—the size (in bytes) of each page, when data is paged in the virtual memory system. The default page size is 1024 bytes.
- **page swap time**—the time a page remains memory resident before being swapped to disc. The page swap time is measured in ticks, with 0 telling the system to dynamically compute the default.
- **page frame pool size**—the maximum size (in bytes) of the memory allocated to the page frame pool (no matter how many memory blocks are free), for the virtual memory system. If the default value is 0, a new value is calculated to optimize the page frame pool. This value then becomes the default. When you ask for default values, the system sets the default back to 0.
- **display memory size**—the size, in pages (one page of display memory equals 24 displayed lines) of the display memory of the Model 520's display. On the HP 98700 ITE, one page is 46 lines of 120 characters. Two pages (48 lines) are the default.

  When HP-UX is installed on a Model 520, the computer's display and keyboard act as a "terminal". As a terminal, it needs memory to hold the information to be displayed. Increasing the size of the display memory increases the amount of data that can be scrolled on the display. Each page of display memory allocated reduces the amount of memory available to the system by about 6 Kbytes.

  If, at power-up, the system does not have enough memory available to allocate the requested amount of display memory, the system notifies you and allocates one page of display memory.

  The primary status of the Model 520 "terminal" identifies the amount of display memory in the "terminal". In the terminal's primary status byte, each page of display memory allocated appears to be 2 Kbytes, instead of 6 Kbytes, as indicated in the preceeding paragraph.

NOTE

The display memory size attribute is ignored when HP-UX is running on a Model 530 or 540 computer.

- **local time zone**—the number of minutes between your local time and Greenwich Mean Time. This is the one configurable attribute that does not deal directly with memory allocation in your system. The default is 300 minutes (Eastern Time Zone).

- **stack size**—the maximum size (in bytes) of a process' stack segment. The default of 0 tells the system to dynamically compute this value.
- **working set ratio**—a decimal value (representing a percentage), specifying the minimum number of virtual pages that are guaranteed to any process in memory.

  Suppose that you are running a program that, if completely loaded into the system, required 1000 pages to be allocated to your partition. Also suppose that the working set ratio is 0.04. This guarantees that 0.04×1000 pages, or 40 pages, are mapped to your process' partition. If more pages are available in the page frame pool, they may be mapped to your process' partition as well. However, when pages are needed by another partition (and none are available in the page frame pool), you are guaranteed that your process' partition will have at last 4% of its maximum need (40 pages). The default is 0.002 (0.2%).
- **process-per-user maximum**—an integer specifying the maximum number of processes a single user can have. The default is 500 processes.

The system is configured with the *uconfig(1M)* command. This command writes the new configuration values to the specified boot area of the boot device (the mass storage device from which the system can be booted). This means that **to see the effect of the new configuration, you must shut the system down and re-boot** (thus loading the kernel and the new configuration values from the boot area).

The operating system being used to modify configuration parameters in a boot area must be the same version as the operating system residing in the boot area being modified.

To configure the system (with other than the default configuration attribute values), you must create a text file. Each line in the text file must have the form:

```
id  value  [# comment]
```

where `id` is a fixed name identifying the attribute you wish to change, and `value` is the new value of the attribute. You can attach comments to any line in the file (and thus make a note to yourself of the reason for the value change) by entering the comment after a `#` character.

A text file containing all configurable parameters and their current values, can be easily created by redirecting the output of *uconfig* with no arguments:

```
uconfig > this.config
```

This file can then be edited to contain the desired parameter values. Once the file is created, it can be copied to the boot area of a device with the `uconfig` - command. See *uconfig(1M)* in the *HP-UX Reference* manual.

Setting the Default Configuration Parameters

The *HP-UX Reference* manual describes each id and their range of acceptable values for each attribute. Should you accidentally set values that reduce the performance of your system, you can restore the system's default configuration values by simply entering (where `boot_device` is the name of the special file of the boot device):

```
uconfig -d <boot_device> Return
```

Controlling Disc Use

As system administrator, you should keep track of the amount of disc space available to users and the distribution of free disc space across file systems. The following procedure will help you evaluate your disc use and identify future disc needs.

1. The *du(1)* command should be executed regularly (weekly or bi-weekly) and the output kept in an accessible file for later comparison. This method lets you spot users who are rapidly increasing their disc usage.
2. Use the *find(1)* command to locate large or inactive files. For example, the following entry records in the file `aging_files` the names of files neither written nor accessed in the last 90 days:

   ```
   find / -mtime +90 -atime +90 -print > aging_files
   ```

3. Use the *df(1)* command to list the amount of free disc space on a volume.

Some files, if present, are written to automatically by certain HP-UX commands (to monitor system use and for general house-keeping). Some files are created automatically by the commands that require them. Both of these sorts of files are called logging files. If not periodically checked and cleared, these files simply continue to grow.

Here are some typical logging files:

- */etc/utmp* (binary) - current login status.
- */etc/wtmp* (binary) - history of logins, logouts and date changes.

- */etc/mnttab* (binary) - mounted devices (not the official list kept by the system in the kernel).
- */usr/adm/sulog* (ASCII) - history of use of the *su* command.
- */usr/lib/cron/log* (ASCII) - history of actions by *cron.*

The only way to avoid logging is to link log files to */dev/null.*

Creating File Systems

The HP-UX system provides a command, *sdfinit(1M)* which "makes" (creates) a file system (structured directory format volume).

Follow these steps to create a file system for the Series 500 HP-UX system:

1. Connect the mass storage device on which the file system will exist to your HP-UX system. See the "Adding/Moving Peripheral Devices" section earlier in this chapter and the installation manuals supplied with your Series 500 computer and/or the mass storage device for hardware installation details.
2. If a special (device) file does not exist for the mass storage medium, create one using the *mkdev* script or the *mknod(8)* command and the directions in the "Adding/Moving Peripheral Devices" section of this chapter.
3. Once the mass storage device is properly installed, run the media initialization utility *sdfinit(1M).* This utility initializes the medium on which the file system will reside. If the medium has been initialized before, you may skip this step. The format is:

   ```
   /etc/sdfinit /dev/fname blocksize bootsize interleave
   ```

   where */dev/fname* is special file describing the new media, and *blocksize, bootsize,* and *interleave* are optional arguments. For more on these arguments, see Chapter 3 (Concepts) of this manual or the *HP-UX Reference* manual.
4. Next, mount the new file system. This is described in the section "Mounting and Unmounting Volumes". The basic steps are:

   ```
   # create a directory on which to mount the new file system
     mkdir /mount_directory

   # mount the new file system onto '/mount_directory'
     mount /dev/fname /mount_directory
   ```

where */dev/fname* is the name of the special file associated with the mass storage medium.

7. Create the lost+found directory. The program *fsck* requires that a directory *lost+found* be present on every file system, so that *fsck* can put stray files there. This directory must be created, then it must have some empty entries in it. This is accomplished by creating many files (100 will do) and then removing them. This creates "slots" that *fsck* can use. Use the following series of commands (where *mount_directory* is the directory on which the file system is mounted as shown above):

```
# create the lost+found directory
  mkdir /mount_directory/lost+found

# go to the lost+found directory
  cd /mount_directory/lost+found

# create approximately 100 files
  touch `(cd /bin; ls)`

# remove the files we just created
  rm lost+found/*
```

8. Add the new file system to */etc/rc.*

   If the newly created file system is intended as a permanent addition, you may wish to modify */etc/rc* so the new file system will be mounted when the system is booted.

Creating Groups/Changing Group Membership

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. The general form of the entry for a group is:

group_name:password:group_id:member1, member2, ..., memberN

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

The *password* field is not currently used. However, to prevent non-group members from switching to this group (with the *newgrp(1)* command), place an asterisk (*) in this field.

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

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

To alter 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.

Note that a user may belong to many different groups. By using the *newgrp(1)* command, the user can change his effective group ID to that of another group. A user's default group is specified by his group ID entry in */etc/passwd.*

Mounting and Unmounting Volumes

When HP-UX is installed, only one file system (the root file system) exists. You may create, modify, and delete files from this system. By default (i.e., installation), the HP-UX file system exists on this one volume and therefore, on one CS/80 or SS/80 mass storage medium. It is possible to have other file systems on **different** volumes; any other mass storage device supported by Series 500 HP-UX can be used as an additional file system. To accomplish this, the additional file system(s) are attached to either the root of the file system or other mounted file systems. The process of attaching additional and functionally independent file systems to the root file system is called **mounting** and is achieved with the *mount* command. The process of removing independent file systems from the root file system is called **unmounting** and is achieved with the *umount* command.

Once a block special (device) file exists for the new disc device, (and the disc is initialized, of course), the volume the device contains can be mounted. See the "Adding/Removing Peripheral Devices" section in this chapter for information on creating special (device) files. The mounting operation makes any files on the new (mounted) volume become part of the file system hierarchy. Files can then be created, modified and deleted on this new volume. When you are finished with the files on that volume, it can be unmounted. Unmounting a volume removes its files from the file system hierarchy. More specifically, the association between the mounted volume and the root of the file system is broken (disconnected). The files themselves are untouched and remain on the mass storage medium; they may be accessed by re-mounting the volume.

It may occur to you that the ability to mount and unmount volumes might be a good way to provide a backup capability for regular users on the system. A user could then keep a copy of important programs or data on a removable medium (such as a flexible disc). Actually, using *cpio* (or the *tar* command) is a superior method of backing up a few files on a flexible disc. See the *cpio(1)* and *tar(1)* entries in the *HP-UX Reference* for details.

To Mount a Volume

The volume to be mounted must have been initialized at some point with the supplied media initialization utility *sdfinit(1M)*. Initializing the volume with this utility insures the correct format for the Series 500 HP-UX file system.

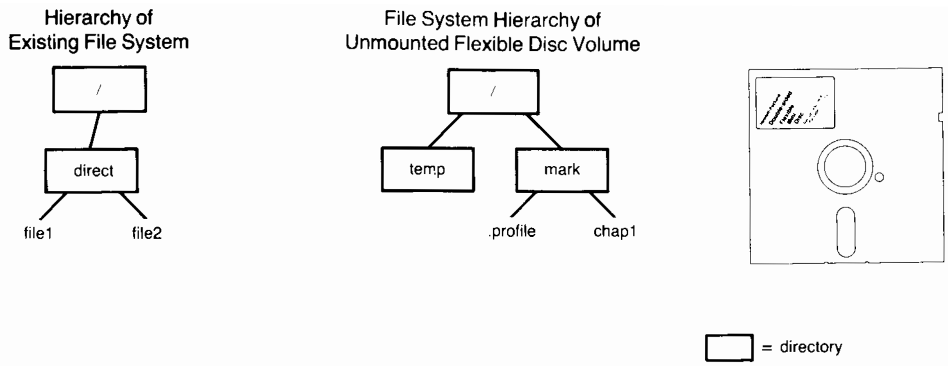
You must create a special (device) file (if one does not yet exist) for the mass storage device containing the volume which is to be mounted. The special file must be a block special file; see the "Adding/Removing Peripheral Devices" section in this chapter for details.

Before attempting to mount a volume, be certain the mass storage device associated with the volume is powered up and is on-line. If the volume is on removable medium (such as a flexible disc), insert the volume in the mass storage device at this time. Do not remove the flexible disc until it is unmounted.

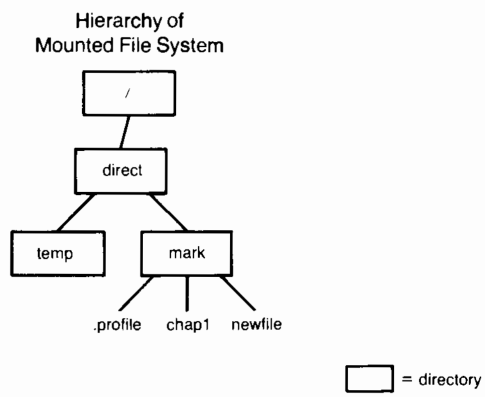
Decide what directory in the HP-UX file system will be used to mount the volume. It is best if you choose a directory that is at the root of the HP-UX file system. That is to say, its complete pathname should consist of only the / character followed by the name of the directory.

When a volume is mounted, the file system it contains is attached to a directory in the existing file system. Any files that the directory previously contained appear to be **temporarily** replaced by the file system contained on the mounted volume. Because of the confusion that may result (from the temporary "disappearance" of files in the mount directory), it is considered intelligent practice to use an empty directory created specifically for mounting.

The diagrams that follow, for the sake of illustration, show a volume mounted on a directory that does contain files but this is **not** standard practice. Consider a directory called */direct* that contains two files, `file1` and `file2`. Assume that you want to mount a flexible disc (that contains a hierarchical file system of its own as shown in the following illustration) on the */direct* directory. The process of mounting the volume, modifying files on that volume, and unmounting the volume is shown in the following illustrations.



The preceding diagram shows the */direct* directory on the existing mounted file system. It also shows the file system hierarchy on the as-yet-unmounted flexible disc volume.

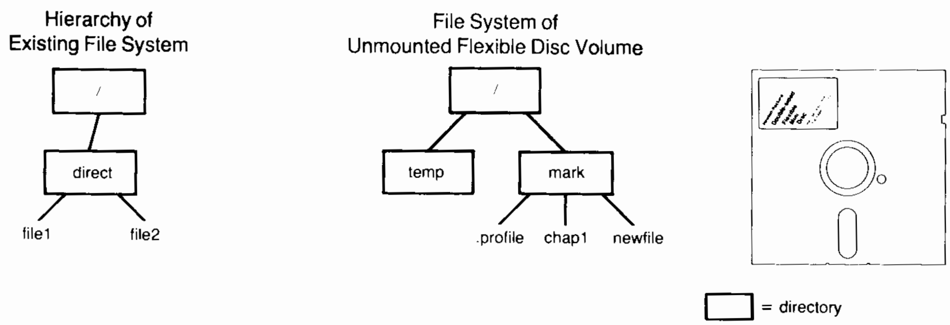


The preceding diagram shows the file system hierarchy once the flexible disc volume is mounted on the */direct* directory. The file `newfile` has been added to the flexible disc volume after it was mounted. Notice that the files `file1` and `file2` previously available in the */direct* directory are no longer accessible; the files are still there, they just cannot be accessed until the flexible disc volume is unmounted.

The flexible disc volume was mounted with a command of the form:

`/etc/mount` */dev/fd_name* `/direct` [Return]

where */dev/fd_name* is the special (device) file associated with the mass storage device containing the flexible disc and `/direct` is the directory on which the flexible disc volume is mounted.



The preceding diagram shows the */direct* directory on the existing mounted file system after the flexible disc volume is unmounted. The `file1` and `file2` files may once again be accessed. The diagram also shows the file system hierarchy on the unmounted flexible disc volume.

The flexible disc volume was unmounted with a command of the form:

`/etc/umount` */dev/fd_name* [Return]

where */dev/fd_name* is the special (device) file associated with the mass storage device containing the flexible disc volume.

NOTE

Always unmount a volume before removing it from its mass storage device. Removing a mounted volume from its mass storage device before unmounting it is likely to corrupt the volume.

To Unmount a Volume

Use the following procedure to unmount a volume.

1. Make sure that all files on the volume are static; no one should be accessing any file on the volume. Attempting to unmount a volume that has open files (including your current working directory) causes the *umount* command to fail without unmounting the volume.
2. Enter the following:

   `/etc/umount` *special_fname* [Return]

   where *special_fname* is the pathname of the special (device) file of the device associated with the mounted volume. This command fails if there are open files on the volume you are attempting to unmount.
3. When the shell prompt (`$`) is again displayed on your screen, it indicates that the volume is unmounted. If the volume is on a removable medium (such as a flexible disc), the medium can now be removed safely.

Special Considerations

You can't unmount a volume that has open files. The following situations are the most frequent cause of open files on a volume:

- having your current working directory on a volume causes an open file on the volume.
- if a program stored on a volume is a shared program and the use count of that program is not zero, the file associated with the program is still open.
- if a file has been accessed and the sticky bit is set in the file's protection mode, the use count of the file is always greater than zero; the file is always open.

Setting the System Clock

Occasionally, the system clock needs to be set or reset. There is no need to reset the system clock if you have shut the system down—your Series 500 computer has a battery which keeps the clock current. The system clock should always have the correct time and date because a number of commands make use of the clock to accomplish their tasks.

Only the effective super-user can change the system clock. To set the current time and date:

1. Login as the super-user `root`.
2. Insure that the time zone environment variable TZ is set properly (for more information refer to *tzset* under the *ctime(3)* entry in the *HP-UX Reference* manual), if your area experiences time zone changes.

   Typically, TZ's value is set with a variable declaration (as shown below) in the file */etc/profile*. It can also be set from an application program with the *tzset* library routine.

   As shipped to you, the system is set up to run in the Mountain Time Zone. To change the time zone to your time zone, modify the */etc/profile* file to contain two entries of the form:

   ```
   TZ=XXXHYYY
   export TZ
   ```

   where `XXX` and `YYY` are three letter representations of the standard and daylight time zones for your area and `H` represents the difference between current local time and Greenwich Mean Time, in hours. The `export TZ` line will remain the same regardless of the time zone. For example, in Denver, Colorado you would enter the following:

   ```
   TZ=MST7MDT
   export TZ
   ```

   where MST stands for Mountain Standard Time and MDT stands for Mountain Daylight Time. Here are some other examples:

   - For St. Clair Shores, Michigan: `TZ=EST5EDT`
   - For Norman, Oklahoma: `TZ=CST6CDT`
   - For Seattle, Washington: `TZ=PST8PDT`
   - For Hawaii: `TZ=HST10` since Hawaii has no Daylight Savings Time.

3. 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} Return
   ```

   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.

When *date* is executed it echoes the time and date on your screen.

Special Considerations

The *make* program (see the *make(1)* entry in the *HP-UX Reference*) is quite sensitive to a file's time and date information and to the current value of the system clock. While setting the clock forward will not effect *make*, **setting the clock backward by even a small amount may cause *make* to exhibit extremely bizarre behavior**. Avoid setting times earlier than the current system clock's value.

As mentioned in the "Backing Up and Restoring the File System" section in this chapter, the process of making incremental backups depends heavily on the correctness of the date. This is because incremental backups are always made in relation to a dated file. This is yet another reason to keep the date correct on your system.

Altering the system clock may also cause some unexpected results for routines scheduled by *cron*; see the *cron(1)* entry in the *HP-UX Reference*. When setting time back, *cron* doesn't 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 (which is **not** advised), *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.

If you are only changing the clock by a small amount (20 - 30 minutes), *cron*'s behavior should not present a problem and no corrective action is necessary. However, if you are changing the clock by a larger increment, perform the following steps:

1. Log in as the super-user.
2. Enter the following command:

   `ps -ef` [Return]

   to run the process status command and locate the *cron* process information supplied by *ps*.
3. Terminate the *cron* process by entering the command:

   `kill -9` *pid* [Return]

   where *pid* is the process id associated with *cron*. The *ps(1)* command displays this number in a column labeled `pid`.
4. Change the time and date with the *date* command.
5. Start the *cron* process running again by entering the command:

   `cron` [Return]

Setting Up the LP Spooler

HP-UX provides a series of commands, collectively referred to as the LP Spooler, to configure and control line printer spooling. Line printer spooling is a mechanism by which printing requests and their associated files get stored temporarily in a spool directory until they can be printed. The LP Spooler can be customized to spool to different printers and allows printers to be grouped into various classes to increase the overall efficiency of the system. Some of the LP commands are available to all users; others, only to the system administrator. The LP Spooler replaces the *lpr(1)* command; *lpr(1)* is a script which involves *lp(1)*, acting like the original line printer spooler. The LP Spooler is a superset of *lpr* capabilities.

This section presents some terminology and an overview of the spooler's operation followed by a brief description of all LP commands: those available to all users and those available only to the super-user (the user `root`) or LP Spooler administrator (the user `lp`). 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 **request** is a combination of one or more files to be printed and all associated information such as destination, number of copies, and other *lp(1)* options. When *lp(1)* is invoked, it associates a unique ID with the request and passes the request to the **LP scheduler** (invoked by the *lpsched(1M)* command). The LP scheduler routes the request to the proper **interface program** to do the actual printing on a device; the program functions as an interface between *lpsched(1M)* 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 models for creating new interface programs.

The *lp(1)* command directs output to the default destination unless a destination is specified when *lp(1)* is executed. The **default destination** may be set or changed by the system administrator. A **destination** is either a printer or a class and a **class** is a name given to a list of printers. Each class must contain at least one printer although a printer may belong to zero, one, or more classes. If the destination is a specific printer, the output gets handled only by that printer. If the destination is a class, the output gets handled by the first available printer belonging to that class.

A complete LP Spooler configuration for a system consists of devices, destinations (printer names and classes), interface programs, and the LP Spooler commands in the *usr/bin* and *usr/lib* directories.

The LP Spooler distinguishes between logical destinations and physical destinations. Logical destinations are defined using the *lpadmin(1M)* command whereas physical destinations are defined using *mknod(1M)* — which associates physical devices with special (device) files. A single physical destination may be associated with one or more logical destinations. *Lp(1)* requests are directed to a logical destination as long as it has been set to accept requests (see *accept(1M)*). When a corresponding physical destination (a printer) is available and has been enabled (see *enable(1)*), the request is transferred to it.

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 User's Guide* or the *HP-UX Reference* manual.

- **cancel(1)** Cancels requests to an LP Spooler line printer made with the *lp(1)* command. The user may address a specific printer or a specific request ID number. See the *lp(1)* entry in the *HP-UX Reference.*
- **disable(1)** Disables one or more physical printers such that they will not print *lp(1)* requests. See the *enable(1)* entry in the *HP-UX Reference.*
- **enable(1)** Activates one or more physical printers to print *lp(1)* 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(1)* 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 available only to 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* manual.

- **accept(1M)** Allows *lp(1)* 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(1M)* 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(1M)* is used. See the *lpsched(1M)* entry in the *HP-UX Reference.*
- **lpsched(1M)** Schedules requests taken by *lp(1)* 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(1)* 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.*

Installing the LP Spooler

To install the LP Spooler, first log in as the super-user (`root`).

Three configuration files **must** be checked and/or modified for the LP Spooler to work properly: */etc/passwd*, */etc/group*, and */etc/rc*. The contents of these configuration files depend on the version of your HP-UX system. See the section "Updating the HP-UX System" in this chapter and the file */etc/newconfig/Update_info* for details on different revisions of HP-UX. Note that the */etc/newconfig* directory is only shipped with updates. If you have just **installed** a new system, the files should be in the */etc* directory and contain the appropriate information; if you have just **updated** an existing system, you need to update your existing files (*passwd*, *group*, and *rc*) with the information contained in the files in the */etc/newconfig* directory.

Check the files for the following information and, if necessary, update your existing files:

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

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

  providing ownership of the LP Spooler to the user `lp`. There may be other users also associated with the group *bin* in your particular configuration.

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

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

  providing group ownership of the LP Spooler to the user `bin`.

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

  ```
  # Start lp printer scheduler
          rm -f /usr/spool/lp/SCHEDLOCK
          /usr/lib/lpsched
          echo line printer spooler started
  ```

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

You may want to add password protection to the `lp` user. To do this, log in as the user `lp` and execute the *passwd* command; see the *passwd(1)* entry in the *HP-UX Reference* for details.

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 *mknod* command. The major number depends on the interface and protocol used; the minor number depends on the interface and select code to which the printer is connected; see the */etc/mkdev* script or the "Adding/Moving Peripheral Devices" section for details.

Assume, for example that you want to set up an HP 2934A printer as a spooled device. The printer is on the HP-IB at select code 5, bus address 1. To install the printer, type the following commands (followed by Return):

```
/etc/mknod  /dev/lp2934  c  22 0x050100
chown  lp  /dev/lp2934
chgrp  bin  /dev/lp2934
chmod  600  /dev/lp2934
```

This sequence creates the special (device) file for the HP 2934 printer and changes the file's owner to `lp`, its group to `bin`, and its protection mode to read/write access for the owner only.

Once this is done and the LP Spooler is configured (described in the following section), spooling requests can take the form:

```
pr myfile | lp -dlp2934
```

which pipes the output of *pr(1)* (the file `myfile`) to *lp* and specifies the destination as the HP 2934 printer.

NOTE

If you have a system printer, you should always name its corresponding special file */dev/lp*, because some commands use this special file as a default. You can create an individual special file for your favorite printer and give it the pathname */dev/lp* (there is one created for you during system installation). Alternately, you can take an existing special file for the printer and create a link to it from the pathname */dev/lp*.

Configuring the LP Spooler System

The */etc/mklp* script is available which you can edit and execute. This section describes each step which also appears in the script. Configuring the LP Spooler system requires the following steps:

1. Log in as super-user and shut down the LP scheduler with:

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

2. Determine for each line printer, how the LP Spooler system will communicate with the printer. This is done by specifying a **model** script when you invoke *lpadmin(1M)*.

   Models are shell scripts that interface between *lpsched* and devices. Several model scripts are shipped with your system and are located in the */usr/spool/lp/model* directory. As shipped to you, this directory includes model scripts for a generic "dumb" printer, the HP 2225A, HP 2631G, HP 2686A, HP 2688A, HP 2934A, and the HP 9000 Model 520 internal thermal printer. These model scripts must have a permission mode of 644 and be owned by *lp* and group *bin*. Refer to the */etc/mklp* script for a description of the provided models.

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

3. When you have selected a model, execute the *lpadmin* command with its **-p** option to name a printer. 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
   ```

   where:

   `-plp` specifies the printer. The logical destination name is `lp`.

   `-v/dev/lp` specifies the full path name of the printer's special (device) file — the physical destination.

   `-mhp2934a` specifies a model in the */usr/spool/lp/model* directory.

   `-h` specifies that 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
   ```

   Note that `lp` is the name of the logical destination. In step 3, the *lpadmin* command associated it with a particular physical destination by specifying the device file name.

5. Select a printer to be the system default. For example, if the printer `lp` is to be the default, execute:

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

6. Now restart the LP scheduler with:

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

   and see that the LP spooler's "scheduler" is properly running by executing:

   ```
   lpstat -t
   ```

7. If the scheduler is not running, you must remove the file SCHEDLOCK before it will work properly. Do this by typing:

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

   and then repeat step 6 above. The `SCHEDLOCK` file acts as a "semaphore" to keep more than one scheduler from running at any given point in time. The *lpshut* command automatically removes the `SCHEDLOCK` file when it terminates the LP scheduler.

Other LP Spooler Adminstrator Duties

There are several other activities that you may need to carry out as the system administrator of the LP Spooler system:

- determining the current status of the LP Spooler system;
- starting and stopping the LP scheduler;
- grouping printers into classes;
- removing destinations (printers and classes of printers);
- moving requests to other destinations.

Determining LP Spooler Status

The command *lpstat(1)* has options that provide a variety of information about your LP Spooler system. Used without any options, *lpstat* prints the status of all requests that you have made to *lp* and the `-t` option gives complete LP Spooler status information. For example,

```
lpstat -t
```

results in output similar to:

```
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
```

The options that you can specify with *lpstat* are:

-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*.

-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) file names.

-t Print all status information.

-u[*list*] Print the status of requests for particular users specified by the login named in *list*. If you do not specify *list*, the status of all users' requests is printed.

-v[*list*] Print the pathnames 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.

You can specify any combination of the above options on an *lpstat* command line.

In addition to using zero or more of *lpstat*'s options, you can also follow the command with particular request IDs, in which case *lpstat* provides status information about those requests.

Controlling the LP Scheduler

The LP 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.

To start the LP scheduler running, use:

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

The scheduler must be running for the LP Spooler to be available for use. However, you **must** shut down the scheduler before using either *lpadmin* or *lpmove*. To shut down the scheduler, use:

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

Remember to re-start the scheduler once you are through using *lpadmin* or *lpmove*.

Building Printer Classes

A **class** is a name given to a group of one or more printers. 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 the class `class1` (unless it already exists) and inserts the printer `lp` into it.

Removing Destinations

LP Spooler destinations (printers, classes, or both) are removed with the *lpadmin* command. To remove a printer from a specific class, use *lpadmin*'s **-r** option:

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

Removing the last remaining member of a class causes the class itself to be deleted. In the example above, since `lp` is the only member of `class1`, the class is deleted.

To remove an entire class of printers, use *lpadmin*'s **-x** option:

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

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

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

NOTE

No printer or class of printers can be removed if it has any pending requests. You can use *lpmove* or *cancel* to move or delete the requests.

Moving Requests

Occasionally it is useful to move requests from one destination to another, such as when one printer is down for repairs. The *lpmove* command is provided for this purpose. Before using the command make sure that *lpsched* is not running. To shut down the LP scheduler, execute:

```
/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 with the ID `lp1-103` to printer `lp2`:

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

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

```
lpstat -alp2
```

to see if `lp2` can accept requests before actually redirecting requests to it.

Shutting Down the System

Powering down the computer (or an "on-line" mass storage device) can cause the file system to become corrupt. The *shutdown* command 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.

Whenever shutting down the system, **always** run the file system check program (*fsck(1M)*) to insure the integrity of the HP-UX file system. See Appendix A, "Using the FSCK Command" in this manual and *fsck(1M)* in the *HP-UX Reference* for instructions on using the command.

The *shutdown(1M)* command — among other things — warns all users that the system is going to be shut down, forces the contents of the file system's I/O buffers to be written to the disc (with the *sync(1)* command) and takes the system into the single-user state.

To run the *shutdown* command, perform the following steps.

1. Login as the super-user.
2. Enter:

   `cd /` [Return]

   to move to the root directory of the file system and then:

   `/etc/shutdown` *grace_period* [Return]

   where *grace_period* is the number of seconds you want *shutdown* to wait between notifying all system users of the impending shutdown (via *wall*) and terminating all processes.
3. *shutdown* prompts to see whether you wish to send the standard file save 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]. Then hold the [CTRL] key depressed as you press [D] to signify the end of the message.
4. After waiting the specified amount of time, *shutdown* asks if you want to continue. An affirmative response causes the command to prompt to see if you want to run the file save routine and/or the file system integrity check program, *fsck*. Supply the appropriate response. When *shutdown* completes its task, it displays a message telling you to halt the system when you are ready. You may then power down the system.

Another command, *killall* is provided to help you shut the system down in a reasonable manner. However, *killall* is not as thorough as *shutdown* and should only be used on small or single-user HP-UX systems. See the *killall(1M)* entry in the *HP-UX Reference* for details on using this command.

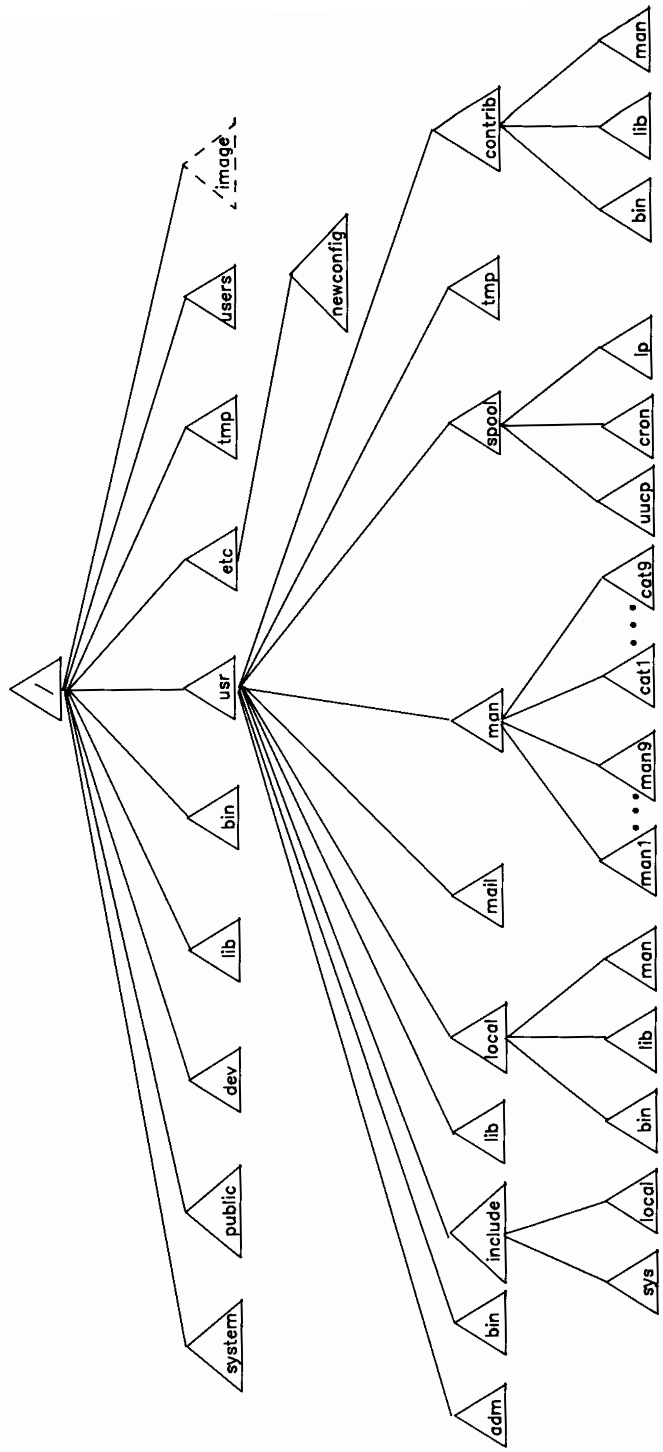
Climbing the HP-UX Tree

As you must know by now, the file system of HP-UX is organized in a tree structure. The base of the tree is the root (although one wonders why we don't call it the trunk) of the file system, and the file name / is associated with the root. Under the root (and not above), are ten standard directories created when you installed your system: **bin**, **dev**, **etc**, **image**, **lib**, **public**, **system**, **tmp**, **users**, and **usr**.

This section describes the basic purpose of the major directories in your HP-UX tree. You will find this useful as you add files and modify your system in the future. As you read the descriptions, reference them to the figure which follows.

- **/bin**—contains frequently used commands, and those required to boot, restore, recover and/or repair the system.
- **/dev**—contains special device files used to communicate to peripherals. For more information, see *mknod(1M)*.
- **/etc**—all system administrative commands and configuration files reside here.
- **/etc/newconfig**—new versions of customizable configuration files and shell scripts are stored here following an update. You should keep these files intact here for future reference.
- **/image**—all files associated with the *IMAGE/Query* optional product are stored here.
- **/lib**—frequently used object code libraries and related utilities are placed in this directory.
- **/public**—used for free access of files to other systems via *uucp* or LAN.
- **/system**—contains object code for drivers; also contains the boot area.
- **/tmp**—a place to put temporary files (those normally with short lifetimes and which may be removed without notice).
- **/users**—user home directories go below this directory.
- **/usr**—less frequently used commands and other miscellaneous files are stuck under this directory.

- **/usr/adm**—system administrative data files lay here.
- **/usr/bin**—less frequently used commands and those not required to boot, restore, recover, and/or repair the system go here.
- **/usr/include**—high-level C language header files (shared definitions).
- **/usr/include/sys**—low-level (kernel-related) C language header files.
- **/usr/include/local**—localized C language header files.
- **/usr/lib**—less frequently used object code libraries, related utilities, and miscellaneous data files go here.
- **/usr/local**—localized files should be placed here.
- **/usr/local/bin**—localized commands should go here.
- **/usr/local/lib**—localized object code libraries are placed here.
- **/usr/local/man**—put any on-line manual pages for localized systems in this directory.
- **/usr/mail**—where your mail box resides.
- **/usr/man**—all on-line documentation shipped with your system can be found here.
- **/usr/man/man1 ... man9**—the unformatted version of *man(1)* pages.
- **/usr/man/cat1 ... cat9**—*man(1)* pages already processed to speed access go here.
- **/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(1)*.
- **/usr/spool/cron**—spooled jobs for *cron(1)* and *at(1)*.
- **/usr/spool/lp**—control and working files for the lp spooler go here.
- **/usr/tmp**—an alternative place (to */tmp*) in which to place temporary files; this directory is usually used when there are many files and/or the temporary files may be very large.
- **/usr/crontrib**—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 on-line documentation for any contributed files, is placed in this directory.


/
image
users
tmp
etc
usr
bin
lib
dev
public
system
newconfig
contrib
tmp
spool
man
mail
local
lib
include
bin
adm
man
lib
bin
lp
cron
uucp
cat9
cat1
man9
man1
man
lib
bin
local
sys

Updating Your HP-UX System

This section describes the steps necessary to update your Series 500 HP-UX system, as well as how to install optional products such as the SRM access utilities (Shared Resource Manager), LAN (Local Area Network) and the IMAGE database system. Since the process of updating or installing optional products involves changes to the HP-UX kernel, you should carefully follow the preparatory steps below before proceeding. Note that the procedure is exactly the same for updating and for optional product installation. You should read the definitions and discussion in the first part of chapter 2 ("Installing HP-UX").

Preparing to Update

As we just mentioned, the HP-UX kernel is modified when updating your Series 500 system. Because of this, it is always possible to crash your disc while updating. By following these simple steps, you will be able to minimize any risks to the integrity of your computer system and data:

Back up your file system

In the event you should make a mistake while updating your computer and possibly corrupting your file system, you should be able to recover all of your data **if you have adequately backed up your Series 500**. See the *Backing up and Restoring the File System* section of Chapter 5 ("The System Administrator's Toolbox") for more information on how to archive your system.

Clean up processes

Log into `root` at the system console. All update procedures take place at the system console. After giving all other users on your system adequate warning, you should have all other users log off. To cleanly shut down your system, you must kill any processes which are running other than *init*, *sh* and *ps*. At the system console type the command:

```
ps -ef[Return]
```

and look for the process *cron*. If *cron* is running, using the *kill(1)* command, signal it to quit. Leaving *cron* running while updating is dangerous because it could spawn a new process which could affect the update process. Also, if *cron* is running you will not get the new copy of *cron* from the update, and will lose the facility. The update programs also assume that your default shell is a Bourne shell; if it is not, you should make it the default using the *chsh* command.

Next you will want to shut down the LP spooler, if it is running. Type the command:

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

to stop the spooler. For more information on the LP spooler see Chapter 5 ("The System Administrator's Toolbox").

After shutting down the LP spooler, you will want to prevent any other users from logging onto the system. Assuming you have made the changes outlined in Chapter 4 ("System Boot and Login"), you can do this by typing:

```
init 1 Return
```

At this point you should be the only user logged onto the system!

Shut down the LAN

If your system is part of a Local Area Network (LAN), you should ensure that your connection to the network is shut down. Unless you do this, someone else on the network could either jeopardize your update, or they could lose critical data. To shut down your connection to a LAN, type:

```
nusers -1 Return
ncontrol /dev/lan reset Return
```

where `/dev/lan` is the special device file describing each LAN connection. This should safely free your LAN connections, and insure the data integrity of others on each network. However, LAN is never completely shut down until your system is re-booted without executing an *npowerup.*

Do an fsck

Now that your shell is the only process on the system, you should do a file system check using the *fsck(1M)* command. For more on how to use the *fsck* command see Appendix A ("Using the FSCK Command").

If the *fsck* showed no problems with your file system, and your shell is the only process on the system when you do a `ps -ef`, you are now ready to begin the update and optional product installation process.

Updating Your System

You should now be running in a single-user mode, and logged in as `root`, the super-user.

NOTE

In order to successfully update your system, you must be using an HP supported terminal as your system console. The update program uses special screen control functions and softkeys which are only on supported terminals.

Begin the Update

You should have one tape which contains the HP-UX operating system update and another tape for each optional product you ordered. Locate the write-protect mechanism (labeled "SAFE") on the top, rear, left-hand corner of the cartridge tape. **The arrow on the protect screw should point toward the word SAFE.** If it does not, use a coin or screwdriver to turn the protect screw such that the arrow points toward the word SAFE. Place this tape in the CS/80 data cartridge drive connected to your system with the SAFE label in the rear left hand corner. Only the BUSY indicator should now be lit. The drive will begin a cartridge tape conditioning sequence that takes approximately two minutes. **Do not proceed until the busy light remains off.**

Loading Tools

The update procedure essentially involves two steps: reading a series of tools necessary for updating from the tape, and then executing the actual */etc/update* procedure. To do this type in:

```
lifcp -a /dev/rmtc:GETTOOLS /tmp/gettools
chmod 700 /tmp/gettools
/tmp/gettools /dev/rmtc
rm /tmp/gettools
```

where the device `/dev/rmtc` is the special device file name assigned to the same CS/80 data cartridge drive you just inserted the update tape in. Executing */tmp/gettools* causes any new tools related to the update process to be extracted from the tape and put into your current file system. This could take from one to several minutes to complete.

You need to know the major and minor numbers of the tape drive used to read the update tape, and for the disc being updated as well. Be sure you know the correct values of these numbers before you continue. You can obtain this information by typing:

```
ll /dev/rmtc /dev/rhd[Return]
```

where `/dev/rmtc` is the source tape drive and `/dev/rhd` is the destination raw hard disc. Make sure you write these values down, because the */etc/update* program uses screen menus—you won't be able to "scroll" back to find these numbers.

Finally, begin the actual update program by typing:

```
/etc/update[Return]
```

Main Update Utility Menu

Upon executing */etc/update* your system will do a soft reboot. This reboot is done to insure that any hidden processes are removed. You should see your normal boot messages, and then the screen should clear and the following menu should appear. This is the main update utility menu. All update procedures are treated as sub-tasks from this menu.

```
******************************************************************
                HP-UX UPDATE UTILITY -- MAIN MENU
******************************************************************
                          Select Choice


Source device is:      /dev/update.src   Major: -1  Minor: fffffff
Destination device is: /dev/update.dest  Major: -1  Minor: fffffff

READ table of contents
EXIT update
CHANGE source device
CHANGE destination device
```

Note the four softkeys at the bottom of your screen:

- **NEXT** or [f2] will move the highlight to the next item in each menu. When you began */etc/update* the line `READ table of contents` should have been highlighted. Pressing **NEXT** when you are at the bottom of the list causes it to wrap around to the first item.

- **PREVIOUS** or `f4` will move the highlight to the previous item in each menu (the item listed above the current item). Pressing **PREVIOUS** when you are at the top of the list causes it to wrap around to the last item.
- **SELECT** or `f6` will execute the currently highlighted option.
- **QUIT** or `f8` will exit the */etc/update* program at any time.

If you have a Model 520 you will need to press the "users keys" to get these soft keys.

Selecting the Source Device

You may notice that for both the source and destination device, the major number is listed as -1 and the minor number as fffffff. This is done to prevent you from accidentally accessing the wrong device. Since these are impossible values, you must enter the correct major and minor numbers, which you should have written down earlier.

By using the **NEXT** and **PREVIOUS** keys, choose the "CHANGE source device" option on the main menu and press **SELECT**. You will see the top half of the following menu:

```
*************************************************************
          HP-UX UPDATE UTILITY -- CHANGE DEVICE MENU
*************************************************************


Current source device address is:  Major: -1  Minor: fffffff

                    NEW Major Number?  1
                    NEW Select Code?   5
                    NEW Bus Address?   4
                    NEW Unit Number?   0
                    NEW Volume Number? 0
```

As you are prompted for each item, enter the source device's major number, select code, bus address, unit and volume numbers in **decimal format**. In our example the major number was `1`, and the minor number was `0x050400` in hexadecimal format (note that you should not type in any leading zeros for these values). Once you have entered the volume number, you will be returned to the main menu. Notice that the new source device values are now shown. Check that the values shown on the menu match those you have written down. It is possible that you could make a mistake while converting from hexadecimal to decimal format.

Selecting the Destination Device

Just as you entered a new source device address, you will need to change the destination device address. In the main menu (which should now show the correct source device address), select the `CHANGE destination device` menu using the softkeys. You should now see a screen very similar to the `CHANGE source device` menu.

In exactly the same manner as you just changed the source device, enter the new destination device major and minor numbers, which you have written down. We will use major number `1` and minor number `0x050500` in our example. Once you have entered the last digit, the `NEW Volume Number`, the */etc/update* program will attempt to *mount(1)* the device you have listed at that address. If it is the root device (which is the normal case), the following prompt will appear near the bottom of the console:

```
Cannot mount the destination device.
Is this the ROOT device?  ('y'or'n') >>
```

Because the method of updating the root volume versus another CS/80 disc is different, you must tell the program that you are indeed updating the root volume by typing `y`. Otherwise, you should enter an `n`. If you have more than one Series 500 HP-UX system, you might find it easier to update by mounting the root volume of each system to one central system, and selectively updating each disc in turn.

Reading the Table of Contents

Once you have entered the correct addresses for the source and destination devices, you see these values reflected in the main menu, similar to our example which follows. If either of these values are wrong, you can go back to change either the source or the destination device. **Do not continue if you are unsure of these values!** Use the **QUIT** softkey if you are not sure of these device addresses, and go back to the "Main Update Utility Menu" section of this chapter to begin again.

```
******************************************************************
                 HP-UX UPDATE UTILITY -- MAIN MENU
******************************************************************
                           Select Choice


Source device is:      /dev/update.src   Major:  1  Minor: 50400
Destination device is: /dev/update.dest  Major:  1  Minor: 50500

READ table of contents
EXIT update
CHANGE source device
CHANGE destination device
```

Assuming that your source and destination addresses are correct, you must next read the list of products on the tape which you may update with. You will need to read each update and/or optional product tape you received in order to install all of these products. Using the **NEXT** and **PREVIOUS** keys, choose the `READ table of contents` option on the main menu and press **SELECT**. */etc/update* will now read the update tape to get a list available options, a procedure which takes a couple of minutes. You should see the following screen:

```
****************************************************
      HP-UX UPDATE UTILITY -- READING FILE MENU
****************************************************
                      reading


Be sure media is in source device, and hit RETURN...
```

After */etc/update* has read the tape, a new main menu will appear at your console. Notice a partition name on the upper left segment of the screen, and the list of product "file sets":

```
Partition:  "16-USER-5.00"            Media Number:  "1"
**********************************************************
           HP-UX UPDATE UTILITY -- MAIN MENU
**********************************************************
                      select choice



Load "97089C" file set
Load "50954A" file set
Load "97069M" file set

Process ALL file sets
READ table of contents
EXIT update
CHANGE source device
CHANGE destination device
```

At this point you have several options:

- You may decide to load all the file sets at one time. Using the softkeys, select the `Process ALL file sets` option. This procedure could take 20 to 60 minutes to load depending upon the processor and peripherals being used.
- You may decide to just load the core system (basically the update for HP-UX itself). Using the softkeys, select the option similar to `Load "97089C" file set.`
- You may want to load a particular optional product, such as LAN. Select the option similar to `Load "50954A" file set` which has the product number of the option you ordered.
- You may wish to load the core system and certain optional products. Select the options you want.
- You may wish to abort the entire update procedure. Select `EXIT update`.

Note that in each `xxxxxx` option, the `xxxxxx` is the product number for that file set. Loading file sets takes a while, so now is a good time to get some coffee and read any update information which accompanied your tape.

/etc/update keeps track of the products that have been loaded by recording this information in the */etc/filesets* directory. Each time the update is used to load a product it looks here to see if that product has previously been loaded. The program will immediately load this file set if it has never done so previously, but if it has, */etc/update* will inform you that the product exists on your system and asks you if you want it removed. Answering [n] will prevent this file set from being loaded. [y] will remove the old file set, and re-load the new one.

NOTE

When loading the HP-UX operating system you should answer [y] for each piece queried, unless you are sure that it is okay to not re-load a particular piece.

When the selected option is complete, the main menu will be re-displayed, but with one or more of the load options removed. Those options corresponding to the file sets you have already loaded will not be displayed. If you decided to load all the file sets, there will be no load options displayed.

Completing the Update

Having loaded all the file sets you want in this session, you will want to leave the */etc/update* utility. Select the `EXIT update` option on the main menu. The program will inform you that it is unloading the update tape, which will take a few minutes. If the HP-UX operating system, (the core system), was loaded to your root device, the system will reboot. This is because you have the old kernel and commands in memory, with the new operating system on disc.

The reboot of your system may appear to take slightly longer than normal. This is because the first time you reboot after doing an update, scripts execute which customize your system for the updated products you installed.

Following this reboot process, you should log in as `root` and follow the instructions supplied in the *Installation Notes* or in the `/etc/newconfig/Update_info` file. Also look for any other update files in the */etc/newconfig* directory.

Modifying the Boot Area

There are times when you will want to add or remove code segments (usually drivers) from the boot area of your HP 9000 Series 500 computer. When you installed your HP-UX, system the boot area was configured to match the equipment you had connected at that time (and this may not be totally true). For example, there are three different console drivers. If you installed HP-UX from an HP 98700 display device, then you will have the code necessary to support driver 29. At some time later, though, if you want to use an HP 2623 terminal as the console you will need the code for driver 31 in the boot area.

Before modifying your system, you **must** bring your system into a quiescent state. See the "Shutting Down the System" section in this manual for more information on this procedure. Hewlett-Packard recommends that you also do an *fsck(1M)* before altering your kernel.

To add or delete segments from the boot area (kernel), you must be logged into `root` (the super-user account), and be in state `s`.

Adding Segments

To install an optional driver, type in:

```
oscp -a /system/xxxxxxxx/yyyyyyyy /dev/rhd
```

where *xxxxxxxx* is the product number of the code you wish to install, *yyyyyyyy* is a segment which resides under that directory, and `/dev/rhd` is the boot volume character special file. As an example, let's say you wish to add the Local Area Network segment:

```
oscp -a /system/97059A/localnet.opt /dev/rhd
```

Once you have added code to the kernel, you will need to reboot the system. Since the kernel is loaded into memory at the time you boot, no change will be reflected until you stop and restart the system (or execute `stopsys -r`).

Removing Segments

There are two basic steps you must follow to safely remove a kernel segment: first split out the good section into a file (*new.kernel*), and then replace the old boot area with your newly created kernel. To accomplish this first step, type:

```
oscp -s /dev/rhd
```

where `/dev/rhd` is the volume containing the boot area. The *oscp(1M)* command is interactive—it will prompt you to find out what to do with each kernel code segment:

```
SEGMENT   OFFSET    BYTES   SYSTEM/TYPE      SEGMENT          REVISION
     0        0     4188   HP-UX STANDARD   SYSTEM POWERUP    04.02
File to append segment to? (system) new.kernel
Appending to file ('new.kernel')
```

You must tell *oscp* to put the segments you **want to keep** in the file *new.kernel*. After you specify *new.kernel*, you will be able to type **Return** to get this file as the default—until you specify a different file.

When you get to the segment you wish to remove, respond with a new file name. By redirecting the segment to */dev/null*, you will eliminate that segment (the original should still be in the */system* directory):

```
     43   301356    12476   HP-UX STANDARD   INTERNAL HP-IB    04.02
File to append segment to? ('new.kernel')  /dev/null
Appending to file ('/dev/null')
```

After rerouting the undesired segment to */dev/null*, if you get another segment you wish to keep, you **must** enter *new.kernel* once again. In general, make sure that you route all the segments you want to keep into the *new.kernel* file.

You may wish to remove driver segments to decrease the size of the boot area. This is desirable, since the kernel always remains in memory. For example, if you had the segment for Local Area Net in your current boot area, and you do not use LAN, you are wasting in excess of 300 kbytes of memory.

To actually update the operating system in the boot area, you simply enter:

```
oscp -m new.kernel /dev/rhd
oscp:  please enter new system ID (non-blank ID required):
NEW AND IMPROVED
```

This overwrites the boot area on */dev/rhd* with the file you just created, *new.kernel*. Notice that *oscp* asked for a new ID field to label the new boot area with. We chose `NEW AND IMPROVED` for this example, but you will probably want something a little more descriptive.

Once you have overwritten code to the kernel, you will need to reboot the system. Since the kernel is loaded into memory at the time you boot, no change will be reflected until you stop and restart the system (or execute **`stopsys -r`**).

Checking the Boot Area

At any time you may desire to simply list what is in your boot area. HP-UX has a command which does this—*osck(1M)*. You may execute *osck* at any time; the system need not be in a quiescent state:

```
osck -v /dev/rhd
```

You may also desire to redirect the output to a file or printer.

For more information on the boot area, see Chapters 3 and 4 of this manual, and the *oscp(1M)* entry in the *HP-UX Reference*.

System Accounting **6**

Multi-user HP-UX allows concurrent sharing of computer resources among multiple users: several users can be logged in, all sharing disc space, memory, and the CPU. On multi-user systems, HP-UX System Accounting provides the means to:

- monitor disc 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 consumption

What Is in This Chapter?

HP-UX System Accounting allows you to accomplish accounting tasks through a number of versatile commands. This chapter illustrates the use of these commands and contains the following sections:

- "Overview of System Accounting" provides the background information necessary to understand how to use System Accounting.
- "Daily Usage and Installation" shows the routine daily usage of System Accounting and shows you how to install it.
- "Disc Space Usage Accounting" illustrates the use of the accounting commands that monitor disc space utilization on a per-user basis.
- "Connect Session Accounting" describes the commands that record and report connect session accounting information.
- "Process Accounting" shows how to generate per-process accounting data and reports.

- "Charging Fees to Users" is the section where you learn how to charge fees to users.
- "Summarizing and Reporting Accounting Information" shows how to generate the main daily and monthly accounting reports that are used to monitor system performance and bill users.
- "Updating the Holidays File" describes how to set up the file */usr/lib/acct/holidays* on your system.
- "Fixing Corrupted Files" Occasionally, during day-to-day usage of System Accounting, certain files may become inconsistent or messed up; this section shows how to fix these files.
- "Sample Accounting Shell Scripts" provides listings of shell scripts that you might find useful on your system.

In addition to these sections, the appendix "System Accounting Files" contains brief definitions of all the files used by System Accounting.

NOTE

Much of the material in this chapter assumes greater knowledge of HP-UX than is required of the "average" user. In particular, System Accounting borrows many concepts from the previous chapters "Concepts", and "System Boot and Login." If you are unfamiliar with the concepts and terminology in those chapters, then you should review them.

Overview of System Accounting

In this section, the instrinsics of System Accounting are examined. 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 used often in System Accounting; understanding these definitions is essential to successfully using accounting capabilities.

connect session

This denotes the period of time in which a user is connected to the system. It starts when the user logs in and finishes when the user logs out.

cron

This process wakes up every minute to execute commands at specified dates and times, according to instructions in files contained in the directory */usr/spool/cron/crontabs*. See the *cron(1M)* and *crontab(1)* entries in the *HP-UX Reference* for more details.

/etc/rc

This is the system initialization shell script. The actions that it performs depend on the state in which it is invoked. To automatically start System Accounting whenever the system is switched to multi-user mode, a command must be added to *rc*. See the chapter "System Boot and Login" in this manual, and *rc(8)* in the *HP-UX Reference* for more details on the use of *rc*.

/etc/shutdown

A shell script that has the primary function of terminating all currently running processes in an orderly and cautious manner. A command that turns off System Accounting should be placed in this file. See *shutdown(8)* for details on this shell.

kernel

This is the "core" of the HP-UX Operating System—the interface between the shell and the computer hardware. The kernel is responsible for managing the computer's resources; for example, it allocates memory and schedules processes for execution. In addition, the kernel manages all the processes that execute in the system.

When System Accounting is installed and running, the kernel has the additional responsibility of creating accounting information (see the definition for process accounting records) for all terminating processes.

prime/non-prime connect time

Prime time is the time during the day when the computer system is most heavily used—for example, from 9:00am to 5:00pm. Non-prime time is the remaining time during the day when the system is less heavily used—from 5:00pm to 9:00am in this example.

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 during the weekends (Saturdays and Sundays) and on any holidays specified in the *holidays* file.

---

process

A process is the environment in which a program (or command) executes. It includes the program's code, data, status of open files, value of variables, and the directory in which the process resides. For example, whenever you execute an HP-UX command, you are creating a process; whenever you log in, you create a process. For additional information on processes, read the chapter "Concepts."

process accounting records

Once System Accounting is installed and turned on, the following occurs: whenever a process terminates, the kernel writes a process accounting record for the terminating process into the current process accounting file, */usr/adm/pacct* by default (you can specify that a file other than *pacct* be used as the process accounting file, if you want).

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. Examples of information contained in process accounting records are:

- the user ID of the process's owner
- the name of the command that spawned the process
- the amount of time it took the process to execute

For greater detail on the contents and format of process accounting records, see *acct(5)* in the *HP-UX Reference Manual.*

total accounting records

These records, created by various accounting commands, contain summary accounting information for individual users. These records provide the basic information for many reports generated by System Accounting. Some examples of information contained in these records are:

- the ID and user name of the user for whom the total accounting record was created
- the total number of processes that the user has spawned during the accounting period for which the total accounting record was created
- fees for special services rendered to this user

The exact contents and format of total accounting records can be found in *acct(5).* In addition, commands covered in later sections of this chapter show how these records are created and used by System Accounting.

Introduction to Commands

System Accounting provides many versatile commands to accomplish numerous varied tasks. There are commands that create data, commands that display data, commands that remove data, commands that merge data, and commands that summarize and report data. In addition, the output of one command may be the input to others, and so on.

Accounting commands can be logically categorized into six basic command groups:

- installation
- disc usage accounting
- connect session accounting
- process accounting
- fee charging
- summarizing and reporting accounting information

Descriptions of these command groups, along with a brief synopsis of each command, follow:

Installation

These commands insure that System Accounting is properly installed. They are used to turn accounting on when HP-UX is powered up and turn accounting off when the system is shut down. They may also do some file cleanups. Two such commands exist:

- **startup**—starts accounting when HP-UX is switched to multi-user mode. Invoked from */etc/rc*.
- **shutacct**—turns off accounting when HP-UX is turned off via the */etc/shutdown* shell.

Disc Usage Accounting

In general, these commands produce disc usage accounting information: they show disc space usage (in blocks) for individual users. They also produce total accounting records. There are four commands:

- **acctdusg** and **diskusg**—both commands show how many blocks of disc space users are consuming. They differ in command options, and the manner in which they produce the information—*acctdusg* takes its input from a list of path names created by *find*, and *diskusg* looks at the inodes of the file system to create its output.
- **acctdisk**—this command produces total accounting records. Its input is supplied (either directly or indirectly) from *acctdusg* or *diskusg*.
- **dodisk**—Produces total accounting records by using the *diskusg* and *acctdisk* commands. *dodisk* is normally invoked by *cron*.

Connect Session Accounting

Independently of System Accounting, the programs **login** and **init** record connect sessions by writing records into */etc/wtmp*. Accounting commands can display or fix this file, and can produce total accounting records for this file. There are five commands:

- **fwtmp**—displays the information contained in *wtmp*.
- **wtmpfix**—normalizes connect session records that span date changes (see *date(1)*). Also validates login names in connect session records.
- **acctcon1**—summarizes *wtmp* in ASCII readable format, producing one line per connect session.
- **acctcon2**—takes input of the format produced by *acctcon1* and produces total accounting records as output.
- **prctmp**—used to display the session record file created by *acctcon1*. (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 *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 insure that *pacct* doesn't become too large. There are eight process accounting commands in all:

- **accton**—turns process accounting on or off, depending on whether or not a filename argument is supplied with the command. If no filename is given, then process accounting is turned off; the kernel stops writing process accounting records to *pacct*. If a filename is specified, then the kernel starts writing process accounting records to the specified filename.

  accton uses the system call ***acct(2)*** to turn process accounting on or off. In addition, only the super-user 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 disc 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 argument */usr/adm/pacct*; *turnacct off* turns process accounting off by calling *accton* with no filename argument; *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**—displays process accounting records contained in *pacct* (or any specified file).

- **acctcms**—takes *pacct* as input, and produces summary accounting information by command, as opposed to by process.

- **acctprc1**—produces readable process accounting information, mainly for input into *acctprc2*.

- **acctprc2**—takes input of the form produced by *acctprc1* and produces total accounting records.

Charging Fees

Occasionally, you may want to charge a user for something. For example, you might charge fees to users for fixing any damaged files that they have. The **chargefee** command allows you to charge fees to specific users.

Summarizing and Reporting Accounting Information

This group of commands summarizes and reports the data created through the command groups described above. These are the commands that are probably used most frequently; they represent the highest level of accounting commands. Five such commands exist:

- **prtacct**—takes as input total accounting records and displays the records in ASCII readable format.
- **acctmerg**—combines the contents of separate total accounting files into a single total accounting file. Allows the merging of disc, process, and connect session total acounting records.
- **runacct**—the main accounting shell script. Normally invoked daily by *cron*, this command processes disc, 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, disc, connect session, process, and fee total accounting records are merged to create one total accounting file.
- **prdaily**—invoked by *runacct* to format a report of the previous day's accounting data; the report is stored in the file ***/usr/adm/acct/sum/rpt****mmdd* 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**—invoked once a month (or accounting period), this command summarizes daily accounting files and produces a summary files for the accounting period.

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 1 illustrates how accounting data is created. The diagram is broken into five separate sub-diagrams, each one representing the data flow for a given command group. The following notational conventions are used:

Symbol	**Description**
source ⟹ dest	Wide arrows represent the *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 disc 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 inputs to and outputs from the various commands.

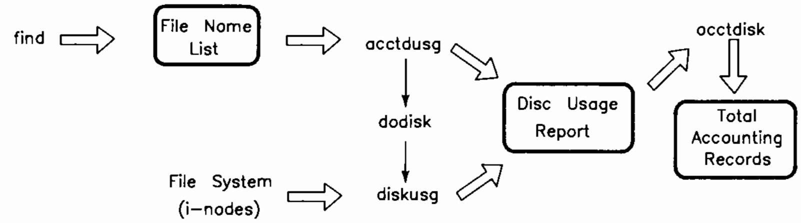
NOTE

The installation commands do not appear in the diagram, because they aren't directly involved in the data creation process; they merely insure that it happens.

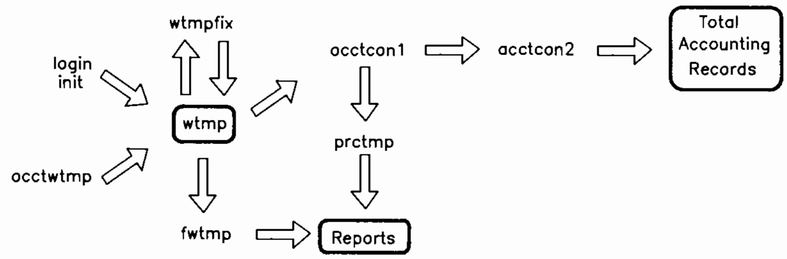
NOTE

The commands *runacct* and *prdaily* are shown as having no inputs. This isn't exactly true: they do have inputs, but they get their inputs by executing other accounting commands. In essence, their inputs are the same basic inputs of the other command groups.

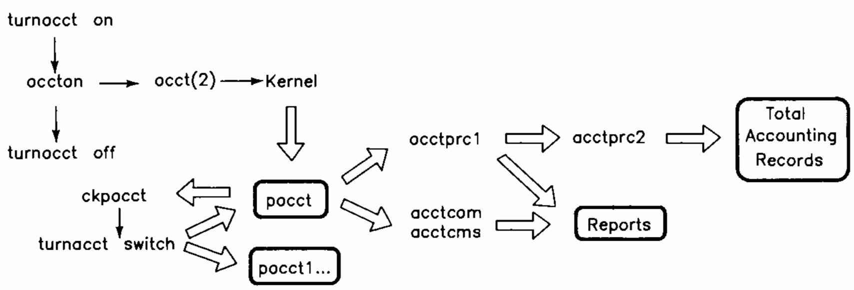
Disc Usage Accounting



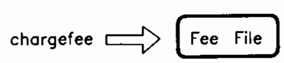
Connect Session Accounting



Process Accounting



Charging Fees



Summarizing and Reporting

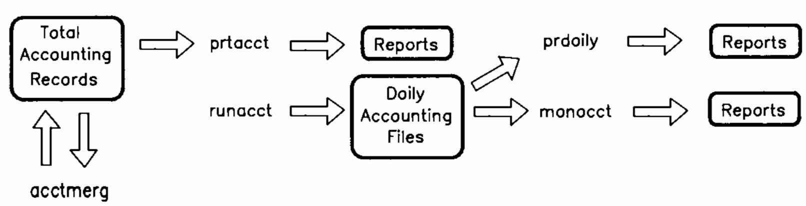


Figure 1. System Accounting Data Flow Diagram

Login and Directory Structure

You now know the basics, but you still can't begin learning the day-to-day usage of accounting commands until you know where 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 multi-level directory structure to organize its many accounting files. Each directory in this structure stores related groups of files, commands, or other directories. (See the appendix "System Accounting Files" for definitions of the accounting data files.)

Figure 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***—cumulative summary files updated by *runacct* are kept here.
- ***/usr/adm/acct/fiscal***—periodic (monthly) summary files created by *monacct* are found here.
- ***/usr/lib/acct***—System Accounting commands reside here.
- ***/etc***—contains *wtmp*, and shell scripts *rc* and *shutdown.*

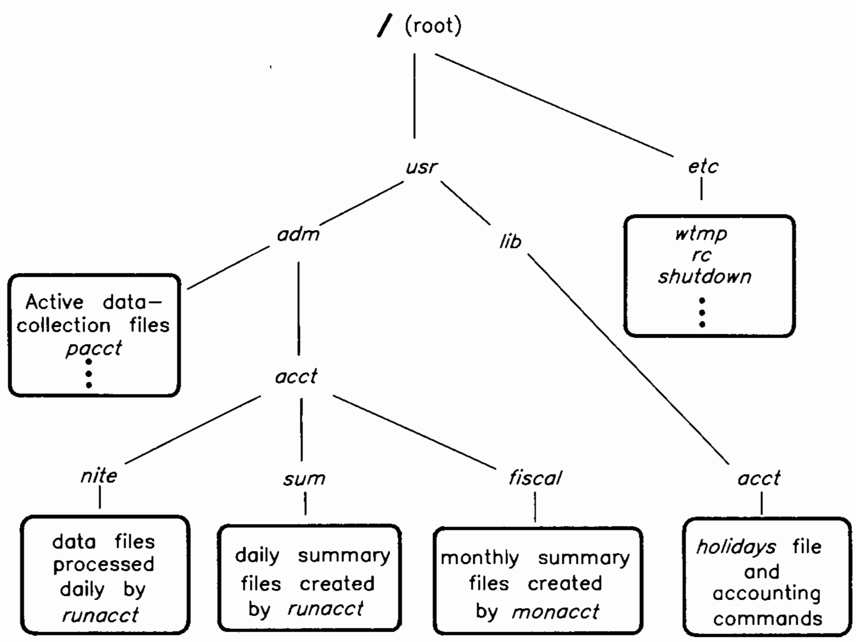


Figure 2. System Accounting Directory Structure

Daily Usage and Installation

Now that the basics have been covered, you can start learning how to use System Accounting on a daily basis. 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 Accounting running on your system.

After reading this section, you should be able to install Accounting on your system. Once properly installed, Accounting will automatically generate 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 multi-user mode, the system initialization shell script *rc* executes the accounting command *startup*. The purpose of *startup* is to start Accounting, and it 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. It 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*. Obviously, this step is omitted the first day that Accounting is installed, because the previous day's accounting information doesn't yet exist. However, after *runacct* has been executed, *prdaily* will generate valid reports.
3. The *ckpacct* command is executed every hour via *cron* to insure 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 smaller *pacct* files is that *runacct* can be restarted faster if a failure occurs while processing these records.

4. The *chargefee* program can be used to charge 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 disc total accounting files. It produces command and resource-usage summaries by login name.
6. When the system is turned off using *shutdown*, the *shutacct* command is executed. The purpose of *shutacct* is to stop Accounting, and it performs the following functions:
   a. Writes a termination record to *wtmp* via the command *acctwtmp*. This record is marked by having "acctg off" in the device name field.
   b. Turns process accounting off by calling *turnacct off*.

How to Install System Accounting

Not all users require accounting services on their systems. For this reason, HP-UX System Accounting is provided as an option: if you want to use Accounting, you must install it yourself. Installation procedure is covered here.

There are four steps in the installation process:

1. Updating */etc/rc*
2. Updating */etc/shutdown*
3. Creating *crontab* entries
4. Setting PATH for accounting commands

Each of these steps must be carried out to insure that System Accounting automatically creates daily and monthly accounting information. Detailed descriptions of each step follow.

Updating /etc/rc

The system initialization shell script *rc* must be updated to automatically start System Accounting when the system is switched into multi-user mode. This requires adding the following entry in the **state 2** section:

```
/bin/su - adm -c /usr/lib/acct/startup
```

Updating /etc/shutdown

To insure that accounting is turned off when the system is brought down via *shutdown*, you must add the *shutacct* command to the *shutdown* shell script. The call to *shutacct* should be placed in the section of *shutdown* where 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 entry for the *shutacct* command should be made as follows:

```
/usr/lib/acct/shutacct
```

NOTE

If you do not use */etc/shutdown* to bring your system down, then you should use some other means—such as *turnacct off* or *shutacct* by itself—to turn system accounting off before shutting down your system.

Creating crontab Entries

To automate the daily and monthly creation of accounting data, you should create a *crontab* file that *cron* can use to automatically run certain accounting commands. This process entails the following steps:

1. Log in to System Accounting as the user *adm*.
2. Use an editor to create the *crontab* file containing the accounting commands that are to be run automatically by *cron*. (The actual entries to make in this file are shown after these steps.)
3. Execute the *crontab(1)* command, specifying the file created in step 2 as input. This step insures that the *crontab* file created in step 2 will be scanned by *cron* every minute. After invoking this command, the step 2 file will be stored in the file:

   ```
   /usr/spool/cron/crontabs/adm
   ```

4. At this point, you are finished creating *crontab* entries. If you ever want to change the entries, simply re-edit the file created in step 2 and use the *crontab(1)* command again.

The following entries, accompanied by a description of each, should be made 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. The above entry executes *runacct* at 4:00am 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 disc space usage for individual users. This entry runs *dodisk* at 2:00am every Thursday morning.

- `5 * * * * /usr/lib/acct/ckpacct`

  To insure that the process accounting file, *pacct*, doesn't get too large, the command *ckpacct* should be executed hourly. This entry invokes *ckpacct* at five minutes into every hour.

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

  The monthly merging of accounting data is facilitated through the *monacct* command. This entry allows *monacct* to generate a monthly total report and total accounting file. *monacct* will be executed at 5:15am on the first day of every month.

NOTE

The dates and times shown in the *crontab* entries above are only suggestions; you can tailor *crontab* entries to suit your needs. However, if you use different entries than those shown here, be sure that *monacct* is run at such a time as to allow *runacct* sufficient time to finish.

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. Path should be set as follows:

```
PATH=/usr/lib/acct:/bin:/usr/bin:/etc:/usr/adm
```

Disc Space Usage Accounting

System Accounting provides the means to monitor disc space utilization for individual users. In this section, disc space usage accounting commands are explained. Before reading this discussion, you may want to review the "File System Implementation" section of the "Concepts" chapter.

Disc usage commands provide two main functions: they report disc usage (in blocks) for individual users and create disc total accounting records (supplied as inputs to commands such as *prtacct* or *runacct*).

Reporting Disc Space Usage

Two commands—*acctdusg* and *diskusg*—report disc usage for individual users; both commands show the number of disc blocks allocated to specific users. However, each command has slightly different options. In addition, each differs in the manner in which it produces accounting information.

acctdusg

acctdusg takes from standard input a list of path names, 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 disc usage information for specific users or files. For example, suppose you want to know how many blocks of disc 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 disc 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 disc 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 *no_owners* — If -u is given, then path names of the files for which no owner is found are written into the file *no_owners*. This option could potentially find users who are trying to avoid disc charges.

-p *p_file* — 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* will use *p_file* instead. This option is not needed if your password file is */etc/passwd*.

The shell script **grpdusg** provided in the appendix "Sample Accounting Shell Scripts" displays disc accounting information for users in a given group. It illustrates the use of the -u option with *acctdusg*.

diskusg

This command reports disc usage information in the same format as *acctdusg*—user ID, user name, and total disc blocks used. However, *diskusg* generates disc accounting information by looking through the **inodes** of a specified special file (see *inode(5)* and the "File System" section of the "Concepts" chapter for more information on inodes and special files.) Therefore, *diskusg* is faster and more accurate than *acctdusg*.

The syntax of the *diskusg* command is:

diskusg [*options*] [*files*]

It generates a disc 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 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 is used to merge 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 disc space accounting charges. When this option is specified, *diskusg* writes records to *stderr* (standard error output) showing the special file name, inode number, and user ID of files that apparently have no owner.

`-i` *fnmlist* Causes *diskusg* to ignore the data on those file systems whose file system name is in *fnmlist*. *fnmlist* is a list of file systems seperated by commas or enclosed within quotes.

`-p` *p_file* This is the same as the `-p` option of *acctdusg*.

`-u` *u_file* This option produces **exactly** 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 the file *u_file*.

The output of *diskusg* is normally used by *acctdisk* to create disc total accounting records. In addition, *diskusg* is normally called by *dodisk*.

The following example creates disc usage information for all users whose files reside on the disc whose device file is */dev/rhd*. **Note that the file system used in this example is the same as was used in the previous *acctdusg* example.**

```
$ diskusg /dev/rhd
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 disc usage totals were generated. However, *diskusg* looked at inodes and saw that `horatio` and `guest` were actually using two blocks for the directories themselves.
4. If two or more users have links to a particular file, then *acctdusg* will prorate disc space usage for the file between each user. For example, if three users had a link to a 300-block file called *skurbnich.dat*, 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

acctdisk takes from standard input records of the format produced by *acctdusg* and *diskusg*. From these records, *acctdisk* produces disc total accounting records that may be inputs to *prtacct* or *runacct*.

The following would write disc 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 disc total accounting records for all users who have files on the disc `rhd`. The total accounting records are written to the file *disktacct*.

```
diskusg /dev/rhd | acctdisk > disktacct
```

acctdisk has no options and is normally invoked by *dodisk*.

dodisk

dodisk is normally invoked by *cron* to create disc total accounting records for daily usage by System Accounting. The syntax for *dodisk* is:

`dodisk` [-*o*] [*files* ...]

In the default case, *dodisk* creates disc total accounting records on the special files whose names are stored in **/etc/checklist**: the special file names 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 *acctdusg* instead of *diskusg*.

If *files* are used, disc 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 file names of mountable filesystems.

NOTE

See the "Daily Usage and Installation" section of this chapter for more information on how *dodisk* should be invoked by *cron*.

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, and
- other status information (see *utmp(5)* 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 – acctwtmp

The command *acctwtmp* allows you to write records to *wtmp* for whatever reason you might have. *acctwtmp* is normally invoked by *startup* and *shutacct* to record when System Accounting was turned on and off respectively. The format of the command is:

`acctwtmp` "*reason*"

where *reason* is string describing the reason for writing the record to *wtmp*. **Note that *acctwtmp*** does not directly write records to *wtmp*: it writes a record containing the terminal name, current time, and reason string to standard output. To actually write the record to *wtmp* you must append the output from *acctwtmp* to the *wtmp* file as follows:

`acctwtmp` "*reason*" >> */etc/wtmp*

The *reason* string may be any combination of letters, numbers, spaces, and the dollar sign ($), but may not exceed **11** characters in length. (*reason* must be enclosed in double quotes as shown.)

Displaying Connect Session Records - fwtmp

To display the contents of *wtmp*, you can use the command *fwtmp*. When no options are used, *fwtmp* takes from standard input records of the format contained in *wtmp*; it writes to standard output the ASCII readable equivalent of the input records. The output of this command can either:

1. be edited, via a HP-UX editor such as *vi(1)*, 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 and is to be converted to binary form. This is essentially the opposite of using *fwtmp* without any options.
`-i`	Both input and output are in ASCII readable 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 the output produced by *fwtmp* and is 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 occured.
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(5)* for more details on this field.
6–7	Exit status for connect session. See *login(1)* and *utmp(5)* for details.
8	Time that entry was made (in elapsed seconds since January 1, 1970).
9–12	The equivalent of column 8 in date/time format showing month, day, time of day (in 24-hour format), and year.

Fixing wtmp Errors – wtmpfix

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 the same user logs out, an entry type of eight (8) 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 preceed their corresponding logout records in chronological order.

Occasionally, this time-stamped ordering becomes inconsistent: logout records might preceed 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 *files* is given, then input is taken from *files*. A dash (-) can be used 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 1 (in the "System Data Flow" section).

acctcon1

acctcon1 converts a sequence of login/logoff records (of the format contained in *wtmp*) read from its standard input to a sequence of records, one per login session. Its input is normally redirected from *wtmp*; its output is columnar ASCII 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 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

$ acctcon1 < /etc/wtmp

520095488    353    sarah   1665    2478    479518335  Tue Mar 12 16:32:15 1985
521012224    352    mike                    479493590  Tue Mar 12 09:40:00 1985
520095488    351    bill    0       5012    479521475  Tue Mar 12 17:24:35 1985
521011712    0      root    41047   6488    479475173  Tue Mar 12 04:32:53 1985
```

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.
7–11	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 echos 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 lines on which users are logged in. When it reaches the end of its input, it 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 causes a line usage summary report to be placed in *file*. This report shows each line's name, number of minutes used, percentage of total elapsed time used, number of sessions charged, number of logins, and number of logins and logoffs. This report can be used to keep track of 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 (`line_use`) 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 (e.g., `acctcon1 -o f_overall`) causes *file* to be filled with an overall record for the accounting period, giving starting time, ending time, number of reboots, and number of date changes.

```
$ acctcon1 -t -l line_use < /etc/wtmp > ctmp
$ cat line_use
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*.

prctmp takes its input from standard input; therefore, to create a *prctmp* report from *acctcon1* information, you can simply pipe the output from *acctcon1* into *prctmp* as follows:

```
$ acctcon1 < /etc/wtmp | prctmp
```

prctmp will respond by generating a report with appropriate headings over the columns of output from *acctcon1*.

acctcon2

acctcon2 creates connect session total accounting records from standard input of the format created by *acctcon1*. In other words, to create connect session total accounting records, simply send the output from *acctcon1* into the input of *acctcon2*.

The total accounting records created by *acctcon2* are sent to standard output. So if you want 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*) doesn't 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 (Figure 1) when reading this section.

Turning Process Accounting On

Before System Accounting can generate process accounting data, process accounting must be turned on; two commands can be used to accomplish this task: *turnacct on* and *accton*. After process accounting has been turned on, the kernel will write a process accounting record, for every terminating process, into the current process accounting file (*pacct* by default).

NOTE

The *startup* command, placed in the system initlization shell script */etc/rc*, automatically turns process accounting on. Therefore, if you have updated */etc/rc* for System Accounting (as described in the section "How to Install System Accounting"), process accounting will automatically be activated, and you should seldom need to use the commands described here.

These commands are described only for your benefit should you ever need to manually turn process accounting on or off.

turnacct on

The command used most often to activate accounting is *turnacct on*; only the super-user and the *adm* login can 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. Check to see if the process accounting file *pacct* exists.
2. If *pacct* doesn't exist, then create a new *pacct* file.
3. Turn process accounting on by invoking *accton* with the filename argument *pacct*.

To execute this command, simply enter `turnacct on` to the HP-UX prompt. Note that only the adm login and the super-user can execute this command.

accton

Again, only the super-user 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 would activate process accounting and make the current process accounting file `/usr/adm/XX107`:

```
$ accton /usr/adm/XX107
```

NOTE

You must make sure that the filename you specify is an existing file; otherwise, *accton* will fail.

Note that in the System Data Flow Diagram (Figure 1), *accton* is shown calling another routine, *acct(2)*. *acct(2)* is the system call that actually tells the kernel to start writing process accounting records. See the *HP-UX Reference* for more details on *acct(2)*.

Turning Process Accounting Off

Two commands are used to turn process accounting off: *turnacct off* and *accton* (with no filename argument). These commands 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 ever use these commands. The reason is that the *shutacct* command, added to */etc/shutdown*, automatically turns process accounting off.

---

turnacct off

turnacct off can be executed by only the super-user and the *adm* login. *turnacct off* turns process accounting off by invoking the *accton* command without the optional filename argument. You execute this command by typing:

```
$ turnacct off
```

accton

When *accton* is invoked without the optional filename argument, process accounting is turned off. You would enter this command as:

```
$ accton
```

As shown in the System Data Flow Diagram (Figure 1), *accton* tells the kernel to stop writing process accounting records by using the system call *acct(2)*.

Checking the Size of pacct

On a multi-user 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 that insure that the default process accounting file *pacct* doesn't become too large. The two commands used for this purpose are: *turnacct switch* and *ckpacct*.

NOTE

The commands described here work only on the default process accounting file, *pacct*.

---

ckpacct

The command *ckpacct* is normally invoked by *cron* every hour to insure that the current process accounting file *pacct* hasn't become to large. The format of *ckpacct* is:

`ckpacct` [*blocks*]

If the size of *pacct* exceeds the *blocks* argument, 1 000 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 number of free blocks in the */usr* file system falls below 500, *ckpacct* will automatically turn off process accounting via *turnacct off*. When at least 500 blocks become available, 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 Manual* for more details.

---

turnacct switch

turnacct switch is used to create a new *pacct* file when the current *pacct* file is too large. The action of *turnacct switch* can be summarized as follows:

1. Process accounting is temporarily turned off.
2. The current *pacct* file is renamed to *pacct**incr***, where ***incr*** is a number starting at **1** and incrementing by one for each additional *pacct* file that is created via *turnacct switch.*
3. Since the old *pacct* file was renamed to *pacct**incr***, a new, current *pacct* file is created.
4. Process accounting is restarted; the kernel starts writing records to the newly created *pacct* file.

The example below illustrates the effect of using the *turnacct switch* command. In the example, *turnacct switch* is executed from the **adm** home directory */usr/adm.* Comment lines begin with a cross-hatch(#) 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 pacct1 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 pacct1
$ #
$ # The current process accounting file is pacct.  The previous
$ # process accounting file is now named pacct1.
$ #
```

Displaying Process Accounting Records – acctcom

The *acctcom* command allows you to display records from any file containing process accounting records. Normally you would use this command to display records from the *pacct* files (*pacct*, *pacct1*, *pacct2* ...).

acctcom is a very versatile command; its syntax follows:

`acctcom` [[*options*][*file*]] ...

If no *file* is specified, *acctcom* uses the current *pacct* file as input. Input can also be taken from standard input. Some of *acctcom*'s options allow you to select only the records that you want to see; other options control the format of the report.

The information contained in this section is organized as follows:

- First, definitions are given for the columnar data produced by *acctcom*.
- Command options that control the format of the report are discussed.
- Options that allow you to select particular records are described.
- Finally, to help you understand how to use *acctcom*'s options, sample *acctcom* reports are shown.

Definitions of Information Produced by acctcom

acctcom generates a columnar report with descriptive headings on each column. Each line of the report represents the execution statistics that a particular process accumulated during its lifetime. The following table defines the standard columns in the report—i.e., the columns that are displayed when none of *acctcom*'s options are specified.

Column Header	Definition
`COMMAND NAME`	The name of the command or program that spawned the process is shown here. Whenever you enter a command, you are spawning a process. For example, if you enter the command `$ ll /usr/lib/acct` you are creating a process with the command name `ll`. If a command requiring super-user privileges is executed, a # appears before the command name.
`USER`	The login name of the user who created the process is displayed here.

`TTYNAME`	This is the name of the 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`	The time that the process began executing (in *hh:mm:ss* format) is displayed here.
`END TIME`	This is the time (*hh:mm:ss*) that the process finished executing.
`REAL (SECS)`	The number of seconds that elapsed from **START TIME** to **END TIME** is shown in this column.
`CPU (SECS)`	This column shows how much of the CPU's time a process used during its execution.
`MEAN SIZE(K)`	This is a rough estimate (in kilobytes) of the amount of memory that a process used during execution.

This estimate is based on the number of 512-byte memory segments held at process exit, rounded up to the next 512-byte unit. HP-UX Accounting reports memory usage only for resident, non-paged segments. This will include the process's stack, global data segment, heap, non-shared code, some operating system data structures, and possibly some data (if not paged or shared).

However, virtual memory statistics ARE NOT INCLUDED!

Note also that only memory that is held for a complete sixtieth of a second will be reported; therefore, **processes that execute for only a very short time (less than 1/60th of a second), may show zero memory usage!**

The table below defines the columns that are not displayed on the standard report, but which can be displayed by 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 super-user privileges show a **2**; super-user commands which do a *fork* without an *exec* show a **3**; otherwise, this column shows a **0**.
STAT	This column displays the system exit status. (This is **not** the status returned by *exit(2)* to a parent process during *wait*). When a process terminates normally, this field shows a **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 will contain a **2**.
HOG FACTOR	The hog factor is computed as the CPU time divided by REAL time; 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: `KCORE MIN = CPU (SECS) * MEAN SIZE(K) / 60`
CPU SYS	This is the portion of total CPU time that was spent executing operating system code, such as system calls (for example, writing to disc).
USER (SECS)	This is the remaining portion of CPU time. User CPU time is the amount of time actually spent executing a process's code (rather than system code).

CPU FACTOR Whenever you execute a command, the CPU spends part of its time actually executing the command's code (user CPU time) and spends the rest of its time performing system functions, such as writing to the disc or terminal (system CPU time). That is, total CPU time is comprised of both *system* and *user* CPU time:

```
CPU (SECS) = CPU SYS + USER (SECS)
```

The CPU factor shows the ratio of user CPU time to total CPU time:

```
CPU FACTOR = USER (SECS) / (CPU SYS + USER (SECS))
```

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 The number of characters (bytes) read and/or written by the command is displayed in this column.

BLOCKS R/W This column shows the 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* will produce a report containing only the default information. Optional information can be displayed only by using the report format options. Definitions of the report format options follow:

Option	Description
`-a`	Causes average statistics to be displayed at the end of the report. The following information is shown: • total number of commands processed (`cmds=xxx`) • average real time per process (`Real=x.xx`) • average CPU time per process (`CPU=x.xx`) • average USER CPU time per process (`USER=x.xx`) • average SYS CPU time per process (`SYS=x.xx`) • average characters transfered (`CHARS=x.xx`) • average blocks transferred (`BLK=x.xx`) • average CPU factor (`USR/TOT=x.xx`) • average HOG factor (`HOG=x.xx`)
`-b`	Using this option will display the process records in reverse order: most recently executed commands will be shown first.
`-f`	Prints the *fork/exec* flag (`F` column) and process exit status (`STAT` column) on the report.
`-h`	Causes the optional `HOG FACTOR` column to be displayed, 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`	Replace the standard `MEAN SIZE(K)` column with `KCORE MIN`.
`-m`	Show the default column `MEAN SIZE(K)` on the report. This option is used to include `MEAN SIZE(K)` when it has been bumped off by another option. For example, `$ acctcom -km` produces a report showing both `KCORE MIN` and `MEAN SIZE(K)`.

`-r` Include the optional `CPU FACTOR` column in the report.

`-t` Show separate system and user CPU times (`CPU SYS` and `USER (SECS)` respectively).

`-v` Using this option will suppress the printing of 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 that are included in the report produced by *acctcom*. The table shown below defines and provides examples for each option:

Option **Description**

`-l` *line* Display only the processes that were executed from the user terminal **/dev/***line*. For example,

```
$ acctcom console
```

would display 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 cross-hatch # (`acctcom -u#`)
- a question mark ? (`acctcom -u?`)

If # is specified as the user name, then only the commands that require super-user priveleges will be displayed by acctcom. If ? is given as the user, then only the processes with unknown process IDs will be displayed.

As an example, the following two commands are equivalent:

```
$ acctcom -u 0
$ acctcom -u root
```

-g *group* — Show only the 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*; then 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 3:30pm:

```
$ 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 12:15am:

```
$ acctcom -e 0:15
```

-S *time* — Select processes **starting** at or after time where *time* is in 24-hour format. The following example would display all the processes that **started** at 1:30:42pm or after:

```
$ acctcom -S 13:30:42
```

-E *time* — Display only the processes that **terminated** at or before *time*, where time is in 24-hour format *hr*:[*min*[:*sec*]]. Note both the -S and -E options with the same *time* argument will cause *acctcom* to display only the processes that existed at the specified *time*. For example, to see all the processes that existed at exactly 30 minutes past noon, you would enter:

```
$ acctcom -S 12:30 -E 12:30
```

`-n` *pattern* — Show only the commands matching *pattern*. *pattern* can be a regular expression as described in *ed(1)*, except that + means one or more occurrences. For example, to display all processes that were created by executing the *ls* command, you would 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, you would type:

```
$ acctcom -n .*m.*
```

`-H` *factor* — Display only those processes whose hog factor exceeds *factor*. For example,

```
$ acctcom -H 0.85
```

would display all the processes that spent over 85% of their execution time in the CPU. You can use this option to find greedy processes—processes that are hogging the CPU.

`-O` *time* — Show only those processes whose system CPU time exceeds *time*, specified in seconds. The following example would be used to determine which processes took more than 8.25 seconds of operating system CPU time to execute:

```
$ acctcom -O 8.25
```

This option could be used to determine which processes are making heavy use of the operating system calls.

`-C` *sec* — Show only the processes whose total CPU time (`SYS + USER`) exceeds *sec* seconds. The next example would display all the processes that used over 5.28 seconds of CPU time to execute:

```
$ acctcom -C 5.28
```

`-I` *chars* — Display only the processes tranferring more characters than the limit given by *chars*. For example,

```
$ acctcom -I 10240
```

will display all the processes that transferred over ten kilobytes of characters ($10\,240 = 10 \times 1\,024$ bytes).

Sample Reports

The following sample report illustrates the use of *acctcom* without any options. The report generated is the standard report produced when no options are specified.

```
$ 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
```

Now display all the processes created between 7:00pm and 7:30pm by the user *root*. 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 to experiment with this command; it is very powerful and can provide you with much useful information if used properly.

Command Summary Report – acctcms

The *acctcms* command takes process accounting records as input; but instead of reporting on the individual processes, *acctcms* generates a report on the commands that generated the process accounting records. The action of *acctcms* can be summarized as follows:

1. *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 the *acctcms* options used, 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 the input process accounting files for which the command summary report is to be generated.

Producing a Readable Report – the -a option

By default, the output of *acctcms* is in internal summary record format; if you display it to your terminal, all you see is gibberish. To get a human-readable report, use the `-a` option.

The `-a` option causes *acctcms* to produce 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	The name of the command for which execution statistics are summarized. Unfortunately, all shell procedures are lumped together under the name **sh**, because only object modules are reported by the process accounting system.
NUMBER CMDS	The total number of times that the command was invoked.
TOTAL KCOREMIN	The total amount 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	The total CPU time that the named command has accumulated.
TOTAL REAL-MIN	Total accumulated real time seconds are displayed in this column.
MEAN SIZE-K	The average amount of memory (in kilobytes) consumed by the command.
MEAN CPU-MIN	The average CPU time consumed per command invocation is shown here; the following equation shows how it is computed: `MEAN CPU-MIN = TOTAL CPU-MIN / NUMBER CMDS`
HOG FACTOR	This column shows the average hog factor over all invocations of the command. It is computed as: `HOG FACTOR = TOTAL CPU-MIN / TOTAL REAL-MIN`
CHARS TRNSFD	This column shows the total number of characters transferred by the command. Note that this number may sometimes be negative.
BLOCKS READ	A total count of the physical blocks read and written by the given 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, several other options can be used 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`. This report can be used to determine which commands are using most of the computer's CPU time.
`-n`	Causes the report to be sorted 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**, `-o` elicits a report only 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 used together with `-a`, a combination prime and non-prime time report is produced. The output of this report is 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 (`pacct?`) and the current process accounting file (`pacct`). The final ASCII report is stored in the file `ascii_summary`.

```
$ acctcms pacct? > old_summary
$ acctcms pacct > new_summary
$ acctcms -as old_summary new_summary > ascii_summary
```

Sample Report

The ASCII reports produced by *acctcms* contain more than 80 characters per line. When these reports are displayed at an 80-column terminal, 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 capabilities, so that all of the report will fit on standard computer paper; or
2. a printer with enough columns to display all the information—for example, 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
COMMAND NUMBER    TOTAL     TOTAL     TOTAL  MEAN     MEAN      HOG    CHARS BLOCKS
NAME      CMDS KCOREMIN   CPU-MIN REAL-MIN SIZE-K  CPU-MIN   FACTOR TRNSFD   READ

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

Two commands—*acctprc1* and *acctprc2*—are used to create total accounting records from the process accounting files. The output from *acctprc1* is supplied as input to *acctprc2* which produces the total accounting records. These commands are normally invoked by *runacct* to produce daily accounting information.

acctprc1

This 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 sixtieth of a second)
- non-prime CPU time, also in ticks
- mean memory size (in memory segment units of 512 bytes each)

The format of *acctprc1* is:

`acctprc1` [*ctmp*]

Input must be redirected from a process accounting file.

The following example creates a file, `ascii_ptacct`, 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 >ascii_ptacct
```

Normally, *acctprc1* gets login names from the password file *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 login 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 login name.

acctprc2

This command reads from standard input records of the form created by *acctprc1*; it then summarizes the records by user ID and name, and then 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 |acctprc2 >ptacct
```

Charging Fees to Users – chargefee

System Accounting provides the capability to charge fees to specific users; the *chargefee* command is used to accomplish this task. *chargefee* allows you to charge generic *units* to a specific login name. The syntax of this command is:

`chargefee` *login_name number*

where *number* is the number of units to be charged to a particular user, and *login_name* is the login name of the user to whom *number* units are to be charged.

NOTE

number can be any whole number in the range -32 768 to 32 767; 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
```

Suppose you inadvertently charged 247 units to the user named **zimblits**, and you want to return his charges to their original value. You would enter the following:

```
$ chargefee zimblits -247
```

Summarizing and Reporting Accounting Information

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 – prtacct

The *prtacct* command allows you to display 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

prtacct 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	The user ID of the owner of the total accounting record—i.e., the ID of the user for whom the total accounting record was created.
LOGIN NAME	The login name of the owner of the total accounting record is displayed here.
CPU (MINS)	The total amount of 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	This represents a cumulative measure of memory and CPU time that a user 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. This information is created through process accounting commands.
CONNECT (MINS)	Identifies the 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. This column is also subdivided into PRIME and NPRIME columns. The connect session accounting commands are the source of this information.
DISK BLOCKS	The total number of disc blocks allocated to the user is shown here. This information is created via disc space accounting commands.
# OF PROCS	The total number process spawned by the user is displayed here. This information is created via the process accounting commands.
# OF SESS	This column shows how many times the user logged in. Connect session accounting commands create this data.
# DISK SAMPLES	This column indicates how many times the disc accounting was run to obtain the average number of disc blocks listed in the DISK BLOCKS column.
FEE	The number of fee units charged via *chargefee* is displayed here.

Example

The following example displays disc total accounting records. First the total accounting records are created via disc 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 "Disc Space Usage Accounting").
2. Only the columns relating to disc space usage have non-zero values, because the total accounting records were created only from disc space usage accounting commands.

```
$ for file_system in `cat /etc/checklist`
> do
> diskusg $file_system >dtmp.`basename $file_system`
> done
$ diskusg -s dtmp.* |sort +0n +1 |acctdisk >disktacct
$ prtacct disktacct "DISC TOTAL ACCOUNTING RECORDS"

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 SAMPLES
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       0
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 – acctmerg

Normally executed by *runacct*, the *acctmerg* command merges separate total accounting files into a single total accounting file. All the total accounting records for a particular user name and ID are merged together to form one total accounting record for the given user name and ID. This command is useful for merging disc, connect session, and process total accounting files together to form a single, comprehensive total accounting file.

acctmerg 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.

Command 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. Using the `-a` option causes *acctmerg* to produce 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 the default case, *acctmerg* assumes that its input files contain total accounting records. If `-i` is specified, then *acctmerg* will expect input files to be in the ASCII format created by the `-a` option.
`-p`	This option simply echos 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 <tot_acct_recs` **Note that `-t` and `-a` can be specified in any order, but they must be specified seperately as shown.**

-u Normally, *acctmerg* merges records that have the same user ID and user name. Using **-u** causes *acctmerg* to merge records on the basis of same user ID only—i.e., disregard the user name as a key on which to merge records.

-v This option causes *acctmerg* to produce 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 disc, process, and connect session total accounting records, merges them together, and stores the merged file in the file `merged_file`.

```
$ # First, create disc space usage total accounting records (dtacct)...
$ #
$ for fs in `cat /etc/checklist`
> do
> diskusg $fs >dtmp.`basename $fs`
> done
$ 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 p_file in pacct*
> do
> acctprc1 <$p_file |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

runacct is the main daily accounting shell procedure. It is normally initiated via *cron* during non-prime hours. *runacct* processes disc, 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 to the reader. (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 purposes. 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.*

runacct 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, a message is written to */dev/console*, mail is sent to *root* and *adm*, locks are removed, diagnostics files are saved, and execution is terminated.

The States of runacct

In order to allow *runacct* to be restartable, processing is broken down into separate reentrant *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	The command *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 `acct_bill` in the appendix "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 disc 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 already 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 being restarted and specifies the month and day for which *runacct* will rerun the accounting. The entry point for processing is based on the contents of *statefile*. To override *statefile*, include the desired 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**—gives a summary of resource usage per individual user: it basically merges all the total accounting records for individual users 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 yardstick for calculating "drain" on a system.

- **Monthly Total Command Summary**—This report is exactly the same as the Daily Command Summary, except that the Daily Command Summary contains command summary information accumulated only since the last invocation of *runacct*, while the Monthly Total Command Summary summarizes commands from the start of the fiscal period to the current date. In other words, the monthly report reflects the data accumulated since the last invocation of *monacct.*

- **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 – prdaily

As *runacct* finishes executing, it deposits a report of the current day's accounting in the file */usr/adm/acct/sum/rpt***mmdd**, 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. Previous days' accounting reports can be displayed by using the *mmdd* option and specifying the exact report date desired.
- The `-l` option prints a report of exceptional usage by login name for the specified date. This option is used to determine 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.
- Valid only for the current day's accounting, the `-c` option is used to get a report of exceptional resource usage by command. This option is used to determine 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 can be edited to reflect your system's needs.

The reports produced by *runacct* were described briefly in the previous sub-section. Now the reports are discussed in more detail.

Daily Line Usage Report

In the first part of this report, the FROM/TO banner should alert you to the period reported on. The times are the date-time that the last report 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 multi-user 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 numbber of minutes that the line was in use during the accounting period.
PERCENT	The percentage of TOTAL DURATION that the line was in use: 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. Generally, interrupts occur on a port when *getty(1M)* is first invoked when the system is brought down to a multi-user state. This column comes into play when # OFF exceeds # ON by a large factor. This usually indicates that the multiplexer, modem, or 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.

During real time, *wtmp* should be monitored as this 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 interrupting is occuring at a furious rate, general system performance 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 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 – monacct

monacct creates monthly summary files and reports; 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.

monacct should be invoked once each month or accounting period. Its syntax is

monacct *number*

where *number* indicates the which month or period it is (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, you must 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, you must 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 the 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 astrisk (*) 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. 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 1985, prime time at 9:00 a.m., and non-prime time at 4:30 p.m., the following entry would be appropriate:

   ```
   1982    0900    1630
   ```

   A special condition allowed for in the time field is that 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 *day_of_year* 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 may become corrupted or lost. Some of these files can simply be ignored or restored from the file save backup. However, certain files must be fixed in order to maintain the integrity of System Accounting.

Fixing wtmp Errors

The *wtmp* files seem to cause the most problems in the daily operation of System Accounting. When the date is changed and HP-UX is switched into multi-user 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 won't be caught by *wtmpfix* and 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 quite important. If *sum/tacct* ever becomes corrupted, then check the contents of *sum/tacctprev* with the command *prtacct*. If it looks 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

grpdusg

This shell script displays disc space usage totals for the users who are members of a specified group. The syntax of this command is:

grpdusg *group_name*

where *group_name* is the name of the group for which disc space accounting information is to be generated.

For example,

```
$ grpdusg pseudo
```

generates disc space usage information for all the users in the group `pseudo`.

The Shell Script

```
# Check for the group-name parameter.
#
if      [ $# -ne 1 ]
then    echo "\nUsage:  grpdusg group-name\n"
        exit 1
fi
echo    "\nOne moment please...\n"
#
# Use the find command to find all the files whose owners are members of
# group-name.  Pipe the output from find into acctdusg which will accumulate
# disc space usage information for the users in group-name.
# NOTE:
#        - accounting data is temporarily stored in _${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 "\nFiles having no owners exist--check _no_owners\n"
else    rm _no_owners
        echo "\nAll files have owners-- _no_owners not created\n"
fi
#
# Use echo and awk to display disc usage totals for this group.
#
echo "\nDisc space usage information (group is ${1}):\n"
awk 'BEGIN {print "\n_UID____USER NAME________BLOCKS"}
     { sum += $3 ;              # add up total disc blocks used
       print $0                 # display information for user
     END { print "\nTOTAL DISC SPACE USAGE= ", sum, "blocks" }' ${fn}tmp
#
# Remove temporary files, then exit.
#
rm ${fn}*
```

acct_bill

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/tacct**mmdd***). Output is written to the file *bills**mmdd***, 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/bills**mmdd***, 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:

- 10 cents for every minute of prime CPU time consumed
- five cents for every minute of non-prime CPU time consumed
- a half cent for every prime kcore minute used
- two-tenths of a cent for every non-prime kcore minute
- a 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 disc 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.

The Shell Script

```
_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.
#
acctmerg -a -t <$_infile >$_outfile  # output TOTAL amount first
acctmerg -a <$_infile >>$_outfile    # append users' total accounting records
#
# 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 used
             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\n", $1, $2, _sum   # display user total
     }' tacct_ASC.tmp_ >$_outfile  # write output from awk to appropriate file
rm tacct_ASC.tmp_                  # remove the temporary ASCII file
```

Using the FSCK Command **A**

Introduction

A file system consistency check (*/etc/fsck*) should be performed whenever the HP-UX Operating System is booted (brought up) and before the system is taken into state 2 (the multi-user state). As shipped, your system should do this automatically via the `bcheckrc` entry in */etc/inittab*. This precautionary measure helps to insure reliable file storage on your system's disc drive. Also, use */etc/fsck* to examine the file system **any** time you suspect problems with the HP-UX file system. The system should **always** be in a single-user state and quiescent (inactive and not being written on) before executing the *fsck* command. Only the system administrator should perform this check. If this check discovers an inconsistency, corrective action must be taken. This article will tell you how to use *fsck* to insure reliable system performance.

Note that *fsck* should be run until it executes without finding any major errors in the file system. Generally only one or two iterations of the command will be necessary to clean up the file system. If, after running *fsck* twice, serious errors still exist in the file system, there is a strong possibility of corruption on the disc. In this case the entire system may need to be restored from the most recent backup tape.

A directory with the name */lost+found* must exist on the file system being examined **before** *fsck* is run. */lost+found* should have been created on your root volume when you installed HP-UX on your system. The *fsck* command uses this directory for any problem files that it finds. The */lost+found* directory must have pre-allocated "slots" made by creating files in the directory and then removing them. To create this directory properly, type in the following commands (the commented lines are explanatory and do not need to be typed):

```
# create the lost+found directory
mkdir /lost+found

# go to the lost+found directory
cd /lost+found

# create approximately 100 files
touch `(cd /bin; ls)`

# remove the files we just created
rm /lost+found/*
```

The remainder of this article explains:

- how the file system is updated
- how the file system can be corrupted
- how corrective actions used by *fsck* can recover a corrupted disc

Updating the HP-UX 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. The problem occurs when this series of updating tasks is interrupted. Since some of these tasks may have completed, it is important to know their execution order so that good decisions can be made when repairing a corrupted disc.

There are five types of file system updates:

- Super-block
- i-nodes
- file-attribute file
- data blocks (directories and files)
- free map

Super-Block

The super-block contains:

- an identification of the disc format (HP-UX)
- a flag describing the integrity of the directory structure
- the root directory name
- the volume block size
- the location and size of the boot area
- the location of the file attributes file
- the largest addressable block

The super-block of a mounted file system is written to the file system whenever a *umount* or *sync* command is issued. The root file system is always mounted.

I-nodes

An i-node contains information about:

- the type of i-node (directory, data, or special)
- the name and i-node of it's parent directory
- the list of blocks and extents claimed by the i-node
- the size of the file
- protection, access, user and group affiliation
- date and time stamps

An i-node is written to the file system upon closure of the file associated with the i-node. All in-core blocks are also written to the file system upon execution of a *sync* system call.

File Attributes File

The file attribute file contains information about:

- the i-node for the root directory
- the free map
- an entry for each file on the file system
- file extents

The location of the file attribute file is maintained in the super-block, and is used in conjunction with i-nodes to maintain the location and integrity of the data in your HP-UX system.

Data Blocks

A data block may contain either file or directory entries. Each directory entry consists of a file name and an i-node number.

Data blocks are written to the file system whenever they have been modified and released by the operating system.

Free Map Blocks

The file attribute file contains a map of the file system (the free map), which is used to determine available file space. In the free map, each bit represents a block on the volume. If a bit is set, then it's corresponding block is being used; if not, then that block is free.

Free map blocks are written to the file system whenever they have been modified and released by the operating system.

Corruption of the File System

A file system can become corrupt in a variety of ways. The most common ways are **improper shutdown procedures** and **hardware failures**.

Improper System Shutdown and Startup

File systems may become corrupt when proper shutdown procedures are not observed:

- not having a quiescent system before stopping the CPU
- forgetting to *sync* the system prior to halting the CPU
- not using *stopsys(1M)* to stop the CPU, or using *stopsys* incorrectly
- physically write-protecting a mounted file system
- taking a mounted file system off-line

File systems may become further corrupted if proper startup procedures are not observed:

- not checking a file system for inconsistencies
- not repairing inconsistencies

Allowing a corrupted file system to be further modified can be disastrous.

Hardware Failure

While your Hewlett-Packard Series 500 computer system and discs are highly reliable, it is good to remember that any piece of hardware can fail at any time. This isn't a prediction of gloom, but merely a word of caution to you as the system administrator, to make small steps of precaution. By following the preventative maintenance outlined in your installation guides and in this manual, you should be able to avert any serious problems. Failures can be as subtle as a bad block on a disc pack, or as blatant as a non-functional disc controller.

Detection and Correction of Corruption

A quiescent file system may be checked for structural integrity by executing *fsck*. The *fsck* command checks on the 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.

When an inconsistency is discovered, *fsck* reports the inconsistency for the system administrator. The system administrator then must choose a corrective action.

Super-Block Consistency

One of the most commonly corrupted items is the super-block. The super-block is prone to corruption because every change to the file system's blocks or i-nodes modifies the super-block.

The super-block 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 *sync* command.

File-System Size and I-node-List Size

The file-system size must be larger than the number of blocks used by the super-block and the number of blocks used by the list of i-nodes. The number of i-nodes must be less than 65,535. The file system size and i-node list size are critical pieces of information to the *fsck* program. While there is no way to actually check these sizes, *fsck* can check for them being within reasonable bounds. All other checks of the file system depend on the correctness of these sizes. *fsck* also checks that the last allocated block can be read as a test for possible corruption of the super-block.

The file attribute file and the i-node list can be checked for:

- incorrect link counts
- blocks not accounted for anywhere
- bad i-node format
- files pointing to unallocated i-nodes
- i-node numbers out of range
- multiply linked directories
- link to the parent directory

Free Map

The free map is part of the **file attribute file** (also known as the file attribute list or FAL), a file which represents all of the inodes on a volume. This map describes all blocks on the volume, including the boot area, super-block, file attribute file, and the rest of the volume. In the FAL, inode 0 (the inode for the FAL) and inode 1 (the inode for the root directory) are first allocated.

Inodes 3 through the end of the free map are allocated to represent all the blocks on the volume. Each block is represented by a single bit in the free map. Since the free map takes up an integral number of inode entries in the FAL, and each inode entry takes 128 bytes (representing $128\times8=1024$ logical blocks on a disc), the number of inode entries that the free map will consume will be:

$$(\text{(number bytes on disc)} \div \text{(block size)} - 1) \div (128\times8) + 1$$

For example, consider an HP 7935 (a 404 Mbyte disc) which has been initialized with a logical block size of 1024. Using our formula, 386 inode entries in the FAL will be devoted to the free map, so inodes 3 thru 388 will be allocated as the free map on that disc.

fsck will check each free map entry for:

- blocks claimed by more than one i-node, or by the free map
- blocks claimed by an i-node or the free list outside the range of the file system

A check is made to see that all the blocks in the file system were found.

Executing the FSCK Command

As we just stated, in order for *fsck* to accurately diagnose (and potentially repair) your system, it **must** run in a quiescent state. Hewlett-Packard recommends that you only run *fsck* when you are in the single-user state (s). Since your system may have files in memory with the sticky bit set (or you may have a process such as *nftdaemon* with Local Area Net, which stays alive even in state s), you should do a soft reboot once you are in the single-user state by typing:

```
stopsys -r
```

For more information on shutting down your Series 500, see the "Shutting Down Your System" section of Chapter 5 (The System Administrator's Toolbox).

Command Options

If you are sure that you are in a quiescent state, you may now begin *fsck*. The command has the following format:

```
/etc/fsck -y -n -s -d special_file
```

where all the arguments are optional. *special_file* is the character special file of the volume you wish to check, such as `/dev/rhd`. Usually you will want to do an *fsck* on all volumes connected to your system. A list of these volumes should be contained in the file */etc/checklist*; if this is the case, you can omit this argument from the *fsck* command line:

```
/etc/fsck
```

The first to options, `-y` and `-n` unconditionally tell *fsck* what action to take should it find inconsistencies, before beginning. Generally, this is undesirable in both cases.

Unconditional Directives

By typing:

```
/etc/fsck -y /dev/rhd
```

you tell the program to take any corrective action, no matter what it finds. Corrective action sometimes involves the loss of data. If you are in the situation of potentially losing critical information, you may want to first consult your Sales Office for help in restoring this information (with tools for gurus such as *fsdb*). Using the `-y` argument prevents you from having this option.

Probably the only time you would ever want to use the `-n` argument is if you schedule *fsck* to execute at night (as in the *backup* command). This should really be avoided, since *fsck* was designed to be run interactively.

Reconstructing the Free Map

In the first pass of executing *fsck* you may find a conflict between the free map and a file. As an example, the file system may think a block on the volume is free, but you know that your mother-in-law's mailing list really resides there. Since you don't want to lose this data (do you?), you need some way to tell the system that a file really is using this block.

To do this, use the `-s` argument to unconditionally reconstruct the free map:

```
/etc/fsck -s /dev/rhd
```

When *fsck* executes, it will calculate what blocks are **really** free and then build a new free map in memory. When this is done, the program will overwrite the old free map (which resides in the file-attribute file).

If `-s` is used to correct a problem on a virtual memory device (which will be the case if you only have one disc on your system), there is a high probability that the final step in *fsck* will fail and you will be forced to re-boot. Should this occur, an appropriate error message will be printed. No damage should occur.

In any situation in which you use the `-s` option, once complete you should always execute *fsck* again to certify that your system is clean.

Verbose Listings

When checking the integrity of your file system, you may desire to have more than the cursory phase summaries printed at your console. This is especially true if you are trying to pinpoint where you are having problems. To do this, *fsck* has been written with the `-d` option.

```
/etc/fsck -d /dev/rhd
```

fsck was designed with five levels of debug information (hence `-d`); you can have additional levels of these messages printed by increasing the number of `-d`'s. Of course since there are five levels of debug information, the most you can do is:

```
/etc/fsck -ddddd /dev/rhd
```

Some caution should be used when specifying the debug option—using more than two `-d`'s will produce copious output (probably more than you would want to wade through, unless you are really serious about reading it). In the discussion which follows, we'll be walking through an *fsck* on an example system, with two levels of debug information.

A Walk Through

To best illustrate the use of the *fsck* command, let's walk through a file system integrity check on a common configuration. This *fsck* was done on a Model 540 (pod mount) system with one HP 7914 disc drive. Since our example system has only one disc, we know that:

- the HP 7914 will be the root file system,
- the root file system will be mounted,
- and this disc will also be the virtual memory device.

Normally you would want to run *fsck* on an unmounted file system, but since we are using the only disc, this is impossible. Because of this, *fsck* will print some warning messages—don't worry about them.

```
/etc/fsck -dd /dev/rhd

Checking /dev/rhd

WARNING: This device is the root device.

WARNING: This device is used by virtual memory.
```

These messages are normal on most systems, since the root and virtual memory volumes will be the device in */etc/checklist.* If you are not in a quiescent state, you may get a warning similar to:

```
WARNING: 22 superfluous processes are running.
For accurate diagnostics, only init, a shell, and fsck should be running.

Quit? yes
```

You do not want to run an *fsck* while in this state, so type `yes` in response to `Quit?` The program will terminate.

Volume Header Information

As the first step of the file system integrity checking process, a summary of volume header (super-block) information will be printed:

```
Volume Header:
        format = 0x700
        corrupt = 0
        block size = 1024
        max block = 129023
        FA file starts at block 69087
```

For more information on the meaning of this information, see the *fs(5)* entry in the *HP-UX Reference* manual.

Phase 1: Checking Directories

After printing the volume header information, *fsck* next creates a list of all extents (groups of contiguous disc blocks), that the system believes are claimed by a file or that are free (as identified by the free map). Each entry in the list consists of the starting address of an extent and the size of the extent.

During this phase of its execution, *fsck* also traverses all directories in the file system, creating a map of all i-nodes pointed to by directory entries (a map of all i-nodes in the file system). Since we are printing two levels of debug information, we will also see which directory is currently being searched:

```
**Checking Directories.
        Finding inodes.
        Traversing directories.
        Processing directory /users/hpux/marka/BASIC/CH11
        Processing directory /users/hpux/clarke
        Processing directory /users/hpux/michael
            .                .                .
            .                .                .
            .                .                .
```

```
Processing directory /users/hpux/michael/man1m
Processing directory /users/hpux/marc
    .                .                .
    .                .                .
    .                .                .
Processing directory /image/demo/LIBRARY
Processing directory /image/docs
Building extent tree.
```

As you can see, printing all directories on the system produces copious output; in our example, *fsck* produced 12 pages of `Processing directory` messages (our example is a 22 user system). Following all these messages are some summary statistics.

```
Free map built.  Number of free blocks = 19255.  Last block = 129023.

Max tree depth = 133

Number of nodes in tree = 15522
```

Don't be alarmed if an *fsck* on your system results in values different from these (even if you have an HP 7914 connected to a Model 540); how a system is used, how big it's boot area is and other factors affect *fsck* results more than the actual hardware. What you should be concerned about are results which totally conflict with anything you've seen in the past.

Phase 2: Checking Blocks

Once phase 1 (checking directories) is completed, phase 2 begins in which *fsck* begins looking for inconsistencies. It examines the list of extents built previously in phase 1. The list identifies all disc allocations. In a consistent file system, this list includes every block on the disc. Any extents not claimed by either a file or the free map are reported at the end of this process along with the total number of unclaimed blocks in the file system.

```
**Checking Blocks.

Number of tree nodes checked:  15522
Number of unclaimed blocks:  23.

MULTIPLE FILES CLAIM THE SAME SPACE ON THIS DISC!!
Multiply claimed extents:
        82344 to 82503, claimed by inodes 2220 and 9340

Unclaimed extents:
        54896 to 54907
        101595 to 101605
```

Similarly, any extents claimed by more than one i-node are reported at this time. The system issues a report as shown above, where `82344 to 82503` identifies the disc blocks in the extent claimed by inodes `2220 and 9340`. When an i-node value equals -1, it specifies that the free map is one of the i-nodes claiming the extent. If one of the i-node values reported is the free map, re-running *fsck* with the `-s` option corrects the situation by re-building the free map. However, if neither i-node is the free map, then two valid files claim the same extent. In this instance you should examine the file associated with each i-node and remove one of the files (usually the oldest file is removed). To find the name of each file associated with each i-node, enter the following commands:

```
/bin/find  /  -inum 2220 -print
/bin/find  /  -inum 9340 -print
```

Once you know the name of each file claiming a common extent, you can examine each file and remove the file(s) that are improperly claiming the extent. When one of the files is removed, its claim on the extent is transferred to the free map. Then, when *fsck* is re-run with the `-s` option, the situation is corrected. *fsck* with the `-s` option must be run immediately after removing the file to prevent the operating system from reallocating its space to yet another file. A consistent file system contains no multiply-claimed extents.

Phase 3: Checking the File Attribute File

In the next phase, *fsck* attempts to locate orphaned files. First, the list of i-nodes created in step 1 above is examined. Any i-node not included in the list is examined to identify the disc space it claims. An i-node can claim space and yet not be recognized as a file if the i-node exists but has no links from a directory. If the space claimed by the i-node is part of the unclaimed blocks list, the file is termed an "orphan" (how terrible). An orphan is an i-node that represents a valid file but has no link from a directory.

```
**Checking the File Attribute File
```

The system next examines the size of the orphaned file (i-node). If the size is 0, the system asks if it may clear the i-node and restore its extents to the free map. An affirmative reply corrects the inconsistency while a negative reply causes no action and leaves the file system corrupt. If the size of the orphan is non-zero, the system asks if it should attempt to restore the file. An affirmative response causes the orphan's i-node to be linked to the *lost+found* directory located at the root level of the volume you are checking. If you are checking the root file system, this directory is called */lost+found* and is shipped with your HP-UX system. If you are checking an unmounted file system volume, then the *lost+found* directory is located at that file system's root level. For example, assuming that you have a file system called "/database", the directories full pathname is "/database/lost+found".

NOTE

If you initialize and mount an independent file system volume to the file system root, you must create the *lost+found* directory for it. Once this directory is created, unmount the volume before using *fsck*. You cannot execute *fsck* on it unless the *lost+found* directory already exists.

Phase 4: Checking Extent Maps

I-nodes are not the only aspects of the file system that can be "orphaned", and the first thing phase 4 does is to check for orphaned extent maps. When an i-node claims more than four extents, it must use one or more extent maps to identify the additional extents. Occasionally, when a file becomes corrupt, the i-node may be cleared but the extent map(s) claimed by the i-node is not cleared. When this happens, an orphaned extent map is created.

```
**Checking Extent Maps.
```

If the orphaned extent map claims blocks listed in the unclaimed block list (created in phase 1), and if all orphaned files have been restored, the system asks if you wish to add the blocks claimed by the orphaned extent map to the free map. An affirmative response causes the claimed blocks to be added to the free map. A negative response causes no action; the file system remains corrupt.

Phase 5: Checking Link Counts

The system begins verifying the number of links to each i-node in phase 5. Each i-node keeps track of the number of directory entries linked to it. Additionally, as *fsck* executes, it creates a separate list with which to cross check the i-node link count. If the link counts disagree, the system displays the messages:

```
**Checking Link Counts.

     Wrong link count for inode 3042.
     It's currently 4 but should be 1.
     Shall I fix it? yes
```

where 4 is the link count maintained by the i-node and 1 is the link count created by *fsck*. An affirmative response to the prompt resets the i-node link count such that it agrees with the count maintained by *fsck*. A negative response prohibits corrective action from taking place; the file system remains corrupt.

Phase 6: Checking The Free Map

If, during any of the previous steps, you have failed to take a suggested corrective action (and thus leaving the file system corrupt), the following message is displayed and the checks described in this step are not performed:

```
**Unable To Check Free Map.
```

This message specifies that the file system still has some uncorrected problems. If the file system is consistent and no further corrections are needed, the following message is displayed and the checks described in this step are not performed:

```
**No Need To Check Free Map.
  There are no unclaimed blocks.
```

This is an informational message stating that nothing is wrong. If all of the corrective actions specified by *fsck* have been taken, and if the free map is not known to be current, *fsck* displays:

```
**Checking The Free Map.
Shall I return lost blocks to the free map?  yes
```

Next, the current state of the file system is compared to the disc's free map. If the system finds an inconsistency between the two, a message is displayed specifying that the free map is out of date. You are then asked if you wish to update it (the free map). An affirmative response causes the system to re-build the free map such that it is consistent with the current state of the file system. A negative response causes no corrective action to be taken; the file system remains corrupt.

Phase 7: Updating Kernel Data Structures

Finally, the system updates the internal information it keeps about the file system root or virtual memory volume, as indicated by the display:

```
**Updating kernel data structures.
```

If this update fails (the update will always fail when checking an unmounted volume), the system displays the message:

```
Kernel failed to update its data structures for /dev/rhd.2
```

If, as in our example system, the volume being checked is the file system root or the virtual memory device, you should re-boot the system at this time. You will get a message similar to this:

```
Kernel reclaimed 23 blocks for existing virtual objects.
```

Do not be alarmed! This is normal for an *fsck* of the root and virtual memory volume. If the update succeeds (or the volume being checked is not mounted), *fsck* next reports:

```
/dev/rhd statistics:
        total number of files:   9376
        total number of blocks:  129024
        number of user blocks:   106654
        number of free blocks:   19278
        percent of disc unused:  14
```

If the file system has not been fully repaired, the following message is displayed:

```
THIS FILE SYSTEM IS NOT COMPLETELY RESTORED.
```

You should fully repair the system before allowing access to it. However, some errors in the file system are more dangerous than others. Unless the file system contains multiply-claimed extents, you can probably use the file system without further corrupting the system. If the file system contains multiply-claimed extents, you must correct the situation (by re-running *fsck*).

NOTE

You should always be able to successfully execute *fsck* without errors before you restore your Series 500 system to normal use.

Occasionally, a file system is too corrupt to be repaired by *fsck*. In this instance, you must initialize the media and then restore the file system from its last backup. If the HP-UX file system is too corrupt to be repaired, you may have to re-install the system and then restore the file system from your last backup. In the event that you do not have the current installation tapes, you should consult your Sales Office for assistance.

FSCK and Virtual Memory

One of the devices that *fsck* can be called upon to check is the virtual memory device (as in our example). Unfortunately, *fsck* requires a great deal of memory for its own use and this means that the file system is often not quiescent when checking the virtual memory device. However, consider the following points:

- *fsck* reads the free map fairly early in the checking sequence before it gets the space to keep track of file system extents. This means that *fsck* thinks that any existing virtual memory extents are unclaimed (and treats them as such). A number of these extents may be associated with *fsck*'s encompassing shell, existing processes, and what little memory *fsck* has already gotten.
- As *fsck* grabs more memory, it is not reflected in its internal copy of the free map.
- When *fsck* rebuilds the free map, all virtual memory extents are marked as "free". Although this is a potentially dangerous situation, *fsck* immediately notifies the operating system, which restores the true status of the virtual memory extents in the free map.

Thus, under normal circumstances, it should be safe to run *fsck* on any virtual memory volume. However, if there is any other virtual memory activity occuring on the volume, especially while the operating system is updating its virtual memory data structures, the volume could be corrupted or destroyed.

Notes

System Accounting Files

B

Descriptions of the different files processed by HP-UX System Accounting are found in this appendix. The files are grouped according to the directory in which they are found.

Files in the /usr/adm directory

Filename	Contents
diskdiag	Diagnostic output from the execution of disc space accounting commands.
dtmp	Output from the *acctdusg* program.
fee	Output from the *fchargefee* 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. *active***MMDD** 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 being executed by *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

Filename	**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

Filename	Contents
cms?	Total command summary for month *?* in internal summary format.
fiscrpt?	Report similar to *prdaily* for the month *?*.
tacct?	Total accounting file for the month *?*.

Notes

System Loader Messages C

The **system loader** is a program which permanently resides in the computer and causes the computer to search for and load an operating system. If the computer is unable to locate and load the TEST System, (or any operating system), a message is displayed. Each of these messages is explained below. Possible causes for many of the messages are provided. If the message begins with `ERROR:`, the system halts after issuing the message. If the message begins with `NOTE:`, the message provides information and the computer continues operating.

If the message you receive indicates a hardware failure, run the Module Self Test or, if present, the System Integritry Test before calling your HP Customer Engineer for service.

Often, the computer attempts to identify the device to which it was "talking" when the message was generated. The trailer `SELECT CODE NN` is appended to the message to indicate which select code (I/O port) of the computer caused the message. Select codes 0 through 7 are on the computer and are controlled by the first **I/O processor** (IOP). Select codes 8 through 15 are on the first I/O expander and are controlled by the second IOP. Select codes 16 through 23 are on the second I/O expander and are controlled by the third IOP (only 2 IOP's can be used with the Model 550).

How the select codes 0 through 7 are associated with the slots in your computer's I/O card cage depends on what model you have:

- The Model 520 has four I/O slots available which are associated with select codes 2 through 5, the top one being select code 2. Select codes 0, 1, 6, and 7 refer to the internal peripherals.
- The Models 530 and 540 have 7 I/O slots available which are associated with select codes 0 through 6, the top one being select code 0. Select code 7 refers to the internal SCM board.
- The Model 550 has 7 I/0 slots available which are associated with select codes 0 through 6, as well as an internal HP-IB card. If the internal HP-IB is used, it is counted as select code 5 by default and you only have 6 available slots left (however, the HP-IB can be set to be on select codes 0 thru 6). Select code 7 refers to the internal SCM board (the internal physical printed circuit board actually contains both the HP-IB and SCM functions).

On an I/O expander, the upper left slot, when viewed from the rear, is always the lowest number select code on that I/O expander.

As an example, suppose the message:

```
NOTE:  BAD CARD OR DEVICE:  SELECT CODE     18
```

is printed. This indicates that the failure is associated with select code 18, which is the third slot on the second I/O expander. In this case, either the I/O card in that slot has failed its self-test or if it is an HP-IB card connecting a mass storage device, the mass storage device failed its self-test.

Messages

`Loader XXX`—informational message identifying the revision of the system loader. This message is usually followed by a single line message identifying the operating system the computer is attempting to load.

`Testing Memory ...`—informational message that follows the `Loader XXX` message indicating that the loader is performing memory tests and configuring memory. This can take up to 15 seconds.

`Looking for System ...`—informational message that follows the `Testing Memory ...` message indicating that the loader is searching for an operating system.

`Please mount next volume.`—informational message. The loader is ready to load another portion of the operating system. Mount the volume containing an un- loaded portion of the operating system. Volumes may be mounted in any order without affecting the loading process.

`SYSTEM NOT FOUND; WILL RETRY IN XXX`—unable to find an operating system on any mass storage device. The loader will attempt to find an operating system again in `XXX` seconds. Possible causes: mass storage device not powered up, no media in mass storage device, wrong disc in disc drive, computer or mass storage device hardware failure, media failure, incompatible loader/system revision numbers, etc.

`BAD SYSTEM FILE:  SELECT CODE NN`—operating system loaded; however, an error has been detected in the operating system code during loading. Possible causes: corrupt system, media failure, mass storage hardware failure or computer hard- ware failure.

`NOT ENOUGH USABLE MEMORY; TOTAL IS XXXX`—The amount of usable memory is too small to load the operating system. The total amount of good memory is: `XXXX` bytes. The amount of memory available for the operating system is `XXXX`−98 304 bytes. Possible causes: corrupt system or hardware (memory) failure.

`BAD CARD OR DEVICE: SELECT CODE NN`—informational message. A hardware failure has been detected (interface card or mass storage device did not pass Module Self-Test). The loader continues searching for an operating system.

`MEDIA/DEVICE NOT READY: SELECT CODE NN`—while loading, the media (volume) was removed from the device (e.g. a floppy disc was pulled out of a disc drive), the device went offline, or a hardware problem caused the device to become otherwise "not ready".

`UNRECOVERABLE DATA: SELECT CODE NN`—part of the operating system is not readable. Possible causes: media failure or mass storage hardware failure.

`END OF VOLUME: SELECT CODE NN`—attempt to address or read past the end of a volume. Possible causes: corrupt system, media failure or mass storage device hardware failure.

`CTRLR/UNIT FAULT: SELECT CODE NN`—hardware passed initial self-test: however, it failed while being used to load the operating system. Possible causes: computer (interface card) hardware failure or mass storage device hardware failure.

`IO TIMEOUT: SELECT CODE NN`—mass storage device failed to respond fast enough while attempting to load from it. Possible causes: computer hardware failure or mass storage device hardware failure.

`CS80 DEVICE: SELECT CODE NN`—indicates a mass storage device hardware failure.

`TAPE DEVICE: SELECT CODE NN`—usually indicates a tape device (HP 7970, HP 7974, HP7978) hardware failure. Can also indicate a failure on the HP 27110A HP-IB Interface. Tape errors covered are: "Command Rejected", "Interface Busy", "Rewinding", "Tape Run-Away", "Data Timing Error", and "Command Parity Error".

`HPIB CARD: SELECT CODE NN`—transaction to the indicated HPIB interface card was terminated due to a probable interface card failure.

`KBD/SCM NOT FOUND.`—indicates a computer hardware failure (keyboard) on a Model 520 Computer; computer hardware failure (system control module) on a Model 530, 540 or 550 Computer.

`BAD IO BUS: SELECT CODE NN`—indicates a computer hardware failure on the computer's first I/O Processor.

`BAD NVM: SELECT CODE NN`—indicates that NVM (Non-volatile Memory) failed self-test. Possible cause: computer hardware error.

`BAD RTC: SELECT CODE NN`—indicates that the computer's built-in real time clock is not functioning.

`BAD SP: SELECT CODE NN`—indicates that the Model 530/540 Computer's service processor failed self-test.

Index

a

b

C

d

e

f

g

h

i

k

l

m

n

o

p

r

S

t

u

V

W