Using HP-UX

HP 9000 Computers

E0997

HEWLETT® PACKARD

HP Part No. B2355-90164
September 1997

Printed in: USA

Legal Notices

Warranty

Restricted Rights Legend

Trademark Acknowledgement

Contents

Contents

Contents

Contents

Contents

Figures

Figures

Tables

Tables

About this Manual

Printing History

The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. The manual part number will change when extensive changes are made.

Manual updates may be issued between editions to correct errors or document product changes. To ensure that you receive these updates or new editions, see your HP sales representative for details.

August 1992...Edition 1...B2910-90001. This edition incorporates material from *A Beginner's Guide to HP-UX*, Edition E0191, with the addition of updated material for HP-UX 9.0, HP VUE, the System Administration Manager, and Instant Ignition. This manual applies to HP 9000 Series 300, 400, and 700 computers.

January 1995...Edition 2...A1700-90014. This edition includes information covering the 10.0 release of HP-UX. References to HP VUE have been removed because HP VUE information is contained in *Using Your HP Workstation* and the *HP Visual User Environment 3.0 User's Guide*. This manual applies to HP 9000 Series 800 computers.

September 1997...Edition 3...B2355-90164. This edition includes information covering the 11.0 release of HP-UX for HP 9000 computers.

How to Use this Guide

This guide covers everything you need to know to begin using your new Hewlett-Packard computer. It provides step-by-step instructions for basic tasks such as copying files, printing documents, and sending electronic mail.

This guide contains the following information:

- *Chapter 1, Getting Started* provides an overview of your system and explains how to log in and log out.
- *Chapter 2, Working with Files and Directories* shows you how to create, view, print, and remove files; create and remove directories; and navigate between directories.
- *Chapter 3, Using Your Shell* explains how to use command syntax, redirect command input and output, and set your login environment.
- *Chapter 4, Using the vi Editor* describes how to create and edit text using the `vi` editor.
- *Chapter 5, Using Electronic Mail* shows you how to send and receive messages electronically.
- *Chapter 6, Communicating over a Network* describes how to use and communicate with remote systems.
- *Chapter 7, Making Your System Secure* covers system security and changing file and directory permissions.
- *Appendix A, HP-UX Quick Reference* contains tables summarizing useful HP-UX commands.
- *Appendix B, Doing Advanced HP-UX Tasks* provides pointers for more information about advanced and system administration tasks not covered in this guide.
- *Appendix C, Scheduling Commands* describes ways to run programs automatically at specified times.
- *Appendix D, Using the Key Shell* discusses an alternative command intepreter (shell) that is very user friendly.
- *Glossary* explains common HP-UX terms.

Comments about this manual (no techical questions, please) can be directed to: editor@fc.hp.com. Please consult your HP service and support representative for technical support questions. Thanks.

Typographic Conventions

This guide uses the following typographic conventions:

Boldface	Words defined for the first time appear in boldface. For example, an **argument** is the part of a command line that indicates what file or directory the command is to act on.
`Computer`	Computer font indicates literal items displayed by the computer. For example: `file not found`
`User input`	Bold, computer text indicates literal items that you type. For example: **`cd`**
Italics	Manual titles and emphasized words appear in italics, as do values that you supply. For example, in the command below you would substitute an actual directory name (such as `mydir`) for *directory_name*. `cd` *directory_name*
Enter	Text in a bold, sans serif font denotes a keyboard key. A notation like **CTRL**+**Q** indicates that you should hold the control key down, then press **Q**.
Softkey	Select an on-screen item or a corresponding softkey. For example, `Help` shown at the bottom left side of the screen means that pressing the softkey corresponding to that position on the screen (f1) will cause a help screen to be displayed.

1 Getting Started

Welcome

Your new Hewlett-Packard computer uses the HP-UX operating system. HP-UX is a versatile operating system that meets the computing needs of diverse groups of users. You can use HP-UX simply to run applications, or you can develop your own applications in its rich software development environment. In addition, HP-UX offers powerful subsystems, such as electronic mail, windows, networking, and graphics.

Although some would say that the HP-UX operating system needs a "Survival Guide" more than a "Users Guide", it's really not that difficult to learn commands that will help you work more productively. Take your time and don't worry if you don't understand everything the first time you see it. Just keep in mind that most people don't like typing, so many commands and methods are shorter than you may think.

Overview of Your System

Installation

Your system should be installed and ready to use. If you have not yet installed your system, please see the *Hardware Installation Guide* or *Owner's Guide* that came with it.

Also refer to the *Owner's Guide* for instructions on starting your system, performing initial configuration, and adding new user accounts.

User interfaces

There are two common ways to access HP-UX.

- With a Graphical User Interface (GUI) such as HP’s Common Desktop Environment (CDE) where you use a mouse and click on icons, or,
- With a textual user interface where you type simple commands into a text window.

This manual describes the latter method even though the next item briefly describes the CDE.

CDE

The HP Common Desktop Environment is a powerful graphical environment that provides an interface to HP-UX. It has several components that help you use your system faster and more intuitively:

- Windows let you run more than one application at a time.
- Workspaces allow you to have multiple work areas.
- Icons let you directly manipulate files and applications.
- Front Panel controls and toolboxes for easy access to applications.
- Extensive online help, provided by the HP Help Manager.
- Easy customization for colors, fonts, and other aspects of the appearance and behavior of your workstation's interface.
- Multimedia (images and audio) applications for playing, recording, and editing audio, and capturing and viewing images.

For information about HP CDE, see the *HP CDE User’s Guide*.

Having now touched upon the easy way of working with HP-UX, we’ll go back down to the low-level textual methods which exist underneath the glossy user interface.

Multi-User Systems

HP-UX is a multi-user operating system More than one person can be using the system at the same time. To prevent users from interfering with each other's work, most users cannot make changes to certain parts of the operating system. Because of this, there has to be at least one person who does have the ability to change the system. That person is generally called the "superuser", "root user", or system administrator.

System administrator

Throughout this guide, you will see the term **system administrator**. The system administrator is someone who manages your system, taking care of such tasks as adding peripheral devices, adding new users, and doing system backups. In general, this person (who may also be called the system operator or the "superuser") is the one to go to with questions about implementing your software. You may find it helpful to take along cookies or other snack food when visiting your system administrator.

However, if you are the only user on a system, you will need to be your own system administrator. In that case, whenever this guide refers you to the system administrator, you should be able to get help from the system administration manuals that you purchased with your system (especially *Managing Systems and Workgroups*) and from other documents that may be installed on your system. Additionally, your HP support engineer can also provide installation and maintenance help, in accordance with your support contract.

SAM

Your system includes a powerful administration tool called System Administration Manager (**SAM**) that has complete online help to guide you through system administration tasks. Use SAM only if a system administrator is *not* available. To start SAM, **log in** as the **superuser** and type `/usr/sbin/sam`, then press **Enter** (see the next section for details on performing this task). For more information about SAM, see the *Managing Systems and Workgroups* manual.

Trusted Systems

In most cases the standard level of HP-UX system security is adequate to keep your information safe and documents private. However, some companies or agencies may wish to use a higher level of security. If you are working in such an organization, consult you group about using this optional feature.

C2 security

Your HP-UX system can be configured as a C2-level trusted system, as described in Section 2.2 of the Department of Defense Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, December 1985.

When appropriately configured as a trusted system, HP-UX provides additional security features such as discretionary access control and system auditing.

Security policy

A “security policy” is a statement of the rules and practices that regulate how an organization manages, protects, and distributes sensitive information. HP-UX C2-level security expands on existing HP-UX security mechanisms and provides procedures and guidelines to help enforce your company’s security policy.

The Hewlett-Packard C2-level trusted system is made up of the HP-UX operating system configured in trusted mode and its commands, utilities, and subsystems along with supported hardware. The system’s application interface must work correctly and satisfy the computing needs of its system users. The system’s security features provide the mechanisms necessary to enforce your site’s security policy and protect system users and their data from threats.

C2-level security features are documented as needed within this manual and additional security information is included in Chapter 7, “Making Your System Secure.”

Logging In and Out of HP-UX

Login accounts

This section explains how to log in, log out, and create user accounts (if necessary) using the command line.

Get your **user name** and **password** from the system administrator or the support person who installed your system.

Superuser

The root user, or **superuser**, is a special user. When logged in as the superuser, you have the permission required to perform all system administration tasks. Normally, the system administrator is the only one who logs in as the superuser. If you are not the system administrator, you will not log in as the superuser to perform daily tasks.

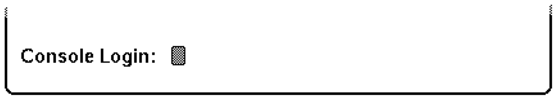
Root login

However, the very first time you (or anyone else) log into your new workstation, you must do so as `root` (`root` is the user name for the superuser). This is because no other **user accounts** have been created yet. Once accounts have been created for other users, you should log out as superuser, then log back in as one of those users.

Logging In

When **Logging In** you will see a login prompt:

Figure 1-1 **Type your user name at the login prompt.**



```
Console Login: ▩
```

Trusted systems restrictions

Your system administrator may set an authorization list for a terminal limiting who can log on to specific terminals. You will only be able to log on to terminals to which you have access in this case. The system administrator can also set time-of-day restrictions on accounts, and you cannot log in if you try to log on at a time you are not authorized to do so. Your system administrator should inform you of any restrictions on your account.

Logging In

Logging in gives you a secure way to access your system. You log in by typing a user name and a personal password.

1. If you have a user account, type your **user name** after the `login:` prompt and press **Enter**.

   For example:

   `leslie` **Enter**

   If you don't have a user account yet, ask your system administrator to create one, or follow the instructions in your *Owner's Guide*. Until you get a user account, you may log in as superuser.

   To log in as superuser, type the following at the `login:` prompt:

   `root` **Enter**

2. If you are logging in with your own user name, or if a password has been set for `root`, you must enter the correct password at the `Password:` prompt. Type the password and press **Enter**.

   The password does not appear on the screen.

3. The copyright notice appears briefly, followed a message asking about your console type. Type **`y`** and press **Enter**.

4. If you logged in with your user name, lines similar to these appear:

```
TERM = (hp)
$
```

Superuser prompt

If you logged in as superuser, lines similar to these appear:

```
Value of TERM has been set to "hp".
WARNING: YOU ARE SUPERUSER !!
#
```

Remember to get a user account for yourself (ask your system administrator, or see your *Owner's Guide*).

You are now logged in.

At this point there are literally hundreds of commands you can use to perform many types of tasks. Too bad you don't know any yet. Keep reading and you'll learn many of the basic commands.

Using HP-UX Commands

Shell

Most of this guide assumes you will be using the POSIX command interpreter, commonly called a **shell**. (Other shells are available and are discussed later in this book.) With a shell, you run a command by typing the command's name on a command line and pressing the **Enter** key. Note that **Enter** is the same as **Return** on some keyboards.

Prompt

The command prompt indicates that the shell is ready for data input. Most users have a "$" as a prompt. The system administrator usually has a "#" as a prompt.

Thus, when you see a shell and see a prompt, it looks like this:

```
$
```

For example, you can run the `date` command by typing:

date(1)

```
$ date Enter
Thu Aug 21 15:26:23 MDT 1997
$
```

Congratulations, you've entered your first command. You might enjoy trying these other "safe" commands: cal, `id`, `who`, `whoami`, and `pwd`.

NOTE

What Does That "(1)" Thing Mean?

The "(1)" indicates the type of HP-UX command. Type "(1)" commands are general commands that any user can execute. System maintenance command are marked as "(1M)" commands. Usually only the system administrator can use type "(1M)" commands. We'll learn more about this in later chapters.

The rest of this guide explains how to use HP-UX commands, and Chapter 3, "Using Your Shell," explains shells. To get visual help directly from the screen while you are working with commands, see the following sections.

Logging Out

After you are done working for the day, you should log out. Logging out prevents other users from accessing your system. However, it does not prevent others from accessing your work over the network—to protect your work, see Chapter 7, “Making Your System Secure.”

1. If you are using any applications, save your work, then exit the application.
2. At the shell prompt, type `exit` and press **Enter**. If you have several work sessions (shells) opened, you may have to type `exit` several times before you return to the login prompt. If you are using CDE, you can press the “Exit” button to logout.
3. Once you have logged out, the system should display a “login:” prompt or display a login screen.

On or off?

After logging out, *do not* turn your computer off. Your computer is a multi-user system, and other people may be using it. If you turn it off, you will deny them access to the computer, and may cause them to lose some of their work. They may find interesting and unique ways of expressing their displeasure with your actions.

shutdown(1M)

If you have to turn off your computer, see *Managing Systems and Workgroups* for information on shutting down. Typically, the superuser will use the shutdown(1M) command or the reboot(1M) command to shut down a system.

Some of the newer workstations and servers will safely shut themselves down by simply pressing the power switch. Older systems will typically “crash” if you remove power without going through the shutdown procedure. If a system crashes, it usually means that it will take more time, sometimes hours, to repair the file system and return to a usable state. If possible, ask someone if it’s safe to power-down your system before attempting to determine it for yourself.

Modifying System Parameters

Use this section only if you need to add or modify the system parameter information that was set the first time you turned on your system (see your *Owner's Guide* for details). Such modifications should be made as soon as possible after initial installation.

Log in as the superuser and type the following command:

set_parms(1M)

`/sbin/set_parms` *option* **Enter**

Where *option* is one of the following:

Option...	Modifies or sets...
`hostname`	System host name
`timezone`	Time zone
`ip_address`	Internet Protocol address
`addl_netwrk`	Additional network parameters
`font_c-s`	Network font service

Any changes you make in `set_parms` will take effect after rebooting the system.

You can also use SAM to add or change most of this information.

Changing Your Password

Typically the system administrator assigns the first password to an account. It is wise to change the password to one that only you know as soon as is conveniently possible. For security reasons, you should frequently change your password.

A password must contain at least six characters. At least two characters must be alphabetic, and one of those characters must be a number or a special character (such as a dash (-), an underline (_), or an asterisk (*)). The password cannot contain your **user name**, or even a reversed version of your user name (for example, if your user name is `bif` your password cannot contain `fib`). Also see “Choosing a Secure Password” in Chapter 7, “Making Your System Secure.”

Examples of valid passwords are: `wild-life`, `!secret`, and `*fuzzy*`.

When you log in to the system, you may see a message informing you that your password is about to expire. In this case, you must change your password:

To change your password, from a command line shell prompt, you can use the `passwd` command to set or change a password. Type:

passwd(1)

```
$ passwd Enter
```

You will be prompted for your old password. Then you will be prompted to enter and re-enter your new password. The re-entered password must match the first entry.

Make sure you do not forget the password you use. Writing down your password on your forehead or anywhere else will defeat the purpose of the password. You will need to keep it secret. If you forget your password, contact your system administrator, or log in as the superuser and set a new password with the SAM utility.

Trusted systems

On a trusted system, you will be required to change your password periodically. This is called password aging. Also, typically you have three tries to log in successfully. If you still fail to log in, you may not be able to log in again at that time. It is possible that your system administrator may have configured your workstation so it locks you out for some period of time after some number of failed login attempts. For additional information, see Chapter 7, “Making Your System Secure.”

Finding Information

Manuals

This section lists some common manuals.

System installation

- If you need help with system hardware installation, see the *Owner's Guide* for your system.
- If you need help with peripheral installation, see the *Owner's Guide* for your system and/or the *Configuring Peripherals* manual.
- If you have not yet installed your HP-UX system, see the current *Installing HP-UX* manual.

HP-UX usage and administration

- For general usage of HP-UX, continue reading this guide.
- For HP-UX system administration and troubleshooting information, see the *Managing Systems and Workgroups* manual.

  For most system administration tasks, you can use the SAM tool. SAM contains an extensive online help system to help you perform system administration tasks.

HP CDE usage and administration

- For basic HP CDE information, see *CDE User's Guide*.
- For advanced HP CDE configuration and system administration, see the *CDE Advanced User's and System Administrator's Guide*.

X Window System

- See *Using the X Window System*, HP part number B1171-90076.

manuals(1)

For a complete list, see the *manuals*(1) manual reference page. To do that, see the next section.

Within the United States, you can order any of the following manuals by calling Hewlett-Packard at 1-800-227-8164. In other countries, contact your nearest HP Sales and Support office.

CD-ROM

These manuals are also available on CD-ROM, with the optional HP-UX Instant Information product. For further information on HP II, contact your HP Sales and Support representative.

Online Documents

There are also online documents to be found on your system. Many of these documents are in the `/usr/share/doc` directory.

Manual Reference Pages

Man pages

The *HP-UX Reference* contains reference entries (also called manual pages or “man pages”) for nearly every HP-UX command.

These manual reference pages provide command syntax and a detailed description of the command and its options and arguments. The description may include examples of command usage and provide other information such as system files used and related commands.

To display the man pages from the command line, type `man` *command_name* at the command prompt. For example, to learn more about the `cp` command type:

man(1)

```
man cp
```

After a few seconds, an information display appears. For command syntax, refer to the “`SYNOPSIS`” section of the man page. Brackets, [ ], in a syntax statement indicate that the enclosed parameter is optional.

Printing man pages

To print a man page, type the following:

```
man command_name | col -b | lp
```

The `col -b` filters and formats the man page, and the `lp` command sends it to the default printer.

NOTE

What does that vertical-bar character “|” mean?

The “|” character represents a **pipe** command. A pipe is used to connect the output of one command as the input of another command.

You can even look at the `man` man-page to learn more about the `man` command itself:

man(1)

```
man man
```

```
 man(1)                                      man(1)

 NAME
   man - find manual information by keywords;
         print out a manual entry

 SYNOPSIS
   man -k keyword...
   man -f file...
   man [-] [section[subsection]] entry_name...

 DESCRIPTION
   man accesses information from the online
   version of the HP-UX Reference.  It can be
   used to:

- More -(11%)
```

more(1)

The message `- More -(11%)` means you have viewed 11% of the file, and 89% remains. (Some systems will just display `- More -`). At this point, you can do any of the following:

- Scroll through the file a page at a time by pressing the space bar.
- Scroll through the file a line at a time by pressing **Enter**.
- Quit viewing the man page by pressing **Q**.

2 Working with Files and Directories

Many tools

HP-UX provides numerous tools to work with files and directories. There are commands to create, remove, group, move, and maintain both files and directories.

A **file** is a named area on your system containing stored information.

A **directory** is a kind of file that can contain other files and directories.

To do this ...	See ...
Creating files	page 34
Listing files	page 35
Naming files	page 36
Viewing or printing files	page 38
Renaming, copying, removing files	page 41
Comparing, joining files	page 43
Understanding hierarchies	page 45
Understanding pathnames	page 50
Creating directories	page 54
Changing directories	page 56
Moving files between directories	page 58
Copying directories	page 60
Removing directories	page 61
Using filename wildcards	page 63
Searching inside of files	page 65
Finding files	page 67

Creating a File

You can use the `cat` command to create a file containing text. For example, to create the file “myfile”, use the `cat` command as follows:

cat(1)

```
$ cat > myfile
```

After you type this command, the cursor sits on the first line of the empty file. Type your text and press **Enter** at the end of each line. To exit the file, hold down **CTRL** and press **D**. The `cat` command returns you to the command line prompt.

You can use the `cat` command to create your own version of `myfile`. For example, you might create the file as follows:

```
$ cat > myfile
The text I am typing will be stored in "myfile". Enter
I press RETURN at the end of each line. Enter
When I'm finished, I hold down the CTRL key and press D. Enter
CTRL-D
```

NOTE

It’s easier to use an editing program.

You can also create and edit files using a text editor such as `vi`. To learn how to use this editor, see Chapter 4, “Using the vi Editor.”

NOTE

What Does That “(1)” Thing Mean?

As we mentioned in the last chapter, the “(1)” indicates the type of HP-UX command. If it is confusing, you may wish to read about manual pages at the end of the previous chapter.

Listing Files

To verify that `cat` created `myfile`, run the `ls` command, which lists the names of your files. Running the `ls` command with the file name will confirm that the file exists, but won't list other files.

ls(1)

```
$ ls myfile
myfile  The ls command lists myfile.
```

To see additional information about a file, use the `ll` (*l*ong *l*isting) command.

ll(1)

```
$ ll myfile
-rw-r--r--   1 myname   mygroup    146 Aug  4 14:13 myfile
```

The mysterious characters at the beginning of the line (-rw-r--r--) indicate the owner's, the group's and other people's permissions to read and/or write the file. The "1" indicates how many links (or names) are associated with this file, the "myname" is typically your login name, the "mygroup" is the name assigned by the administrator to the group of users who work with you, the "146" is the number of bytes (characters) in the file, followed by the date the file as last modified, followed by the file's name.

For more information about the `ll` command and access permissions, see Chapter 7, "Making Your System Secure."

Viewing files

Viewing the file's contents is discussed in "Viewing and Printing Files" in this chapter.

Naming Files

When you choose a file name, you need to follow certain rules regarding the length of the name and the types of characters you include.

Guidelines for File Names

When you choose a file name, remember these rules:

- Generally, file names can contain up to 256 characters (or bytes, in non-ASCII character sets). These characters can be any combination of the following:
  - Uppercase or lowercase letters (`A` through `Z`; `a` through `z`)
  - Digits (`0` through `9`)
  - Special characters, such as: `+`, `-`, `_`, `.`

  Based on these rules, the following are valid file names:

Valid names

```
money          Acct.01.87     CODE.c
lost+found     112.3-data     foo_bar
```

- HP-UX interprets uppercase and lowercase letters differently in file names. Thus, the following file names all are different:

```
money      Money      MoneY      MONEY
```

NOTE

Short File-Name Systems

On some computers, file names cannot be longer than 14 characters. If you are not sure if your computer can support longer file names, check with your system administrator. You could also create a file with a very long name and see if its name becomes truncated.

Invisible File Names

A file name in which the first character is a dot (.) is an **invisible file name**, since the `ls` command does *not* normally display it. Use invisible file names if you don't want or need certain files displayed when you run `ls`.

To illustrate, you have an invisible startup file that the system runs when you log in, a **login script**. It is used to customize your working environment. To learn more about login scripts, see “Using Login Scripts to Set the System Environment” in Chapter 3, “Using Your Shell.”

To cause `ls` to list invisible file names, including the name of your login script, run it with the `-a` option:

```
$ ls -a                  Use -a to see invisible file names.
.profile    myfile        This is the POSIX Shell, so .profile is shown.
```

Invisible files, or “dot files” are often used to store configuration information. Be careful not to remove these files without understanding what they doing for you.

Viewing and Printing Files

Using the `more` command, you can view a text file one screenful at a time. If your system is appropriately configured, you can print a text file using the `lp` command.

Viewing a File with more

The `more` command displays a text file's contents on the screen. For example, the following `more` command displays the contents of `myfile` (which you created in “Creating a File”):

more(1)

```
$ more myfile
The text I am typing will be stored in "myfile".
I press RETURN at the end of each line.
When I'm finished, I hold down the CTRL key and press D.
```

If the file contains more lines than are on your screen, `more` pauses when the screen is full. With a longer file, press **space** to continue looking at additional screens, and press **Q** when you are finished. Then `more` returns you to the system prompt.

Try running `more` on the system file `/etc/passwd`:

```
$ more /etc/passwd
root:XOSDMfBA.hqs6:0:3::/:/usr/bin/sh
daemon:*:1:5::/:/usr/bin/sh
bin:*:2:2::/bin:/usr/bin/sh
adm:*:4:4::/var/adm:/usr/bin/sh

More(4%)
```

The “`More(4%)`” message at the bottom of the screen means you have viewed 4% of the file thus far, and 96% of the file remains to be viewed. At this point, you can do any of the following:

- Scroll through the file a page at a time by pressing the space bar.
- Scroll backwards a page at a time by pressing **B**.
- Scroll through the file a line at a time by pressing **Enter**.
- Quit viewing the file and leave `more` by pressing **Q**.

Displaying the First and Last Lines of a File

We often want to see just the beginning (head) of a file or just the end (tail) of a file.

- To see the first line of a file without using a text editor, use the `head` command:

  head(1)

  ```
  $ head filename
  ```

  This will display, by default, the first ten lines of *filename*, including blanks. For example:

  ```
                CONFERENCE NOTES

  Attendees:

   Mary
   Sam
   Nina
   George
   Raphael
   Sergei
  ```

- To see the last ten lines (default value) of your file, use the `tail` command:

  tail(1)

  ```
  $ tail filename
  ```

  You will see the last ten lines (including blanks) of *filename*.

Both `head` and `tail` take numeric arguments. For example, use this command to display the first 25 lines of `file1`: `head -25 file1`

Printing a File with lp

You can print a text file using the `lp` (*l*ine *p*rinter) command. For example:

lp(1)

```
$ lp myfile
```

The `lp` command displays a message indicating that it sent your file to the printer. For example:

```
request id is lp-number (1 file)
```

The *number* is an ID number assigned to the print job by the `lp` command. If you don't see this message, or if you get an error message, consult your system administrator. You should get a printout with your username displayed on the first page. The time required for a printout depends on the number of tasks being run by the system and the speed of the printer itself.

To configure printers and set up the `lp` spooler, use the System Administration Manager (SAM). For command-line printer configuration, see the *Configuring HP-UX for Peripherals* manual. For command-line spooler configuration, see the *System Administration Tasks* manual.

Getting Printer Information with lpstat

To display a report on the printer status, including the order of your print job in the printer queue, type:

lpstat(1)

```
$ lpstat -t
```

Canceling a Print Request with cancel

To cancel a print request, enter the `cancel` command with the ID number of your request:

cancel(1)

```
$ cancel request_id
```

Renaming, Copying, and Removing Files

To change a file's name, use the mv ("*move*") command; to make a copy of a file, use the cp ("*copy*") command; to remove a file, use the rm ("*re*m*ove*") command.

Renaming Files with mv

Using the mv command, you can rename the file myfile to foofile as follows:

mv(1)

```
$ mv myfile foofile
```

To verify that mv renamed the file, use the ls command:

```
$ ls
foofile
```

To rename foofile back to myfile, type:

```
$ mv foofile myfile
$ ls        Using ls, verify that the action was successful.
myfile
```

CAUTION

If you move a file to an existing file, the existing file will be lost.

When renaming files, take care not to rename a file to the name of a file that already exists in that directory. If you do this, the file that already has the name will be lost. To ensure that you do not accidentally remove an existing file, use the -i option. For example:

```
$ mv -i myfile foofile
```

In this case, if foofile exists, the above command asks for confirmation before removing it.

The mv command can also be used to move files to different locations on the system. See "Moving and Copying Files between Directories".

Copying Files with cp

Copy a file when you want to make a new version of it while still keeping the old version around. For example, to make a new copy of `myfile` named `myfile2`, type:

cp(1)

```
$ cp myfile myfile2
```

Now when you use the `ls` command, you will see the following:

```
$ ls
myfile      myfile2
```

Use `more` to view `myfile2`. You will find that it is the same as `myfile`.

CAUTION

If you copy a file to an existing file, the existing file will be lost.

To ensure that you never accidentally overwrite an existing file, use the `-i` option. For example, if you attempt to copy `myfile` to `myfile2` in the current directory and `myfile2` already exists, `cp` asks for permission to overwrite `myfile2`:

```
$ cp -i myfile myfile2
overwrite myfile2? (y/n)
```

Removing Files with rm

If you have files that are no longer needed, you should remove (delete) them. Deleting unnecessary files leaves more room for other files on your system. For example, suppose you have finished using `myfile2`, and it is no longer needed. To remove `myfile2`, type:

rm(1)

```
$ rm myfile2
```

To see that `myfile2` was removed, use `ls`:

```
$ ls  The directory listing shows the remaining file.
myfile
```

NOTE

To cause the `rm` command to prompt you for permission before deleting any file, use the `-i` option:

```
$ rm -i myfile
myfile: ? (y/n)
```

See “Removing Directories” for information on how to remove directories and contents.

Comparing the Contents of Two Files

If two text files are known to be similar, and you want to determine what the differences are or which one has been changed:

1. First run `ll` and look at the date and time fields showing when each file was last saved. For example:

   ```
   -rw-r--r--   1 jim      users      1759  Mar 17 15:53 test1
   -rw-r--r--   1 jim      users      2130  Mar 17 15:47 test2
   ```

   `test1` was saved more recently than `test2`, because it has the more recent time (and its size was also changed).

2. You can determine the differences between `test1` and `test2` by running the `diff` command:

diff(1)

```
$ diff test1 test2
```

For example, if `test1` contains:

```
You are in a maze of
twisty little passages
which are all alike.
```

And `test2` contains:

```
You are in a maze of
twisty little passages
which are all different.
```

The command will indicate the differences it found, by line number, and point (with < and >) to which file the difference occurred in:

```
3c3  The relevant line numbers
< which are all alike.  The version in test1
---
> which are all different.  The version in test2
```

Note how the diff command is telling you that if you were to remove "<" the one line, and add ">" the other line, the two files would be the same.

Joining Two Files

To append to an existing file, you use the `cat` command with two greater-than signs (>>). The file name following the >> identifies the file to which the contents of the first file is appended. If that file exists, the new data is appended to the end of the file. If the file does not exist, it is created. The command format is:

cat(1)

```
$ cat filename2 >> filename1
```

where *filename2* is the file whose output is redirected, and *filename1* is the name of the file that is appended to.

This also works with the output of commands. The following example executes the `date` command with the output redirected to append to the `whoison` file:

```
$ date >> whoison        Append output to whoison.
$ more whoison           Display contents of whoison.
pat    console  Oct  4 08:50   Output from previous example.
terry  tty01    Oct  4 11:57
kim    tty02    Oct  4 08:13
Tue Oct  4 13:20:16 MDT 1994   Newly appended output from date.
```

Understanding a Directory Hierarchy

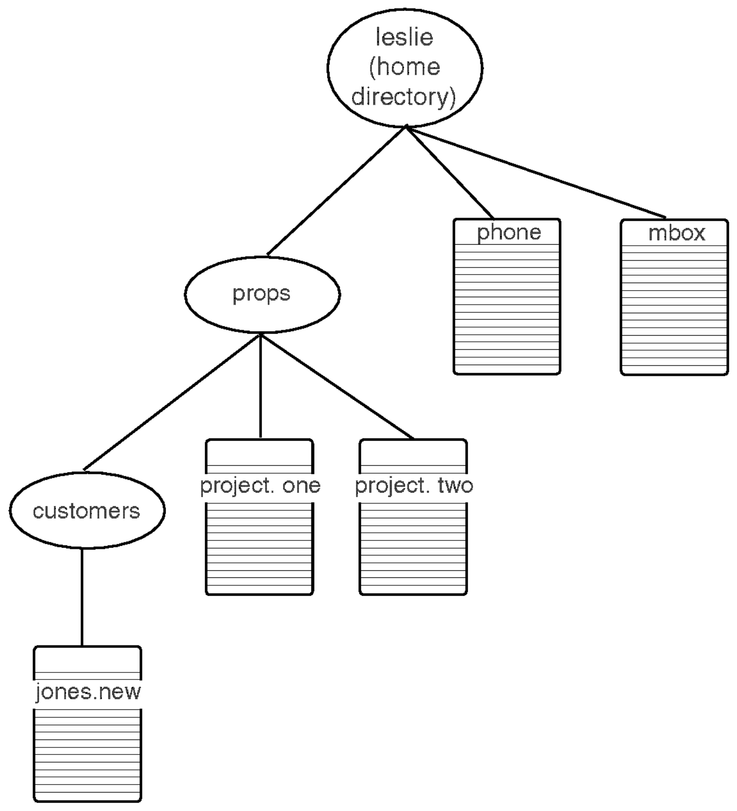
Directory tree

HP-UX directories can contain files and other directories. Therefore, a directory usually has a **parent directory** “above” and may also have **subdirectories**, or child directories “below” within it. Similarly, each subdirectory can contain other files and also can have more subdirectories. Because they are hierarchically organized, directories provide a logical way to organize files.

With the help of directories, you can organize your files into manageable, logically related groups. For example, if you have several files for each of several different projects, you can create a directory for each project and store all the files for each project in the appropriate directory.

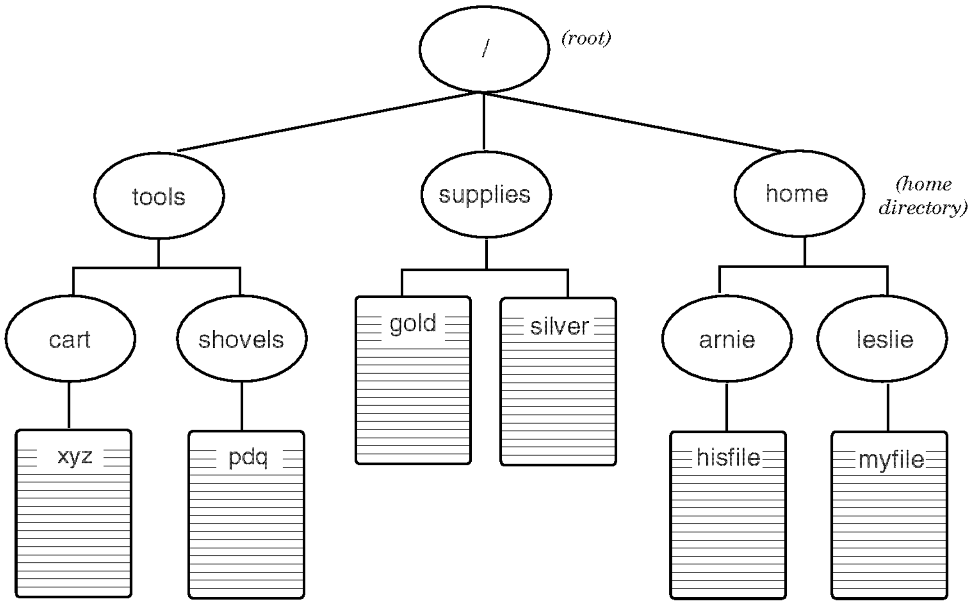
The structure of an HP-UX directory resembles an inverted tree. These directories (shown in the figure below as ovals) usually contain more directories; thus, the typical home directory develops a branching tree structure.

Figure 2-1 A Typical HP-UX Directory Structure



Each directory also contains files (represented below as boxes), which hold actual text, data, or code. At the top of the inverted tree structure is the **root directory**, represented in path names as `/`. This figure shows a broader part of a system's directory structure.

Figure 2-2 **A System Directory Structure**

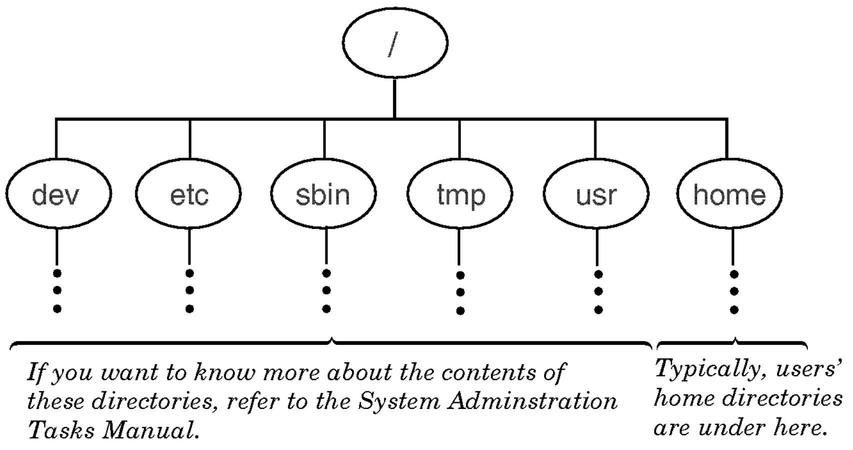


Determining Your Location in an HP-UX Directory Hierarchy

This section discusses the HP-UX directory structure and how you specify the location of a file in the structure. All directories fall under the topmost **root** directory, which is denoted by a slash (/). When you use HP-UX, you are working in a directory called the **current working directory**. And when you log in, HP-UX places you in your **home directory**.

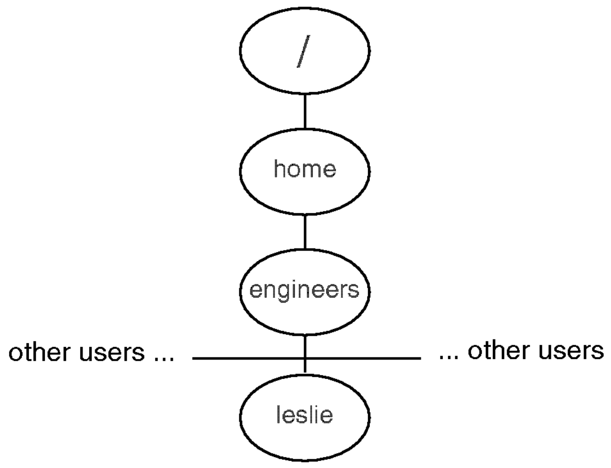
The figure below shows the two highest levels of a typical HP-UX directory structure. Each directory, including the root, may contain logically organized files, as well as more directories.

Figure 2-3 **The HP-UX Directory Structure**


/
dev
etc
sbin
tmp
usr
home
If you want to know more about the contents of these directories, refer to the System Adminstration Tasks Manual.
Typically, users' home directories are under here.

Here is a sample directory hierarchy for a user named Leslie. When Leslie logs in, she is in her home directory, `leslie`.

Figure 2-4 Leslie's Home Directory



pwd(1)

To determine your location in the directory hierarchy, use the pwd (*p*rint *w*orking *d*irectory) command. The pwd command displays the “path” from the root directory to your current working directory.

For example:

```
$ pwd
/home/engineers/leslie
```

Specifying Files and Directories

When specifying files in your current working directory, you can refer to them just by their file names. But when referring to directories and files outside your current working directory, you must use **path names**, which tell HP-UX how to get to the appropriate directory.

Absolute Path Names

Absolute path names specify the path to a directory or file, starting from the root directory at the top of the inverted tree structure. The root directory is represented by a slash (/). The path consists of a sequential list of directories, separated by slashes, leading to the directory or file you want to specify. The last name in the path is the directory or file you are pointing to.

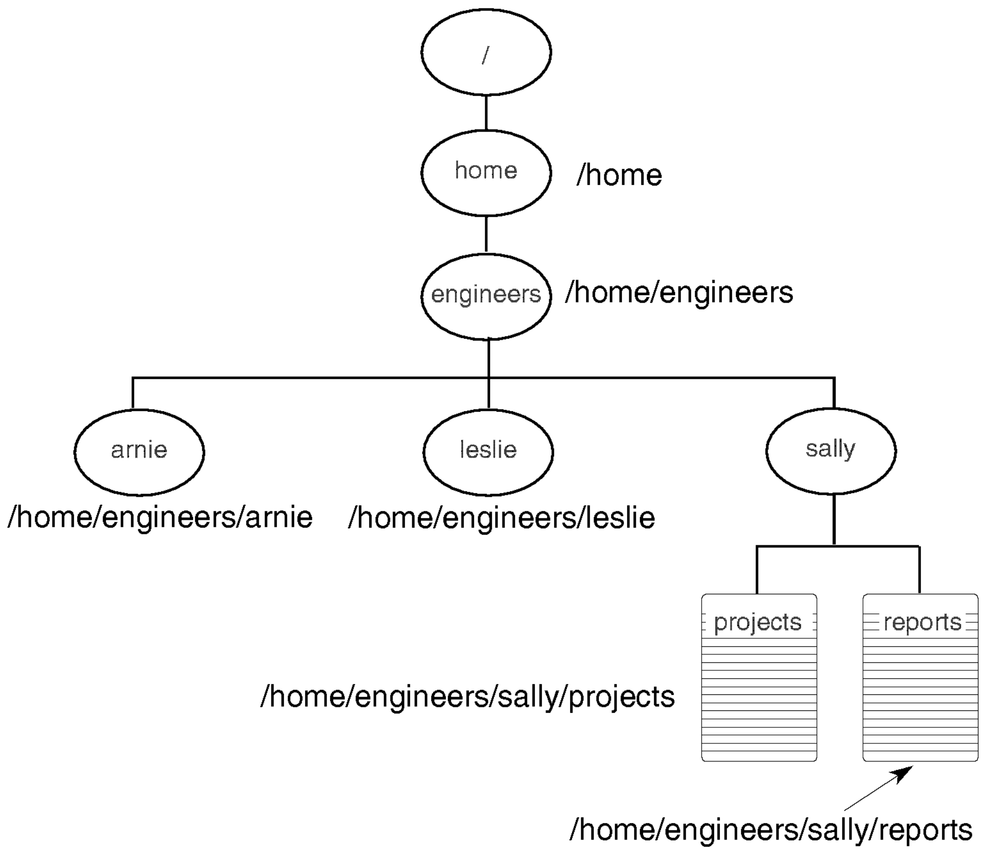
Here is an example of an absolute path, displayed with the `pwd` command:

```
$ pwd
/home/engineers/leslie
```

This specifies the location of the current directory, `leslie`, by starting from the root and working down.

The figure below shows the absolute path names for various directories and files in a typical directory structure:

Figure 2-5 Absolute Path Names



Relative Path Names

You can use a *relative* path name as a shortcut to the location of files and directories. Relative path names specify directories and files starting from your current working directory (instead of the root directory).

This relative path name...	Means...
`.`	The current directory.
`..`	The parent directory (the directory above the current directory).
`../..`	Two directories above the current directory.
directory_name	The directory below the current directory.

For example, suppose the current directory is `/home/engineers/leslie`. To list the files in the directory above (which is `/home/engineers`), enter:

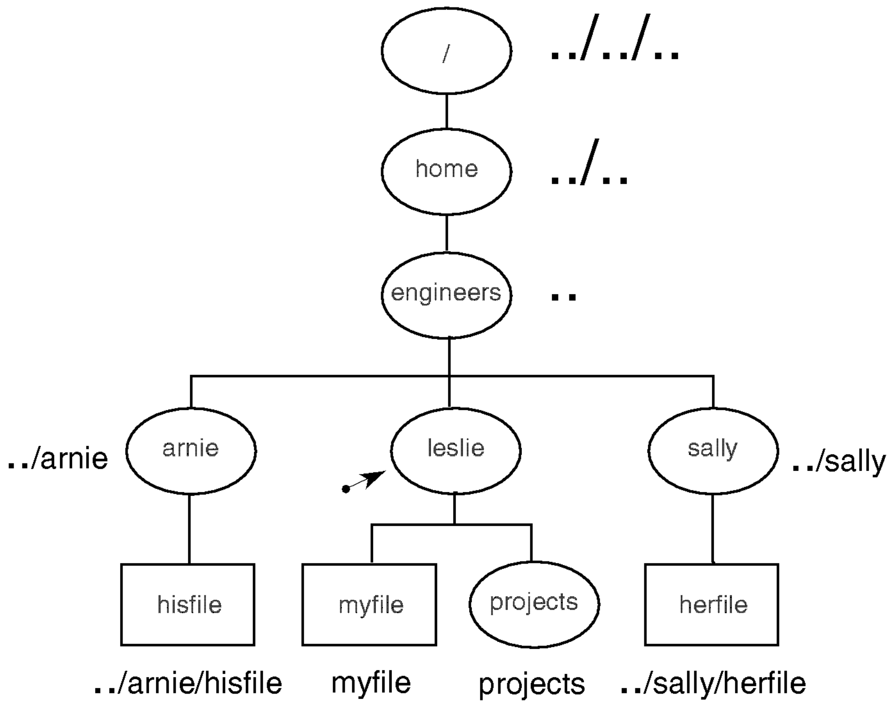
```
$ ls ..
arnie  leslie  sally
```

To list the files in a directory immediately below your current directory, simply enter the directory name. For example, to list the files in the `projects` directory, below the current directory `/home/engineers/leslie`, enter:

```
$ ls projects
$           The projects directory is empty.
```

The following figure shows relative path names for various directories and files starting from the current directory, `/home/engineers/leslie`.

Figure 2-6 Relative Path Names from /home/engineers/leslie



Creating Directories

To create a directory, use the `mkdir` (*m*ake *dir*ectory) command. After you create a directory, you can move files into it, and you can even create more directories underneath it. For example, to create a subdirectory in your current working directory named `projects`, type:

mkdir(1)

```
$ mkdir projects
```

To verify its creation, you can use either the `ls` or `lsf` command to list the contents of the directory. Both commands display the new directory, but `lsf` appends a slash (/) to the end of directory names to differentiate them from file names. For example:

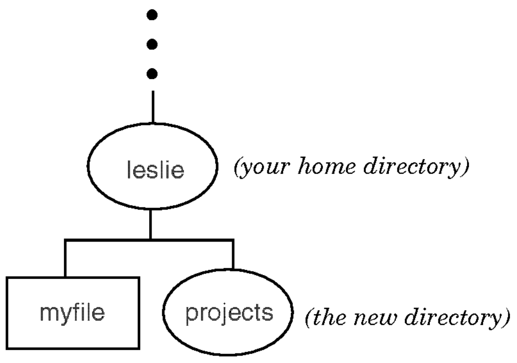
```
$ ls
myfile  projects
```
It did what you expected.

lsf(1)

```
$ lsf
myfile  projects/
```
The lsf command appends a slash to directory names.

This figure shows the resulting directory structure under `/home/engineers`:

Figure 2-7 Creating the “projects” Directory



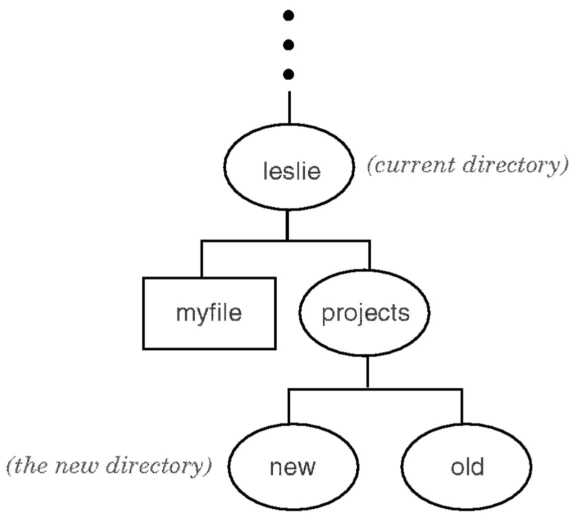
Use `mkdir` as follows:

```
$ mkdir new_dir_path
```

where *new_dir_path* is the path name of the directory you want to create. For example, to create two directories named `old` and `new` under the `projects` directory, type:

```
$ mkdir projects/old
$ mkdir projects/new
$ lsf projects
new/      old/
```

Figure 2-8 Structure after Creating New Directories



Changing Your Current Directory

To change your **current working directory**, use the `cd` command. For example, `cd projects` moves you into the directory `projects` (which you created in “Creating Directories”).

To verify your location, use the `pwd` command, which displays your current directory. For example, if your home directory was `/home/leslie` and you ran the “`cd projects`” command, `pwd` would display the following:

```
$ pwd
/home/leslie/projects
```

To move into the directory `new` under `projects`, type:

cd(1)

```
$ cd new
$ pwd               Verify where you are.
/home/leslie/projects/new
```

Remember that `..` is the relative path name for the parent directory of your current working directory. So to move up one level, back to `projects`, type:

```
$ cd ..
$ pwd          Show your current working directory.
/home/leslie/projects       It was successful.
```

NOTE

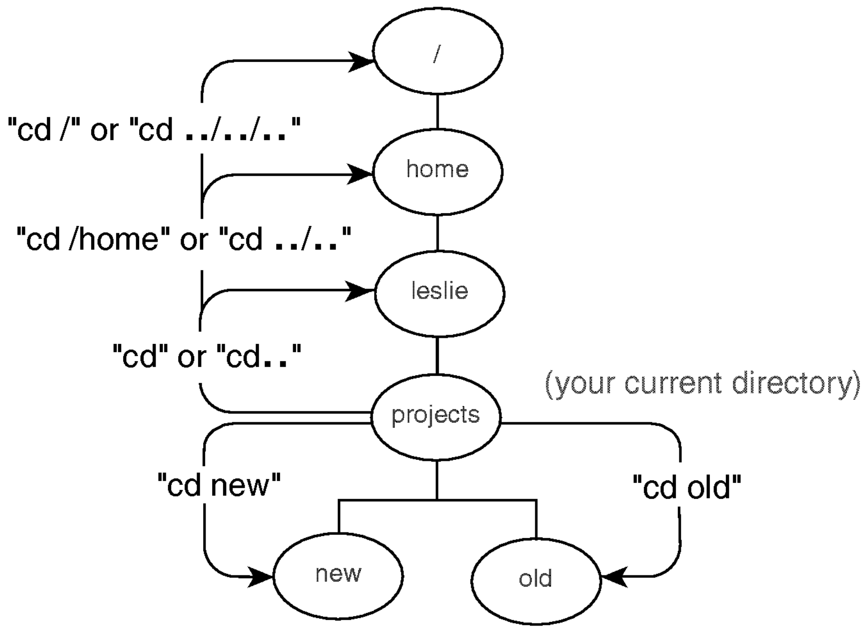
Finding your way home:

Experiment with the `cd` and `pwd` commands to move around your directory structure. If you become lost, don't panic; just remember that you can type `cd` to return to your home directory. For example:

```
$ cd
$ pwd     Are you back home?
/home/leslie            Yes!
```

The following figure illustrates how various `cd` commands change your current working directory. The example assumes you're starting at the directory `/home/leslie/projects`, and that your home directory is `/home/leslie`.

Figure 2-9 Effect of Various "cd" Commands



Absolute path

You can also get to any directory using its absolute path name. For example, to change to the `projects` directory in the example hierarchy, enter:

```
$ cd /home/leslie/projects
```

Moving and Copying Files between Directories

The `mv` command lets you move a file from one directory to another. With the `cp` command, you can copy a file into a different directory.

Moving Files

To move files from one directory to another, use the `mv` command.

```
$ mv  from_path to_path
```

where *from_path* is the file name or path name of the file you want to move, and *to_path* is the name of the path where you are moving the file. For example, to move `myfile` into the `projects` directory, type:

```
$ cd                          Move to home directory first.
$ mv myfile projects
```

A single dot (`.`) for a path name represents your current working directory. So, to move `myfile` from the `projects` directory back to your current working directory, type:

```
$ mv projects/myfile .      Don't forget the dot.
```

CAUTION

If you rename a file to an existing file, the existing file will be lost.

When renaming files, take care not to rename a file to the name of a file that already exists in that directory. If you do this, the file that already has the name will be lost. To ensure that you do not accidentally remove an existing file, use the `-i` option. For example:

```
$ mv -i myfile /home/leslie/foofile
```

If `/home/leslie/foofile` exists, the above command asks for confirmation before removing it.

Copying Files

To copy a file into a different directory, use the `cp` command.

```
$ cp from_path to_path
```

where *from_path* is the file name or path name of the file you want to copy, and *to_path* is the path name of the directory or file to which you are copying.

For example, to make a copy of `myfile` named `myfile2` in the `projects` directory, type:

```
$ cp myfile projects/myfile2
$ lsf
myfile   projects/  The file myfile still exists.

$ lsf projects
myfile2  new/  old/  The copy (myfile2) is in projects.
```

To make a new version of `myfile2` named `myfile3` in your current directory, type:

```
$ cp projects/myfile2 myfile3
$ lsf
myfile    myfile3    projects/
```

CAUTION If you copy a file to an existing file, the existing file will be lost.

To ensure that you never accidentally overwrite an existing file, use the `-i` option. For example, if you attempt to copy `/home/leslie/myfile` to `myfile2` in the current directory and `myfile2` already exists, `cp` asks for permission to overwrite `myfile2`:

```
$ cp -i /home/leslie/myfile myfile2
overwrite myfile2? (y/n)
```

Copying Directories

To copy entire directories, use the `-r` option of the `cp` command.

For example, if you have a directory called `mydir` that contains `myfile` and `newfile`, you can copy the directory to a new directory called `mydir2`. `mydir2` will also contain a copy of `myfile` and `newfile`. Use the following command:

```
$ cp -r mydir mydir2
```

The `-r` option copies any files and subdirectories below the specified directory.

NOTE

Where did the directory go?

If the destination directory already exists, the directory you are copying will become a **subdirectory** under the existing destination directory. If the destination directory does not exist, it will be created.

Removing Directories

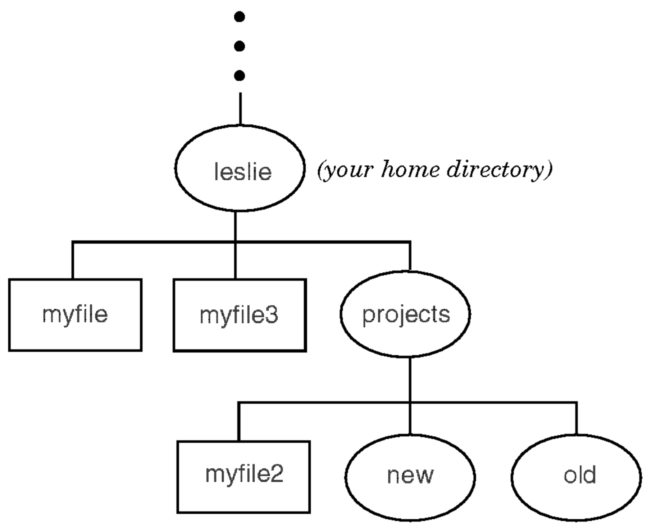
You can remove an empty directory with the `rmdir` command. To remove a directory and all of its contents in one step, use the `rm` command with the `-rf` option.

After you have removed a directory, you can no longer use it, and it will no longer appear in an `ll` or other listing of the directory above it.

Removing a Directory with rmdir

Before removing a directory with `rmdir`, you must remove any visible or invisible files and any directories under it. For example, suppose you want to remove the `projects` directory and its files:

Figure 2-10 The "projects" Directory Structure



To remove this structure, run the following sequence of commands:

```
$ cd              Move back to your home directory
$ lsf             List the files and directories.
myfile myfile3 projects/
$ rmdir projects      Try to remove projects.
rmdir: projects not empty   It won't let you.
$ cd projects     Change directory to projects.
$ lsf             List its contents.
myfile2  new/ old/
$ rm myfile2      Remove file myfile2.
$ lsf             Check if it is gone.
new/   old/
$ rmdir new       Remove directory new. If it's empty, rmdir removes it.
$ lsf             Check if it is gone.
old/
$ rmdir old       Now remove the old directory. If empty, rmdir removes it.
$ lsf             There is no message; the action was successful.
$ cd              Now move back to your home directory.
$ rmdir projects
$ lsf              Verify that it worked.
myfile    myfile3
$
```

Removing Everything with rm -rf

To avoid having to empty a directory before removing it, you can remove a directory *and all its files and directories* in one action by typing the following:

```
$ rm -rf dirname
```

CAUTION Use `rm -rf` with great caution, since it does remove a directory and all its contents, irretrievably, in one action.

File Name Shorthand: Wildcard Characters

Wildcard characters provide a convenient shorthand for specifying multiple file or directory names with one name. Two of the most useful wildcard characters are * and ?. The * matches any sequence (string) of characters (including no characters), and the ? matches any one character.

The * Wildcard

The * wildcard means "any characters, including no characters." Suppose you have created the following files in your current working directory:

```
$ lsf
myfile    myfile2    myfile3    xenic    yourfile
```

To list only the file names beginning with "`myfile`", type:

```
$ lsf myfile*
myfile    myfile2    myfile3
```

To list file names containing "`file`", type:

```
$ lsf *file*
myfile    myfile2    myfile3    yourfile
```

The ? Wildcard

The ? wildcard means "any single character." Although you probably won't use the ? wildcard as much as *, it is still useful. For instance, if you want to list only the files that start with `myfile` and end with a single additional character, type:

```
$ lsf myfile?
myfile2   myfile3
```

The ? wildcard character matches *exactly one character*. Thus, `myfile` didn't show up in this listing because it didn't have another character at the end.

Using the * Wildcard Character with mv, cp, and rm

Wildcard characters are often useful when you want to move or copy multiple files from one directory to another. For example, suppose you have two directories immediately below your current directory, named `new` and `old`, and these directories contain the following files:

```
$ lsf new
myfile    myfile2
lsf old
myfile3   myfile4
```

To move all the files from the directory `new` into the directory `old`, type:

```
$ mv new/* old
$ lsf new              The files are no longer in new.
lsf old
myfile myfile2 myfile3 myfile4   They are in the directory old.
```

You can do a similar operation with the `cp` command. For example, to copy all the files from `old` into `new`, type:

```
$ cp old/* new
```

Similarly, you can use wildcard characters with the `rm` command. For example, to remove all the files in the directory `new`, type:

```
$ rm new/*
```

CAUTION

It's easy to remove too much.

When using wildcards, be careful not to accidentally remove files you need. Sometimes it's helpful to substitute the `ls` command for the `rm` command if you want to see which filenames match a pattern.

For More Information...

See the *regexp*(5) man page for general features of `*` and `?`. For additional features relating to individual shells: if you use the POSIX Shell, see *sh-posix*(1) man page; if you use the C shell, see the *csh*(1) man page.

Searching for Text Patterns using grep

You can use the `grep` (“`g`lobal `r`egular `e`xpression `p`rint”) command to search for a text pattern within a file or to display the names of files that contain a specified text pattern. This command is useful when you want to search for information in files or directories.

The `grep` command looks at each line of one or more files for a text string that matches a specified pattern. When it finds a matching text string, it displays the line in which the matching string is found.

Searching a File for a Text String

Suppose you have a mailing list called `mailist` with the contents shown below:

```
Smith, Joe     2345 Pine St.     Santa Clara, CA
Walsen, Stacey  493 Winkle Ave.  San Jose, CA
Diaz, Robert   6789 Pine St.     Santa Clara, CA
Wang, Michael  1832 Jackson St.  Santa Clara, CA
```

If you want to extract the addresses of all the people on Pine Street, enter:

grep(1)

```
$ grep Pine mailist
```

The `grep` command lists all lines in `mailist` that contain the string `Pine`. The output is:

```
Smith, Joe     2345 Pine St.     Santa Clara, CA
Diaz, Robert   6789 Pine St.     Santa Clara, CA
```

To make the search case-insensitive, use the `-i` option. For example:

```
$ grep -i pine mailist
```

Searching Multiple Files

The `grep` command can be useful in other ways. Sometimes, you want to find information, but are not sure in which file it is located.

Suppose you have three mailing lists, and cannot remember which contains Stacey Walsen's address. Enter:

```
$ grep 'Walsen, Stacey' mailist mailist2 mailist3
mailist: Walsen, Stacey  493 Winkle Ave. San Jose, CA
```

The `grep` command displays the line containing Stacey's address and the file in which it was found. Note that because it contains a space, the string must be surrounded by single quotes (`'Walsen, Stacey'`).

To search the entire current directory for this information, enter:

```
$ grep 'Walsen, Stacey' *
```

See the *grep*(1) man page in the *HP-UX Reference* for more information on using the `grep` command.

Searching for Files using find

You can use the `find` command to search through a directory and its subdirectories for files meeting certain criteria. You can then execute a command on the files you've found.

Finding Files that Match a Pattern

Although the syntax of `find` can be complex, it may help you use HP-UX more productively. It is a powerful and flexible command. However, it may run slowly, especially if it is searching many directories.

Suppose you want to display all files in the current directory and its subdirectories that begin with d. Enter:

find(1)

```
$ find . -name 'd*'
```

The dot (.) causes `find` to search the current directory and its subdirectories. The `-name` option followed by a file name or a file name pattern (in this case `d*`) tells `find` to search for all file names that match that pattern. In this example, `find` will look for all file names beginning with `d`.

Note that `d*` is enclosed by single quotes '`d*`'. If you use a file name pattern in the `find` command, you must quote it so that the shell will interpret it correctly.

Finding Files that are Newer than a Certain File

Suppose you want to display all files modified after a certain file. To display all files newer than `myfile` in the `/home/leslie` directory and its subdirectories, enter:

```
$ find /home/leslie -newer myfile
```

This example can be read as follows: in directory `/home/leslie` and its subdirectories, find all files modified after `myfile`. (To determine when a file was last modified, use the `ll` command.)

Running Commands on Files

You can execute commands on files located with the `find` command. Let's say you want to remove all files with a `.tmp` extension in the current directory and its subdirectories. Enter:

```
$ find . -name '*.tmp' -exec rm {} \;
```

This example finds and displays on the screen all files in the current directory and its subdirectories that end in `.tmp`, then deletes these files. The `-exec` option causes the following command (`rm`) to be executed. The brackets { } represent the files found with the `find` command. The semi-colon that ends the `exec` string is escaped with a backslash (\;).

Using Logical Operators

The syntax of `find` includes the logical Boolean operators NOT, AND, and OR.

To find files that do not match a specific pattern, use the logical NOT operator, the exclamation mark (!). After using this operator, you must use options to define file attributes such as file name. Then, files are found that do *not* have the attributes you specify.

For example, to find all files in `/tmp` that are *not* owned by `leslie`, use this command:

```
$ find /tmp \( ! -user leslie \)
```

The \ escapes the parentheses so that they are not interpreted as special characters by the shell.

To find files that have two distinct attributes, use the logical AND operator, *expression* `-a` *expression*. For example, to find all directories in `/` that are owned by `leslie`, use this command:

```
$ find / \( -type d -a -user leslie \)
```

To find files that have either or both of two attributes, use the logical OR operator, *expression* `-o` *expression*. For example, to remove all files ending with `.o` or named `a.out` that have not been accessed for a week, use this command:

```
$find / \( -name a.out -o -name '*.o' \) -atime +7 -exec rm {} \;
```

For More Information...

See the *find*(1) man page for more information on using the `find` command.

Chapter Command Summary

To Do This...	Type This...
Create a file	`cat > filename`
Terminate keyboard input for `cat`	**CTRL-D**
List visible files in current directory	`ls`
List all files in current directory	`ls -a`
List files; show directories with "/"	`lsf`
View a file	`more` *filename*
Print a file	`lp` *myfile*
Get information on a print job	`lpstat`
Cancel a print job *lpno*	`cancel` *lpno*
Rename (move) a file	`mv` *fromfile tofile*
Copy a file	`cp` *fromfile tofile*
Delete (remove) a file	`rm` *filename*
Change directory	`cd` *directory_path*
Change to home directory	`cd`
Display working directory	`pwd`
Remove an (empty) directory	`rmdir` *directory_name*
Remove a directory and contents	`rm -rf` *directory_name*
Search a file for a text pattern	`grep '`*text*`'` *filename*
Search for a file and display output on screen	`find` *dir_path* `-name` *filename*

3 Using Your Shell

Command Interpreter

HP-UX provides several command interpretation programs called shells. A shell is the interface between HP-UX and you. The shell interprets the text you type and the keys you press, then directs the HP-UX operating system to take an appropriate action.

To do this …	See …
Understand command syntax	page 72
Understand processes	page 74
Understand input, output, and error	page 76
Understand pipes	page 83
Changing your shell	page 86
Editing in a shell	page 90
Recalling previous commands	page 92
Setting the login environment	page 94
Using login scripts	page 97
Setting variables	page 99
Setting the search path	page 101
Setting terminal characteristics	page 104

Understanding Command Syntax

HP-UX has many useful commands that will help you handle data and text, do system administration tasks, and find information. Most of these commands are easy to enter, that is, they are either a command without any arguments (`whoami`), or a command whose only argument is a file name (`mkdir projects`). HP-UX commands can also be more complex, having additional options, arguments, or both.

Options change a command's behavior. For example, in Chapter 2, “Working with Files and Directories,”, you used the `-a` option to change the behavior of the `ls` command so you could list invisible file names. In general, command options are preceded by a dash (`-`). **Arguments** provide additional information needed by the command, such as the name of the file on which to run the command.

Examples Using Options

When used without options, the `rm` command removes a file without verifying whether you really want to remove it. Suppose, for example, your current working directory contains these files: `myfile`, `myfile1`, `myfile2`, `myfile3`, and `myfile4`. You could remove all these files by typing this command:

```
$ rm my*
$        All the files are removed, no questions asked.
```

To cause `rm` to prompt you for verification before removing each file, use the `-i` (interactive) option:

```
$ rm -i my*
myfile1: ? (y/n) y  Type y to remove the file.
myfile2: ? (y/n) y
myfile3: ? (y/n) y
myfile4: ? (y/n) n     Or, type n to leave it alone.
$ ls
myfile4  myfile4 was not removed.
```

If you are using `rm` non-interactively and the file does not have write permission (for example, with `-r--r--r--` permission, as displayed in a long listing), then a message something like this will be displayed:

```
filename:  444 mode ?  (yes/no)
```

Respond with `y` if you want to remove the file.

Examples Using Arguments

The `cal` command displays an English calendar for the current month. With multiple command arguments, you can specify which calendar month and year to display. For example, to display a calendar for February 1998, type the `cal` command as follows:

cal(1)

```
$ cal 2 1998
   February 1998
 S   M Tu  W Th  F  S
 1   2  3  4  5  6  7
 8   9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
```

Be sure to include the century, `19`. If you use `98` as the argument, you will get a calendar for the year 98 A.D.

Enclosing Arguments in Quotes

When a *single* command argument contains embedded blanks, you must enclose it between quotes ('word1 word2'). For example, the following `grep` command displays all lines in `myfile` containing “`I am`”:

grep(1)

```
$ grep 'I am' myfile
The text I am typing will be stored in "myfile".
```

Running Multiple Commands on the Same Command Line

Occasionally, you may find it useful to run two or more commands on the same command line. To do so, separate the commands with a semicolon, as illustrated below:

Using ;

```
$ whoami ; date
leslie     Output from whoami
Tue Sep 16 12:01:55 MDT 1997      Output from date
```

You can also connect commands, using the output of one as input to another. See “Piping Command Output and Input”.

Understanding Processes

The shell interprets your keyboard commands for the HP-UX operating system to act on. When you log in, you are said to be “in” a **shell**. After the shell interprets a command line, HP-UX loads into memory the corresponding program. When a program is running, it is called a **process**. HP-UX assigns every process a unique number, known as a **process identifier** (**PID**).

How Processes are Created

ps(1)

When you log in, HP-UX starts your shell. During login, HP-UX copies the shell program from system disk into memory. When it is in memory, the shell begins executing, and it becomes a process that lasts until you log out. **Process**, then, refers to the copied program that is actively executing in memory, while **program** is the file stored on the disk.

Similarly, the commands you type create processes. After you type a command line, the following events take place:

1. The shell interprets the command line and searches the disk until it finds the requested program.
2. The shell asks HP-UX to run the program; then control transfers from the shell to HP-UX.
3. HP-UX copies the specified program from a disk file into memory. When the program resides in memory, it begins executing — and a process is created.
4. Each process is assigned a **Process Identifier** or **PID**. You can find out what processes are currently running on your system by typing `ps -ef`.
5. When a program finishes executing, control transfers back to the shell, and the process disappears.

Stopping a Process with kill

Normally, processes can be terminated by entering the following, where *PID* is the identification number for the process you want to get rid of.

kill(1)

```
$ kill PID
```

The PID for the process is determined by running `ps -ef` and noting the name and process ID.

NOTE

Stopping a process with extreme prejudice.

In some cases, the process may choose to ignore the polite `kill` signal, and you will find it still running after you have issued the `kill` command correctly. If this happens, you can send a more forceful signal. Enter the following:

```
$ kill -9 PID
```

Run `ps -ef` to confirm that the process has been deleted.

There's a very small chance that if a process is not even listening for a signal, it will not terminate even when sent a "`kill -9`" signal. In this case, the process cannot be killed, and it is usually referred to as a "zombie" process.   The only way to make it go away is to reboot.

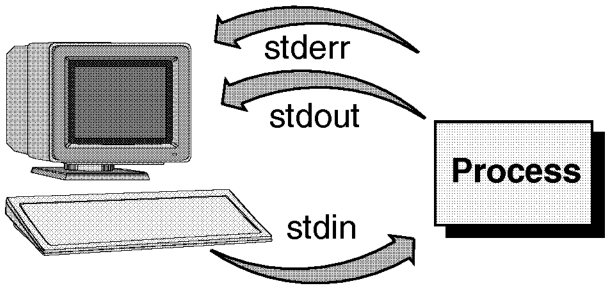
Understanding Standard Input, Standard Output, and Standard Error

Each process opens three standard “files”: standard input (`stdin`), standard output (`stdout`), and standard error (`stderr`). Programs use these as follows:

- **Standard input** is the place from which the program expects to read its input. By default, processes read `stdin` from the keyboard.
- **Standard output** is the place the program writes its output. By default, processes write `stdout` to the terminal screen.
- **Standard error** is the place the program writes its error messages. By default, processes write `stderr` to the terminal screen.

The figure below illustrates the relationship of these files to the process.

Figure 3-1 Standard Input, Standard Output, and Standard Error



Writing Standard Output to a File

The shell lets you redirect the standard output of a process from the screen (the default) to a file. Redirecting output lets you store the text generated by a command into a file; it's also a convenient way to select which files or devices (such as printers) a program uses.

In its simplest form, the command syntax is as follows:

stdout

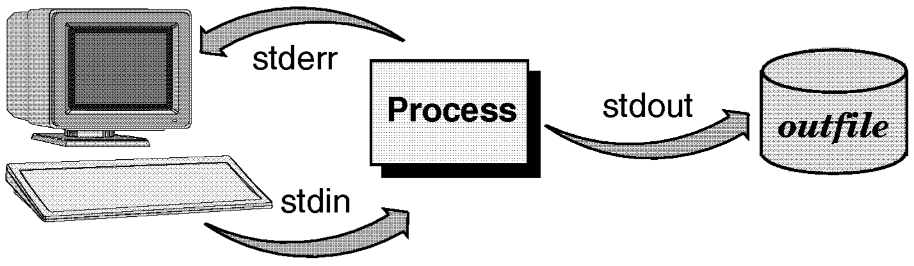
command **>** *outfile*

where *command* is the command whose output is redirected, and *outfile* is the name of the file to which the process writes its standard output. If the output file exists, its previous contents are lost. If the file does not exist, it is created.

To append the output to an existing file, use two greater-than signs (>>) pointing to the file to be appended on.

This figure illustrates where `stdin`, `stdout`, and `stderr` go when output is redirected to a file.

Figure 3-2 Standard Input, Output, and Error When Output Is Redirected



The example below shows output redirection using the `who` command, which displays a list of users currently logged in to the system. Instead of displaying the users on the terminal screen, the output is redirected to the file `whoison`.

who(1)

```
$ who > whoison     Redirect output to whoison.
$ more whoison      Display contents of whoison.
pat    console  Oct 9 08:50
terry  tty01    Oct 9 11:57
kim    tty02    Oct 9 08:13
```

Using Files for Standard Input

The shell lets you redirect the standard input of a process so that input is read from a file instead of from the keyboard. To redirect the input of a process, separate the command and the input file name with a less-than sign (<) directed at the command name. You can use input redirection with any command that accepts input from `stdin` (your keyboard).

In its simplest form, the command syntax is as follows:

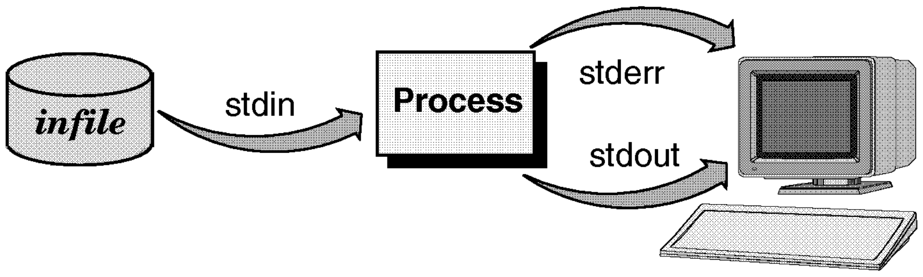
stdin

command **<** *infile*

where *command* is the command whose input is redirected, and *infile* is the name of the file from which the process reads standard input. The file must exist for the redirection to succeed.

The figure below illustrates where `stdin`, `stdout`, and `stderr` go when input is redirected from a file.

Figure 3-3 **Standard Input, Output, and Error When Input Is Redirected**



In the following example, standard output from the `who` command is redirected to a file named `savewho`. Then, the `more` command displays the contents of `savewho`. Finally, standard input for the `wc` (*w*ord *c*ount) command is redirected to come from the `savewho` file:

Using ">"

```
$ who > savewho  Redirect output to savewho
more savewho  Display contents of savewho
pat   console Oct 9 08:50
terry tty01   Oct 9 11:57
kim   tty02   Oct 9 08:13
```

Using "<"

```
$ wc -l < savewho  Redirect input from savewho
4  The answer
```

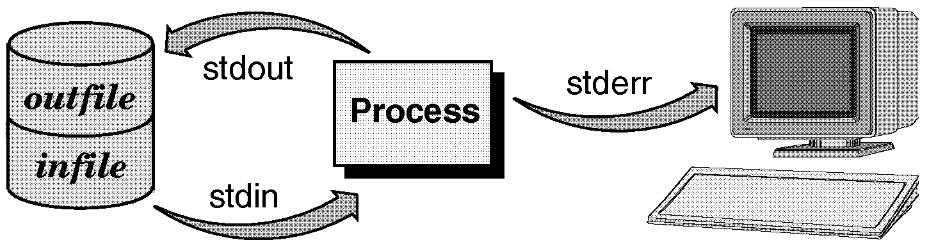
In the preceding example, the `wc` command with the `-l` option counts the number of lines in the input file. Because input is redirected from `savewho`, this number equals the number of users logged in to the system when the `who` command was executed.

Redirecting Both Standard Input and Standard Output

You can redirect both the standard input and the standard output of a single command. However, do *not* use the same file name for standard input and standard output, since the original contents of the input file are lost.

The following figure illustrates where `stdin`, `stdout`, and `stderr` are directed when both output and input are redirected from and to files.

Figure 3-4 Redirecting Both Input and Output



Using the Default Standard Input and Standard Output

The following example uses the `sort` command to sort text typed at the keyboard. Typing **CTRL-D** ends standard input. The standard output displays on the terminal screen as follows:

sort(1)

```
$ sort
muffy
happy
bumpy
CTRL-D                End of standard input.
bumpy
happy
muffy                 End of standard output.
```

Redirecting Standard Input

In the following example, input is redirected:

```
$ more socks         Display contents of socks.
polka dot
argyle
plaid

$ sort < socks       Redirect input from socks and sort the contents.
argyle
plaid
polka dot
```

In the preceding example, the `sort` command uses a file named `socks` as input. The standard output displays on the terminal screen.

Using Both Standard Input and Standard Output Redirection

The next example combines both input and output redirection:

```
$ sort < socks > sortsocks  Use both input and output redirection.
$ more sortsocks            Display contents of sortsocks.
argyle
plaid
polka dot
```

In this example, the `sort` command reads input from the `socks` file and writes output to the `sortsocks` file; thus, standard output (unlike the first two examples) does not display on your screen.

Piping Command Output and Input

The shell lets you connect two or more processes so the standard output of one process is used as the standard input to another process. The connection that joins the processes is a **pipe**. To pipe the output of one process into another, you separate the commands with a vertical bar (|). The general syntax for a pipe is as follows:

command1 **|** *command2*

where *command1* is the command whose standard output is redirected or piped to another command, and *command2* is the command whose standard input reads the previous command's output. You can combine two or more commands into a single pipeline. Each successive command has its output piped as input into the next command on the command line:

command1 **|** *command2* **|** . . . **|** *commandN*

In the following example, output from the `who` command is again stored in the file `savewho`. Then, the `savewho` file is used as input to the `wc` command:

wc(1)

```
$ who > savewho        Redirect output of who to file savewho.
$ wc -l < savewho      File savewho is input to wc command.
4                    Sample result.
```

With a pipeline, these two commands become one:

```
$ who | wc -l
4
```

As this example illustrates, using pipes eliminates the need for temporary intermediate files. Instead, the standard output from the first command is sent directly to the second command as its standard input.

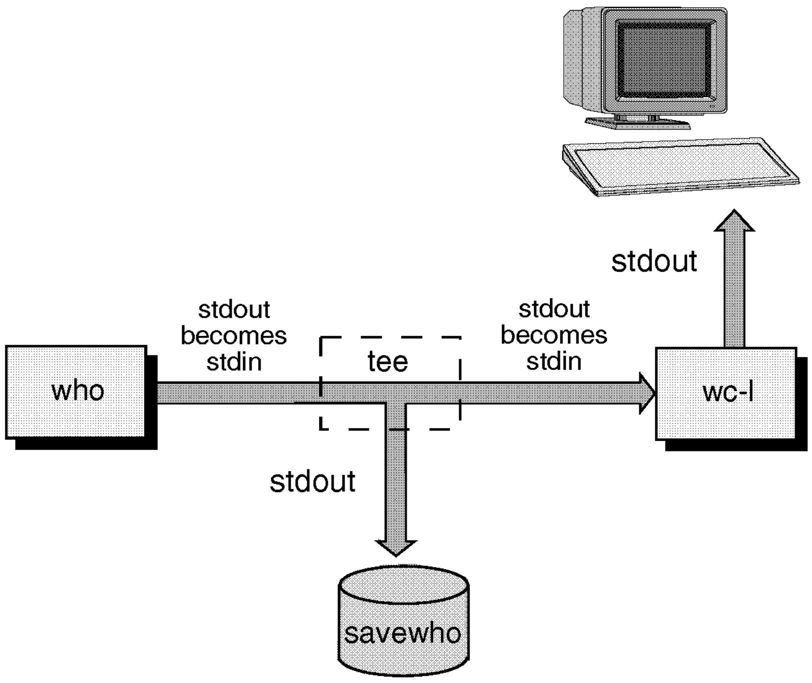
Using the tee Command with Pipes

The tee command lets you divert a copy of the data passing between commands to a file without changing how the pipeline functions. The example below uses the who command to determine who is on the system. In the example, which is further illustrated in the figure below, the output from who is piped into the tee command, which saves a copy of the output in the file savewho, and passes the unchanged output to the wc command:

tee(1)

```
$ who | tee savewho | wc -l
4
$ more savewho
pat         console      Oct  9 08:50
terry       tty01        Oct  9 11:57
kim         tty02        Oct  9 08:13
```

Figure 3-5 Standard Input and Output with Pipes and tee Command



For More Information...

HP-UX provides filter programs that are useful in pipelines. These programs accept text as input, transform the text in some way, and produce text as output. Filter commands include `adjust`, `awk`, `more`, `cut`, `grep`, `head`, `more`, `pr`, `rev`, `sed`, `sort`, `spell`, and `tail`. For information on these commands, see their man pages.

Shell Features: Determining and Changing Your Shell

HP-UX gives you your choice of several different shells. This section discusses the POSIX and Bourne shells. Details on the C shell can be found in the *Shells: User's Guide*.

Each of these shells has different characteristics, and you can increase the speed and efficiency with which you interact with HP-UX if you learn to use some of the built-in features of the shell of your choice.

With the POSIX shell, you can edit your command line and recall previous commands. Your shell environment can be “customized” using **shell variables** and **login scripts**.

Using simple commands, you can determine which shell you are running or change your shell temporarily or permanently. See “Determining Your Login Shell” for a listing of both the file name for each shell and the default system prompt.

NOTE

Choosing your default shell.

As of the HP-UX 10.0 release, the OSF POSIX Shell replaces the Korn Shell and Bourne Shell. Thus, `/usr/bin/sh` will be the POSIX Shell, and `/usr/bin/ksh` will be linked to `/usr/bin/sh`. However, `/usr/old/bin/sh` will contain the Bourne Shell for those users who still need it.

The following table lists features that may help you decide which shell to use:

Table 3-1 Comparison of Shell Features

Features	Description	POSIX Key	Bourne	C
Command history	Allows commands to be stored in a buffer, then modified and reused.	Yes	No	Yes
Line editing	Ability to modify the current or previous command lines with a text editor.	Yes	No	No
File name completion	Ability to automatically finish typing file names in command lines.	Yes	No	Yes
`alias` command	Lets you rename commands, automatically include command options, or abbreviate long command lines.	Yes	No	Yes
Restricted shells	A security feature providing a controlled environment with limited capabilities.	Yes	Yes	No
Job control	Tools for tracking and accessing processes that run in the background.	Yes	No	Yes

Determining Your Login Shell

The command `echo $SHELL` displays the file name of the shell you entered when you logged in.

```
$ echo $SHELL
/usr/bin/sh
```

The `echo` command displays the contents or value of a variable named `SHELL`. The `SHELL` variable contains the name of the file that contains the shell program that you are running. In this example, it is `/usr/bin/sh`, the file that contains the code for the POSIX Shell.

The following table lists both the file name of each shell and the default system prompt. (The superuser prompt for each is #.)

Table 3-2 Shell File Names and Default Prompts

Shell	File Name	Prompt
POSIX	`/usr/bin/sh`	`$`
C	`/usr/bin/csh`	`%`
Bourne (obsolete)	`/usr/old/bin/sh`	`$`
Korn (replaced by POSIX shell)	`/usr/bin/ksh` (linked to `/usr/bin/sh`)	`$`

Temporarily Changing Your Shell

Unless you are in a restricted shell, you can temporarily change your shell by using this command:

shell_name

where *shell_name* is the name of the shell (for example, `sh`, or `csh`). Temporarily changing your shell lets you experiment in other shells. By typing the name of the shell you want to run, you *invoke* (enter) that shell, and the correct prompt is displayed. To return to your original shell, type either `exit` or **CTRL-D**.

The following example begins in the POSIX Shell, enters the C Shell, and returns to the POSIX Shell:

csh(1)

```
$ csh               Enter C Shell.
% ps                Execute the ps command.
   PID TTY          TIME COMMAND
  6009 tty01        0:00 csh    Notice that both the C and
  5784 tty01        0:00 sh     POSIX Shel l processes are running
  6010 tty01        0:00 ps
% exit               Exit C Shell.
$                    POSIX Shell returns.
```

Permanently Changing Your Shell

To permanently change your *login shell* (the default shell you get when you log in), use the `chsh` (`ch`*ange* `sh`*ell*) command:

chsh(1)

```
$ chsh  username  full_shell_name
```

where *username* is your user name and *shell_path_name* is the full path name (for example, `/usr/bin/sh`) of the shell you want as your default. “Determining Your Login Shell” contains the full path names for each of the shells. After you use the `chsh` command, you must log out and log in again for the change to take effect. For example, if `terry` changes the default login shell to the C Shell, the command reads:

```
$ chsh terry /usr/bin/csh
%
```

Editing the Command Line

In the POSIX shell, you can correct errors in a command line before you enter it, using line editing commands or edit keys. You can also recall a previous command and edit it. See “Recalling Previous Commands” later in this chapter.

Using vi Line Editing Commands

Chapter 4, “Using the vi Editor,” explains how to use the `vi` screen editor with text files. The `vi` editor is also used to edit command lines.

To enter the `vi` line editor mode while in the POSIX shell, press **ESC** to change from the usual “typing mode” into “edit mode.” Use editing commands to move the cursor or delete characters. Return to “typing mode” by entering the `vi` commands `i` or `a` to insert or append text.

The following table lists some `vi` editing commands.

Desired action	`vi` command
Move back one character	**`h`**
Move forward one character	**`l`**
Move back one word	**`b`**
Move forward one word	**`w`**
Move to the beginning of the line	**`^`**
Move to the end of the line	**`$`**
Delete the character under the cursor	**`x`**

The editor command set is governed by the setting of the `EDITOR` variable. Some possibilities are `vi` or `emacs`. Setting the `EDITOR` variable also depends on the `VISUAL` variable being defined.

To use the `vi` editor temporarily, type `set -o vi`. To turn off the `vi` editing mode, type `set +o vi`. To set the `EDITOR` variable automatically each time you log in, see “Setting the Login Environment”.

An Example of Line Editing with the vi Command Set

Activate the `vi` command set (if it is not already set at login by your login script):

```
$ set -o vi
```

Type this next line *but do not* press **Enter**:

```
$ ll /dve | grep '^d' | more
```

The second element should have been `/dev`. Correct the error by following these steps:

1. Press **ESC**. The cursor moves back one space (beneath the `e` in `more`). The line editor is now in "command mode."

   ```
   ll /dve | grep '^d' | more
   ```

2. Press **H** repeatedly to move the cursor to beneath the `v` in `/dve`.

   ```
   ll /dve | grep '^d' | more
   ```

3. Press **X**. The character `v` disappears, and the rest of the line shifts one space to the left to fill in. The cursor is now under the `e` in `/de`.

   ```
   ll /de | grep '^d' | more
   ```

4. Press **A**. The cursor moves one space to the right. The line editor is now ready to "append" text to the line.

   ```
   ll /de_ | grep '^d' | more
   ```

5. Press **V**. The character `v` is inserted after `/de`, completing the correction.

   ```
   ll /dev | grep '^d' | more
   ```

6. Press **Enter** to execute the command line.

Recalling Previous Commands

The POSIX shell stores the commands you execute in a **command history**. You can retrieve these commands, modify them, and re-execute them. For information on the C Shell implementation of command history, see the *Shells: User's Guide*.

For example, make sure you are in the POSIX Shell by typing `/usr/bin/sh`

Execute some commands, as a test. Then, to re-execute a previous command:

1. Make sure you have set `vi` as the command line editor (enter `set -o vi` on the command line for the login session, or make the appropriate entries in your `.profile` to set and export the `EDITOR` variable).
2. Press **ESC**.
3. Then press **K** repeatedly to scroll back to the previous command that you want.
4. Or, press **J** to scroll forward through the command history list.
5. Once you have found the command you want, you can edit it just as if it were the current command.
6. You can then execute whatever is on the command line by pressing **Enter**.

The POSIX Shell “remembers” the last 128 command lines you typed in and can display all or any of them. For example, type in some commands:

```
$ date
Thu Sep  8 15:01:51 MDT 1994
$ pwd
/home/terry
$ hostname
hpabc
```

Now type in this command:

```
$ history -3
121     date
122     pwd
123     hostname
124     history -3
```

Notice that the POSIX Shell displays the last three commands (`date`, `pwd`, and `hostname`) *and* the `history -3` command. You can increase the amount of the command history shown by using a larger negative number to follow `history`. For example, this will display the last 100 commands if there are 100 commands in the history:

```
$ history -100 | more
```

If there are fewer than 100 commands in the history, the full contents of the history will be displayed. The output of history is piped to the `more` command so you can see one screenful of the history commands at a time.

For More Information...

For more details on the command history in the POSIX Shell, see the relevant tutorial in the *Shells: User's Guide*. For more information on the Key Shell, see Appendix D, “The Key Shell.”

Briefer presentations are available in the *sh-posix*, *keysh*, and *csh* entries in the respective man pages.

Setting the Login Environment

When you log in, your shell automatically defines a unique working **environment** for you, which is maintained until you log out. Your environment defines such characteristics as who you are, where you are working, and what processes you are running. These characteristics are defined by values assigned to environment variables.

Your shell environment is analogous to an office environment. In the office, physical characteristics like lighting and temperature are similar for everyone. But many factors in your office environment are unique to you, such as your routine tasks and your individual workspace. Thus, your work environment is different from that of your co-workers—just as your shell environment is different from theirs.

The login Program

When you log in, HP-UX runs a program named `login`. This program starts your session using data stored in the `/etc/passwd` file, which contains one line for each system user. This file includes your user name, password (in encrypted form), home directory, and the shell to run when you log in. If `/etc/passwd` doesn't specify a shell, the POSIX Shell (`/usr/bin/sh`) is selected.

The `login` program does the following:

- Display the `Password:` prompt (if you have a password).
- Verify your user name and password in the `/etc/passwd` file.
- Assign default or user-defined values to the shell environment.
- Start executing the shell process.

Environment Variables

The shell environment defines how HP-UX interacts with you. The environment's characteristics are defined by **environment variables**, which consist of a name and a value. For example, the directory in which you begin each session is your **home directory**; its environment variable is the variable named `HOME`, and its value is assigned during the login process. Throughout this section, the value of `HOME` is equal to `/home/terry`.

Here are some environment variables set during the login process. Note that most of these will already be set in your default `.profile` file.

HOME

- Defines the user's home directory; the default directory for the `cd` command (for example, `/home/terry`).
- Default value assigned during login.

LOGNAME

- Contains the user name (for example, `terry`).
- Default value is *username*

MAIL

- Determines where the system looks for mail. Set based on the user name (for example, `/var/mail/terry`).
- Typical default value is `/var/mail/`*username*

PATH

- Sets the directories through which the system searches to find and execute commands.
- Typical default values include the following paths:

  `/usr/bin:/usr/bin:/usr/contrib/bin:/usr/local/bin:/usr/lib`

SHELL

- Determines which shell to run. Set to the last field in the `/etc/passwd` file entry for the user logging in. If this field is not defined, the default value is used.
- Typical default value is `/usr/bin/sh`

TERM

- Specifies the kind of terminal for which output is prepared.
- Typical default value is `hp`

TZ	• Provides the current time zone and difference from Greenwich Mean Time. Set to Mountain Standard Time by default; your system administrator should change the value if you are in another time zone. Set by the script `/etc/profile`. • Typical default value is `MST7MDT`
EDITOR	• Determines the default editor. • Typical default value is `vi`
DISPLAY	• Specifies window display host. Use on a remote system to display windows locally. • Typical default value is `DISPLAY=:0`

Using Login Scripts to Set the System Environment

During the login process, HP-UX prompts you for your user name and password (if applicable) before displaying a shell prompt. HP-UX also notes which shell you have selected to run, starts your shell process, and sets up your environment referring to *login scripts*. A login script is a file that lets you customize your environment.

A login script contains commands that let you define your system environment. When you log in, default values are assigned to environment variables. Login scripts provide an automatic way to change the value of these variables every time you begin a session.

Two types of login scripts are used:

- A system script for all users of a particular shell on your system or HP-UX cluster.
- Local login scripts in your own home directory.

Typically, a system administrator maintains the system login scripts. These scripts set up a default environment for everyone on that system. The POSIX shell uses a system login script named `/etc/profile`.

Once your account is set up, you maintain the local login scripts in your home directory. The local scripts allow you to set up an environment specific to your needs. The Bourne Shell looks for one script: `.profile`. The POSIX shell uses two login scripts: `.profile` and the one referred to by the ENV variable.

Default versions of the login scripts are placed in your home directory when your account is set up. Default versions are also in the `/etc` directory. For reference, the default `.profile` script for the POSIX shell is `/etc/skel/.profile`.

Why Use Login Scripts?

Login scripts provide a convenient way to set up the shell environment to suit individual needs. For example, the script can change the value of the search path used to find commands, change the shell prompt, set the terminal type, or simply cause the shell to greet you with a friendly message of your choosing.

Customizing your login script is not required, and the login script your system administrator provides should set up the most critical shell parameters.

A Summary of Login Scripts

The following table summarizes the login scripts for each shell. All the scripts run when you first log in. For more information on the POSIX, C, Key, and Bourne Shells, see the *Shells: User's Guide*.

Table 3-3 **Shells and Their Login Scripts**

Shell	System Login Script	Local Login Script
POSIX	`/etc/profile`	`$HOME/.profile`
C	`/etc/csh.login`	`$HOME/.cshrc` `$HOME/.login`
Bourne (obsolete)	`/etc/profile`	`$HOME/.profile`

Setting and Referencing Variables

Your shell uses both environment variables and shell variables to define your environment. Your login shell uses **environment variables** and passes them to all processes and subshells that you create. **Shell variables** are known only to your current shell and are not passed to subshells.

$

The POSIX shell sets variables using an assignment statement and an optional `export` command. In all shells, you refer to the *value* of a variable by placing a dollar sign ($) in front of the variable name.

Assigning Values to Variables

In the POSIX shell, variables are assigned (or set). They can also be created, if necessary. Both tasks are done with an assignment statement:

name=*value*

The *name* is the variable name and *value* is the value assigned to the variable. No spaces are allowed between *name* and = or between = and *value*.

In the following example, the shell prompt (`PS1`) is reset so that it reads:

```
Ready ==>
```

If `PS1` is a shell variable, the subshell (created by typing `sh`) does not know the new value. If you export `PS1`, the value of `PS1` passes to the subshell:

```
$ PS1="Ready ==> "     Set shell variable PS1.
Ready ==> sh           Type in subshell name.
$ exit                 Subshell now has default prompt; exit to original shell.
Ready ==> export PS1   Set environment variable with export.
Ready ==> sh           Enter subshell.
Ready ==>                Subshell knows the new value of PS1.

Ready ==> exit         Leave the subshell.
Ready ==> PS1=”$ “     Set environment variable with export.
$                      Back to normal.
```

Referencing the Values of Variables (Parameter Substitution)

All three shells use **parameter substitution** for referencing the value of variables. Parameter substitution means that the variable's value is substituted for the variable name. Parameter substitution occurs when a dollar sign ($) is placed in front of the variable name.

For example, earlier you learned to determine your login shell with the command `echo $SHELL`:

```
$ echo SHELL        Because $ is omitted, the word SHELL is echoed.
SHELL
$ echo $SHELL       The $ is included, so the value of SHELL is echoed.
/usr/bin/sh
```

The `echo $SHELL` command uses parameter substitution. The shell substitutes the value of the environment variable named `SHELL` into the `echo` command because the dollar sign ($) precedes the variable name.

For More Information...

To learn more about parameter substitution, refer to *sh*, *sh-posix*, *keysh*, or *csh* man pages or to the *Shells: User's Guide*.

Finding Commands with Search Paths

PATH

When you type a command, HP-UX must be able to find the directory containing the command before it can run the command. The PATH environment variable contains a list of directories you want HP-UX to search when looking for commands. Your PATH should contain all the directories necessary to locate all the commands that you use.

PATH Variable Format

The PATH variable is read from your `.profile` or `/etc/profile` login script. It contains a list of directories to search, separated by colons. There should be no spaces surrounding the colons. You can also use the `echo` command to determine the current value of PATH, as follows:

```
$ echo $PATH
/usr/bin/sh:/usr/bin:/usr/local/bin
```

This line means that when you type a command, the shell first searches for the command in the `/usr/bin/sh` directory, in the `/usr/bin` directory, and then in the `/usr/local/bin` directory. If the command is not found in any of these directories, the shell displays this message:

```
command_name: Command not found.
```

Changing PATH

If the shell can't find a command that you know exists, you have two options:

1. Type the full path name of the command. For example, if you wish to execute a command called `prog`, and it resides in the directory `/home/sue/bin`, type this:

   ```
   $ /home/sue/bin/prog
   ```

2. *Or*, change the value of the PATH variable to add the command path, a better long-term solution if you use the command frequently.

The following table shows the path names of the most frequently used directories.

Directory	What It Contains
`/usr/bin`	Frequently used HP-UX commands.
`/usr/sbin`	Commands the system administrator uses.
`/usr/bin/sh`	POSIX Shell
`/usr/contrib/bin`	Contributed programs not supported by Hewlett-Packard.
`/usr/local/bin`	Programs and commands written at your location.
`$HOME/bin`	A directory you might create for your own shell scripts and programs.

CAUTION Because of the potential security risk, do not put your current directory (usually represented as `.`) as the first element in PATH. Leave the current directory out of your PATH, or include it only as the *last* element.

Remember that directories in PATH are searched in the order in which they appear (left to right). In general, put the most frequently used directories first in the path — unless two commands in the search path have the same name (for example, `/usr/bin/rm` and `$HOME/bin/rm`). In this example, if you want the shell to find your version of `rm` first, put `$HOME/bin` before `/usr/bin` in PATH.

The following example shows how to alter PATH to include $HOME/bin before any other directories, and to include the current directory as the last directory in the search path (this example assumes you're using the POSIX shell):

```
$ echo $PATH
/usr/bin/sh:/usr/bin:/usr/bin:
/usr/contrib/bin:/usr/local/bin
$ PATH=$HOME/bin:$PATH:.
$ echo $PATH
/home/terry/bin:/usr/bin/sh:/usr/bin:
/usr/bin:/usr/contrib/bin:/usr/local/bin:.
```

Including "." as the last element makes the current directory the last one searched.

Setting PATH as an Environment Variable

Normally, you set PATH as a environment variable, so it is set to the appropriate value when you log in. In the Bourne and POSIX shells, you can change PATH in the .profile script and export it. You can find out more about these scripts in *Shells: User's Guide*.

Setting Terminal Characteristics

For most effective use of your terminal, HP-UX must know the type of terminal or graphics display you're using. If no terminal type is provided, the default value is `TERM=hp`. The `tset` command sets terminal characteristics.

The default local login script prompts you to enter your terminal type as follows:

```
TERM = (hp)
```

Pressing **Enter** sets the `TERM` environment variable to `hp`, the default value. This value works with Hewlett-Packard terminals, but it may not let you take full advantage of your terminal or graphics display features. Entering a different value sets the `TERM` environment variable to that value.

ttytype(1)

The *ttytype*(1) command can be used to help you determine your terminal type. See the man page for details.

Selecting a Value for the TERM Variable

HP-UX supports many terminal types. The `/usr/share/lib/terminfo` database tells HP-UX how to communicate with each terminal type. When you assign a value to `TERM`, the value must equal a value in the `terminfo` database.

For example, the files listed under `/usr/share/lib/terminfo/2` show all acceptable `TERM` values that begin with 2 (this is only a partial listing):

```
$ ls /usr/share/lib/terminfo/2
2382   2397a    2621a    2623p  2626-x40  2640a
2392   2500     2621k45  2624   2626A     2640b
2392A  2621     2621nl   2624a  2626P     2644
2392a  2621-48  2621nt   2624p  2626a     2645
2393   2621-ba  2621p    2625   2626p     2647
2393A  2621-fl  2621wl   2626   2627      2647F
```

Here are some common terminal and graphics display settings for Hewlett-Packard equipment. When more than one choice is listed, all choices are equivalent.

If You Are Using a ..	Set TERM to ..
terminal	the terminal's model number; for example `2622`, `hp2622`, `262x`, or `2392`
HP Vectra PC	`2392`
medium resolution graphics display (512x600 pixels)	`300l` or `hp300l`
high resolution graphics display (1024x768 pixels)	`300h` or `hp300h`
HP 98550 display station (1280x1024 pixels)	`98550`, `hp98550`, `98550a`, or `hp98550a`
HP 98720 or HP 98721 display station (1280x1024 pixels)	`98720`, `hp98720`, `98720a`, `hp98720a`, `98721`, `hp98721`, `98721a`, or `hp98721a`

Setting TERM with the tset Command

The `tset` command (with the -s option) sets the value of `TERM` and initializes your terminal characteristics. If you always log in using the same terminal type, you may change your `.profile` to eliminate the `TERM` prompt. Your `.profile` contains a line similar to:

```
eval ` tset -s -Q -m ':?hp' `
```

This command displays the `TERM` prompt. To customize the command, replace `?hp` with your terminal type.

For example, the following command initializes your terminal as a high-resolution graphics display (`300h`), but the `TERM` prompt itself does not display:

```
eval ` tset -s -Q -m ':300h' `
```

If you use more than one type of terminal (such as one at work and one at home), you can modify your `tset` command to include multiple terminal types. See *tset*(1) in the *HP-UX Reference* for more information.

Chapter Command Summary

To Do This...	Type This...
Delete (remove) a file interactively	`rm -i` *filename*
Run several commands on same line	*command*`;`*command2*
Temporarily change to Key Shell	`/usr/bin/keysh`
Find command information online	`man` *command_name*
See what processes are running	`ps -ef`
Stop a process	`kill` *PID*
Stop an unresponsive process	`kill -9` *PID*
Redirect standard output to a file	*command* `>` *outfile*
Append standard output on a file	*command* `>>` *outfile*
Redirect input from a file to a command	*command* `<` *infile*
Redirect both standard input and output to a file	*command* `<` *infile* `>` *outfile*
Connect ("pipe" between) two processes	*command1* `|` *command2*
Save command output to a file and send to another command	*command1* `|` `tee` *file* `|` *command2*
Determine what shell you are in	`echo $SHELL`
Temporarily change to POSIX Shell	`/usr/bin/sh`
Temporarily change to C Shell	`/usr/bin/csh`
Permanently change to another shell	`chsh` *username shell_path_name* (then log out and log in again)
Set command-line editor	`set -o` *editor_name*

Edit your command line (once editor is set)	Press **ESC**; use `vi` commands to move cursor and enter text
Recall a previous command line	In `vi` mode, press **ESC**; press **`k`** (backwards) or **`j`** (forward) to move through command history file
Execute a previous command line	Press **Enter** when desired command line is displayed
Set a variable value	*VARIABLE_NAME*`=` *variable_value*
Display PATH setting	**`echo $PATH`**
Set terminal parameters	**`tset`** *options term_type*

4 Using the vi Editor

Editing Text

The `vi` (*Vi*sual) editor is the default text editor for your HP-UX system. The `vi` editor is a powerful, versatile editing tool.

The `vi` editor is included with every HP-UX system and most other versions of UNIX as well. It is not difficult to learn or use. This chapter will teach you the basics of it's operation.

Starting the vi Editor

Start `vi` by entering the following command at the prompt:

vi(1)

```
$ vi filename
```

If a file called *filename* exists, you will see the first screen of that file. If the file does not exist, it is created, and you will see a blank screen.

For the impatient reader who does not want to read the next eight pages before learning how to exit vi, you will want to know the secret key sequence: **Esc**`:q!` which will exit vi and discard the edits you made.

NOTE Alternative editors do exist.

If you do not wish to use `vi`, you can use the optional, freely available Emacs editor. Emacs is a widely used public domain editor that provides many versatile functions. GNU Emacs, and other related software, is available from:

```
Free Software Foundation, Inc.
675 Massachusetts Avenue
Cambridge, MA   02139-3309
USA

+1-617-876-3296
gnu@prep.ai.mit.edu
ftp://prep.ai.mit.edu/pub/gnu/GETTING.GNU.SOFTWARE
```

You can also get information on Emacs from *GNU Emacs: UNIX Text Editing and Programming*, Addison-Wesley, 1992.

Command Mode and Text Entry Mode in vi

The `vi` editor has two basic modes for manipulating text:

- Command mode
- Text entry mode

When you enter `vi`, you will be in command mode until you enter one of the text entry codes, such as `i` or `a`, which are explained in this section.

In text entry mode, you can backspace and type over text you have just entered (by pressing **CTRL-H** or **Backspace**). But, if you want to move around otherwise in your text and execute other text manipulation commands, you will have to press **ESC** to return to command mode.

If You Make Mistakes

Use the following procedures to correct mistakes:

- If you type an error while entering text, press **Backspace** to back up over the error, and then re-type the correct text.
- The `u` undo command (lowercase `u`) reverses the last change made to your text. The `U` undo command (uppercase `U`) reverses all the changes made to a *single line* since you began editing that line.
- If you type several errors and cannot recover, exit `vi` without saving the file, and start over. To do this, press **ESC**. Then type `q!` **Enter**.

CAUTION

Save your work often.

While you work in a file, save your changes frequently (every 5 to 10 minutes). Regular saving helps you avoid losing your changes if there is a power failure or other accident. See “Saving Your Work and Exiting vi”.

Of course you probably won’t listen to us until you lose a day’s worth of work because the electricity went off suddenly. Then you will develop the habit of frequently saving your work every 5 to 10 minutes.

Entering and Deleting Text

Press **ESC** to ensure that `vi` is in command mode. Then you can execute any of the following commands (among others). The text entry commands put `vi` in text mode; the deletion commands do not.

Type This...	To Enter Text...
`i`	Preceding the cursor. Everything after cursor moves to the right.
`I`	Before the first character of the line.
`a`	After the current cursor position. Cursor moves to the right, and text is inserted as with `i`.
`A`	At the end of the line.
`o`	Open a blank line below cursor for text entry (lowercase `o`).
`O`	Open a blank line above cursor for text entry (uppercase `O`).

Type This...	To Delete...
`x`	Character highlighted by the cursor. Does not put document in text mode.
n`x`	*n* characters, starting at the cursor.
`dw`	From cursor to beginning of next word or first punctuation.
`dd`	Deletes current line.
`dG`	All lines to end of file, including current line.

When you enter commands in `vi`, letter case (caps or small letters) does matter. For example, lowercase `i` and uppercase `I` represent two different commands. Therefore, if the cursor doesn't move as it should, make sure the **Caps** key is not locked on, or see your system administrator.

Positioning the Cursor

These keys move the cursor as follows (press **ESC** first for command mode):

To Do This…	Type This…
Move the cursor right.	`l` or the right arrow key
Move the cursor left.	`h` or the left arrow key
Move the cursor up.	`k` or the up arrow key
Move the cursor down.	`j` or the down arrow key

NOTE

Working with Line Numbers:

To move to a specific line, use `G`, a “goto” command. For example, suppose you are editing a file and want to go to line 799. You type `799G`, and the cursor moves to line 799. Similarly, to go to line 1 of the file, type `1G`. To move the cursor to the last line, simply type `G`.

To find the current line number, press **CTRL-G**; to display line numbers along the left margin of your file, type `:set number`.

Scrolling through Text

To scroll, press **ESC** to ensure that you're in command mode, then type the appropriate key while holding down the **CTRL** key.

To Scroll…	Type This…
Backward to previous screen.	**CTRL-B**
Backward one half screen.	**CTRL-U**
Backward one line.	**CTRL-Y**
Forward to next screen.	**CTRL-F**
Forward one half screen.	**CTRL-D**
Forward one line.	**CTRL-E**

Finding Text Patterns

To search forward from the current cursor position, use this command:

/pattern **Enter**

where *pattern* represents the specific sequence of characters you want to search for.

To search backward from the current cursor position, use this command:

?*pattern* **Enter**

When you press **Enter**, `vi` searches for the specified pattern and positions the cursor at the first character in the pattern sequence. For example, to search forward for the word `place`, type:

`/place` **Enter**

If `vi` finds `place`, it positions the cursor at the `p`. To search for additional occurrences of `place`, press either **n** or **N**:

- **n** continues searching in the same direction for `place`.
- **N** reverses the direction of the pattern search.

If `vi` does not find the pattern you specified, it displays the following message at the bottom of your screen, and the cursor is not moved:

```
Pattern not found
```

Searching for Special Occurrences

In the previous example, `vi` finds *any* sequence containing the pattern `place`, including `displace`, `placement`, and `replaced`.

- To find the single word `place`, type the pattern with a space before and after it (the   represents a space):

  `/ place ` **Enter**

- To find `place` occurring only at the beginning of a line, precede the pattern with a caret (^):

  `/^place` **Enter**

- To find `place` occurring only at the end of a line, follow the pattern with a dollar sign ($):

  `/place$` **Enter**

Using "^"

To search literally for such a character as a caret (^) or a dollar sign ($), precede the character with a backward slash (\). The backward slash tells `vi` to search for a special character.

Using "$"

Special characters are those (such as `^` , `$` , `*` , `/` , and `.`) that have special functions for `vi`. For example, a `$` normally means "go to the end of the line," but if the `$` is preceded immediately by a `\`, the `$` is simply another ordinary character.

Using "\"

For example, `/(No \$ money)` searches forward for the pattern `(No $ money)`. The escape character (\) immediately preceding the `$` tells `vi` to search literally for a dollar sign.

Replacing Characters

To replace a single character of text, press **ESC** to enter command mode, position the cursor over the character you want to replace, and type `r` while in command mode. Then type the replacement character. The `r` command lets you substitute only one character. After you have replaced the character, you are back in command mode.

Substituting Characters

To substitute one or more characters for a single character, type `s` while in command mode. Unlike the `r` command, the `s` command puts you in insert mode and lets you substitute more than one character for a single character.

When you type the `s` command, a dollar sign ($) appears in place of the character. After you type the desired character (or characters), press **ESC**.

To substitute for more than one original character, precede the `s` command by the number of characters.

Saving Your Work and Exiting vi

You can save your work with or without quitting `vi`. Press **ESC** to ensure that `vi` is in command mode.

To Do This...	Type This...
Save without quitting `vi`	`:w`
Save and quit `vi`	`:wq`
Quit `vi` without saving changes	`:q!`
Save under another file name	`:w` *filename*
Save in an existing file and overwrite that file	`:w!` *filename*

To print your files, see “Viewing and Printing Files” in Chapter 2, “Working with Files and Directories.”

Using Options to Change Your vi Environment

To customize `vi`, you can set (or unset) any of several options. When you enter `vi`, options are assigned certain default values.

When exiting `vi`, all options return to the default, so you will need to reset your options each time you enter `vi`. See the next section for how to make your options permanent.

To see all your default options, type:

`:set all` **Enter**

To change these values, use the `:set` command:

`:set` *option* **Enter**

where *option* is the name of the editor option you want to use (see the following table for descriptions of some of these options).

To unset (discontinue) an editor option, type `no` before the option:

`:set no`*option* **Enter**

Table 4-1 Editor Options

Option	Abbrev.	Default	Effect When Set
`all`	~	~	Lists all editor options on the screen.
`autoindent`	`ai`	`noai`	Begins each new line of text in the same column as the previous line (useful for programmers).
`ignorecase`	`ic`	`noic`	Causes `vi` to ignore uppercase and lowercase during searches.
`number`	`nu`	`nonu`	Numbers each line of text.
`readonly`	~	`noreadonly`	Enables write protection on the file you are editing. This ensures that you don't accidentally change or destroy the file's contents.
`showmatch`	`sm`	`nosm`	Shows the opening parenthesis, brace, or bracket when you type the corresponding closing parenthesis, brace, or bracket. This option is useful when you are typing mathematical expressions or writing programs in a language that uses parentheses, braces, or brackets.
`showmode`	~	`noshowmode`	Displays a message like "`INPUT MODE`" or "`REPLACE MODE`" at the bottom of the screen whenever you are in either of these modes.
`wrapmargin`	`wm`	`wm=0` (zero)	Changes the right margin. *n* equals the number of spaces in the right margin. For example, if you're using an 80-column terminal, then `:set wm=8` sets the right margin at column 72.

Making Your vi Environment Permanent

To avoid setting options or defining abbreviations or macros each time you enter `vi`, place all options and definitions you normally use into an `.exrc` file in your home directory. This file is read automatically each time you enter `vi` and its contents make your customized `vi` environment permanent.

To create or change the `.exrc` file, follow these steps:

1. Type `cd` at the HP-UX prompt to ensure that you're in your home directory; then use `vi` to create or edit the `.exrc` file:

   ```
   $ cd
   $ vi .exrc
   ```

2. Type the options, word abbreviations, and macros you want to make permanent (don't precede the commands with a colon).

3. Type `:wq` to save the text and exit `vi`.

After creating the `.exrc` file, you can access it whenever you want to change your `vi` environment. Any of the editor options discussed in the previous section can be placed in this file.

An Example of Changing Your .exrc File

In “Using Options to Change Your vi Environment”, you saw examples of some of the options that change the overall behavior of `vi`. You can also make `vi` recognize short forms of commonly used expressions by using `ab` to define an abbreviation.

If you include the following options and abbreviations in your `.exrc` file:

```
set wm=8
set showmode
ab eeg Electrical Engineering
```

Then you have changed your `vi` environment so that each time you enter `vi`, you can expect the following:

- The right margin automatically contains eight spaces (changing the default from zero), and a carriage return will occur after approximately 72 spaces.
- The lower right part of the screen will show “INPUT MODE” when you are in any of the text insert modes.
- Whenever you enter `eeg`, this abbreviation will automatically expand into the words `Electrical Engineering`.

For More Information…

For more information on this versatile editor, see *The Ultimate Guide to the vi and ex Text Editors*.

Chapter Command Summary

To Do This…	Type This…
Enter `vi` and create or use existing *file*.	`vi` *file* **Enter**
Insert text before the cursor.	`i`
Append text after the cursor.	`a`
Delete one character.	`x`
Return to command mode.	**ESC**
Move the cursor right.	`l` or right arrow
Move the cursor left.	`h` or left arrow
Move the cursor up.	`k` or up arrow key
Move the cursor down.	`j` or down arrow key
Exit `vi` without saving changes.	`:q!` **Enter**
Write (save) the current file.	`:w`
Write the current file and quit (exit) `vi`.	`:wq`
Write the current file to *filename*	`:w` *filename*
Overwrite contents of *filename* with the current file.	`:w!` *filename*
Write lines *x* through *y* of current file to *filename*.	`:`*x,y* `w` *filename* (*x,y* are specific line numbers or place markers)
Insert contents of *filename* into the current file.	`:r` *filename*
Run an HP-UX command while in `vi`	`:!`*command*
Print the current file (see “Viewing and Printing Files” in Chapter 2, “Working with Files and Directories.”)	`:!lp %`

5 Using Electronic Mail

e-mail

With an electronic mailer program, you can send messages to other users on your system. If your system is configured to a network, such as a local area network (**LAN**), you can send mail messages to users on other systems.

If you are connected to a larger network such as the Internet, you can communicate with users world-wide. Consult your system administrator to determine where you can send electronic mail.

Starting the elm Mailer

elm(1)

One of the popular HP-UX mailer programs is called `elm` (the *el*ectronic *m*ailer). Other mailers, such as `mail`, and `mailx`, are also available.

1. To start the `elm` mailer, type `elm` at the system prompt.

   ```
   $ elm
   ```

2. If this is the first time you have started `elm`, you will be asked two questions about creating directories and folders. Answer `y` (yes) to both.

3. To get help in `elm`, press **?** at the command prompt.

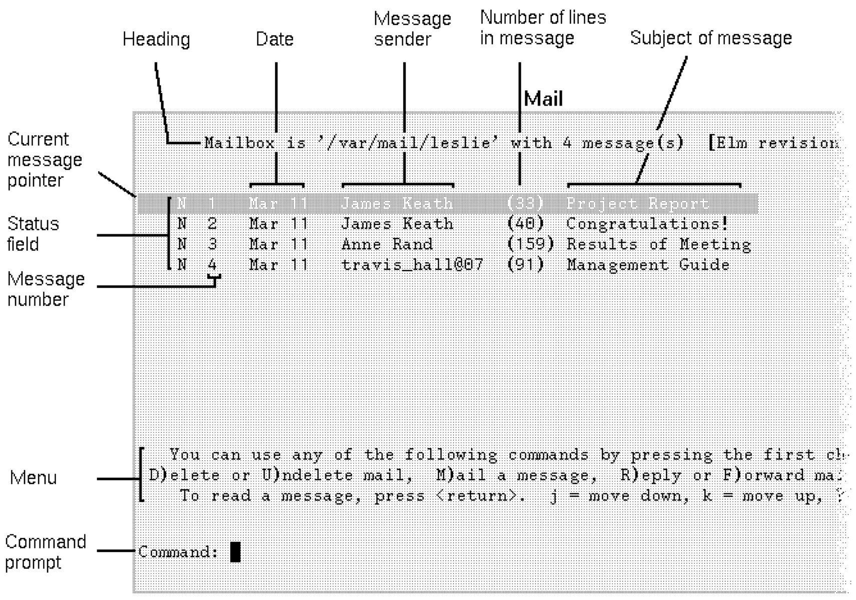
   - For a summary of all of the commands you can use from `elm`, press **?** again.

   - For help on a particular command, type the first letter of the command (for example, press **R** to get information about the `reply` command).

Understanding the Main Screen

The main screen of `elm` has several components, shown in the following illustration.

Figure 5-1 **The elm mailer lets you send and receive messages.**



```
Mailbox is '/var/mail/leslie' with 4 message(s)  [Elm revision
N  1   Mar 11   James Keath      (33)  Project Report
N  2   Mar 11   James Keath      (40)  Congratulations!
N  3   Mar 11   Anne Rand        (159) Results of Meeting
N  4   Mar 11   travis_hall@07   (91)  Management Guide

    You can use any of the following commands by pressing the first ch
D)elete or U)ndelete mail,  M)ail a message,  R)eply or F)orward ma
    To read a message, press <return>.  j = move down, k = move up, ?

Command: █
```

The main screen has these elements:

Heading	The first line on the screen displays the current mailbox, the number of messages, and the current `elm` revision number.
Date	The date when the message was sent.
Message sender	The mail address of the person who sent the message.
Number of lines	The total number of lines in the message.
Subject of message	A description of the contents of the message.

Current message pointer	The highlight indicates the current message.
Status field	The status or characteristics of each message. The field can be blank, or contain the following common status characters: `N`—new message `NU`—new urgent message `D`—message is to be deleted `M`—multi-part message
Message number	This is used to specify a message.
Menu	This three-line menu at the bottom of the screen shows the commands available.
Command prompt	In response to the “Command:” prompt, type any of the commands in the menu.

Entering elm Commands

Even if you are not familiar with mailers, you can easily use `elm` by following the instructions displayed on each screen.

To enter an `elm` command, type the first letter (uppercase or lowercase) of the command. Commands in `elm` are not case-sensitive.

For a summary of elm commands, press **?** within `elm`, then press **?** again.

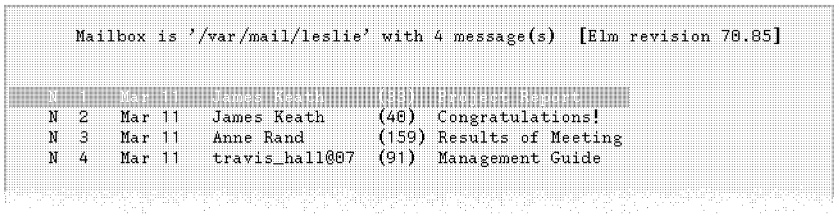
Also see “Chapter Command Summary” at the end of this chapter.

Reading Your Mail

To read your mail, start `elm` by typing `elm` at the command prompt.

If you have mail, `elm` displays a list of mail messages. You can read the current message or pick a specific message to read. Messages appear in a display similar to the following:

Figure 5-2 **Elm lists your mail messages.**



```
     Mailbox is '/var/mail/leslie' with 4 message(s)  [Elm revision 70.85]


   N  1   Mar 11   James Keath       (33)  Project Report
   N  2   Mar 11   James Keath       (40)  Congratulations!
   N  3   Mar 11   Anne Rand         (159) Results of Meeting
   N  4   Mar 11   travis_hall@07    (91)  Management Guide
```

- To read the current (highlighted) message, press **Enter**.

  (You can configure `elm` to use a > to indicate the current message. See "Customizing elm".)

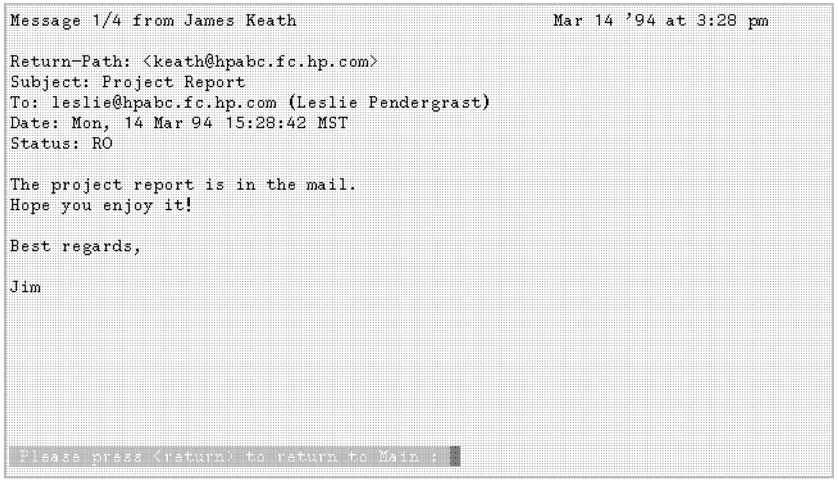
- If the message has multiple parts, indicated by an "M" in the status field, press **V** one or more times to "View" the sub-parts of the message.
- To advance to the next message, press **J**.
- To move to the previous one, press **K**.
- To jump to a specific message in the header, type the number of the message and press **Enter**.

CAUTION Use only one mailer at a time.

Problems with system behavior may result if you attempt to use one of the other HP-UX mailers, such as `mailx`, while you are also working in `elm`.

This example shows the output from reading message 1.

Figure 5-3 An example message.



```
Message 1/4 from James Keath                                Mar 14 '94 at 3:28 pm

Return-Path: <keath@hpabc.fc.hp.com>
Subject: Project Report
To: leslie@hpabc.fc.hp.com (Leslie Pendergrast)
Date: Mon, 14 Mar 94 15:28:42 MST
Status: RO

The project report is in the mail.
Hope you enjoy it!

Best regards,

Jim






 Please press <return> to return to Main :
```

To return to the `elm` main screen, press **Enter**.

NOTE Seeing more mail headers:

Only ten message headers appear on the screen at one time. If you have more than ten messages you can display them as follows:

- To see the next page of message headers, press **+**.
- To see the previous page, press **-**.
- To move to the first message in the list, press **=**.
- To move to the last message in the list, press *****.

Sending Mail to Users on Your System

One of the easiest ways to learn how to send a mail message is to send one to yourself.

1. Start `elm` by typing `elm` at the command prompt.
2. To mail a message, press **M** as the response to "Command:"

   ```
   Command: m
   ```

3. `elm` responds with a prompt requesting the mail address of the recipient. (To use **aliases** for recipients, see "Using Mail Aliases".)

   On your local system, your mail address is the same as your user name. Type your user name and press **Enter**. For example:

   ```
   Send the message to: leslie
   ```

   To send a message to more than one person, simply specify each recipient's name.

   ```
   Send the message to:  mike leslie tj
   ```

4. `elm` then responds with a subject line prompt. Type the subject line of the message and press **Enter**. For example:

   ```
   Subject of Message: Important Message to Myself
   ```

5. `elm` responds with a prompt for the carbon copies. Type the mail addresses of any users who should receive a carbon copy of the message, then press **Enter**.

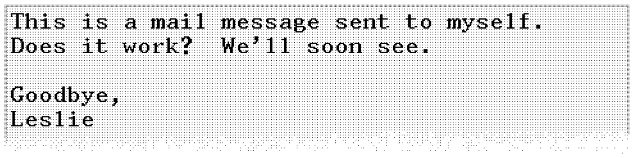
   In this example, since you're sending a message to yourself, you don't want to send any additional copies so press **Enter**.

(continued)

6. `elm` brings up a `vi` editor window. Type the message. For information about using the `vi` editor, see Chapter 4, “Using the vi Editor.”

   This screen shows a sample message.

Figure 5-4 **An example message.**



```
This is a mail message sent to myself.
Does it work?  We'll soon see.

Goodbye,
Leslie
```

   To configure your workstation to bring up a different editor, see “Customizing elm”.

7. When you are done typing, save the message.

8. After you exit the editor, you will see the following message on the screen.

```
Please choose one of the following options by the first character:
E)dit message, edit the H)eaders, S)end it, or F)orget it. s
```

9. To change any of the header information for your message (for example, to add recipients to the `Cc` list, change the message subject, or mark the message as Urgent), press **H**.

10. To send your message, type **s**

11. After you have sent the mail message, the `elm` main screen reappears and displays the message “`Mail sent!`”. It might take a few seconds for the system to deliver the message.

Sending Mail to Users on Other Systems

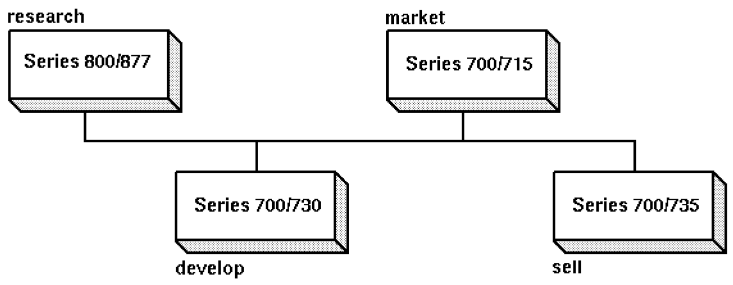
If your system is connected to other systems over a **LAN** (local area network) or other networking facility that supports `elm`, you can also send mail to users on other systems.

Host Names

Every workstation connected over a network has a unique **host name** (also known as a node name). To send mail to a user on another workstation, you must know the host name of the user's workstation.

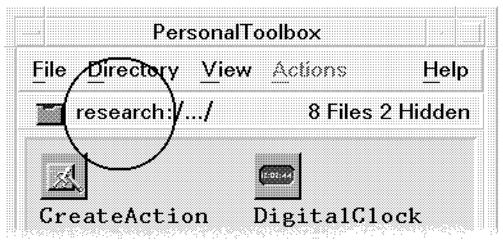
This figure shows an example LAN with four workstations connected. The host names are `research`, `market`, `develop`, and `sell`.

Figure 5-5 **Example host names on a network.**



To determine your system's host name, use the `hostname` command.

Figure 5-6 **The host name is circled.**



Mail Syntax when Mailing to Other Systems

The general syntax used when mailing to a user on another workstation is either:

user@host	simple format
user@host.organization.domain	Domain naming

host is the host name of the workstation the person is on, and *user* is the person's unique user name.

If the user you are sending to is at another organization, the syntax may also include that person's *organization* name (such as `ucsd` or `hp`), and the *domain* to which the organization belongs (such as `edu` for educational institutions, and `com` for commercial businesses).

Your system administrator can tell you which syntax to use.

Examples

Suppose you want to send mail to user john, who uses the workstation named sell. Then, in response to the elm prompt, you would type the following:

```
john@sell
```

To send mail to `arnie` on your workstation, `john` on the `sell` workstation, and `leopold` on the `research` workstation, use the following addresses:

```
arnie  john@sell  leopold@research
```

To send mail to `mark` who works for a university and `davidm` who works for the XYZ company, use the following addresses:

```
mark@hub.ucsd.edu davidm@xyz.com
```

In the first part of this example, `mark` is the user name, `hub` is the hostname of Mark's workstation, `ucsd` is the abbreviation for “University of California at San Diego”, and `edu` is the domain name for educational institutions. In the second part, `davidm` is the user name, `xyz` is the abbreviation for the XYZ Company, and `com` is the domain name for companies and businesses.

Using Mail Aliases

Shortcuts for mailing

You can set mail **aliases** for people you mail to frequently. Aliases are shortcuts that you define so you don't have to type a recipient's complete mail address.

`elm` provides two types of aliases:

User alias	For private use. Individual users create and maintain their own user aliases. To create user aliases, see “Creating Mail Aliases”.
System alias	For use by all users with the same host. The system administrator must create and maintain the system aliases.

Understanding Mail Aliases

Aliases allow you to send messages to several recipients by specifying a single name, or to send messages without specifying a complete mail address.

For example, for Christine Lawson at mail address `christine@market.elm.com` you might create the alias `chris`. To send a message to Christine, you could enter her alias at the `To:` prompt:

```
To:  chris
```

`elm` automatically converts it to the proper address and full name:

```
To: christine@market.elm.com (Christine Lawson)
```

Here are more example aliases. (These aliases are shown in the format that `elm` uses to store them in the `/`*HomeDirectory*`/.elm/aliases.text` file. This file is automatically appended to when you create aliases.)

```
writers = My writing group = john jtl michele ken willa bg
friends = David and Mark = davidm@xyz.com mark@hub.ucsd.edu
tom     = Tom Middleton = tmm@eff.org
```

Thus, sending a message to `writers` means that `john, jtl, michele, ken, willa,` and `bg` on the local system all receive the message. Sending a message to `friends` means that `davidm@xyz.com` and `mark@hub.ucsd.edu` receive the message. Sending a message to `tom` is shorthand for sending to `tmm@eff.org`.

Creating Mail Aliases

1. Press **A** in response to the “Command:” prompt on the `elm` screen. The “Alias commands” sub-menu appears.
2. Press **M** at the “Alias:” prompt to make a new alias.

NOTE TIP for making aliases:

To make an alias for the person who sent the current message, use `a` instead of `m`. This command reads in the sender's address automatically in step 5.

3. Enter the desired alias name, and press **Enter**. For example:

   ```
   Enter alias name: chris Enter
   ```

NOTE TIP for extra aliases:

You can set more than one alias name. For example, if you enter `chris` and `cl` in the example under “Understanding Mail Aliases”, you can use either alias to send mail to Christine Lawson.

4. Enter the full name at the next prompt, and press **Enter**.

   For example:

   ```
   Full name for chris: Christine Lawson Enter
   ```
5. Enter the address in response to the next prompt, and press **Enter**.

   For example:

   ```
   Enter address for chris: christine@market.elm.com Enter
   ```
   Then the sub-menu "Alias commands" returns.
6. Press **Enter** to set the alias. The main screen returns.

NOTE

TIP for group aliases:

To create an alias for a group of users, enter more than one address. For example, if you set:

In step 3	`Enter alias name: friends`
In step 4	`Full name for friends: Incredible Three`
In step 5	`Enter address for friends: chris, john, alex@market`

You can send a message to `chris, john,` and `alex` by responding to the “To:” prompt with:

`To: friends` **Enter**

Note that you must make the aliases `chris` and `john` before making the group alias. If aliases have not been made, you must enter the user's full address, as is done with `alex@market`.

Listing and Deleting Aliases

You can view your aliases and delete those you no longer need.

1. Press **A** in response to the “Command:” prompt on the `elm` screen. The “Alias commands” sub-menu appears.
2. To check the address of a specific alias, press **P** and enter the alias name.
3. To list user aliases, press **U**.
4. To list system aliases, press **S**.
5. To delete a specific alias, press **D** and enter the alias name.
6. To return to `elm`'s main screen, press **R** or press **Enter**.

Replying to Messages

When you receive a message, you may want to reply to it right away.

1. Move the message pointer to highlight the message to which you want to reply.
2. To reply to just the sender of the message, press **R**.

   Or, to reply to everyone who received the message, press **G**.

   The following message appears at the bottom of the screen.

   ```
   Command: Reply to message      Copy message (y/n)? n
   ```

   (If you pressed **G**, “Group reply” appears after “Command:”.)
3. Press **Enter** if you don't need to include the original message in your reply.

   Press **Y** if you do need to include it.
4. The bottom part of the screen now appears similar to this:

   ```
    Command: Reply to message To: charlie (Charlie Pike)
   Subject of message: Re: Meeting Schedule Changed!
   ```

   The “To:” and “Subject of message:” fields are automatically filled in. The “Re:” in front of the subject indicates that the message is a reply.

   If the message you are replying to has no subject, “Re: your mail” is used in the subject field.

   You can change the subject field by backspacing over it and typing a new subject.
5. `elm` responds with a prompt for the carbon copies.

   Type the mail addresses of any users who should receive a carbon copy of the message, then press **Enter**.
6. `elm` brings up a `vi` editor.

   If you chose yes in reply to the “Copy message” prompt, the editor will be invoked with the original message copied into the editing buffer. `elm` inserts the default prefix character (>) and a blank at the beginning of each line of the copied message.

   Type an introductory message.

For help with using the `vi` editor, see Chapter 4, “Using the vi Editor.”

To configure your workstation to bring up a different editor, see “Customizing elm”.

7. When you are done typing, save the file.
8. After you exit the editor, you will see the following message on the screen.

```
Please choose one of the following options by the first character:
E)dit message, edit the H)eaders, S)end it, or F)orget it. s
```

9. To change any of the header information for your message (for example, to add recipients to the `Cc` list, change the message subject, or mark the message as Urgent), press **`H`**.
10. To send your message, type **`s`**
11. After you have sent the mail message, the `elm` main screen reappears and displays the message “`Mail sent!`”.

Forwarding Messages

You may receive messages that you want to forward to someone else, with or without adding your own comments.

1. Move the message pointer to the message you want to forward.
2. Press **F**. The following message appears at the bottom of the screen.

   ```
   Command: Forward          Edit outgoing message (y/n)? y
   ```
3. Press **N** if you don't want to edit the forwarded message.

   Press **Enter** if you do want to edit it.
4. The bottom part of the screen changes to:

   ```
   Command: Forward          Edit outgoing message (y/n)? Yes

   Send the message to:
   ```
5. Enter the mail address of the person you want to forward the message to. To use aliases for recipient names, see “Using Mail Aliases”.

   For example, if you enter `donesky`,

   ```
   Command: Forward                            To: donesky

   Subject of message: AI Competition (fwd)
   ```
   “(fwd)” is attached to the subject automatically. It tells the recipient that this is a forwarded message.

   If the message you are forwarding has no subject, “Forwarded mail” is used in the subject of message field. The subject field can be edited.
6. `elm` responds with a prompt for the carbon copies.

   Type the user addresses of any users who should receive a carbon copy of the message, then press **Enter**.
7. `elm` brings up a `vi` editor window.

   If you chose yes in reply to the “Edit outgoing message” prompt, the editor will be invoked with the original message copied into the editing buffer. `elm` inserts the default prefix character (>) and a blank at the beginning of each line of the copied message.
8. Type an introductory message.

For information about using the `vi` editor, see Chapter 4, “Using the vi Editor.”

9. When you are done typing, save your file.
10. After you exit the editor, you will see the following message on the screen.

```
Please choose one of the following options by the first character:
E)dit message, edit the H)eaders, S)end it, or F)orget it. s
```

11. To change any of the header information for your message (for example, to add recipients to the `Cc` list, change the message subject, or mark the message as Urgent), press **`H`**.
12. To send your message, type **`s`**
13. After you have sent the mail message, the `elm` main screen reappears and displays the message “`Mail sent!`”.

Saving Messages to a File

You may want to save important messages for future reference.

1. To save the current message to a file, at the “Command:” prompt, type:

   ```
   Command: s
   ```

2. The following prompt appears:

   ```
   Command: Save Message
   File message in: =/username
   ```

   - If you press **Enter**, the message is saved in a file named with the sender's *username* in the `Mail` directory in your home directory.

     The equal sign (=) is shorthand for `/`*HomeDirectory*`/Mail`.

   - To save the message in another file, enter the name of the file (the name you enter replaces `=/`*username*). For example:

     ```
     Command: Save Message
     File message in: =/oldnews
     ```

     If the user is Leslie, the current message is saved in the file `oldnews` in the `/home/leslie/Mail` directory. If the `oldnews` file already exists, the message will be appended to the contents of the file. If the `oldnews` file doesn't already exist, it will be created.

3. To save the message in another directory, enter the **path name** of the directory and the name of the file.

   For example:

   ```
   Command: Save Message
   File message in: ~/messages/oldnews
   ```

   If the user is Leslie, the current message is saved in the file `oldnews` in the `/home/leslie/messages` directory. If the file already exists, the message will be appended to the contents of the file. If the `oldnews` file doesn't already exist, it will be created.

CAUTION

Saved messages automatically deleted.

After you save a message in a file, the message is marked with a `D` for deletion. When you exit `elm`, the message is automatically deleted.

Deleting Mail Messages

After you have read your mail messages, you may want to delete them.

1. To delete a mail message, highlight the message and press **D**.

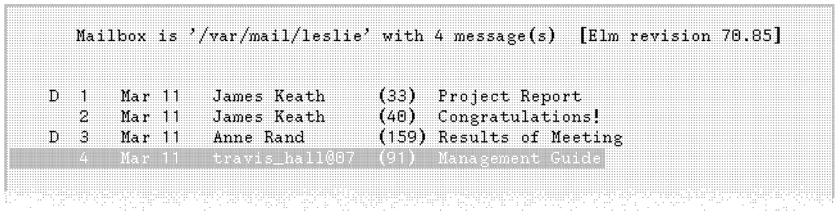
   A D appears to the left of the message to show that it is marked for deletion.

2. To move to the next and previous messages, use **J** and **K**.
3. Delete the marked messages at the "Command:" prompt in one of two ways:

   - To delete the marked messages *and* quit `elm`, press **Q**. `elm` will ask you to confirm this action.
   - To delete the marked messages *without* quitting elm, press **$**.

The following screen shows two messages marked for deletion:

Figure 5-7 Messages are marked with a "D" for deletion.



```
     Mailbox is '/var/mail/leslie' with 4 message(s)  [Elm revision 70.85]


  D  1   Mar 11    James Keath      (33)  Project Report
     2   Mar 11    James Keath      (40)  Congratulations!
  D  3   Mar 11    Anne Rand        (159) Results of Meeting
     4   Mar 11    travis_hall@07   (91)  Management Guide
```

Exiting the elm mailer

When you are in `elm`, exit by typing **Q** at the “Command:” prompt.

If you have mail, `elm` responds as follows:

```
Command: q     Keep mail in incoming mailbox ? (y/n) y
```

If you press **Y** or press **Enter**, any messages in the incoming mailbox will remain there. You will see these messages when you start `elm` again.

If you press **N**, messages are stored in your home directory in an alternate mailbox, called `mbox`.

Mailing a Directory and Contents

Multiple files

Sometimes you may need to mail multiple files to another user. You can use the HP-UX `shar` **utility** to bundle files and directories into a single distribution package. You can then use `elm` to mail the package.

The files can contain any type of data, including executable programs that otherwise cannot be mailed. The resulting package can be edited, for example, to add a message at the beginning.

The optional MPower product allows you to send audio, video, and fax messages through electronic mail.

1. To bundle files, type the following command in a terminal window:

   shar(1)

   ```
   $ shar filename > packagename
   ```

   You can use more than one *filename*, separated by spaces. The output of `shar` appears only on your screen unless you redirect the output with > *packagename*.

   For example, to archive all files with a `.tag` suffix in the current directory in a file called `myarchive`, type:

   ```
   $ shar *.tag > myarchive Enter
   ```

   The asterisk * is a "wildcard" that matches any combination of characters.

2. To unpack a `shar` package:

   - Edit the file and delete any mail headers or messages at the top of the file.
   - Use the `sh` command with the package name as follows:

     ```
     $ sh package
     ```

     When unpacking, the files and directories in the package are written to their original directory path names.

Using Options for Packing Files

You can also use options with the `shar` command. For example, in this command:

```
$ shar -CvZ *.tag > myarchive Enter
```

The `-C` option adds a line saying `- cut here -` before the archive. The `-v` option lists the file names as they are packaged. The `-Z` option compresses the files so that they require less disk space. For more information on `shar` options, see *shar*(1) in the *HP-UX Reference*.

To pack files from many directories and unpack them into one directory, use the `-b` option. The original path names are ignored.

For example, the following command packs files from two directories:

```
$ shar -b /home/ann/list /home/ann/pics/pic.xwd > package Enter
```

When `package` is unpacked, `list` and `pic.xwd` are placed in the current directory. Without `-b`, the original directories would be recreated during unpacking.

Mailing a Package

1. Start `elm` by typing `elm` at the command prompt.
2. Press **M** to mail a message.
3. Respond to the prompts for recipient, subject, and copies.
4. When the editor window opens, read in the package file.

   In `vi`, press **ESC**, then type `:r` *filename*.
5. Type an introductory message, if desired.
6. When you are done typing, save your file.
7. After you exit the editor, you will see the following message on the screen.

   ```
   Please choose one of the following options by the first character:
   E)dit message, edit the H)eaders, S)end it, or F)orget it. s
   ```
8. To change any of the header information for your message (for example, to add recipients to the `Cc` list, change the message subject, or mark the message as Urgent), press **H**.
9. To send your message, type `s`
10. After you have sent the mail message, the `elm` main screen reappears and displays the message "`Mail sent!`".

Customizing elm

Set options

The `elm` mailer has different options you can set to make it more convenient for you to use. Among features you can change are the menus that appear on the screen, the printer your mail is sent to, and the order in which your mail is listed in your mailbox. These are entered automatically in the `.elm/elmrc` file. This file is created by `elm` and contains your default settings and customizations.

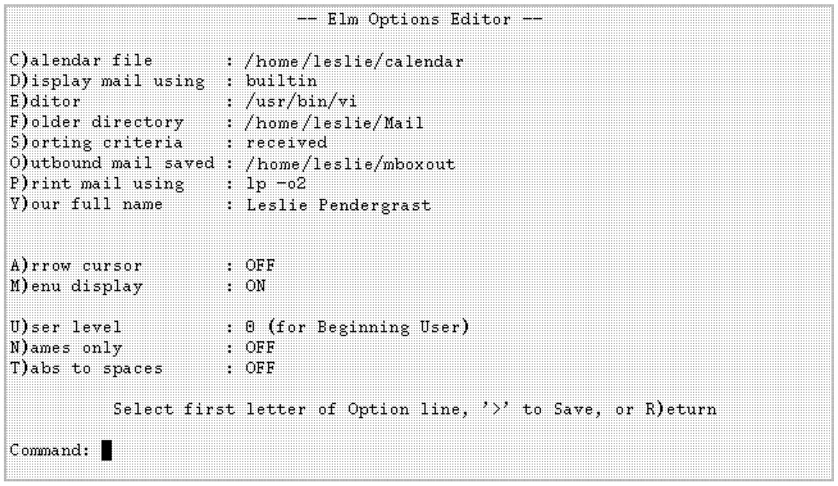
Using the Options Editor

To start the `elm` options editor, press **`o`** at the `elm` command prompt:

```
Command: o
```

You will see a screen similar to the following:

Figure 5-8 **You can configure elm using the options editor.**



```
                                -- Elm Options Editor --

C)alendar file        : /home/leslie/calendar
D)isplay mail using   : builtin
E)ditor               : /usr/bin/vi
F)older directory     : /home/leslie/Mail
S)orting criteria     : received
O)utbound mail saved  : /home/leslie/mboxout
P)rint mail using     : lp -o2
Y)our full name       : Leslie Pendergrast


A)rrow cursor         : OFF
M)enu display         : ON

U)ser level           : 0 (for Beginning User)
N)ames only           : OFF
T)abs to spaces       : OFF

          Select first letter of Option line, '>' to Save, or R)eturn

Command: █
```

Using the screen, you can change the 12 predefined settings in the `.elm/elmrc` file. For information on changing other settings, see *Mail Systems: User's Guide*.

1. To select an option to set, type the first letter of the option. For example, to change the way messages are sorted, press **S**.

   To get information about a specific option in the option menu, type `?` and then type the first letter of the option.

2. Type the new option setting, then press **Enter**.
3. When you are finished setting options, press **>** to save your changes.

   If you do not save the changes, the `.elm/elmrc` file will not be changed and the changes will only apply to the current mail session.

4. Exit the options editor by pressing **Enter**.

Example: Changing the Order of Your Mail Messages

1. To change the order in which your mail messages are listed in your mailbox, press S (the first letter of “S)orting criteria”) in the options editor.

   ```
   Command: s
   ```

   A message indicates how messages are currently sorted. For example:

   ```
   This sort will order most-recently-sent to least-recently-sent
   ```

2. To see different choices for sorting your messages, press the space bar.
3. When you see the method you want, press **Enter**. For instance, press **Enter** when you see:

   ```
   This sort will order by sender name
   ```

4. Press **>** to save the change.
5. To return to your mailbox, press **Enter** again. The messages in your mailbox will now appear in alphabetical order by sender name.

Changing the User Level of elm

`elm` provides three user levels:

- Level 0, for beginning users.
- Level 1, for users familiar with `elm`.
- Level 2, for expert users.

The user level affects:

- Commands displayed in the command menu.

  The frequency of use of each command differs with user level. `elm` considers this difference and tailors the command menu for each user level.

- The simplicity of the message.

  For example, `elm` displays “To:” instead of “Send the message to:” in the higher user levels.

- The handling of a message that was aborted (forgotten).

  The aborted (forgotten) message is an outgoing message that is canceled when the “F)orget it” action is taken. In user level 1 or 2, the canceled message is stored. This allows you to retrieve the message before exiting `elm`.

To change the default user level (Level 0, beginner) to an advanced user level (Level 1 or 2), follow these steps:

1. Press **o** (letter O, not zero) at the main screen prompt, “Command:” The options editor screen appears.
2. Press **u** at the “Command:” prompt.
3. Select an advanced user level (1 or 2) by pressing the space bar, then press **Enter**.
4. Save the setting by pressing **>**.

For more information on using and customizing `elm` and other mail systems in HP-UX, see *Mail Systems: User's Guide*.

Chapter Command Summary

Table 5-1 Elm Commands

To do this `elm` command...	Use...
Delete the messages marked for deletion without quitting.	`$`
Get help on `elm` commands.	`?`
Send a command to the shell without leaving `elm`.	`!`
Set up mail aliases.	`a`
Change the mailbox.	`c`
Mark messages for deletion.	`d`
Forward the current message to another user.	`f`
Send a group reply to all recipients of the original message.	`g`
Move the message pointer to the next message (below).	▼
Move the message pointer to the previous message (above).	▲
Send mail to a specified user or users.	`m`
Set different mail options, including the sorting method for messages, the destination of printed messages, the type of menus displayed, and so on.	`o`
Print messages. (You can change the destination of printed messages using the `o` command listed above.)	`p`
Quit `elm` with the option of changing the contents of the mailbox.	`q`
Reply to the author of the current message.	`r`
Save a message to a file.	`s`
View sub-parts of multi-part messages.	`v`
Exit `elm` without making any changes.	`x`

6 Communicating over a Network

Remote computing

Sometimes you may need to get a data **file** that's on another computer, or to work on a computer that's not at your desk.

The easiest way to accomplish these tasks is to use a network. Local area networks (**LAN**) and wide area networks (**WAN**) allow you to access data and use information distributed among many computers.

If your system is on a network, you can transfer files to and from other computers, log in remotely, and run applications or commands on remote computers.

To do this ...	**See ...**
Using HP-UX network services	page 154
Using Global networks	page 155
Transferring files with ftp	page 156
Copying files with rcp	page 163
Logging in with rlogin	page 169
Running commands with remsh	page 172

HP-UX Network Services

Your HP-UX system can use a variety of networking services to enable you to transfer copies of files to other computer systems. These services can also enable you to log onto remote machines on the network and run commands and processes remotely.

Remote File Systems: NFS

The HP Network File System (NFS) services allow many systems to share the same files. Since NFS is independent of the operating system, it can provide data-sharing among heterogeneous systems. Explicit file transfers across the network are unnecessary. Since access techniques are transparent, remote file access remains similar to local file access.

For more information about setting up NFS-mounted file systems, ask your system administrator, or see *Installing and Administering NFS Services* and the *Managing Workgroups and Clusters*.

Using Global Networks

If your computer has access to a network, you may be able to connect to global and national networks. These networks allow you to communicate with people all over the world by sending electronic mail and reading electronic bulletin boards.

Here are some of the largest academic and research computer networks:

- The Internet
- BITNET
- JANET
- NFSNET
- USENET

Ask your system administrator to determine if you have access to one of these networks.

For information on the Internet, see publications such as *The Internet Yellow Pages* (Osborne), *The Internet Navigator* (Wiley and Sons), or *The Whole Internet* (O'Reilly and Associates). For more general network information, see the *User's Dictionary of Computer Networks*, an annual publication of the University of Texas System Office of Telecommunication Services. This guide provides descriptions of networks, lists of host systems, site contacts, and organizations.

Transferring Files Remotely with ftp

ftp(1)

The `ftp` (File Transfer Protocol) program allows you to do the following tasks:

- Copy files over a network connection between your local system and remote systems.
- Manage files on remote systems for which you have a valid login account.

Some systems are set up to allow anonymous access to “public” files. This capability is referred to as anonymous `ftp`.

Preparing to Use ftp

hosts(4)

- If your system has an `/etc/hosts` file, the system administrator should ensure that it contains entries for each remote systems with which you will communicate. Each entry contains the following information:

  internet_address   official_name   alias

  For example:

  ```
  15.15.232.18     hpabc.fc.hp.com   hpabc
  ```
- Have the system administrator for the remote systems arrange to give you a password and an account, or a login to someone else's account, so that you can log in on the remote systems. (If the remote system allows anonymous `ftp`, you do not need an account on that system.)

nslookup(1)

Instead of a using a local host file, there may be a list of hosts available through the *nslookup*(1) command.

For example:

```
$ nslookup hpabc
```

If nslookup finds the remote system, you should be able to ftp to it.

Starting ftp

1. To invoke `ftp` and connect to a remote system in one step, type the following:

   ```
   $ ftp remote_hostname Enter
   ```

2. `ftp` confirms the connection with the remote system and prompts you for a remote login name:

   ```
   Name (remote_hostname):
   ```

3. To log in with the same remote login name as your local login name, just press **Enter**.

   Otherwise, type your login name for that system and press **Enter**.

   To access an anonymous ftp account, use the login name “anonymous” or “ftp”.

4. `ftp` prompts you for a password:

   ```
   Password (remote_hostname):
   ```

   Type the password associated with your remote login name and press **Enter**. For security reasons, the password will not appear on the screen.

   To access an anonymous ftp account, use any non-null password (by convention, the password should be your e-mail address).

5. `ftp` confirms this action with a message:

   ```
   Password required for remote_login_name
   User remote_login_name logged in.
   ```

6. To see a list of available commands, press **?** at the `ftp>` prompt.

   To get help on a particular command, press **?** and type the command name.

Listing and Creating Directories

While connected to a remote computer with `ftp`, you can view the contents of directories and move between directories.

If the remote computer has been configured correctly, you can also create and remove directories.

To do this...	Type...
Display the name of the current remote working directory	`pwd`
Display the name of the current *local* working directory	`!pwd`
Change the working directory on the remote system to *remote_directory*	`cd` ***remote_directory***
Change the working directory on the local system to *local_directory*	`lcd` ***local_directory***
List the contents of the current remote directory	`ls`
Create a remote directory	`mkdir` ***remote_directory***
Delete an empty remote directory	`rmdir` ***remote_directory***
Delete a remote file	`delete` ***remote_file***

Transferring Files from a Remote System

Use `get` to transfer files from a remote system to your local directory.

1. If you are going to transfer binary files, such as graphics or executable programs, type **`bin`** at the `ftp>` prompt.
2. At the `ftp>` prompt, type:

   `ftp>` **`get`** *remote_file local_file* **Enter**

   The *remote_file* can be the name of a file in the remote **working directory**, or a **relative** or **absolute** path from that directory.

   If you do not specify *local_file*, the local destination file name will be the same as the remote source file name.

   a. `ftp` copies the remote file to the local file name.

   b. If the remote file is not in the current working directory on the remote system, *remote_file* is the absolute path name or **relative path name** for that file. In that case, `ftp` copies the file to a file name with the same path on your local system.

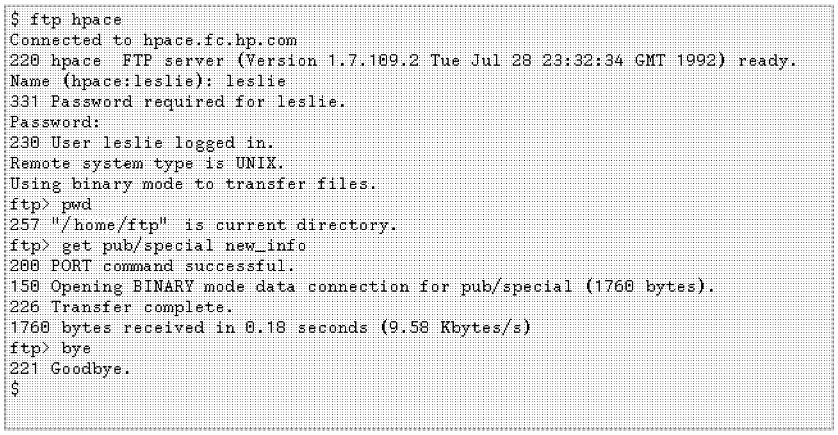
   c. If there is no matching path, `ftp` gives you a message, "`No such file or directory`".

   d. If the destination file already exists, `ftp` overwrites its contents with the contents of the remote file.
3. During a successful copy, `ftp` displays messages confirming the copy and the length of time required.

Example

This example shows user leslie getting the remote file special from the remote directory /home/ftp/pub and placing it on the local system as new_info.

Figure 6-1 **Use ftp to get files from remote systems.**



```
$ ftp hpace
Connected to hpace.fc.hp.com
220 hpace  FTP server (Version 1.7.109.2 Tue Jul 28 23:32:34 GMT 1992) ready.
Name (hpace:leslie): leslie
331 Password required for leslie.
Password:
230 User leslie logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> pwd
257 "/home/ftp"  is current directory.
ftp> get pub/special new_info
200 PORT command successful.
150 Opening BINARY mode data connection for pub/special (1760 bytes).
226 Transfer complete.
1760 bytes received in 0.18 seconds (9.58 Kbytes/s)
ftp> bye
221 Goodbye.
$
```

Transferring Files to a Remote System

Use `put` to transfer files from your local system to a remote system.

1. If you are going to transfer binary files, such as graphics or executable programs, type `bin` at the `ftp>` prompt.
2. At the `ftp>` prompt, type:

   `ftp>` **`put`** *local_file remote_file* **Enter**

   The *local_file* can be the name of a file in the local **working directory**, or a **relative** or **absolute** path from that directory.

   If you do not specify *remote_file*, the remote destination file name will be the same as the local source file name.

   a. `ftp` copies the local file to the remote file name.

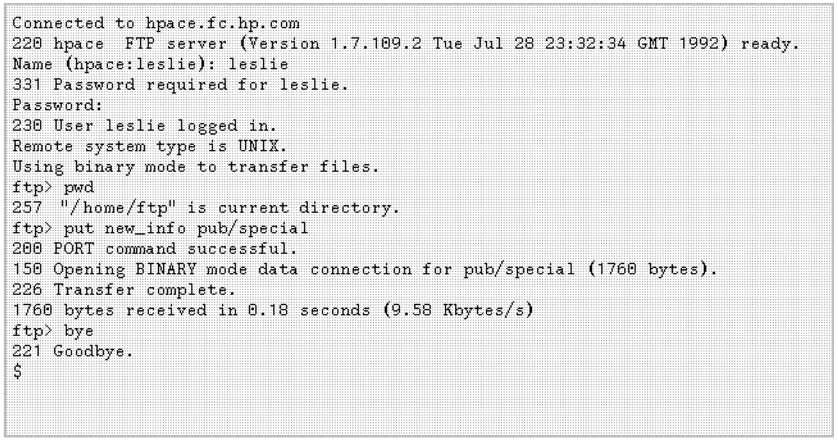
   b. If the remote file is not in the current working directory on the remote system, *remote_file* is the absolute path name or **relative path name** for that file.

   c. If the destination file already exists, `ftp` overwrites its contents with the contents of the local file.
3. During a successful copy, `ftp` displays messages confirming the copy and the length of time required.

Example

This example shows user leslie putting the local file new_info onto the remote system as the file special in the remote directory /home/ftp/pub.

Figure 6-2 **Use `ftp` to put files on remote systems.**



```
Connected to hpace.fc.hp.com
220 hpace  FTP server (Version 1.7.109.2 Tue Jul 28 23:32:34 GMT 1992) ready.
Name (hpace:leslie): leslie
331 Password required for leslie.
Password:
230 User leslie logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> pwd
257  "/home/ftp" is current directory.
ftp> put new_info pub/special
200 PORT command successful.
150 Opening BINARY mode data connection for pub/special (1760 bytes).
226 Transfer complete.
1760 bytes received in 0.18 seconds (9.58 Kbytes/s)
ftp> bye
221 Goodbye.
$
```

Exiting ftp

To close the connection with the remote system and exit `ftp`, type:

`ftp>` **bye** **Enter**

Copying Files Remotely with rcp

You can use the HP-UX `rcp` (Remote Copy) command to copy files or directories to and from a remote system or to copy among remote systems.

Preparing to Use rcp

If your system administrator has already configured your system to use `remsh`, you can use `rcp` without any additional setup.

To use `rcp`, you will need the following:

- **Read permission** on the files you want to copy, and read and search (execute) permission on all directories in the directory path.
- An account (login) on the remote system.

rhosts(4)

A `.rhosts` file in the remote system's **home directory** containing the names of your local system and your local login name.

For example, an entry in the `.rhosts` file on the remote system might be:

```
hpabc leslie
```

where `hpabc` is the name of your local system and `leslie` is your local login name. This allows `leslie` on `hpabc` to copy files back and forth to the remote system containing the `.rhosts` file.

NOTE

It is important to protect your remote `.rhosts` file and home directory to prevent unauthorized users from gaining `rcp` access to your remote account.

- Make sure you own the file.
- Make sure that you (the `owner`) have read and write permission on the `.rhosts` file, and that `group` and `other` have no permissions.
- Protect your remote home directory so that `owner` has read, write, and **execute permission**, `group` has read and execute permission, and `other` only has execute permission.

For information on file ownership and permissions, see "Protecting Your Files and Directories" in Chapter 7, "Making Your System Secure."

- A `.rhosts` file on your local system. This contains the names of all the systems you will copy files to and from.

  For example:

  ```
  hpqrs  leslie
  hpxyz  leslie
  ```

- If your system has an `/etc/hosts` file, the system administrator should ensure that it contains entries for the remote hosts with which you will communicate.

  The `/etc/hosts` file has a line containing the following information about each remote system:

  internet_address   official_name   alias

  For example:

  ```
  15.15.232.18     hpabc.fc.hp.com   hpabc
  ```

  Instead of a using a local host file, there may be a list of hosts available through the *nslookup*(1) command.

  For example:

  ```
  $ nslookup hpabc
  ```

  If nslookup finds the remote system, you should be able to rcp to it.

Copying Files to a Remote System

To copy from your system to a remote system, use the following:

rcp(1)

`rcp` *local_file remote_hostname*:*remote_file* **Enter**

Note that, if *local_file* is not in your current directory, you will need to supply the relative path (from your current directory) or the **absolute path name** (from `/`), in addition to the local file name.

Specify the complete (absolute) path for the *remote_file* on *remote_hostname* only if you want to place it in a directory other than the remote home directory.

Examples

To copy myfile from your current directory to a remote system called hpxyz:

`rcp myfile hpxyz:/home/leslie/otherdir` **Enter**

In this case, `myfile` will be copied as `myfile` into the remote subdirectory, `otherdir`. If you had only supplied the remote host name, `rcp` would have copied `myfile` into the remote home directory, also as `myfile`.

You can also include a file name in the destination. For example, to copy to a system named `hpxyz`:

`rcp myfile hpxyz:/home/leslie/otherfile` **Enter**

In this case, you have copied `myfile` as `otherfile`, in the remote directory `leslie`.

Copying Files from a Remote System

To copy a file from a remote system into your local directory, use the following syntax:

```
rcp remote_hostname:remote_file local_file  Enter
```

Example

To copy myfile from your account in a remote system hpxyz into your current directory:

```
rcp hpxyz:/home/leslie/myfile .  Enter
```

The dot (`.`) is shorthand for “current directory”. In this case, `myfile` will be copied from the remote directory into your current directory as `myfile`.

If you want to copy the file to a new name, supply the destination file name.

If you want to copy `myfile` into another directory in your home system, use a path name, absolute or relative, as shown:

```
rcp hpxyz:/home/leslie/myfile otherdir/  Enter
```

Or, if you want to copy the file to another file name in another directory:

```
rcp hpxyz:/home/leslie/myfile otherdir/otherfile  Enter
```

Copying Directories to a Remote System

To copy a local directory with all its files and subdirectories to a remote system, use `rcp` with the `-r` (recursive) option.

The syntax is as follows:

`rcp -r` *local_dir remote_hostname*:*remote_dir* **Enter**

If *local_dir* is not in your current directory, you will need to supply the **relative path name** (from your current directory) or the **absolute path name** (from `/`, the top of the directory hierarchy), in addition to the local directory name. Also, if *remote_dir* is not in your home directory, the *remote_dir* will require a relative path (from your home directory) or an absolute path (from `/`).

For more information, see “Specifying Files and Directories” in Chapter 2, “Working with Files and Directories.”

Example

To copy an entire subdirectory called work to a directory called products in your home directory on a remote computer called hpabc, type the following:

`rcp -r work hpabc:/home/leslie/products` **Enter**

This command creates a directory named `work`, with all its contents, in `hpabc:/home/leslie/products` (provided that `/home/leslie/products` already exists on `hpabc`).

The example assumes that you are in the local directory containing `work`. Otherwise, you would have to give a relative or absolute path to that directory, such as `/home/leslie/work`.

Copying Directories from a Remote System

To copy a remote directory with all its files and subdirectories to a local directory, use `rcp` with the `-r` (recursive) option in the following syntax:

`rcp -r` *remote_hostname*:*remote_dir local_dir* **Enter**

Example

To copy a remote directory called work to your current directory, type the following:

`rcp -r hpabc:/home/leslie/work .` **Enter**

The dot (`.`) indicates the current directory. The `work` directory will be created in this directory.

Logging In to Another Computer with rlogin

If you have an account on a remote system, you can use `rlogin` to log in to that system by supplying your remote login name and password. You can then work on that system, running commands and applications just as you would on your home system.

Preparing to Use rlogin

Note that if your system has already been configured to use `rcp` or `remsh`, you can use `rlogin` without any additional setup.

- To use `rlogin`, you will need an account (login) on the remote system.
- If your system has an `/etc/hosts` file, the system administrator should ensure that it contains entries for the remote systems with which you will communicate.

  The `/etc/hosts` file has a line containing the following information about each remote system:

  internet_address   official_name   alias

  For example:

  ```
  15.15.232.18     hpabc.fc.hp.com   hpabc
  ```

  Instead of a using a local host file, there may be a list of hosts available through the *nslookup*(1) command.

  For example:

  ```
  $ nslookup hpabc
  ```

  If nslookup finds the remote system, you should be able to access it.

Logging In on a Remote System

1. Type this command:

   rlogin(1)

   ```
   rlogin remote_hostname Enter
   ```

   The *remote_hostname* is the name of an appropriately configured remote system. This system is named in your `/etc/hosts` file.

   To log in as another user on the remote system, use the `-l` *username* option. By default, you log in with the same user name you are logged in with on your local system. This option is useful if you are using someone else's computer and want to log back in to your own system.

   For example, the following command lets user `leslie` log in to remote system `hpabc` from a local system where another user is already logged in.

   ```
   rlogin hpabc -l leslie Enter
   ```

2. Enter your remote password.

   The remote system logs you in with the login message and the remote prompt.

If you make an error in entering your password, the remote system gives you the error message, `Login incorrect`, and prompts you for your login and password again.

Logging Out and Exiting the Remote System

You can log out of the remote system just as you would from your home system, by typing:

```
exit Enter
```

Typing **CTRL-D** also logs you out on most system.

At this point you are logged out of the remote system, disconnected, and returned to HP-UX on your local system, which displays a message and your local prompt:

```
Connection closed.
$
```

Temporarily Returning to Your Local System

To execute a command on your local system while you are in `rlogin`, type the `rlogin` escape character (normally a ~) followed by `!` and the command to execute locally. (The "~" will be invisible until you type the "`!`" after it.) After the command has executed, `rlogin` returns you to the remote system.

Example

To print the current working directory on your local system while logged in on a remote system, use the following command. In this case, the current local directory is /home/leslie.

```
~! pwd Enter
/home/leslie
[Returning to remote]
```

Press **Enter**, or enter a command to redisplay the remote system prompt.

Running a Command Remotely with remsh

The `remsh` command enables you to execute a command on a remote system without logging in to that system.

Preparing to Use remsh

The remote system must be configured as follows:

- You must have an account on the remote system with the same login name as your local login name.
- The name of your local system and your local login name must be in a `.rhosts` file in your home directory on the remote system.

NOTE

A /*HomeDirectory*/`.rhosts` file creates a significant security risk.

It is important to protect your remote `.rhosts` file and home directory to prevent unauthorized users from gaining `rcp` access to your remote account.

- Make sure you own the file.
- Make sure that you (`owner`) have read and write permission on the `.rhosts` file, and that `group` and `other` have no permissions.
- Protect your remote home directory so that `owner` has read, write, and **execute permission**, `group` has read and execute permission, and `other` only has execute permission.

See “Protecting Your Files and Directories” in Chapter 7, “Making Your System Secure.” for information on permissions.

For example, if your local system's name were `hpabc.hp.com` and your local login name were `jim`, you would create on the remote system a /*HomeDirectory*/`.rhosts` file with the following entry:

```
hpabc.hp.com  jim
```

See the *Using Internet Services* manual for more details on using and configuring `remsh`.

Running a Command Remotely

NOTE

Do not use `remsh` to run an interactive command, such as `vi` or `more`. With some interactive commands, `remsh` hangs. To run interactive commands, log into the remote system with `rlogin`.

At your HP-UX prompt, enter:

remsh(1)

`remsh` *remote_hostname* *command* **Enter**

where *remote_hostname* is the name or alias of a remote system and *command* is a command to execute on the remote system.

X Window

You can also set `remsh` to display the windowed command output on your local system.

At your HP-UX prompt, enter:

`remsh` *remote_hostname* *command* **`-display`** *system:display.screen*

where:

remote_hostname	The name or alias of a remote system.
command	The program you want to run on the remote system.
system:display.screen	The system and display the results are to be displayed on. *screen* is optional.

Examples

To copy the file special to the file special.old in your home directory on hpabc, use this command:

`remsh hpabc cp special special.old` **Enter**

`remsh` executes the command on the remote system, and then your local system redisplays its prompt.

To run `xload` on the remote system named `there` and direct output back to your system, `here`, use this command:

`remsh there -n /usr/bin/X11/xload -display here:0 &` **Enter**

The `-n` option closes the standard input and prevents `remsh` from using input not intended for it.

Chapter Command Summary

Table 6-1 Networking Commands

To Do This...	Type This...
Start `ftp` and connect to *remote_hostname*	`ftp` *remote_hostname*
Get help in `ftp`	`?` or `?` *command*
Copy files from *remote_hostname* to current directory, in `ftp`	`get` *remote_file*
Copy files from your local current directory to the current directory on *remote_hostname* in `ftp`	`put` *local_file*
List the contents of the current remote directory	`ls`
Change the current remote directory to *remote_dir*	`cd` *remote_dir*
Change the current local directory to *local_dir*	`lcd` *local_dir*
Exit `ftp`	`bye`
Copy *local_file* to a remote system, using `rcp`, with full path names.	`rcp` *local_file* *remote_hostname*`:`*remote_file*
Copy a file from a remote system to your local directory, using `rcp`, with full path names.	`rcp` *remote_hostname*`:`*remote_file* *local_file*
Copy a directory structure from your local system to a remote system	`rcp -r` *local_dir* *remote_hostname*`:`*remote_dir*
Copy a directory structure from a remote system to your local system	`rcp -r` *remote_hostname*`:`*remote_dir* *local_dir*
Log in on a remote system	`rlogin` *remote_hostname*

To Do This...	**Type This...**
Set the display to a local system	`DISPLAY=`*hostname*`:0; export DISPLAY`
Exit `rlogin`	`exit`
Run a command on a remote system	`remsh` *remote_hostname command*
List the contents of a remote home directory	`remsh` *hostname* `ls`

7 Making Your System Secure

Security Policy

HP-UX provides many security features to protect files from unauthorized access. Although the system administrator is primarily responsible for the security of your system, you need to follow good security practices to maintain security on your system. The degree to which you need to enforce security measures depends on where you work, the security policy in your workplace, and the type of information with which you work.

NOTE

About Trusted Systems.

Your HP-UX system can be configured as a C2-level trusted system, as described in Section 2.2 of the Department of Defense Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, December 1985.

Your system administrator can inform you if your system is a trusted system. When appropriately configured as a trusted system, HP-UX provides additional security features such as discretionary access control and system auditing.

Security Strategies

This chapter summarizes the security strategies you should follow to help keep your system secure:

- Become familiar with the security policies of your workplace.
- Keep your terminal secure.
- Choose a secure password, and protect your password after you have chosen it.
- Be aware of who has permission to access your files and directories, and be able to control such access.

NOTE Security requires ongoing attention.

It may be impossible to have a 100% secure system under all circumstances. This chapter provides some guidelines for securing your system. However, even these cannot guarantee you a completely secure system.

Securing Your Terminal

When you are working with sensitive material, take care to position your terminal so the screen is not visible to others. Never leave your terminal unattended. Log off (`exit`) when you leave your terminal.

Guidelines for Securing Your Terminal

When working with sensitive material, take these security precautions:

- Position your terminal so the screen points away from open windows and doors.
- Never leave your terminal in a non-secure state:
  - Exercise care when logging in. Make sure no unauthorized person is observing you while you're entering your password.
  - Log off if you will be away from your terminal for a time.
  - Clear your display, even if you leave your terminal for a brief period. Type `clear` at the command line prompt. (Note that the `clear` command clears only the current screen; one can still scroll up and see previous screens.)

NOTE Other security requirements.

Check the security policies in your workplace. You may be required to log off whenever you leave your terminal, even if only for a brief period.

Working in an Audited Environment

HP-UX provides the capability to audit computer use, both on an individual and system-wide basis. Depending on how your system is configured, your actions may be recorded by an audit program. This subsystem monitors user actions at your terminal and records security-relevant information.

Choosing a Secure Password

When you choose a password, you want to ensure that no one can guess what you chose. If someone knows your password, that person may log in and access your files. This section offers suggestions on how to select and protect your password. These guidelines are of particular significance if you work with sensitive material.

What is a Secure Password?

When selecting a password in a secure environment, follow these guidelines:

- Choose a password that is not publicly associated with you (your personal or professional life, your hobbies, and the like):
  - Don't use your name, your spouse's name, your children's names, or your pets' names.
  - Don't use the name of your street or your car.
  - Don't use phone numbers or special dates (anniversaries, birthdays, and the like).
  - Don't use your address, social security number, or license plate numbers.
- Choose a password that is not listed in the dictionary (spelled either forwards or backwards). Password-cracking programs can use dictionary lists.

What *can* you use as a password? Here are a few suggestions:

- Make up a nonsense word.
- Make up an acronym.
- Misspell a word intentionally.
- String together syllables from a favorite song or poem.

NOTE Password requirements.

HP-UX requires that your password be six to eight characters long. At least two of these characters must be letters (uppercase or lowercase); at least one character must be either a numeral (the digits `0` through `9`) or a special character (such as `-`, `_`, or `$`).

See “Changing Your Password” in Chapter 1, “Getting Started,” for some examples.

Trusted System Passwords

On a trusted system, the system administrator controls how passwords are generated. The following options are available:

- User-generated passwords: You select your own password but passwords are run through a screening program that checks the password against the dictionary, a list of login names, login name permutations, repeated characters, and palindromes.
- System-generated passwords using letters only: The system assigns passwords using alphabetical characters only.
- System-generated passwords using a combination of letters, numbers, and punctuation characters: The system assigns passwords using alphanumeric and punctuation characters.
- System-generated passwords using pronounceable English phrases: The system assigns passwords that you can pronounce.

NOTE Trusted System passwords can be up to 80 characters long.

Protecting Your Password

When you have chosen your password, follow these guidelines to ensure that no one discovers it:

- Never write down your password.
- Don't tell others your password.
- Don't let others watch as you type your password.
- Don't store your password in the function keys of a terminal.
- Change your password occasionally (for example, once every three or four months).

  See “Changing Your Password” in Chapter 1, “Getting Started,”if you need information on how to change your password.
- If you use more than one computer, use a different password for each system.

You should always report any changes in status and any suspected security violations to your system administrator.

Protecting Your Files and Directories

Three classes of users can access files and directories: owner, group, and other. For each of these classes of users, there are three types of **access permissions**: read, write, and execute.

Who Has Access?

The three classes of users are:

- **Owner** — Usually the person who created the file.
- **Group** — Several users that have been grouped together by the system administrator. For example, the members of a department might belong to the same group.
- **Other** — All other users on the system.

What Kind of Access?

The access permissions on a file or directory specify how it can be accessed by the owner, group, and other user classes.

Table 7-1 A Comparison of Permissions for Directories and Files

Permission	Means This For a Directory	Means This For a File
read (`r`)	Users can view names of files and directories in that directory.	Users can view the contents of the file.
write (`w`)	Users can create, rename, or remove files or directories contained in that directory.	Users can change the contents of the file.
execute (`x`)	Users can view the contents of files within the directory, and run commands, scripts, and programs within that directory.	Users can execute (run) the file (if it is an executable file or script) by typing the filename at the command line prompt.

You should always be aware of the permissions assigned to your files and directories. Check your files and directories periodically to make sure appropriate permissions are assigned. If you find any unfamiliar files in your directories, report them to the system administrator or security officer.

Always carefully consider the permissions you allow on your files and directories. Give others access to them only when you have good reason to do so (if you are working on a group project, for example, your group may need access to certain files or directories).

Displaying Access Permissions

ll(1)

The `ll` (*l*ong *l*isting) command displays the following information:

- Whether the item is a file or directory.
- The access permissions for each of the three classes of users (owner, group, and other).
- Number of links.
- Name of the owner.
- Name of the group.
- Size in bytes.
- Date and time of last modification. If the time of last modification was more than six months ago, the year is substituted for the hour and minute of the modification time.

Displaying File Permissions

To see the permissions, owner name, and group name on `myfile`, for example, type the following:

```
$ ll myfile
```

When you press **Enter**, you should see something like this:

```
-rw-r--r--   1  leslie  users  154   Nov 4 10:18  myfile
    |              |       |      |     |             |
permissions      owner   group  size  date        file name
```

The first dash on the left indicates that `myfile` is a file (if `myfile` were a directory, you would see a `d` in place of the dash).

Here is a closer view with all permissions indicated (note that the permissions are in sets of three):

```
 rwx   rwx   rwx
  |     |     |
owner group other
```

If a permission is not allowed, a dash appears in place of the letter. In the example above (`-rw-r--r--`), owner (`leslie`) has read and write permission (`rw-`); group (`users`) and other have only read permission (`r--`).

Displaying Directory Permissions

To display permissions showing owner, group, and other for a specific directory, use the `ll` command with the `-d` option.

For example, to see the permissions on the `projects` directory below the current directory, type the following:

```
$ ll -d projects
```
Follow the ll command with a -d and the directory name.

When you press **Enter**, you should see something like this:

```
drwxr-x---  1  leslie  users      1032 Nov  28 12:38 projects
```

The first character (`d`) in the long listing above indicates that `projects` is a directory. The next nine positions (three sets of three) indicate the read (`r`), write (`w`), and search (`x`) permissions for owner, group, and other.

If a permission is not allowed, a dash appears in place of the letter. Here is a closer view with all positions indicated:

```
      d        rwx     rwx     rwx
      |         |       |       |
  directory   owner   group   other
```

Then, in the original example above (`drwxr-x---`):

The owner (`leslie`) has read, write, and search permission (`rwx`); group (`users`) has read and search permission (`r-x`); other has no access (`---`) to the `projects` directory.

Guidelines for Access to Sensitive Files

Make sure that permissions assigned to sensitive files and directories are appropriate. Here are some general suggestions:

- Only you should be able to write to your home directory.
- Only you should be able to write to the files used to customize your home environment, for example, `.login` and `.profile` (`.profile` is discussed in Chapter 3, “Using Your Shell,”, in this manual, and in the *Shells: User's Guide*).
- Only you (and the pseudo-group “mail”, assigned to the mailer) should be able to write to your mailfile `/var/mail/`*username*.

For More Information...

For information on access control lists (ACLs), which allow finer control of access to files, see the *acl*(5) man page and *Managing Systems and Workgroups*.

Changing File or Directory Ownership

To change the owner of a file, use the `chown` (“`change owner`”) command. The file's **owner** and the **superuser** are the only ones who can change a file's permissions.

For example, to give user `george` ownership of the `scores` file, use this command:

chown(1)

```
chown george scores
```

The owner can either be a decimal user ID or a **login** name found in the `/etc/passwd` file. You can see the file's current owner by typing `ll` *filename*.

NOTE You can change both owner and group at the same time.

For example, this command changes both the owner of the file and the group to which the file belongs:

```
chown george:team scores
```

Changing Who Has Access to Files

To change who has read, write and execute permission on your files, use the `chmod` (*ch*ange *mod*e) command. In general, give others access to your files only when you have good reason to do so (if you are working in a group project, for example, your group may need access to certain files).

Using chmod to Set File Permissions

You can specify permissions for `chmod` using the letters `u`, `g`, and `o`, as symbolic code for the `o`wner ("user"), `g`roup, and `o`thers (the *class*). This "symbolic mode" is easy to remember, since the symbols `r`, `w`, and `x` (the *mode*) are used directly as arguments in the command.

The `chmod` syntax uses the `+`, `-`, and = signs. The syntax is:

`chmod` *class*[±=]*mode*, [...] *filename*

For example, you can use the symbolic mode to create `rw-r--r--` permissions by specifying the symbols `rw`, `r`, and `r` directly in the `chmod` command. "User" is represented by `u`, "group" by `g`, and "other" by `o`. To assign the permissions absolutely, use the = sign in the argument. Unspaced commas separate class permissions:

```
chmod u=rw,g=r,o=r myfile
```

When permissions are being set the same, you can also combine the arguments as:

```
chmod ugo=r myfile
```

With only read permission on `myfile`, no one can write to it. Also, if you now try to remove `myfile`, the `rm` command asks you whether you really want to remove the file:

```
rm myfile
myfile: 444 mode? (y/n) nIf you do not want to remove it, enter n.
                            If you do want to remove it, enter y.
```

To create `rw-------` permissions and set "no permissions" for the classes `g` and `o`, use = with no symbol following:

`chmod u=rw,g=,o=` *filename*

Permissions are added with the + sign. Again, separate each class-permission with a comma and no space:

`chmod u+rw,g+r,o+r` *filename*

You can also subtract permissions from `u`, `g`, or `o`, using `-`, to restrict the level of permission from a previous “higher” level. For example, if you had set `rwxrw-rw-` and you wanted to change this to `rwx------`:

```
$ chmod g-rw,o-rw filename
```

However, unless you began with no permissions you may find that using `+` or `-` has added to, or subtracted from, some previously existing permissions for that file. Run the `ll` command to check this. If in doubt, set the permissions absolutely by using `=`.

Later, if you want to permit yourself and members of your group to read from and write to `myfile`, use `chmod` as follows:

```
$ chmod ug=rw,o=r myfile
```

The `ll` command now should show:

```
-rw-rw-r--  1  leslie  users      154 Nov  4 10:18 myfile
```

Here is a summary of the various `chmod` commands you can use to protect `myfile`.

To Set Permissions so that...	Type This...
Only you can read from `myfile`, and no one (including you) can write to it. Set permissions to `-r--------`.	`chmod u=r,g=,o= myfile`
Everyone can read from `myfile`, but no one can write to it. Set permissions to `-r--r--r`.	`chmod ugo=r myfile`
Only you can write to `myfile`, but everyone can read it. Set permissions to `-rw-r--r--`.	`chmod u=rw,go=r myfile`
Only you and members of your group can write to `myfile`, but everyone can read it. Set permissions to `-rw-rw-r--`	`chmod ug=rw,o=r myfile`
Everyone can read from or write to `myfile`. Set permissions to `-rw-rw-rw-`.	`chmod ugo=rw myfile`
Only you can read from or write to `myfile`, but no one else can. Set permissions to `-rw-------`	`chmod u=rw,go= myfile`

Changing Who Has Access to Directories

In addition to changing permissions on files, the `chmod` command can also change permissions on directories. For example, you can protect a directory so that no one can alter its files.

The following examples assume that the directory `projects` exists under your current working directory.

To Set Permissions to…	Type This…
Allow other users to list and access the files in `projects`, but not to create or remove files from it. Set permissions to `drwxr-xr-x`.	**`chmod u=rwx,go=rx projects`**
Allow all users to list, create, remove, and access files in `projects`. Set permissions to `drwxrwxrwx`.	**`chmod ugo=rwx projects`**
Allow only yourself to list, create, remove, and access files in `projects`. Set permissions to `drwx------`.	**`chmod u=rwx,go=- projects`**

When determining who should be allowed to use your directories, be aware that *anyone who can write to a directory also can remove or rename a file in that directory* — even if that person cannot write to the file.

For More Information…

This section covered some of the most common uses of the `chmod` command for protecting files and directories. To learn more about `chmod`, see the *chmod*(1) man page.

Controlling Default Access Permissions

In the previous two sections, you learned how to change the permissions on individual files and directories using the `chmod` command. You should also be aware of the default permissions assigned to all of your files and directories at the time you create them. You can list or change the default permission settings by using the `umask` command.

umask(1)

Default file permissions are assigned by the system whenever you create a new file or directory, and these are governed by your `umask` setting. The default `umask` setting is 0, which means that new files are created with read/write permission for everyone (`-rw-rw-rw-`) and new directories are created with read/write/search permission for everyone (`drwxrwxrwx`).

To restrict these default permissions for newly-created files and directories, use the `umask` command.

When a new file is created, each bit in the file mode creation mask that is set causes the corresponding permission bit in the file mode to be cleared (disabled); hence the term mask. Conversely, bits that are cleared in the mask allow the corresponding file mode bits to be enabled in newly created files.

The `umask` command built into the POSIX and Key shells accepts symbolic mask values (as well as the obsolescent numeric form). These symbolic mask values are similar to those used with the `chmod` command (see *chmod(1)*).

`umask` syntax is as follows:

`umask` *who operator permissions*

where the parameters have the following meaning:

who	One, two, or all of the following characters: • `u` (for user permissions) • `g` (for group permissions) • `o` (for other permissions) • `a` (short form for `ugo`) If the *who* character is omitted, *operator* and *permissions* apply to all categories (same as `a` or `ugo`).
operator	One of the characters `+`, `-`, or `=`. • `+` means clear the file mode bits represented by the accompanying *who* and *permissions* values in the mask, thus enabling corresponding permissions in newly created files. • `-` means set the file mode bits represented by the specified *who* and *permissions* values in the mask, thus denying corresponding permissions in newly created files. • `=` means clear the file mode bits specified by the corresponding *who* and *permissions* values and set all others.
permissions	One of the characters or character combinations `r`, `w`, `x`, `rx`, `wx`, `rw`, or `rwx`, specifying read, write, and/or execute (search) permission for the corresponding *who* and *operator*. If *permissions* is not specified, no change is made to the existing file creation mode mask for the corresponding *who*.

For example, to set the `umask` value to produce read, write, and execute permissions for the file's owner and read-only permission for all others (`-rwxr--r--`) on newly created files, you would enter this:

```
$ umask u=rwx,g=r,o=r
```

To set the `umask` value to produce read, and write permissions for the file's owner, read-only for users in the same group, and no access for others (`-rw-r-------`), enter the following:

```
umask a-rwx,u+rw,g+r
```

To determine your current `umask` setting, type:

```
$ umask -S
```

NOTE

Don’t lock yourself out.

You should not set a `umask` value that restricts your access permissions to your own files. A number of HP-UX utilities, such as `vi`, assume that you can always access newly-created files. Such files might include the temporary files that `vi` creates. These utilities may malfunction when used under such a restrictive `umask` setting.

If you set `umask` at a shell prompt, it will apply to shells and subshells in the current login session only. It won't apply to future login sessions. To apply a `umask` setting automatically at login, add the `umask` command to your `.profile` (POSIX and Bourne Shell users) or `.login` file (C Shell users).

For More Information...

For more information about the `umask` command, see the *umask*(1) man page.

To learn more about the `.profile` and `.login` files, see the *Shells: User's Guide*.

Privileged Groups

A “privilege” is the ability to ignore access restrictions and change restrictions imposed by security policy and implemented in an access control mechanism. On HP-UX, only superusers and members of certain groups are the only privileged users.

The system administrator can associate a group with a system capability so that members of certain groups can be granted special privileges. These groups are called “privileged groups.”

setprivgrp(1)

All users by default are members of the CHOWN privilege group. People with this privilege can change the ownership of files you own. Your system administrator may limit access to the chown(1) command by setting up privileged groups using *setprivgrp*(1M). In that case, only those in the privileged group or groups can change file ownership using *chown*(1). Refer to the *chown*(1) man page for more information.

getprivgrp(1)

Your system administrator can tell you what type of privileges you have been granted. You can also execute the *getprivgrp*(1) command to determine the special attributes for groups to which you belong:

```
$ getprivgrp [groupname]
```

where the optional groupname is the name of the group for which you want to determine your special attributes. If omitted, the command lists the access privileges for all privileged groups to which you belong. Refer to the *getprivgrp*(1) man page for more information.

Trusted System Access Control

If you are running a C2-level trusted system, there are additional forms of access control you can apply to files and directories.

Using discretionary access control (DAC), owners of objects containing data can allow or deny access to these objects at their own discretion, on a need-to-know basis. Objects are things such as files, devices, or interprocess communications mechanisms that another user's process or your process is attempting to access. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission to any other subject.

On a standard HP-UX system, you can protect objects, such as files, by establishing read, write, and access permissions to these objects. If you are the owner, you can set permissions on objects so that your access is different from other group members; group members can have different access to an object than the rest of the user community. The owner can change these protection attributes so they are more restrictive (controlled access) or more permissive (open access).

You can use the chown and chmod commands to control file and directory access. For more information, refer to the *chown*(1) and *chmod*(1) online man pages. To learn how to change your current group selection, refer also to the *newgrp*(1) man page. The newgrp command changes your group ID without changing your user ID and replaces your current shell with a new one.

On a C2-level trusted system, you can have additional discretionary controls on an object that let you include or exclude access even to specific users. You control who (that is, users or groups of users) has access to files and directories by assigning optional access control lists (ACL) permissions.

Realize that the system administrator can use the *su*(1) command to become another user without logging out. The system administrator or superuser can access all files and perform any task on the system. So file access is not restricted for system administrators. Refer to the *su*(1) man page for more information.

Access Control Lists

Access control lists are key to enforcing discretionary access control on trusted systems. ACLs offer a greater degree of selectivity than HP-UX permission bits.

An access control list is a set of user, group, and mode entries associated with a file that specify permissions for all possible user ID or group ID combinations.

acl(5)

Refer to the *acl*(5) online man page for detailed information about access control lists and definitions of related security terminology.

Listing ACLs

To see who can access certain files and directories and what permissions are allowed, you can use the lsacl command:

```
$ lsacl filename
```

The system responds with a listing in this general form:

```
(user.group,mode) ... filename
```

where (user.group,mode) is an ACL entry and filename is the name of the file or directory for which you want a listing.

For example:

```
$ lsacl filex
(sarah.adm,rw-) (alex.%,r--) (%.mtg,r--) (%.%,---) xfile
```

where:

(sarah.adm,rw-) means user sarah in group adm has read and write permissions (rw-) on filex.

(alex.%,r--) means user alex in any group (%) has read permission (r--) on filex.

(%.mtg,r--) means any user (%) in the group mtg has read permission r--) on filex.

(%.%,---) means no other user from any other group has read, write or execute permission on filex.

Changing ACLs

By adding entries to access control lists, you can allow or deny access to particular users or groups of users. You can set or change ACLs using the *chacl* command.

The general form of the *chacl* command is as follows:

```
$ chacl 'user.group operator mode' filename
```

where:

user is the login name; a % here means all users.
group is the user's group; a % here means all groups.
operator is +, -, or = to add, deny, or specify permissions to existing ACL entries.
mode indicates the permissions allowed (that is, r for read, w for write, and x for execute/search.
filename is the name of the file or directory for which you want to specify access.

For example:

```
$ chacl 'cyc.%=rw' myfile
```

Creates a new ACL entry allowing the user cyc in any group (%) read and write (=rw) access to myfile.

```
$ chacl '%.%+r' status
```

Modifies an ACL entry to allow all users(%) in all groups (%) read access (+r) to the file status.

Refer to the *chacl*(1) man page for more information and examples of setting ACLs.

Additional Notes about ACLs

If a file contains ACLs and you use the chmod command to change file mode access permissions without using the -A option, you will delete any optional entries in the file's ACL.

Trying to delete a base ACL entry will result in no access to a file.

Only the *fbackup* (1M) and *frecover*(1M) file archive utilities handle ACLs correctly. Archive programs such as *ar*(1), *cpio*(1), *ftio*(1), *tar*(1), and *dump*(1M) are unable to handle ACLs on files with optional ACL entries.

Backing Up and Recovering Files with ACLs

On a trusted system, you should only use *fbackup*(1M) and *frecover*(1M) to back up and recover files selectively. These commands retain ACLs that have been applied to the files. Your system administrator can help with the authorized copying of files to tape or disk. It is likely you will have to get permission to use the tape drive or other media and you will need to get the name of the device.

Realize that you need to keep copied files in a safe location. Label tapes and disks. Make sure the files are copied with the correct access permissions if you load them onto another system.

Refer to the *fbackup*(1M) and *frecover*(1M) online man pages for more information.

The command *tar*(1) needs to be used properly to ensure that the DAC permissions are preserved when coping files to and from tape. Be sure to use the -p when taring files from a tape to your system in order to preserve DAC permissions. See the *tar*(1) man page for more information.

Removable Media Security

It is your responsibility to protect the physical security of removable media such as floppy disks and tapes. Do not leave them lying around. It is best to keep them locked up since it is easy for other people to read what you have on these media.

Obtaining Software Security Patches

The U.S. Computer Security Act of 1987 stipulates that if financial loss occurs due to computer fraud or abuse, the company, not the perpetrator, is liable for damages.

To protect system and data integrity, HP recommends that you establish a comprehensive security policy to govern computer use. HP provides up-to-date software patches to close known security problems that allow unauthorized root access to your system.

For information on available HP-UX security patches, contact HP support at:

`support@support.mayfield.hp.com`

or access the World-Wide Web (WWW) using a web browser with this URL:

`http://us-support.external.hp.com`

or, for Europe,

`http://europe-support.external.hp.com`

In addition to HP support options, you can explore on you own at the HP web sites:

`http://www.hp.com`

and

`http://www.software.hp.com`

Chapter Command Summary

To Do This…	Type This…
Display access permissions of files and directories.	`ll`
Add or subtract access permissions.	`chmod` *class±permissions name*
Change access permissions absolutely.	`chmod` *class=permissions name*
View current mask setting.	`umask -S`
Change permissions mask setting.	`umask` *who operator permissions*

A HP-UX Quick Reference

How to use this Reference:

The following table summarizes the most useful HP-UX commands. To make it easier to refer to selected commands, copy or tear out these pages and place them near your display.

1. Type the commands as they are shown in the second column below.
2. Include paths with file names, if you are working with different directories.
3. Follow each command with **Enter**.
4. To get more information on a specific command, type:
   `man` ***command_name*** .

To Do This...	Type This...
Working with Directories	
Show current working directory	`pwd`
Change directory	`cd` ***directory_path***
Change to home directory	`cd`
Create a directory	`mkdir` ***directory_name***
Remove an (empty) directory	`rmdir` ***directory_name***
Working with Files	
Read mail	`elm`
List files and directories in current directory	`ls`
List all files or directories, including invisible (“dot”) files	`ls -a`
List files, and mark directory names with “/”	`lsf`
Compress a file	`compress` ***filename***

To Do This...	Type This...
Uncompress a file	`uncompress` ***filename***
Create or edit a file	`vi` ***file_name***
Display file contents	`more` ***file_name*** (`q` to quit)
Display the first 10 lines of a file	`head` ***file_name***
Display the last 10 lines of a file	`tail` ***file_name***
Copy a file	`cp` ***file_name file_copy***
Move a file to a new file name	`mv` ***old_file new_file***
Append ***file1*** onto the end of ***file2***	`cat` ***file1*** `>>` ***file2***
Remove ***file***	`rm` ***file***
Remove a directory ***dir_name*** and all its files	`rm -rf` ***dir_name***
Check the spelling in a file	`spell` ***file_name***
Printing	
Print a file	`lp` ***file_name***
Determine printer status	`lpstat -t`
Cancel a print request	`cancel` ***request_id***
Finding and Organizing	
Find file(s) beginning with ***x*** in current and subdirectories	`find . -name '`***x***`*'`
Find all occurrences of ***word*** in all files in current directory	`grep` ***word*** `*`
Sort ***listfile*** alphabetically	`sort` ***listfile***
Display date and time	`date`
List all command aliases	`alias`
Find HP-UX command information	`man` ***command_name***
Determine PATH setting	`echo $PATH`

To Do This...	**Type This...**
Determine what shell you're in	`echo $SHELL`
Security Operations	
Create or change password	`passwd`
Display permissions for a file	`ll` ***file_name***
Display permissions for a directory	`ll -d` ***directory_name***
Change file or directory permissions	`chmod` ***class=permissions name***
Change file or directory ownership	`chown` ***user name***
System Operations	
Clear screen	`clear`
Set command-line editor	`set -o` ***editor_name***
Edit your command line (in Korn/Key/Posix Shell set for `vi`)	ESC (use `vi` commands)
Recall previous command line (with `vi` editor)	ESC `k` (back) or `j` (forward)
Execute previous command line	Enter (when line is displayed)
Set terminal type (select ***term_type*** from `/usr/lib/terminfo`)	`TERM=`***term_type***
List current process status and *PIDs*	`ps -ef`
Kill (terminate) a process	`kill` ***PID***
Create or change a password	`passwd`
Redirect input from a file to a command	***command*** `<` ***infile***
Connect two processes with a “pipe”	***command1*** `|` ***command2***

HP-UX Quick Reference

How to use this Reference:

B Doing Advanced HP-UX Tasks

For advanced users

Occasionally you may need to perform more advanced configuration or system administration tasks. These tasks frequently require superuser permission and a more advanced knowledge of HP-UX. To run SAM, ensure you have superuser permission, then enter `/usr/sbin/sam` on the command line.

The following tables list advanced tasks for HP-UX, and provide references to additional information.

Table B-1 Advanced HP-UX Tasks

Task	Where to Look
System Administration Tasks	
Using SAM, the System Administration Manager.	See the *Managing Systems and Workgroups* manual. You can also run SAM and look at the extensive online help available in SAM.
Getting information on system performance.	Run SAM, select and open “Process Management”, then select and open “Performance Monitors”, then select and open one of the available performance tools.
Displaying disk usage.	See the *du*(1) man page.
Recovering disk space.	Run SAM, select and open “Routine Tasks”, then select and open “Selective File Removal”.
Rebooting a system.	See the *Managing Systems and Workgroups* manual.
Changing the system run level.	See the *Managing Systems and Workgroups* manual.
Command Usage Tasks	
Running a command at a specified time	See the *crontab*(1) man page. You can also run SAM, select and open “Process Management”, then select and open “Scheduled Cron Jobs”, then select “Add” from the “Actions” menu.

Task	**Where to Look**
Network Tasks	
Using a remote file system (NFS)	See the manuals *Managing Systems and Workgroups* and *Installing and Administering NFS Services*.
Installing or updating from a network server.	See the *Installing HP-UX 11.0* manual.
File Tasks	
Backing up files or directories.	See the *Managing Systems and Workgroups* manual. You can also run SAM, select and open “Backup and Recovery”, then select and open either “Automated Backups” or “Interactive Backup and Recovery” and enter the appropriate information (see SAM online help for more specific information).
Restoring files or directories.	See the *Managing Systems and Workgroups* manual. You can also run SAM, select and open “Backup and Recovery”, then select and open “Interactive Backup and Recovery”, then choose “Recover Files and Directories” from the “Actions” menu.
Printer and Peripheral Tasks	
Getting information on printers.	See the *lpstat*(1) man page. You can also run SAM, select and open “Printers and Plotters”, then select and open “Printers and Plotters”.
Enabling or disabling a printer.	Run SAM, select and open “Printers and Plotters”, then select and open “Printers and Plotters” again, highlight one of the printers listed, then choose the appropriate action from the “Actions” menu.
Adding or removing printers	See the *Managing Systems and Workgroups* manual. You can also run SAM, select and open “Printers and Plotters”, then select and open “Printers and Plotters” again, highlight one of the printers listed, then choose the appropriate action from the “Actions” menu.

Task	**Where to Look**
Diagnosing a printing error	See the *Managing Systems and Workgroups* manual. You can also run SAM, select and open “Printers and Plotters”, then select and open “Printers and Plotters” again, highlight one of the printers listed, then choose “Show Common Problems” from the “Actions” menu.
Selecting system default printer.	See the *Managing Systems and Workgroups* manual. You can also run SAM, select and open “Printers and Plotters”, then select and open “Printers and Plotters” again, highlight one of the printers listed, then choose “Set as System Default Destination” from the “Actions” menu.
Adding or removing peripherals.	See the manuals *Managing Systems and Workgroups*, and *Configuring HP-UX for Peripherals*. You can also run SAM, select and open “Peripheral Devices”, then select and open the appropriate area for the type of peripheral you want to add.
Configuration Tasks	
Modifying login scripts (`.profile`, `.kshrc`, etc.).	See the *Shells: User's Guide*.
Setting and referencing environment variables.	See Chapter 3, “Using Your Shell,” in this manual and also the *Shells: User's Guide*.
Modifying or viewing user information.	Run SAM, select and open “Accounts for Users and Groups”, then select and open “Users” and use the “Actions” menu to select what you want to do to a user account. Also see the *Managing Systems and Workgroups* manual.
Modifying a user's group ID.	Run SAM, select and open “Accounts for Users and Groups”, then select and open “Users”, highlight the user, then select “Modify Group Membership” from the “Actions” menu. Also see the *Managing Systems and Workgroups* manual.
Removing a user.	Run SAM, select and open “Accounts for Users and Groups”, then select and open “Users”, highlight the user, then select “Remove” from the “Actions” menu.

C Scheduling Commands

Running Commands at Preset Times

Scheduling commands are useful if you want to run resource intensive commands when demands on the system are low or to routinely run commands at certain times. For example, you can schedule a long file to print at midnight or erase temporary files in your home directory everyday.

at(1) The `at` command runs commands in your home directory at the time you specify.

crontab(1) The `crontab` command runs commands in your home directory at regularly specified intervals.

Prerequisites for Using at and crontab

Before you can use crontab or at, your system administrator must set up certain files that allow you permission to run these commands.

Two files, called `at.allow` and `at.deny` in `/usr/lib/cron` determine whether you can use the at command. You can use it if your name is in `at.allow`.

If `at.allow` does not exist, the system checks to see if your name is in `at.deny`. If it is, you are denied access to the at command.

If neither `at.allow` nor `at.deny` exists, only those with superuser privilege can use `at`. If only `at.deny` exists and it is empty, all users can use `at`.

Permission to use `crontab` is determined in the same way except that the files are called `cron.allow` and `cron.deny`.

Refer to the *at*(1) and *crontab*(1) man pages for more information.

Running Commands Using at

Suppose you want to print a large file at night when system usage is low. The following at command sequence prints the file "bigfile" at 4:00 AM.

`at 4am` (Type the at command)

`lp bigfile` (Enter the command you want to schedule for later.)

Ctrl-D (End the command by pressing Ctrl-D.)

You can also specify a date. For example, to print a message on April 10, at 3:30 AM, use the following commands:

```
at 3:30am Apr 10
echo "time to go home" > /dev/console
```

Ctrl-D

To list jobs scheduled with at, enter:

```
at -l
```

You will see output such as:

```
job 8745156.a at wed Sep 17 11:00:00 1997
```

Submitting Batch Jobs

batch(1)

You can also use the batch command to submit a batch file. For example:

```
$ batch
nroff filename > outfile
```

Ctrl-D

This command executes an nroff command when the system is able to handle it. Refer to the *at* (1) man page for more details.

Running Commands Using crontab

You can use the `crontab` command to run commands at regular intervals. For example, you can send a weekly e-mail reminder for a meeting to your mailbox, or erase all “tmp” files everyday.

The `crontab` command creates a file called by your username in the directory `/var/spool/cron/crontabs`. The commands in the file are executed at the specified intervals in your home directory.

A crontab file contains line with six fields each separated by spaces or tabs. The first five fields specify the time the command will be run

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

The sixth field is a string that is executed at the appropriate time.

To create a crontab command file, enter:

```
$ crontab
```

Then type the commands you want to schedule and press **Ctrl-D**.

```
30  8   * * 4  echo “Staff meeting today at 10:00 AM”
 0  0   * * *   rm *.tmp 2 > errfile
```

Ctrl-D

The crontab file is interpreted as follows:

On Thursday at 8:30 AM, crontab sends you a reminder of your 10:00 AM staff meeting. The first field (30) indicates 30 minutes after the hour. The second field indicates the hour (8). The asterisks mean all legal values. The 4 means Thursday.

At midnight everyday, crontab erases files with a *.tmp extension in your directory. Error messages are redirected to a file called `errmsg` in your home directory.

List crontab

To list your current crontab entries, use the `-l` option.

```
$ crontab -l
```

Refer to the *crontab*(1) man page for more information.

D The Key Shell

Using the Key Shell

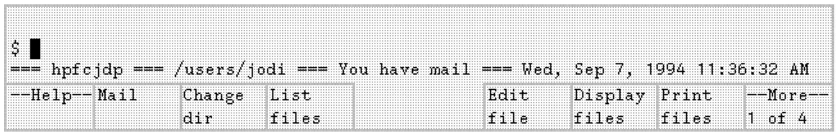
keysh(1)

An optional “friendlier” shell is available for users who want an alternative to using a regular shell. The Key Shell uses softkey menus and context-sensitive help to assist with command options and syntax. The Key Shell automatically translates softkey commands into HP-UX commands when the **Enter** key is pressed.

Using the Key Shell Displays

The Key Shell gives you softkey displays at the bottom of your screen that provide a “menu” of basic shell commands, along with their options in sequence. To start Key Shell, enter the command **`/usr/bin/keysh`**. (Exit this shell by entering: **`exit`**.) You will first see a status line like the following

Figure D-1 **Key Shell Softkey Display**



```
$ █
=== hpfcjdp === /users/jodi === You have mail === Wed, Sep 7, 1994 11:36:32 AM
--Help-- Mail     Change   List               Edit     Display  Print    --More--
                  dir      files              file     files    files    1 of 4
```

You can enter commands from the Key Shell softkey menu or you can enter standard HP-UX commands as usual. If you enter standard HP-UX commands, the Key Shell will often display an appropriate left-to-right set of menu options in the softkey label area at the bottom of your screen. Each label corresponds to a softkey, f1 through f8. The softkeys are separated into groups of four. You may select any or none of the options successively by pressing the corresponding softkey.

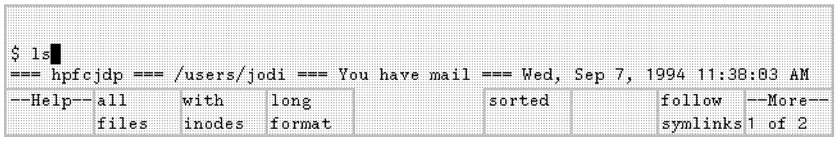
When you want to see more commands, or more options to go with a command you have already chosen, press the More softkey, f8. This will cause the Key Shell to display the next “bank” of softkeys in sequence, eventually cycling back to the first, if you press f8 repeatedly.

After you make a selection by pressing a softkey, your choice will appear on the command line in “English,” just as it appeared in the softkey display, with the correct order and spacing.

Example: Entering a Command with the Key Shell

For example, enter the **ls** command. You will see the following:

Figure D-2 Options Displayed

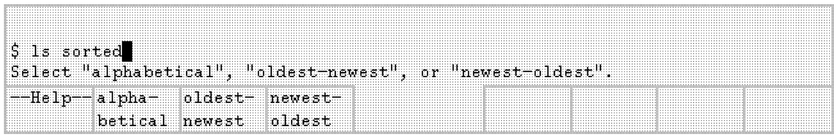


```
$ ls
=== hpfcjdp === /users/jodi === You have mail === Wed, Sep 7, 1994 11:38:03 AM
--Help-- all      with     long              sorted            follow   --More--
         files    inodes   format                              symlinks 1 of 2
```

Many softkey commands require that the user enter a parameter or select an additional softkey before pressing **Enter**. A “prompt line” underneath the command line will indicate whether you need to enter anything else.

If you select `ls`, and then select the “sorted” option, the Key Shell will ask you to specify *how* you want your file listing sorted:

Figure D-3 Required Options Requested

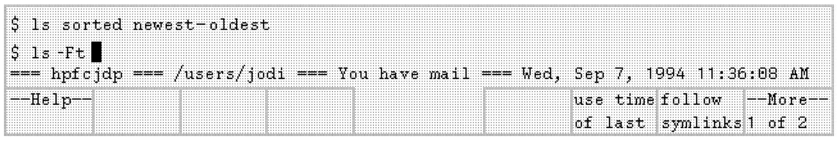


```
$ ls sorted
Select "alphabetical", "oldest-newest", or "newest-oldest".
--Help-- alpha-   oldest-  newest-
         betical  newest   oldest
```

At any time, you can use the Help softkey, f1, to find out more about what functions are available to you.

Suppose you have selected “newest-oldest” for the `sort` option above. You can now enter the finished command line by pressing **Enter**. Or, if you want to preview the HP-UX commands to which these “English” words correspond, you can optionally press **Insert line** and the HP-UX commands will be displayed as shown in the figure below.

Figure D-4 Optional HP-UX Commands Display



```
$ ls sorted newest-oldest
$ ls -Ft
=== hpfcjdp === /users/jodi === You have mail === Wed, Sep 7, 1994 11:36:08 AM
--Help--                                           use time follow   --More--
                                                   of last  symlinks 1 of 2
```

Customizing Your Key Shell Softkeys

You can change the Key Shell's configuration (for example, status line or options) using the Keysh_config softkey in the "More 4 of 4" display. Any changes you make will be automatically saved in the `.keyshrc` file in your home directory. This file will then be replayed upon subsequent invocations of `keysh`.

If they are not already on, some global options that you can change using Keysh_config are:

To Enable	Enter These Softkeys and Press **Enter**
Help softkey	Keysh_config options help on
Automatic prompt messages	Keysh_config options prompts on
Visible HP-UX command translations	Keysh_config options translations on

To turn off any of these options, enter off at the end of the entry sequence instead of on.

Status line indicators you can change, using the Keysh_config softkey, are:

To Enable	**Enter These Softkeys and Press Enter**
Host name	Keysh_config status_line host_name on
User name	Keysh_config status_line user_name on
Current directory	Keysh_config status_line current_dir on
Mail status	Keysh_config status_line mail_status on
Date	Keysh_config status_line date on
Time	Keysh_config status_line time on

Summary of Key Shell Procedures

The general rules for using the Key Shell are:

- Select any desired softkeys from left to right.
- Use the More softkey to see more options.
- Optionally, use the **Insert line** key to preview the translated command-line.
- Use the Help softkey to find out more functions.

If you make an error, use **Backspace** or **CTRL-H** to erase the line back to where you want to re-enter command text.

You can also use the arrow keys, **Clear line**, **Delete line**, **Insert char**, and **Delete char** to manipulate your command line, in addition to using the editor that is set for your POSIX Shell. Note that **Clear line** functions to delete the line only from the cursor position to the end of the line. **Delete line**, however, deletes the entire command line and cancels the command.

For more information, see the *keysh*(1) man page, and the *Shells: User's Guide*.

Glossary

absolute path name The full path name of a file, including all the directories leading to it, starting with root (/) and ending with the file name itself. For example, `/home/michael/myfile` is an absolute path name.

See also **file**, **file name**, **path name**, **relative path name**.

access permissions File characteristics (including read, write, and **execute permission**) that determine if a process can perform a requested operation on the file (such as opening a file for writing).

Hence, access permissions control who can read or alter files or directories. They define read, write, and execute permissions for the file's owner, members of the file's group, and all others.

alias An alternative name for a person or a list of people, used as a shortcut when sending electronic mail.

For example, if you often send mail to someone whose mail address is `christine@market.elm.com`, you could set up the alias `chris`. Then you could send mail just to `chris` instead of typing the entire address.

application See **software application**.

argument The part of a command line that identifies what element (file, directory, etc.) is to be acted upon.

background process A program, usually low priority, run non-interactively by the shell without terminal I/O, while other processing occupies the terminal. Place an ampersand (&) at the end of a command line to cause that command to be run as a background process.

backup A copy of all or part of the file system.

boot To start up your system, loading it into the computer memory.

Bourne Shell A command interpreter. As of the HP-UX 10.0 release, the OSF **POSIX** shell replaces the Korn Shell and Bourne Shell. Thus, /usr/bin/sh will be the POSIX shell. However, /usr/old/bin/sh will still contain the Bourne shell.

Glossary

CD-ROM Compact Disc Read-Only Memory.

CD ROM file system A read-only memory file system on compact disk. You can read data from a CD ROM file system, but you cannot write to one.

cluster A group of workstations connected via a **LAN**. One computer, the cluster server, performs as a server to the cluster. It provides file access, login access, file transfer, printing and other services across the network to the cluster nodes.

command interpreter A program that reads lines of text from standard input (typed at the keyboard or read from a file), and interprets them as requests to execute other programs. An HP-UX command interpreter is called a "shell".

command line prompt A command line prompt shows that the computer is ready to accept your commands. Each terminal window has a command line prompt that acts just like the command line prompt that would be shown if your computer was not running HP CDE. Usually the command line prompt is %, >, or $. You can find the command line prompt by pressing Enter in an HP CDE terminal window.

control key The keyboard key, normally labeled "CTRL", that is used as a modifier key. You hold down this key while pressing another key.

C Shell An HP-UX command interpreter, invoked as `csh`.

current working directory The directory in which you are currently located. **Relative path name** searches begin in this directory. It is also called the working directory.

cursor An image used to indicate the focus of keyboard input. The cursor can have several forms. For instance, the text entry cursor appears as an I.

directory An organizational unit of your workstation's disk drive, composed of files and subdirectories. A directory is analogous to a file folder containing letters (text files), which is contained in a filing cabinet (disk).

Glossary

environment The set of defined shell variables (some of which are PATH, TERM, SHELL, HOME) that define the conditions under which your commands run. These conditions can include your terminal characteristics, home directory, and default search path. These variables are set in your `.profile`.

execute permission Users with execute permission on a file can execute (run) the file as a program by typing the file name at the command prompt. If the file is a directory, they can access the directory's contents.

file The basic named unit of data stored on disk. See also **directory**, **file name**.

file access permissions File name characteristics (including *read*, *write*, and *execute*) that determine whether a process can perform a requested operation on the file (such as opening a file for writing). Access permissions can be changed by a *chmod*(1) command.

file name The name given to a particular file. See also **file**, **absolute path name**, **relative path name**, and **path name**.

file system The organized set of files and directories on a hard disk.

filter A command, such as `cat`, `grep`, or `sort`, that reads data from the standard input, performs a transformation on the data, and writes it to the standard output.

font A complete set of characters (letters, digits, and special characters) of one size and one typeface. "Ten-point, Helvetica, bold" is an example of a font.

foreground process The process occupying the currently active terminal I/O, which may be a window. The shell will not return a prompt until a foreground process has finished executing.

group An association of users permitted to access the same set of files. The members of a group are defined in the files `/etc/passwd`, `/etc/group`, and `/etc/loginggroup` (if it exists) via a numerical group ID. Users with identical group IDs are members of the same group.

group access list The group access list is a set of supplementary group IDs, associated with a process, used in determining resource accessibility.

Glossary

$HOME The value of the environment variable representing the **home directory**.

home directory A personal directory where you keep files and additional subdirectories that belong to you. By default, File Manager and Terminal Emulator windows are set to your home directory when you first open them.

/HomeDirectory/ Symbolizes your **home directory**. For example, if your home directory is `/home/anna/`, then `/`*`HomeDirectory`*`/bitmaps/smile.bm` represents `/home/anna/bitmaps/smile.bm`.

host name The unique identifying name given to a system in a network. There are generally different **host name** domains associated with different networks. Also called node name. For example, `hpabc`.

invisible file name A file name in which the first character is a dot (.). Invisible file names are not displayed by the HP-UX listing commands such as `ls` and `ll` unless you use the `-a` option.

keysh The command for invoking a **Key Shell**.

Key Shell An HP-UX shell which, as an extension of the Korn Shell, uses hierarchical softkey menus and context-sensitive help to aid users in building command lines. Invoked as `usr/bin/keysh`.

Korn Shell An HP-UX shell, featuring command history recall and line-editing. Invoked as `/usr/bin/ksh`. As of the 10.0 Release of HP-UX, this shell is obsolete, replaced by the **POSIX** shell.

LANG An NLS environment variable used to inform a computer process of the user's requirements for "native language," "local customs," and "coded character set."

LAN Stands for Local Area Network. The systems and/or clusters that share data, hardware, and software resources via networking software.

log in To begin a **session** on the computer by entering the necessary information, such as your user name (login name) and password.

Glossary

login The login name by which you are known to the system. This may be any group of characters, as long as it meets system rules.

NFS Network File Services.

NFS file system A file system accessible over a network via the NFS Services product.

NLSPATH An NLS environment variable used to indicate the search path for message catalogs.

operating system The kernel (`/stand/vmunix`), commands, input-output control, system accounting, mass storage, and other services.

owner The owner of a file is usually the creator of that file. Ownership of a file can be changed by the superuser or the current owner.

parent directory A directory that contains other directories, each of which is then called a subdirectory. See also **subdirectory**.

parent process In a shell environment, an existing process that has caused a new process (a **child process**) to be created.

password An encrypted sequence of characters used by HP-UX to identify an authorized user and to permit authorized login on a system.

path name Specifies the location of a particular file or directory within the directory structure by specifying the directories you need to pass through to get there. The directory names are separated by slashes. For example, `/home/michael/myfile` is the path name for `myfile`.

There are two kinds of path names. See also **relative path names** and **absolute path names**, and **file name**.

permissions See **access permissions**.

PID Process identifier (number).

POSIX POrtable Systems Interface, complying with UNIX standards 1003.1 and 1003.2 from IEEE.

POSIX Shell **POSIX**-compliant version of the Korn Shell.

process An invocation of a program. Generally, **process** refers to a program running in memory, while **program** is the code (a sequence of instructions stored on disk that cause the system to perform some function). Several users can access the same program simultaneously. Each generates a separate process from the same program.

process ID A unique identification number assigned to all processes by the operating system. *Also see* PID.

prompt See **command line prompt**.

read permission Users with read permission can view the contents of a file or directory.

regular expression A string of characters that selects text.

relative path name The name of a file, listing all the directories leading to that file in relation to the **current working directory**.

For example, if you are in the `/home` directory, `michael/myfile` is the relative path to `/home/michael/myfile`.

See also **absolute path name**.

root See **superuser**.

root directory The highest level directory of the hierarchical **file system**, from which all other files branch. In HP-UX, the slash (/) character refers to the "root directory." The root directory is the only directory in the file system that is its own "parent directory."

run-level The system state determined at boot that defines, among other things, multi- or single-user status.

SAM The HP System Administration Manager, a tool that allows you to perform many system administration tasks without having to know the specific HP-UX commands that are associated with the task. You must have **superuser** permission to run SAM.

server A computer program that provides file access, login access, file transfer, printing and other services across a network. Sometimes, but not always, a server consists of a dedicated computer.

session Generally describes the time between beginning to use an application and quitting the

Glossary

application. More specifically, used to describe the time between logging in and logging out.

shell An HP-UX command interpreter (Bourne, Korn, Key, POSIX or C), providing a working environment interface for the user. The shell takes command input from the keyboard and interprets it for the **operating system**.

software application A program used to perform a particular task, usually interactively, such as computer-aided design, text editing, or accounting. Style Manager, Text Editor, and File Manager are examples of software applications.

standard error The destination of error and special messages from a program, intended to be used for diagnostic messages. The standard error output is often called `stderr`. Standard error usually appears on the display unless it is directed otherwise.

standard input The source of input data for a program. The standard input file is often called stdin. Standard input is usually supplied by entering data at the keyboard.

standard output The destination of output data from a program. The standard output file is often called `stdout`. Standard output appears on the display unless it is redirected otherwise.

subdirectory A directory that is located in, or anywhere on a path below, another directory, which is then called its **parent directory**. Sometimes called child directory.

superuser A login that allows special permissions for modifying system files that most users do not have permission to modify. Superuser is also called "the root user" or simply "root" since the user ID for superuser is `root`. On most computer systems, only a few users have permission to become superuser.

System Administration Manager See **SAM**.

system administrator The person responsible for system and network installation, updating, maintenance, and security at your site.

user account The system administrator defines a user account for every person authorized to use the system. Each user account contains the name the computer uses to identify the person (user name) and the person's password. See also **user name**, **password**.

user name The name that identifies your account to the `login` program and to the mail systems and other software requiring secure entry. Sometimes called **login**.

utility A program provided with the HP-UX operating system to perform a task, such as printing a file or displaying the contents of a directory.

working directory See **current working directory**.

write permission Users with write permission can change the contents of a file or directory.

Index

Index

Index

Index

Index

Index

Index

Index

Index

Index