UNIX® Essentials Featuring the Solaris™ 9 Operating Environment SA-119

Student Guide With Instructor Notes

Sun Microsystems, Inc.
UBRM05-104
500 Eldorado Blvd.
Broomfield, CO 80021
U.S.A.

Revision A

Table of Contents

About This Course

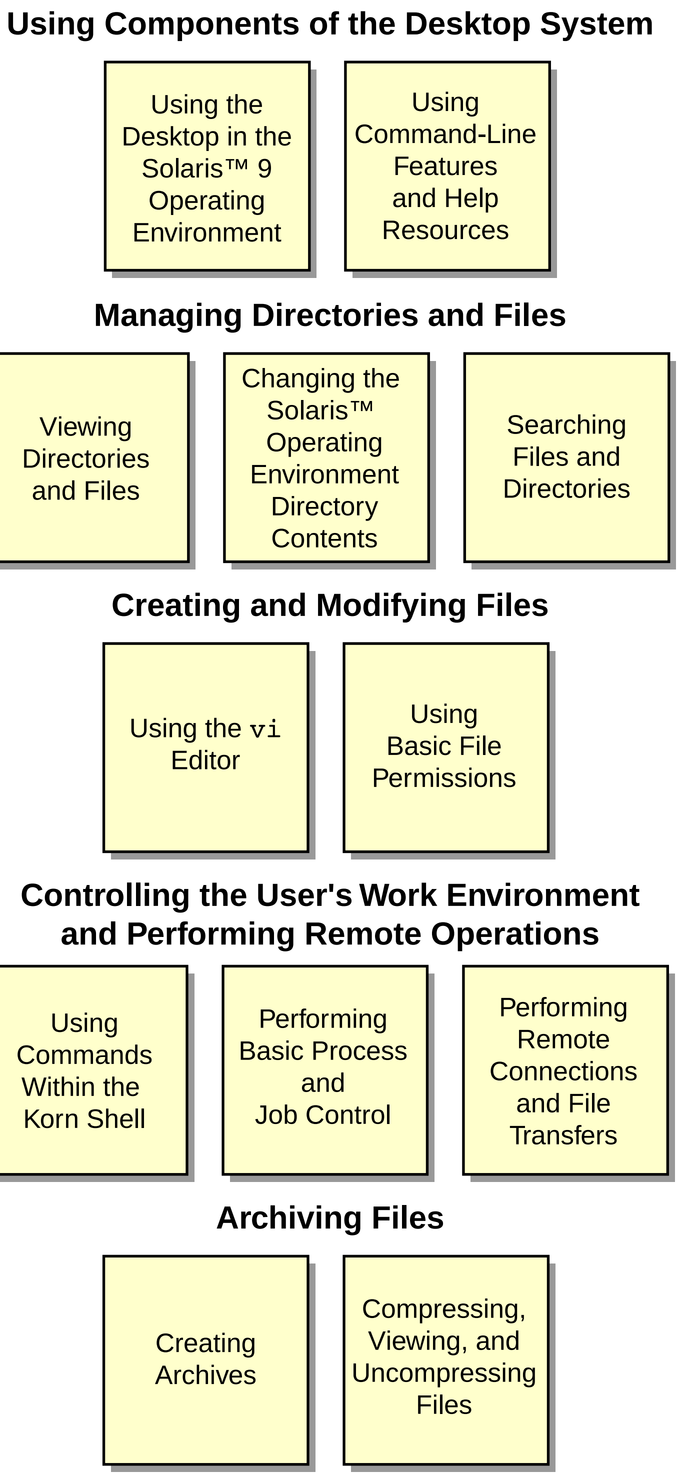
Instructional Goals

Upon completion of this course, you should be able to:

- Use components of the desktop system
- Manage directories and files
- Create and modify files
- Control the user's work environment and perform remote operations
- Archive files

Course Map

The following course map enables you to see what you have accomplished and where you are going in reference to the instructional goals.


Using Components of the Desktop System
Using the Desktop in the Solaris™ 9 Operating Environment
Using Command-Line Features and Help Resources
Managing Directories and Files
Viewing Directories and Files
Changing the Solaris™ Operating Environment Directory Contents
Searching Files and Directories
Creating and Modifying Files
Using the vi Editor
Using Basic File Permissions
Controlling the User's Work Environment and Performing Remote Operations
Using Commands Within the Korn Shell
Performing Basic Process and Job Control
Performing Remote Connections and File Transfers
Archiving Files
Creating Archives
Compressing, Viewing, and Uncompressing Files

Topics Not Covered

This course does not cover the following topics. Many of these topics are covered in other courses offered by Sun Educational Services:

- System administration concepts – Covered in SA-239: *Intermediate System Administration for the Solaris™ 9 Operating Environment*
- Detailed shell programming – Covered in SL-120: *Shell Programming for System Programmers*
- Advanced system administration concepts – Covered in SA-299: *Advanced System Administration for the Solaris™ 9 Operating Environment*

Refer to the Sun Educational Services catalog for specific information and registration.

How Prepared Are You?

To be sure you are prepared to take this course, can you answer yes to the following questions?

- Can you use a keyboard to input commands and control characters?
- Can you use a mouse to point and click in a graphical user interface (GUI)?
- Are you familiar with working in a desktop environment?

Introductions

Now that you have been introduced to the course, introduce yourself to the other students and the instructor, addressing the items shown below:

- Name
- Company affiliation
- Title, function, and job responsibility
- Experience related to topics presented in this course
- Reasons for enrolling in this course
- Expectations for this course

How to Use Course Materials

To enable you to succeed in this course, these course materials employ a learning module that is composed of the following components:

- Objectives – You should be able to accomplish the objectives after completing a portion of instructional content. Objectives support goals and can support other higher-level objectives.
- Lecture – The instructor will present information specific to the objective of the module. This information will help you learn the knowledge and skills necessary to succeed with the activities.
- Activities – The activities take on various forms, such as an exercise, self-check, discussion, and demonstration. Activities are used to facilitate mastery of an objective.
- Visual aids – The instructor might use several visual aids to convey a concept, such as a process, in a visual form. Visual aids commonly contain graphics, animation, and video.

Note – Many system administration tasks for the Solaris™ Operating Environment (Solaris OE) can be accomplished in more than one way. The methods presented in the courseware reflect recommended practices used by Sun Educational Services.

Conventions

The following conventions are used in this course to represent various training elements and alternative learning resources.

Icons

Discussion – Indicates a small-group or class discussion on the current topic is recommended at this time.

Note – Indicates additional information that can help students but is not crucial to their understanding of the concept being described. Students should be able to understand the concept or complete the task without this information. Examples of notational information include keyword shortcuts and minor system adjustments.

Caution – Indicates that there is a risk of personal injury from a nonelectrical hazard, or risk of irreversible damage to data, software, or the operating system. A caution indicates that the possibility of a hazard (as opposed to certainty) might happen, depending on the action of the user.

Typographical Conventions

`Courier` is used for the names of commands, files, directories, user names, host names, programming code, and on-screen computer output; for example:

> Use the `ls -al` command to list all files.
> `host1# cd /home`

`Courier bold` is used for characters and numbers that you type; for example:

> To list the files in this directory, type the following:
> `#` **`ls`**

`Courier italics` is used for variables and command-line placeholders that are replaced with a real name or value; for example:

> To delete a file, use the `rm` *`filename`* command.

`Courier italic bold` is used to represent variables whose values are to be entered by the student as part of an activity; for example:

> Type **`chmod a+rwx`** ***`filename`*** to grant read, write, and execute rights for *`filename`*.

Palatino italics is used for book titles, new words or terms, or words that you want to emphasize; for example:

> Read Chapter 6 in the *User's Guide*.
> These are called *class* options.

Notes to the Instructor

This course includes eSlides for instructors to use as visual aids in the classroom. eSlides, a browser-based visual aid tool, enables delivery of text, graphics, video, and animated media in the classroom.

Module	**Lecture (Minutes)**	**Lab (Minutes)**	**Total Time (Minutes)**
About This Course	45	0	45
Using the Desktop in the Solaris™ 9 Operating Environment	75	30	105
Using Command-Line Features and Help Resources	50	30	80
Viewing Directories and Files	90	30	120
Changing the Solaris Operating Environment Directory Contents	125	30	155
Searching Files and Directories	80	30	110
Using the `vi` Editor	60	30	90
Using Basic File Permissions	95	30	125
Using Commands Within the Korn Shell	120	30	150
Performing Basic Process and Job Control	105	30	135
Performing Remote Connections and File Transfers	80	30	110
Creating Archives	60	30	90
Compressing, Viewing, and Uncompressing Files	45	30	75

Module 1

Using the Desktop in the Solaris™ 9 Operating Environment

Objectives

Upon completion of this module, you should be able to:

- Describe the four main hardware components of a desktop computer
- Describe the three main components of the SunOS™ operating system
- Log in to the system by using the Common Desktop Environment (CDE) and the command line
- Use the CDE

The following course map shows how this module fits into the current instructional goal.

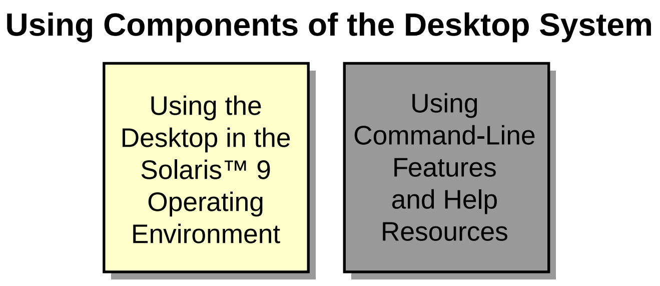

Using Components of the Desktop System
Using the Desktop in the Solaris™ 9 Operating Environment
Using Command-Line Features and Help Resources

Figure 1-1 Course Map

Introducing the Main Components of a Computer

A computer consists of hardware and software that work together to perform tasks. The computer hardware is made up of several components, such as the central processing unit (CPU), random access memory (RAM), and disks. The software, or operating system, is a collection of programs and files that direct and coordinate the hardware activities.

Hardware Overview

Figure 1-2 shows the four main hardware components of a computer, which are the RAM, the CPU, the input/output (I/O) devices, and the hard disk or some other storage device.

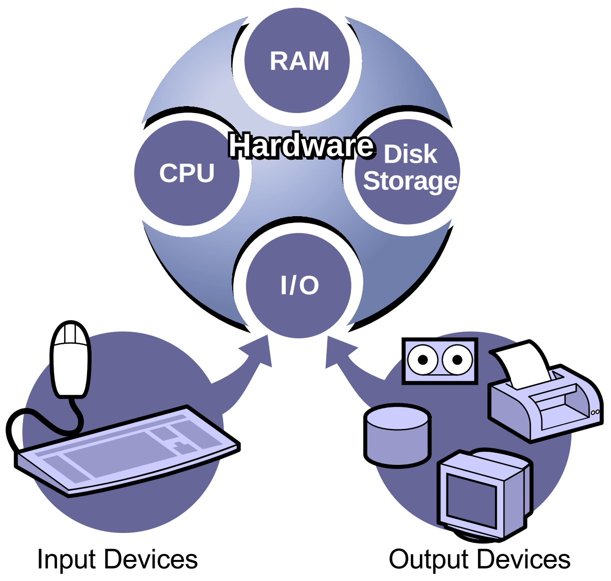


Figure 1-2 Main Hardware Components of a Computer

RAM

RAM is the main computer memory. The statement, "The system has 64 Mbytes of memory," refers to the amount of RAM installed on the computer. Software programs and data must be loaded into RAM before the operating system can process them.

SunOS 5.*x* is a virtual memory operating system. A virtual memory operating system enables programs to execute as if more memory is available than physically exists in RAM. This is accomplished by using disk space as temporary memory storage.

Programs reside on the hard disk. When you execute a program, a copy of the program is loaded into virtual memory, and pieces are copied into RAM as needed during the actual execution of the program. The program remains in virtual memory until it finishes. Upon program termination, the operating system can overwrite the virtual memory space with other programs it needs to execute. If you reboot the system, or if there is a power loss, all data in virtual memory is cleared.

CPU

The CPU is the computer logic chip that executes instructions received from RAM. These instructions are stored in binary code on the hard disk.

Input/Output Devices

I/O devices are components in the computer that communicate with external pieces of equipment. An I/O device reads information from an input device, such as a keyboard or a mouse, into RAM. An I/O device writes information to output devices, such as monitors, printers, or tape drives.

Hard Disk

The hard disk is a magnetic storage device that stores files, directories, and software programs.

The Solaris Operating Environment

The Solaris Operating Environment (Solaris OE) consists of the SunOS 5.*x* operating system, Open Network Computing (ONC+™) software, and the Solaris CDE graphical user interface (GUI).

The SunOS operating system is software that manages system resources and schedules system operations. This operating system interprets instructions from the user or from an application and instructs the computer what to do. The operating system handles data, keeps track of data stored on disks, and communicates with I/O devices, such as monitors, hard disks, diskette drives, printers, and modems.

ONC+ software provides network services, such as Network File System (NFS), which allows file sharing between computers. Other services are Network Information Service (NIS), which provides network information naming services to clients, and Network Information Service Plus (NIS+).

The CDE is the GUI that, for example, displays the Login screen on your monitor. You can use the CDE to access most of the functions of the computer.

Introducing the SunOS™ Operating System

The three main components of the SunOS operating system are:

- The `kernel`
- The shell
- The directory hierarchy

The `kernel`

The `kernel` is the core of the SunOS operating system. Figure 1-3 shows the `kernel`, which manages all the physical resources of the computer, including:

- File systems and structures
- Device management, such as storing data to the hard disk
- Process management or CPU functions
- Memory management

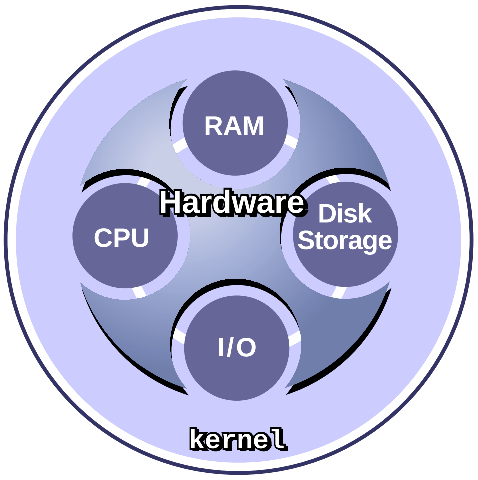


Figure 1-3 Role of the `kernel`

The Shell

Figure 1-4 shows how the shell is an interface between the user and the `kernel`. The shell is primarily a command interpreter. The shell accepts commands you enter, interprets these commands, and then passes them to the `kernel` for execution.

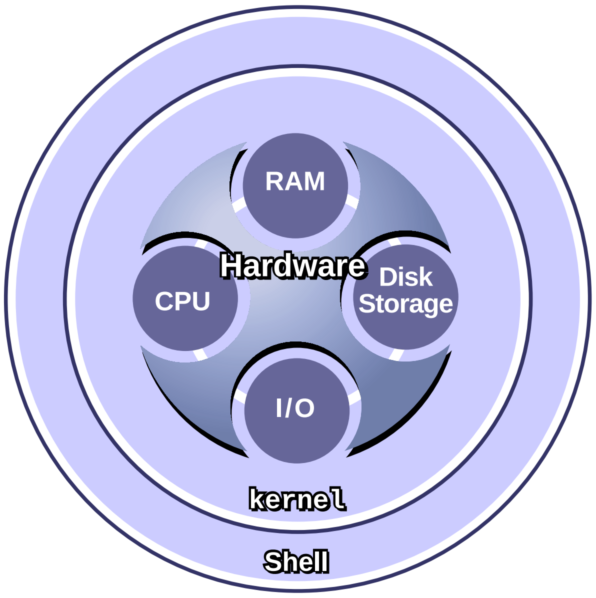


Figure 1-4 Role of the Shell

The Default Shells

The Solaris OE supports three primary shells:

- Bourne shell
- C shell
- Korn shell

Examples given in this course assume that the Korn shell is being used.

The Bourne shell is the original UNIX® system shell. The Bourne shell is the default shell for the `root` user. The `root` user is a special system account with unlimited access privileges for system administrators. The default Bourne shell prompt for a regular user is a dollar sign ($); for the `root` user the default shell prompt is a pound sign (#).

The C shell has a number of features that the Bourne shell does not, such as command-line history, aliasing, and job control. The default C shell prompt for a regular user is the host name followed by a percent sign (`hostname%`); for the `root` user the default shell prompt is the host name followed by a pound sign (`hostname#`).

The Korn shell is a superset of the Bourne shell with C shell-like enhancements and a number of additional features. The Bourne shell is almost completely upwardly compatible with the Korn shell. The default Korn shell prompt for a regular user is a dollar sign (`$`); for the `root` user the default shell prompt is the pound sign (#).

The Alternative Shells

The Solaris 9 OE contains three additional shells:

- Bash – The GNU project's Bourne-Again shell is a Bourne-compatible shell that incorporates useful features from the Korn and C shells.
- Z shell – The Z shell most closely resembles the Korn shell but includes many other enhancements.
- TC shell – The TC shell is a completely compatible version of the C shell with additional enhancements.

The Directory Hierarchy

Figure 1-5 on page 1-8 shows a Solaris OE directory hierarchy. The directory hierarchy contains an organized group of directories and files.

In this example you see the starting point of the `user1` home directory within the directory hierarchy.

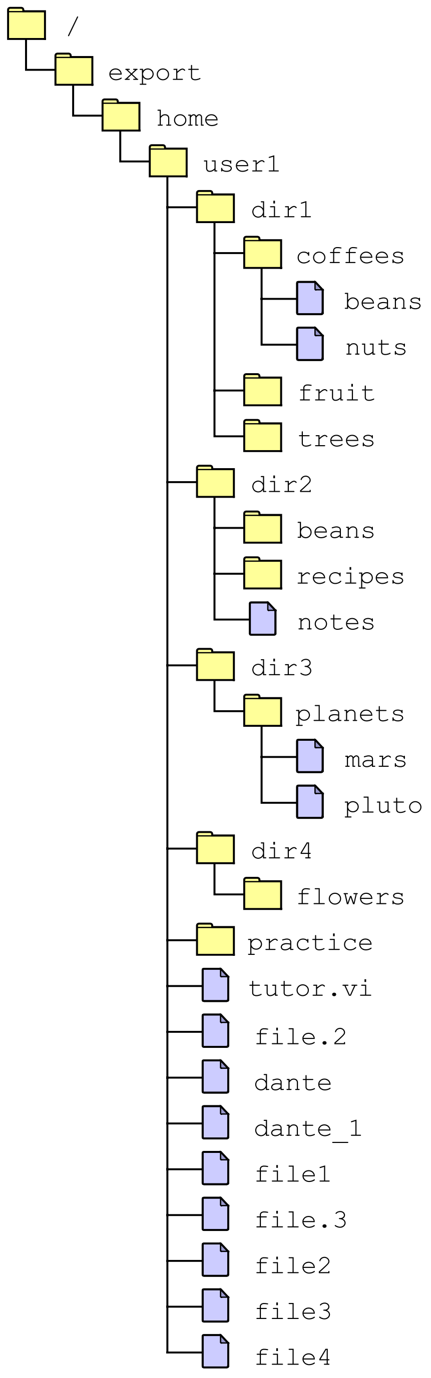


Figure 1-5 Directory Hierarchy

Logging In to the System

The login process provides the system with a way of recognizing and authenticating the user. The CDE Login screen, which appears on your monitor, enables you to log in to the system and use the desktop. Figure 1-6 shows the Login screen.

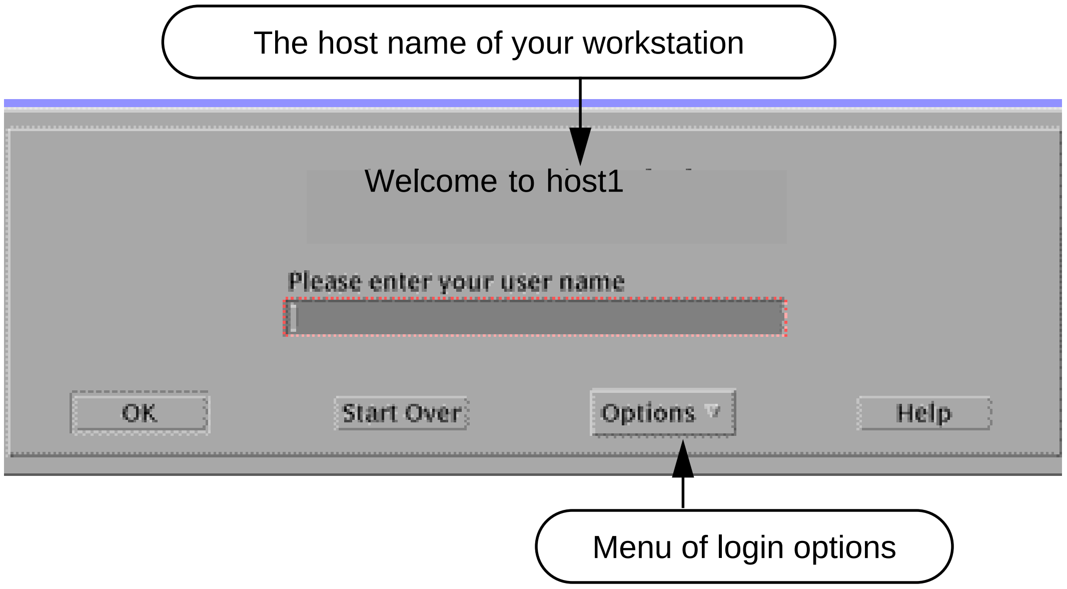

The host name of your workstation
Welcome to host1
Please enter your user name
OK
Start Over
Options
Help
Menu of login options

Figure 1-6 Login Screen

There are several ways to log in to the system. You can log in directly by entering your user name and password, or you can use the Options button. The Options button enables you to log in to a remote host or log in using the command line.

Selecting Login Options With the Options Button

The Options button allows you to select different methods of logging in. Click Options on the Login screen. A menu as shown in Figure 1-7 on page 1-10 lists a hierarchy of login choices. Click the option you want to use.

- ➤ Options
  - ➤ Language
  - ➤ Session
    - ➤ Common Desktop Environment (CDE)
    - ➤ User's Last Desktop
    - ➤ Failsafe Session
  - ➤ Remote Login
    - ➤ Enter Host Name
    - ➤ Choose Host From List
  - ➤ Command Line Login
  - ➤ Reset Login Screen

Figure 1-7 Login Options Menu

Language

The Language option allows you to select a particular language for your session. The system administrator sets a default language for your workstation.

Session

The Session option allows you to select a CDE, User's Last Desktop, or Failsafe Session.

The CDE option opens a desktop environment. If no home session is set, this option opens the desktop as you left it at the end of your previous session. If the Return to Home session is set, this option opens the desktop as you specify by selecting Set Home Session in the Startup Style Manager.

The User's Last Desktop option opens the desktop as you left it at the end of your previous session.

The Failsafe option opens a terminal window instead of starting a desktop window. This method of logging in allows you to fix problems with other sessions.

Remote Login

The Remote Login option enables you to connect to a remote system to start a remote CDE desktop log in. This option enables you to either enter the specific host name of a remote system or to select from a list of available remote systems.

Command Line Login

The Command Line Login option enables you to work in a UNIX command environment. This non-GUI environment is not a CDE desktop session. The CDE desktop is suspended in this mode.

When you log out from a command-line session, the CDE Login screen reappears within 30 seconds.

Reset Login Screen

The Reset Login Screen option restarts the CDE Login screen.

Logging In Using the CDE Login Screen

Assign user names and passwords and have the students log in to their systems.

To log in to a desktop session from the CDE Login screen, execute the following:

1. Type your user name in the text field. Press Return or click OK.
2. Type your password in the password text field. Press Return or click OK.

If the login attempt fails, a dialog box appears stating:

```
Login incorrect; please try again.
```

3. Exit your CDE login session by clicking EXIT, located on the front panel.

Logging In Using the Command Line

To log in to a command-line session, perform the following:

1. Click Options to display the Options pull-down menu.
2. Click Command Line Login. The Login screen disappears.
3. Press Return to get a prompt for a user name entry. If you do not log in within 30 seconds, the Login screen automatically restarts.

Choosing the Command Line Login option suspends the `dtlogin` process and starts a timer for the process to restart. The `dtlogin` screen reappears when that timer expires regardless of whether you pressed Return. The timeout is approximately 30 seconds.

4. Type your user name at the prompt. Press Return.
5. Type your password at the `Password:` prompt. Press Return. The password does not appear on the screen when it is entered.
6. Log out of your command-line session by typing `exit`.

Note – By default, if a user does not have a password, then the user is automatically prompted to enter a new password during the initial login process.

Changing Your Password

Passwords protect user accounts from unauthorized access. Users should change their passwords frequently to further prevents unauthorized access to the system.

In the Solaris OE, a user's password:

- Must be six to eight characters in length
- Should contain at least two alphabetic characters and must contain at least one numeric or special character, such as a semicolon (`;`), an asterisk (`*`), or a dollar sign (`$`)
- Must differ from the user's login name
- Must differ from the previous password by at least three characters
- Can contain spaces
- Cannot be the reverse of the user's login name

These password requirements do not apply to the system administrator's `root` account password or to any password that is created for a regular user by the `root` user.

To change your password, perform the following steps:

1. Move the cursor to an open space on the desktop.
2. Press the right mouse button to open the Workspace Menu.
3. Click Tools.
4. Click Terminal, and a new terminal window opens with a shell prompt in the upper left corner of the window. The shell is ready to receive a command.

5. Type the `passwd` command at the shell prompt. Press Return.
6. When the prompt `Enter existing login password:` appears, enter the current password. Press the Return key.
7. When the prompt `New password:` appears, type the new password. Press Return.
8. When the prompt `Re-enter new Password:` appears, retype the new password. Press Return. The system uses your second entry to verify the new password.

```
$ passwd
passwd: Changing password for user1
Enter existing login password:
New password:
Re-enter new passwd:
passwd: passwd successfully changed for user1
$
```

Using the CDE

The CDE is the standard desktop environment available to Solaris OE users. This section describes how to use the CDE to secure and select sessions, add and delete workspaces, change backgrounds, and manage files.

Securing Your CDE Session

Securing your CDE session prevents unauthorized users from gaining access to the system. The two ways to secure the system are:

- Locking the screen
- Exiting the session

Figure 1-8 shows the Padlock and EXIT buttons on the front panel.

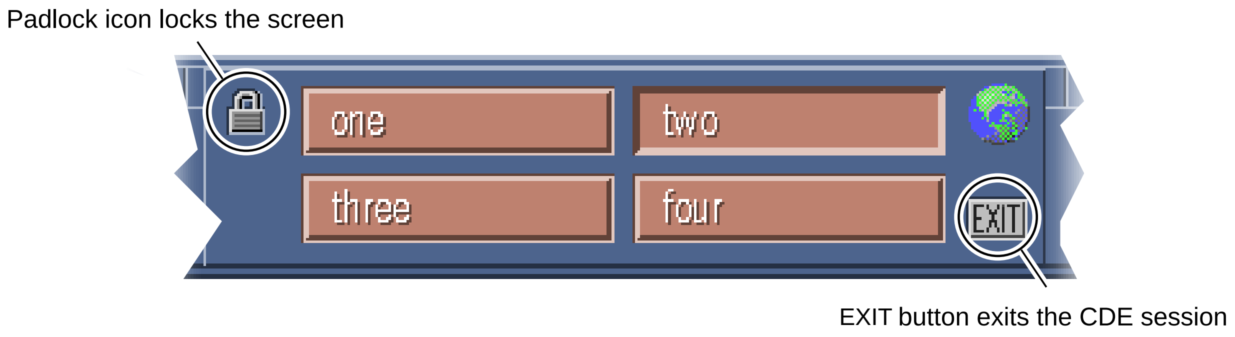

Padlock icon locks the screen
one
two
three
four
EXIT
EXIT button exits the CDE session

Figure 1-8 Two Methods of Securing Your Session

Locking the Screen

Locking the screen prevents unauthorized users from gaining access to your CDE session, while keeping your session intact. To lock the screen perform the following:

1. Click the Padlock icon on the front panel to lock the screen and apply password protection.
2. Type your password and press Return to regain access to your CDE session.

Exiting the Session

Exiting your session ends that session completely. Any data contained in the current set of open applications is lost when you exit. Save all currently unsaved data before exiting the session. To exit using the EXIT button, perform the following:

1. Click EXIT on the front panel. A Logout Confirmation window appears.
2. Click OK or press Return when the OK button is highlighted to confirm that you want to log out.

Selecting CDE Login Sessions

The Options button on the Login screen allows you to select three types of sessions, CDE, User's Last Desktop, or Failsafe, depending on your desktop requirements.

The CDE session is the default and most commonly used desktop. The User's Last Desktop session displays the last saved session for that user. The Failsafe session bypasses the typical CDE startup and displays a terminal window.

Initial Session

When you log in to the desktop for the first time, your desktop display conforms to system default values. Your desktop displays the File Manager and Introducing the Desktop, a help volume.

Current Session

Your running session is always considered the current session. When you exit a session, the system saves the current session by default. The next time you log in to the desktop, the system displays this current session exactly as it appeared when you last logged out. This function can be modified using the Style Manager - Startup utility. This Style Manager can be accessed by clicking the Desktop Controls icon located on the front panel.

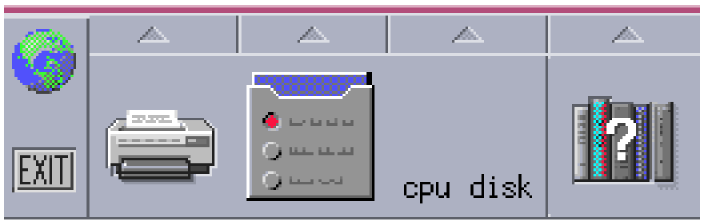


Figure 1-9 Desktop Controls Icon

Home Session

You can also create a session in the same state every time you log in, regardless of the state of the session when the previous user logged out. You save the state of a current session using the Style Manager - Startup settings. This sets the current session as your default session for every time you log in.

Changing the CDE Background

You can change the background for your desktop by clicking on the Backdrop icon in the Style Manager window. The color choices you make also affect the backdrop appearance. You can select a different backdrop for each workspace. Figure 1-10 shows the Style Manager - Backdrop window.

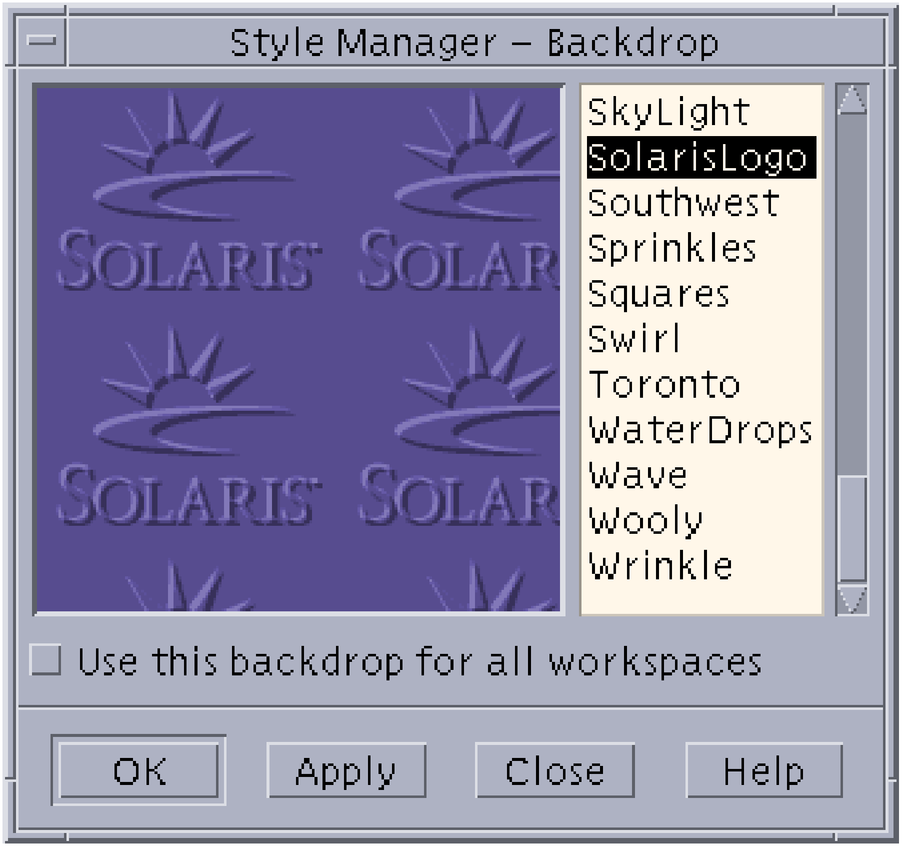


Figure 1-10 Style Manager - Backdrop Window

To change the background on your desktop, perform the following:

1. Click the Desktop Controls icon to display the Style Manager window.
2. Double click the Backdrop icon.
3. Select the backdrop you want to use from the Style Manager - Backdrop window.
4. Click OK.

Accessing CDE Workspaces

There are four default workspace buttons on the front panel. Each workspace is a separate desktop environment. Click the workspace buttons to move from one workspace to another.

You can customize each workspace using the Style Manager. You can also add and delete workspaces. The number of workspaces is reflected by the number of buttons on the front panel.

With the right mouse button, click a workspace button to add, delete, or rename the workspace. Figure 1-11 shows how to manipulate workspaces.

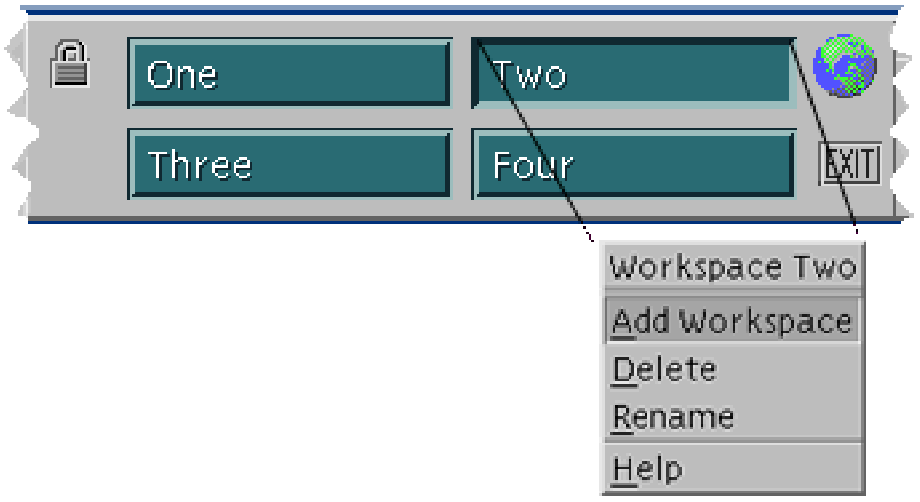


Figure 1-11 Altering Workspaces

Adding a CDE Workspace

To add a workspace, perform the following:

1. Move the pointer to the workspace area on the front panel.
2. Click the workspace called Two with the right mouse button, and select Add Workspace. A new workspace called New appears on the front panel.

Renaming a CDE Workspace

There are two ways to change the name of a workspace.

To change the name of the workspace that is currently active:

1. Move to the workspace you want to rename.
2. Click the workspace name on the front panel.
3. Delete the old name.
4. Type in the new name.
5. Press Return to complete the change.

To change the name of another workspace:

1. Move the pointer to the workspace name on the front panel.
2. Click the right mouse button.
3. Click Rename.
4. Delete the old name.
5. Type the new name.
6. Press Return to complete the change.

Deleting a CDE Workspace

To delete a workspace, perform the following:

1. Move the pointer to the workspace name on the front panel that you want to delete.
2. Click the right mouse button.
3. Select Delete. The workspace is removed from the front panel.

Managing Files Using the CDE

The File Manager allows you to organize files into a hierarchical structure of directories and subdirectories. When you open the File Manager, the default view lists the contents of your home directory. From that directory, you can move up and down in the hierarchy to view other directories.

Figure 1-12 shows the File Manager icon located on the front panel.

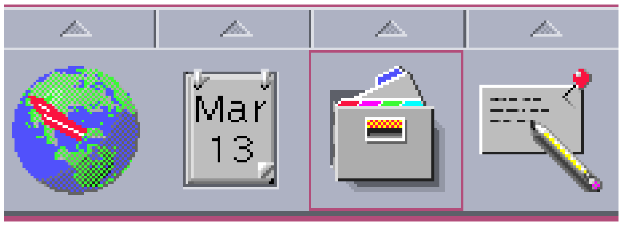


Figure 1-12 File Manager Icon

Your initial CDE session starts the File Manager. To open the File Manager manually, click the File Manager icon located on the front panel.

The path name lists the path through the hierarchy that you must follow to locate a file or directory. The path name of the current directory is displayed above the list of the directory contents. In this example, the path name is `/export/home/user1`.

Figure 1-13 shows the path name and a list of contents in the `user1` directory.

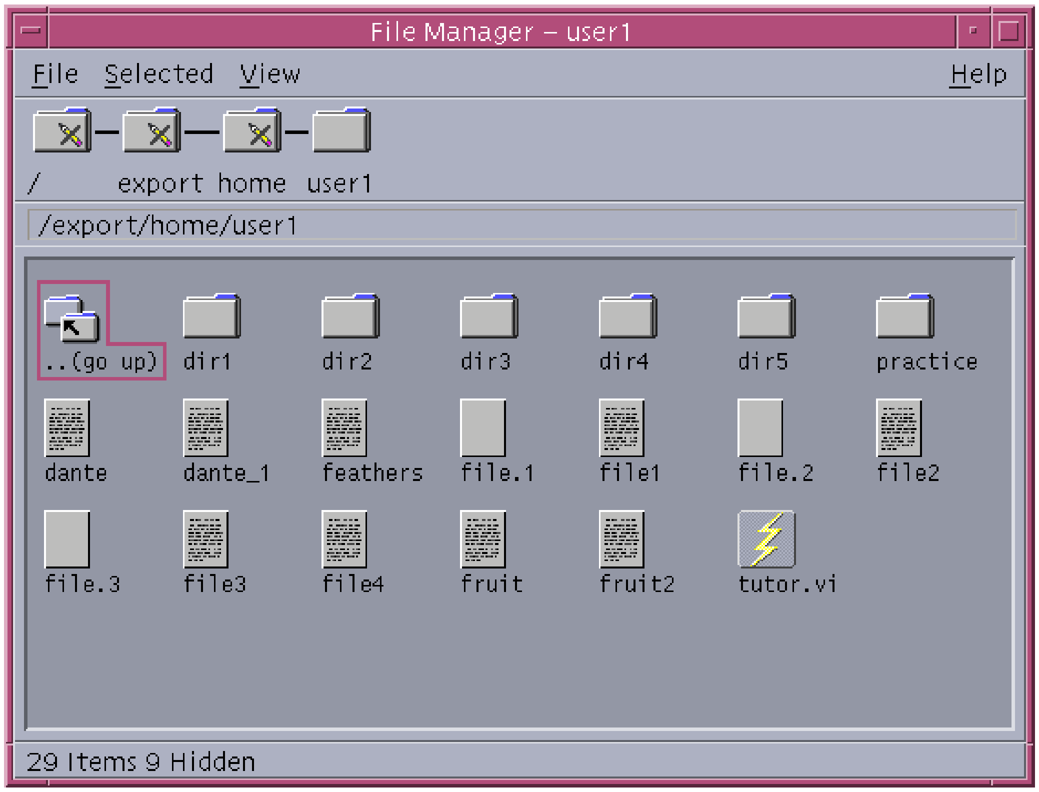


Figure 1-13 File Manager

To move a file from one directory to another, position the mouse pointer over the file icon, hold down the left mouse button and drag the icon to the appropriate directory icon. When the file icon is positioned over the directory icon, release the left mouse button, and the file moves to that directory.

Figure 1-14 shows the `file.1` file moving to the `practice` directory.

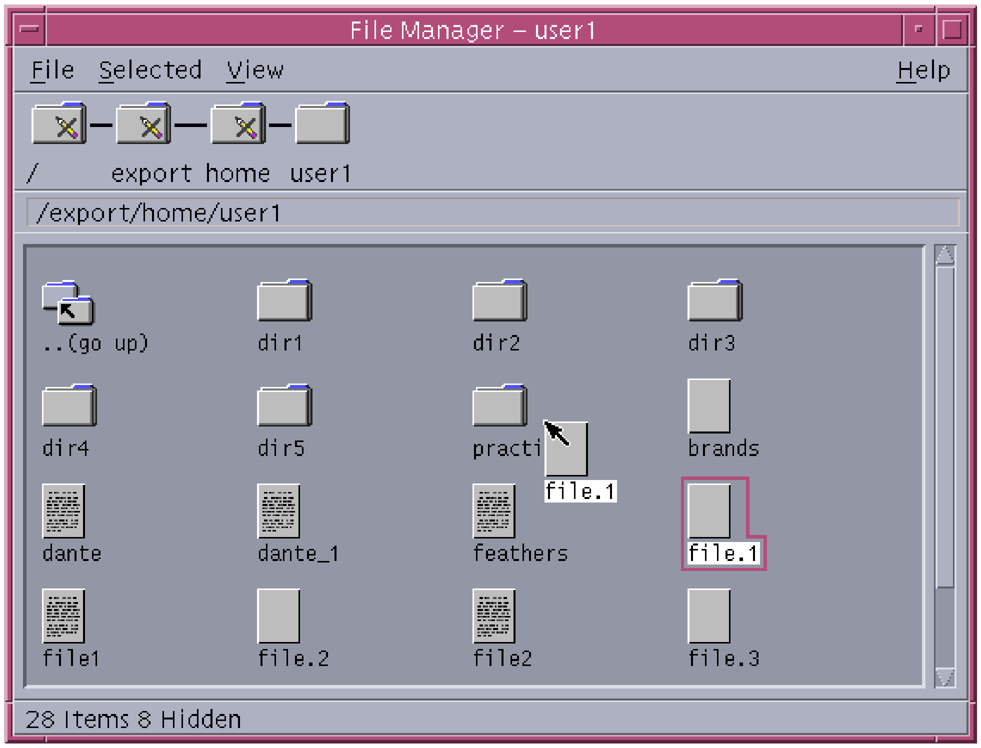


Figure 1-14 Moving Files in File Manager

Note – Figure 1-14 is for illustration only. If you moved the `file.1` file into the `practice` directory, make sure to return the `file.1` file back to its original location.

Exercise: Using the Solaris 9 OE

In this exercise, you complete the following tasks:

- Identify the components of the computer
- Use passwords
- Log in using the Command Line Login option
- Invoke a Failsafe Session
- Lock the CDE screen
- Drag and drop files in a directory

Preparation

Log in or log out of your CDE session or system as required.

Tasks

Complete the following tasks.

Task 1 – Identifying the Components of the Computer

Complete the following steps to identify components of the computer.

1. List the four main hardware components of a computer.

   ____

   ____

   ____

   ____

2. Name the three main components of the SunOS operating system.

   ____

   ____

   ____

3. List the three primary shells supported by the Solaris OE.

 ____

 ____

 ____

4. Match the following terms with their descriptions:

__ Shell	a. Core of the Solaris OE
__ `kernel`	b. Command interpreter
__ Hard disk	c. Storage device

Task 2 – Using Passwords

Complete the following steps to use passwords.

1. Obtain a user name and password from your instructor.
2. Log in to the system using the CDE Login screen.
   a. Type your user name. Press Return.
   b. Type your password. Press Return.
3. Click the right mouse button in the CDE desktop background. The Workspace Menu appears.
4. Select Tools from the Workspace Menu. The Tools menu appears.
5. Select Terminal from this menu. A terminal window appears.
6. Use the `passwd` command to change your password to `mypass1`.

```
$ passwd
passwd: Changing password for username
Enter existing login password:
New password:
Re-enter new passwd:
passwd: passwd successfully changed for username
$
```

7. Click the EXIT button on the front panel to exit the CDE session. A Logout Confirmation window appears.
8. Click OK or press Return to continue to log out.

9. Enter the following incorrect user name and password on the Login screen:

   User name: **`user2`**

   Password: **`wrong`**

   The following message appears:

   ```
   Login incorrect; please try again
   ```
10. Click OK.
11. Click and hold down the Options button on the Login screen.
12. Select the Reset Login Screen option from the Options menu.

Task 3 – Logging In Using the Command-Line Login Option

Complete the following steps to log in using the command line.

1. Click and hold down the Options button on the Login screen.
2. Select Command Line Login. The following message appears.

```
************************************************************************
* Suspending Desktop Login ...
*
* If currently logged out, press [Enter] for a console login prompt.
*
* Desktop Login will resume shortly after you exit console session.
*************************************************************************
*
```

3. Press Return to display the following log in prompt.

```
hostname console login:
```

4. Type your user name. Press Return.
5. Type your new password. Press Return.
6. At the shell prompt, type `exit`. The *`hostname`* `console login:` prompt reappears. After a short time, the Login screen reappears.

Task 4 – Invoking a Failsafe Session

Complete the following steps to invoke a Failsafe session.

1. Click and hold down the Options button on the Login screen.
2. Select Session. The Session menu appears.
3. Select Failsafe Session.

4. Type your user name. Press Return.
5. Type your password. Press Return. A terminal window appears.
6. Using the mouse, move the cursor into the terminal window.
7. At the shell prompt, type `exit`.

Task 5 – Locking the CDE Screen

Complete the following steps to lock the CDE screen.

1. Log in to the desktop using the default CDE session.
2. Locate the Padlock icon on the front panel.
3. Move your pointer over the Padlock icon.
4. Click the Padlock icon to lock the screen.
5. Enter your user password to unlock the screen.

Task 6 – Drag and Drop Files Into a Directory

To drag and drop a file into a directory in the desktop environment, you must create a directory and a file.

1. Open the File Manager if it is not already open on your desktop.
2. In the File Manager, click File.
3. Click New Folder.
4. A New Folder window appears. Name the folder `Folder1` and click OK.
5. In the File Manger, click File.
6. Click New File.
7. A New File window appears. Name the file `newfile` and click OK.
8. Move your pointer over the `newfile` icon.
9. Click and hold the left mouse button.
10. Drag the `newfile` icon over the `Folder1` icon and release the mouse button.
11. Move your pointer over the `Folder1` icon.
12. Click and hold the right mouse button.
13. Select Put in Trash and click OK.

Exercise Summary

Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercise.

Manage the discussion based on the time allowed for this module, which was provided in the “About This Course” module. If you do not have time to spend on discussion, highlight just the key concepts students should have learned from the lab exercise.

- Experiences

Ask students what their overall experiences with this exercise have been. Go over any trouble spots or especially confusing areas at this time.

- Interpretations

Ask students to interpret what they observed during any aspect of this exercise.

- Conclusions

Have students articulate any conclusions they reached as a result of this exercise experience.

- Applications

Explore with students how they might apply what they learned in this exercise to situations at their workplace.

Exercise Solutions

Task 1 – Identifying the Components of the Computer

1. List the four main hardware components of a computer.
   - *RAM*
   - *CPU*
   - *I/O*
   - *Hard disk*
2. Name the three main components of the SunOS operating system.
   - *The* `kernel`
   - *The shell*
   - *The directory hierarchy*
3. List the three primary shells supported by the Solaris OE.
   - *Bourne shell*
   - *C shell*
   - *Korn shell*
4. Match the following terms with their descriptions:

b	Shell	a. Core of the Solaris OE
a	`kernel`	b. Command interpreter
c	Hard disk	c. Storage device

Module 2

Using Command-Line Features and Help Resources

Objectives

Upon completion of this module, you should be able to:

- Construct and execute commands from the command line
- Use online documentation

The following course map shows how this module fits into the current instructional goal.

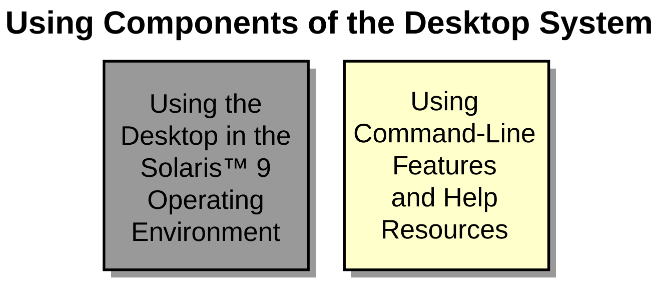

Using Components of the Desktop System
Using the Desktop in the Solaris™ 9 Operating Environment
Using Command-Line Features and Help Resources

Figure 2-1 Course Map

Constructing and Executing Commands From a Command Line

You use UNIX system commands to instruct the system to perform specific tasks. You enter commands on the command line in a terminal window. Command lines consist of commands with or without options and arguments. The structure and order of these components is known as syntax.

To open a terminal window, perform the following:

1. Log in to the system using the CDE Login screen.
2. Move the cursor to an open space on the desktop.
3. Press the right mouse button to open the Workspace Menu.
4. Click Tools.
5. Click Terminal. A shell prompt appears at the beginning of the command line. The shell is ready to receive a command.

Introducing Command-Line Syntax

You can modify how commands function by using options and arguments. Table 2-1 shows these components.

Table 2-1 UNIX System Command Components

Item	Description
`command`	Specifies what the system is to do (an executable).
`option`	Specifies how the command is run (a modifier). Options start with a dash (`-`) character.
`argument`	Specifies what is to be affected (a file, a directory, or text).

The following is an example of the format for commands:

```
command option(s) argument(s)
```

Using Commands

Examples of commands are `uname`, `date`, and `cal`.

The `uname` command provides information about the system. By default, entering this command displays the name of the current operating system.

To display the operating system information, enter the following:

```
$ uname
SunOS
```

The `date` command displays the system's current date and time. To display the date and time, enter the following:

```
$ date
Tue May 28 14:31:55 MDT 2002
```

The `cal` command displays a calendar for the current month and year. To display the calendar, enter the following:

```
$ cal
    May 2002
 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 29 30 31
```

To clear the terminal window, enter the following:

```
$ clear
```

Using Commands With Options

Adding options to a command alters the type of information displayed. Options are case sensitive.

The following is an example of the `uname` command with the `-a` option, which lists information currently available from the system.

```
$ uname -a
SunOS host1 5.9 Generic sun4u sparc SUNW,Ultra-5_10
```

You can use more than one option with a command. You can list multiple options separately or combined after one dash (`-`). The same response is presented regardless of the order of the options in the command line.

The following two examples show the `uname` command with two options. The `-s` option shows the name of the operating system. The `-r` option shows the operating system release level.

The following is an example of separate commands:

```
$ uname -s
SunOS
$ uname -r
5.9
```

The following is an example of the `uname` command with two separate options:

```
$ uname -s -r
SunOS 5.9
```

The following is an example of the `uname` command with two combined options:

```
$ uname -rs
SunOS 5.9
```

Whether you enter the command with separate or combined options, the response for both command lines is the same.

Using Commands With Arguments

Arguments enable you to define exactly what you want to do with a command. The following is an example of the `cal` command with two arguments. The first argument, `12`, specifies the month to be viewed. The second argument, `2002`, specifies the year to be viewed.

```
$ cal 12 2002
    December 2002
 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
29 30 31
```

Using Commands With Options and Arguments

The following is an example of the `ls` command with the `-l` option and the *`filename`* argument. The `ls` command lists the files in a directory. The `-l` option provides additional information about the files. The *`filename`* argument specifies the file to be viewed.

```
$ ls -l file.1
-rw-r--r--   1 user1    staff          0 Oct 19 09:54 file.1
```

Entering Multiple Commands on a Single Command Line

You can enter multiple commands on a command line by using a semicolon (`;`). The shell recognizes the semicolon as a command separator. The shell executes each command from left to right when you press Return. The command format for multiple commands is:

```
command option argument;command option argument
```

The following example shows two commands separated by a semicolon.

```
$ date;uname
Thu May 30 15:24:03 MDT 2002
SunOS
```

The following example shows three commands separated by a semicolon. The `cal` command has two arguments, followed by the `date` command, and the `uname` command with two options.

```
$ cal 05 2002;date;uname -rs
    May 2002
 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 29 30 31

Thu May 30 15:24:03 MDT 2002
SunOS 5.9
$
```

Control Characters

You can use special control characters on the keyboard to stop and start screen output, erase an entire command line, or stop the execution of a command. Table 2-2 shows the control characters.

Table 2-2 Control Characters

Control Characters	Purpose
Control-C	Terminates the command currently running
Control-D	Indicates end-of-file or exit
Control-U	Erases all characters on the current command line
Control-W	Erases the last word on the command line
Control-S	Stops output to the screen
Control-Q	Restarts output to the screen after you have pressed Control-S

To enter a sequence of control characters, hold down the Control key and press the appropriate character on the keyboard for the desired action.

The Control-S and Control-Q characters are rarely needed today because graphical user environments with terminals can scroll back to the previous output.

Using Online Documentation

The Solaris OE provides online manual pages, which describe commands and their usage. Additionally, online help resources provide general desktop information.

Displaying the Online Manual Pages

The online UNIX Reference Manual (man) pages provide detailed descriptions of UNIX commands and how to use them. Use the `man` command to display the man page entry that explains a given command.

The format for the `man` command is:

```
man command
man option command
man option filename
```

For example, to display the man pages for the `uname` command, enter the following:

```
$ man uname
Reformatting page.  Please Wait... done

User Commands                                          uname(1)

NAME
     uname - print name of current system

SYNOPSIS
     uname [ -aimnprsvX ]

     uname [ -S system_name ]

DESCRIPTION
     The uname utility prints information about the current  sys-
     tem on the standard output. When options are specified, sym-
     bols representing one or more system characteristics will be
     written to the standard output.  If no options are specified,
     uname  prints  the  current  operating  system’s  name.  The
     options  print  selected  information  returned by uname(2),
     sysinfo(2), or both.
<output omitted>
```

Scrolling in Man Pages

Table 2-3 shows the keys on the keyboard used to control the scrolling capabilities when you are in the man pages.

Table 2-3 Keys to Control Scrolling in Man Pages

Scrolling Keys	**Action**
Spacebar	Displays the next screen of a man page
Return	Displays the next line of a man page
`b`	Moves back one full screen
/pattern	Searches forward for a pattern
`n`	Finds the next occurrence of a pattern after you have used */pattern*
`h`	Provides a description of all scrolling capabilities
`q`	Quits the `man` command

Searching the Man Pages

There are two ways to search for information in the man pages. You can search for information by section or by keyword.

Searching Man Pages by Section

You can search within a specific section of the man pages by using the `man` command with the `-s` option.

Entries in the online manual pages are organized into sections based on the type or use of the command or file. For example, Section 1 contains user commands, while Section 4 contains information about various file formats.

You can use the `man intro` command to view descriptive information about sections contained in the man pages.

The format for looking up a specific section of the man pages is the `man` command with the `-s` option, the section number, and the command or file name.

For example:

```
man -s number command
```

or

```
man -s number filename
```

The last part of a man page, under `SEE ALSO`, lists other commands or files related to the man page. The number in parentheses reflects the section where the man page is located. You can use the `man` command with the `-l` option to see a list of man pages that relate to the same command or file name.

The following example shows the `SEE ALSO` part of the man page for `passwd(1)`. Note that there is also a man page entry for `passwd` in Section 4.

```
SEE ALSO
    finger(1),  login(1),  nispasswd(1),  nistbladm(1),
    yppasswd(1),  domainname(1M),  eeprom(1M),  id(1M),
    passmgmt(1M),  pwconv(1M),  su(1M),  useradd(1M),  userdel(1M),
    usermod(1M),    crypt(3C),    getpwnam(3C),    getspnam(3C),
    getusershell(3C),  nis_local_directory(3N),  pam(3),  login-
    log(4),  nsswitch.conf(4),  pam.conf(  4),  passwd(4),  sha-
    dow(4), attributes(5), environ(5), pam_unix(5)
```

To view the online man page for the passwd file information, enter the following commands:

```
$ man -l passwd
passwd (1)     -M /usr/man
passwd (4)     -M /usr/man

$ man -s 4 passwd
Reformatting page.  Please Wait... done

File Formats                                            passwd(4)

NAME
     passwd - password file

SYNOPSIS
     /etc/passwd

DESCRIPTION
     The file /etc/passwd is a local source of information about users'
     accounts.  The password file can be used in conjunction with
     other password sources, including the NIS maps passwd.byname and
     passwd.bygid and the NIS+ table  passwd. Programs use the
     getpwnam(3C) routines to access this information.
<output omitted>
```

Searching Man Pages by Keyword

When you are unsure of a command name, you can search for man page entries that are related by using the man command with the -k option and a keyword. The man command output provides a list of commands and descriptions that contain the keyword. The command format for searching with a keyword is:

```
man -k keyword
```

To view commands containing the keyword calendar, enter the following command:

```
$ man -k calendar
<output omitted>
cal cal (1)- display a calendar
calendarcalendar (1)- reminder service
difftimedifftime (3c)- computes the difference between two
        calendar times
mktimemktime (3c)- converts a tm structure to a calendar time
$
```

By default, the keyword lookup feature for searching man pages is not enabled. To enable this feature the `root` user must execute the `catman -w` command. See `catman (1M)` for more information.

```
#catman -w
#catman -w -M /usr/dt/man
#catman -w -M /usr/openwin/share/man
```

If the `catman -w` command has not been run, the user will see No such file or directory error messages when the `man -k` *keyword* command is executed.

Using CDE Help Resources

The CDE help utility is called the Help Manager. Clicking the Help icon on the front panel of your desktop launches the Help Manager. Figure 2-2 shows the Help icon as an image of books with a question mark superimposed.

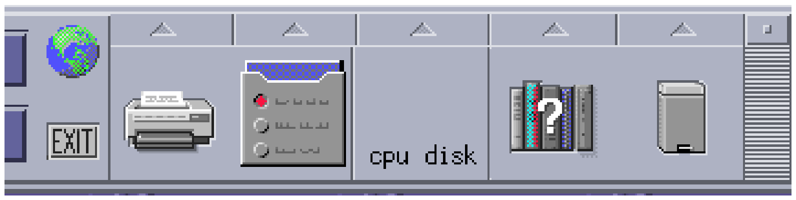
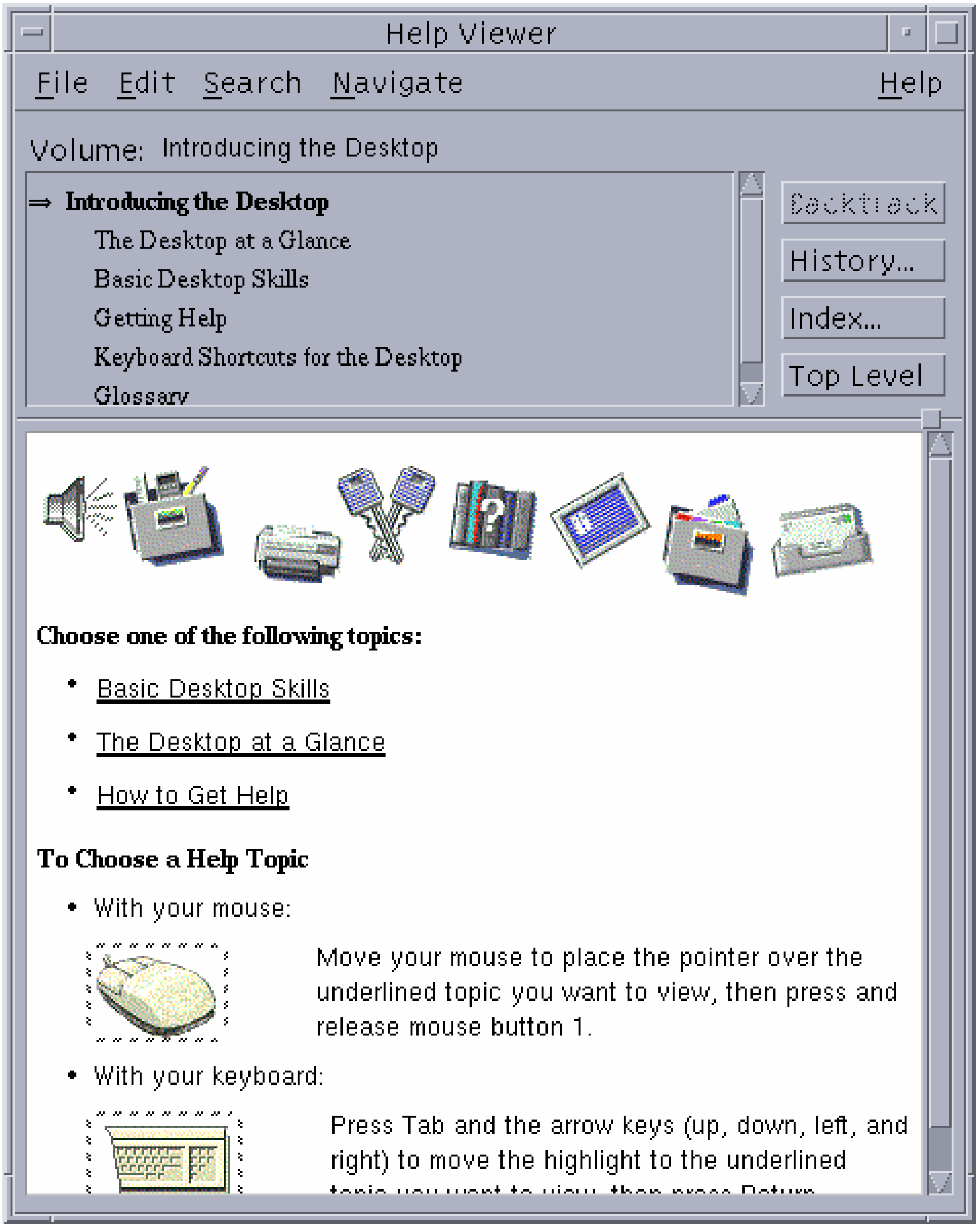


Figure 2-2 Help Icon

The Help Manager displays information in a window labeled Help Viewer. You can search for a specific desktop topic or list the entire index of information from the Help Viewer.

The first time you start a CDE session, the Help Viewer automatically displays the Introducing the Desktop topic.

Figure 2-3 shows this window.


Help Viewer
File Edit Search Navigate Help
Volume: Introducing the Desktop
⇒ Introducing the Desktop
The Desktop at a Glance
Basic Desktop Skills
Getting Help
Keyboard Shortcuts for the Desktop
Glossary
Backtrack
History...
Index...
Top Level

Choose one of the following topics:

- Basic Desktop Skills
- The Desktop at a Glance
- How to Get Help

To Choose a Help Topic

- With your mouse:

  Move your mouse to place the pointer over the underlined topic you want to view, then press and release mouse button 1.

- With your keyboard:

  Press Tab and the arrow keys (up, down, left, and right) to move the highlight to the underlined

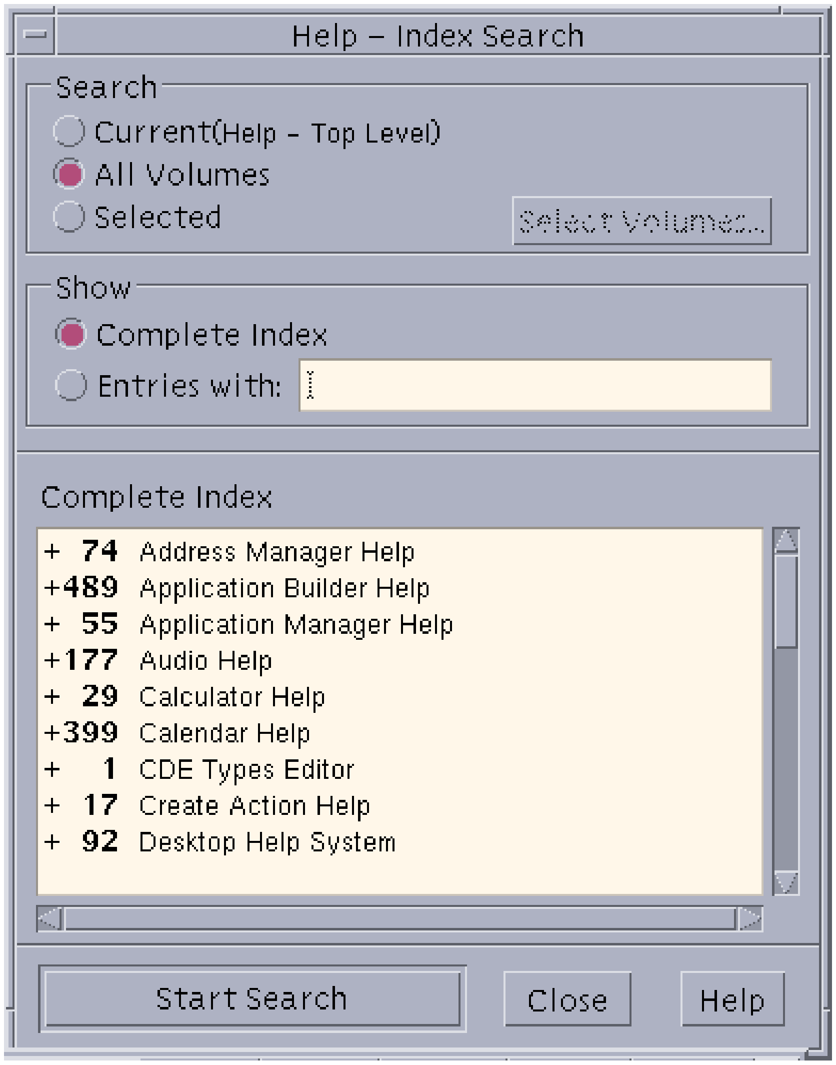
Figure 2-3 Introducing the Desktop Help Window

Note – The initial screen contents of the Help Viewer depend upon the manner in which you start the viewer. The contents of the initial screen differ from the above example when you manually start the Help Viewer.

Click Search in the Help Viewer window. Then select Index from the list to launch the Index Search window. You can search on a particular topic or view the entire index of all volumes.

To view the entire index of all volumes, select the All Volumes button on the Search panel, then select the Complete Index button on the Show panel.

Figure 2-4 shows this window.



Search: [✓ All Volumes]

Show: [✓ Complete Index]

Complete Index

```
+ 74  Address Manager Help
+489  Application Builder Help
+ 55  Application Manager Help
+177  Audio Help
+ 29  Calculator Help
+399  Calendar Help
+   1  CDE Types Editor
+ 17  Create Action Help
+ 92  Desktop Help System
```

Figure 2-4 Index Search Window

Using Online Product Documentation

To start a web browser in the CDE, click the clock icon located on the front panel. See Figure 2-5.

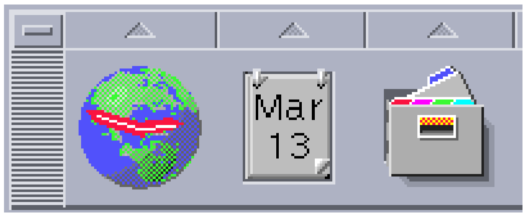


Figure 2-5 Clock Icon

To access comprehensive information on Sun products, visit `http://docs.sun.com` using your web browser. You can search by subject, collection title, product category, and keyword.

Figure 2-6 shows a portion of this web page.

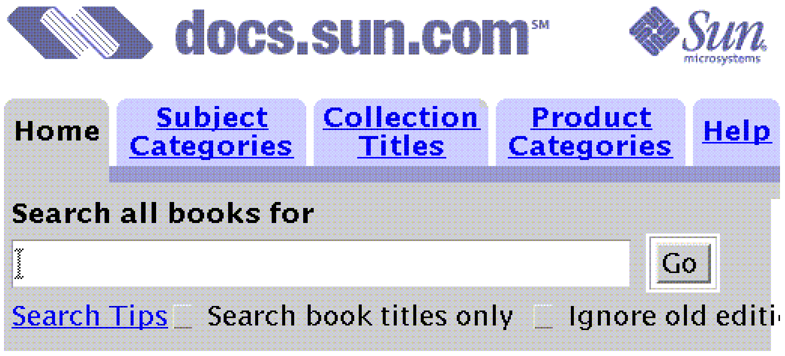


Figure 2-6 Online Product Documentation Site

Click one of the types of searches or type a keyword in the “Search all books for” area to access information.

Exercise: Using Command-Line Features

In this exercise, you complete the following tasks:

- Practice using command-line features
- Practice viewing online documentation

Tasks

Complete the following tasks.

Task 1 – Using Command-Line Features

Complete the following steps to use command-line features.

1. At the CDE Login screen, enter your user name and password, and start a CDE session.
2. Click the right mouse button in the CDE desktop background.
3. Open a terminal window from the Tools section of the Workspace Menu.
4. Name the two components of command-line syntax that can enhance the capability of a command.

   ____

5. What command displays information about the operating system and the workstation name?

   ____

   Execute the `man uname` command, and determine what the options `-s` and `-r` do.

   a. Enter the `uname -s` command. What information appears?

      ____

   b. Enter the `uname -r` command. What information appears?

      ____

6. Display the current time and date on your system. What command did you enter?

   ____

7. Display this month's calendar. What command did you enter?

   ____

8. What special character is used to separate commands on a single command line?

   ____

9. Clear the terminal window. What command did you enter?

   ____

Task 2 – Using Online Documentation

Complete the following steps to access online documentation.

1. Display the man page for the `passwd` command.
   a. Execute the `man passwd` command. The `man` command displays the `passwd` entry for Section 1 of the man pages to locate information on the `passwd(1)` command.
   b. Move to the end of this man page by searching for the word `SEE`. Enter `/SEE` (the forward slash [`/`] character and the word `SEE` are a search pattern).
   c. Execute the `man -s4 passwd` command. The `man` command searches Section 4 of the man pages, the File Formats section, for an entry for `passwd`.
2. Using the `-k` option with the `man` command, find the man page that describes how to clear a terminal window (use the keyword `clear`). How would this command be entered on the command line?

   ____

3. Clear the terminal window.
4. Display the Help Viewer. Click the Help icon located on the Front Panel. There is a question mark on the icon.
5. Display the Index Search window from the Help Viewer window.
   a. Select the Search menu option at the top of the window.
   b. Select Index from either the Search drop-down menu or the Index button located on the viewer screen.

6. Display the entire index of all volumes in Help Manager.
   a. Select the All Volumes button in the Search panel.
   b. Select the Complete Index button in the Search panel.
   c. Select one of the topics listed in the index window to see the help content displayed in the Help Viewer window.
   d. Close the window.
7. What web site enables you to browse Sun product documentation?

   ____

Exercise Summary

Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

Manage the discussion based on the time allowed for this module, which was provided in the "About This Course" module. If you do not have time to spend on discussion, then just highlight the key concepts students should have learned from the lab exercise.

- Experiences

Ask students what their overall experiences with this exercise have been. Go over any trouble spots or especially confusing areas at this time.

- Interpretations

Ask students to interpret what they observed during any aspect of this exercise.

- Conclusions

Have students articulate any conclusions they reached as a result of this exercise experience.

- Applications

Explore with students how they might apply what they learned in this exercise to situations at their workplace.

Exercise Solutions

Task 1 – Using Command-Line Features

Complete the following steps to use command-line features.

4. Name the two components of command-line syntax that can enhance the capability of a command.

   Options and arguments

5. What command displays information about the operating system and the workstation name?

   ```
   $ uname -a
   ```

   a. Enter the `uname -s` command. What information appears?

      The operating system name, for example, SunOS

   b. Enter the `uname -r` command. What information appears?

      The operating system release level, for example, 5.9

6. Display the current time and date on your system. What command did you enter?

   ```
   $ date
   ```

7. Display this month's calendar. What command did you enter?

   ```
   $ cal
   ```

8. What special character is used to separate commands on a single command line?

   The semicolon (`;`)

9. Clear the terminal window. What command did you enter?

   ```
   $ clear
   ```

Task 2 – Using Online Documentation

Complete the following steps to access online documentation.

2. Using the `-k` option with the `man` command, find the man page that describes how to clear a terminal window (use the keyword `clear`). How would this command be entered on the command line?

   ```
   $ man -k clear
   ```

3. Clear the terminal window.

   ```
   $ clear
   ```

7. What web site enables you to browse Sun product documentation?

   http://docs.sun.com

Module 3

Viewing Directories and Files

Objectives

Upon completion of this module, you should be able to:

- Work with directories
- Work with files
- Print files

The following course map shows how this module fits into the current instructional goal.

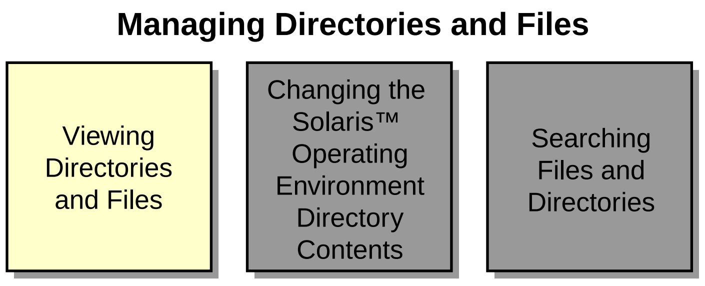

Managing Directories and Files
Viewing Directories and Files
Changing the Solaris™ Operating Environment Directory Contents
Searching Files and Directories

Figure 3-1 Course Map

Working With Directories

There are different commands that enable you to work with directories. Use these commands to display the current directory, view the contents of a directory, and change directories.

Determining the Current Directory

The `pwd` command identifies the directory that you are currently accessing.

To display the current working directory, enter the `pwd` command.

```
$ pwd
/export/home/user1
```

Displaying the Directory Contents

Use the `ls` command to display the contents of a directory. Use the `ls` command without arguments to list the files and directories within the specified directory.

The format for the `ls` command is:

```
ls -options pathname
```

To list the contents of your directory, enter the `ls` command.

```
$ ls
dante     dir2     dir5     file.2   file3    fruit2
dante_1   dir3     file.1   file2    file4    practice
dir1      dir4     file1    file.3   fruit    tutor.vi
```

Figure 3-2 shows a graphical representation of the files and directories in the `/export/home/user1` directory.

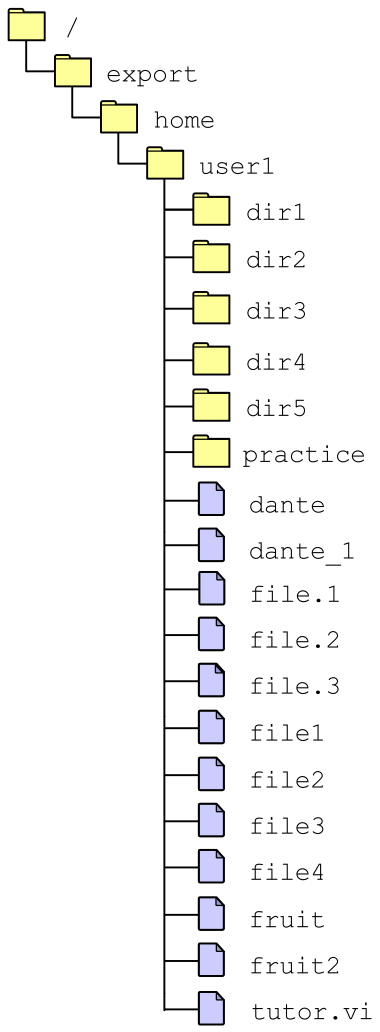


Figure 3-2 Directory List

To display the contents of a specific directory within the current working directory, enter the `ls` command with a directory name.

```
$ ls dir1
coffees   fruit   trees
```

To display the contents of a directory that is not in the current working directory, enter the `ls` command with a path name.

```
$ ls /var/mail
:saved user1
```

Displaying Hidden Files

Some files are hidden from view when you use the `ls` command. Hidden files often contain information that customizes your work environment. Use the `ls -a` command to list all files in a directory, including hidden files. The file names of hidden files begin with a period (`.`).

To display hidden files, enter the `ls -a` command.

```
$ ls -a
.             dir2          .dtprofile    file.3        practice
..            dir3          file.1        file3         .sh_history
dante         dir4          file1         file4         .solregis
dante_1       dir5          file.2        fruit         .TTauthority
dir1          .dt           file2         fruit2        tutor.vi
```

The single period (`.`) represents the current working directory. The double period (`..`) represents the parent directory, which contains the working directory.

Displaying a Long List

Use the `ls -l` command to view detailed information about the contents of a directory.

Figure 3-3 shows the information provided by the `ls -l` command.

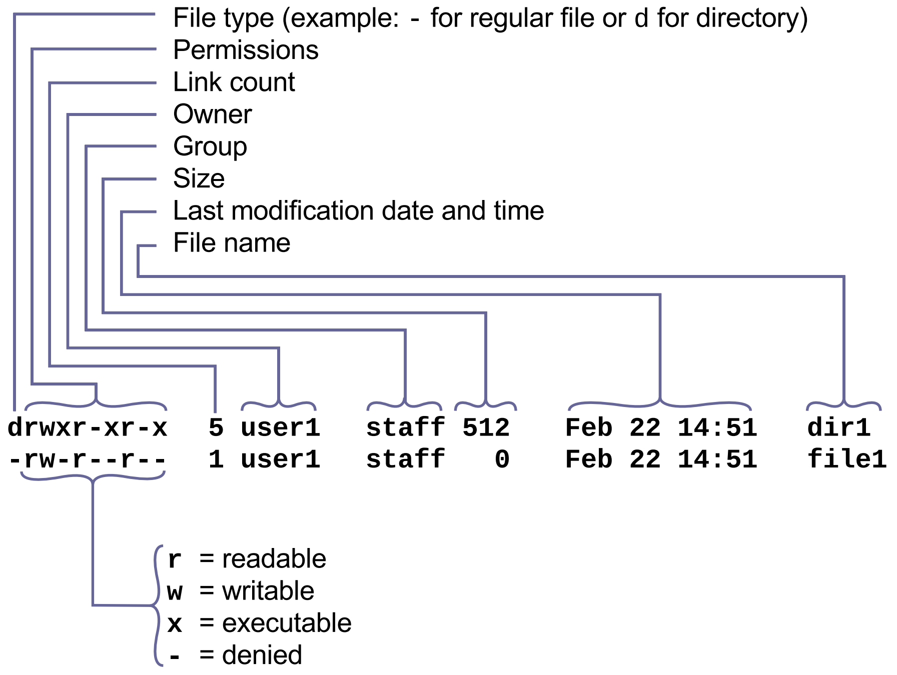

File type (example: - for regular file or d for directory)
Permissions
Link count
Owner
Group
Size
Last modification date and time
File name
drwxr-xr-x 5 user1 staff 512 Feb 22 14:51 dir1
-rw-r--r-- 1 user1 staff 0 Feb 22 14:51 file1
r = readable
w = writable
x = executable
- = denied

Figure 3-3 Long List File Information

The following is an example of the `ls -l` command.

```
$ ls -l
total 90
-rw-rw----   1 user1    staff      1319 Mar 14 10:12 dante
-rw-rw----   1 user1    staff       368 Mar 14 10:12 dante_1
drwxr-xr-x   5 user1    staff       512 Mar 14 10:12 dir1
drwxr-xr-x   4 user1    staff       512 Mar 14 10:12 dir2
drwxr-xr-x   3 user1    staff       512 Mar 14 10:12 dir3
drwxr-xr-x   2 user1    staff       512 Mar 14 10:12 dir4
drwxr-xr-x   2 user1    staff       512 Mar 14 10:12 dir5
-rw-rw----   1 user1    staff         0 Mar 14 10:12 file.1
-rw-rw----   1 user1    staff      1610 Mar 14 10:12 file1
-rw-rw----   1 user1    staff         0 Mar 14 10:12 file.2
-rw-rw----   1 user1    staff       105 Mar 14 10:12 file2
-rw-rw----   1 user1    staff         0 Mar 14 10:12 file.3
-rw-rw----   1 user1    staff       218 Mar 14 10:12 file3
-rw-rw----   1 user1    staff       137 Mar 14 10:12 file4
-rw-rw----   1 user1    staff        57 Mar 14 10:12 fruit
-rw-rw----   1 user1    staff        57 Mar 14 10:12 fruit2
drwxr-xr-x   2 user1    staff       512 Mar 14 10:12 practice
-rwx--x--x   1 user1    staff     28738 Mar 14 10:12 tutor.vi
```

The line containing `total 90` at the beginning of the output indicates the number of 512-byte blocks used at this level of the directory structure.

To view detailed information on the contents of the `dir1` directory, enter `ls -l dir1`.

```
$ ls -l dir1
total 6
drwxr-xr-x   2 user1    staff          512 Feb 22 14:51 coffees
drwxr-xr-x   2 user1    staff          512 Feb 22 14:51 fruit
drwxr-xr-x   2 user1    staff          512 Feb 22 14:51 trees
```

Figure 3-4 shows the list of directories and files in the `dir1` directory.

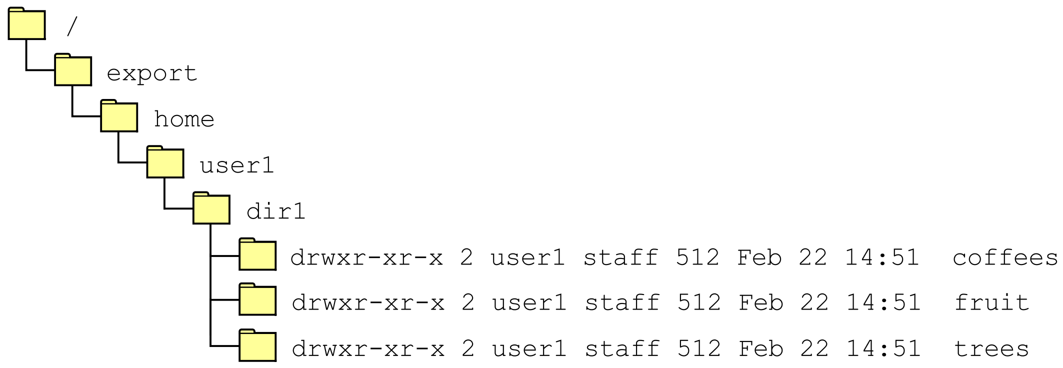

/
export
home
user1
dir1
drwxr-xr-x 2 user1 staff 512 Feb 22 14:51 coffees
drwxr-xr-x 2 user1 staff 512 Feb 22 14:51 fruit
drwxr-xr-x 2 user1 staff 512 Feb 22 14:51 trees

Figure 3-4 Long List

Displaying Individual Directories

Use the `ls -ld` command to display detailed information for the directory only, not its contents.

To obtain detailed directory information for the `dir1` directory, enter the `ls -ld` command.

```
$ ls -ld dir1
drwxr-xr-x 5 user1 staff 512 Feb 22 14:51  dir1
$
```

Displaying a Recursive List

Use the `ls -R` command to display the contents of a directory and the contents of all of the directory's subdirectories. This type of list is known as a recursive list.

To view a recursive list of the contents of the `dir1` directory, enter `ls -R dir1`.

```
$ ls -R dir1
dir1:
coffees  fruit    trees

dir1/coffees:
beans    brands    nuts

dir1/coffees/beans:
beans

dir1/fruit:

dir1/trees:
```

Displaying File Types

There are two ways to display file types. You can use the `ls -F` command or you can use the `file` command.

Using the `ls -F` Command

Use the `ls -F` command to display file types. Table 3-1 shows the symbols used in the `ls -F` output.

Table 3-1 File Type Symbols

Symbol	**File Type**
/	Directory
*	Executable
(none)	Plain text file/ASCII
@	Symbolic link

The following is an example of the `ls -F` command.

```
$ ls -F
dante      dir2/      dir5/      file.2   file3    fruit2
dante_1    dir3/      file.1     file2    file4    practice/
dir1/      dir4/      file1      file.3   fruit    tutor.vi*
$
```

Note – A symbolic link is a special type of file that points to another file or directory.

Using the `file` Command

Use the `file` command to determine certain file types. Knowing the file type helps you decide which command or program you need to use to read the file.

The output from the `file` command is typically one of the following:

- Text – Text files include American Standard Code for Information Interchange (ASCII) text, English text, command text, and executable shell scripts.
- Data – Data files are created by programs. The `file` command indicates the type of data file, such as a FrameMaker document, if the type is known. The `file` command indicates that the file is a data file if the type is unknown.
- Executable or binary – Executable files include 32-bit executable and extensible linking format (ELF) code files and other dynamically linked executable files. Executable files are commands or programs.

The format for the `file` command is:

```
file filename(s)
```

To view the file type for the `dante` file, enter the `file` command with a file name.

```
$ file dante
dante:          English text
```

Changing Directories

When working within the directory hierarchy, you always have a current working directory. When you initially log in to the system, the current directory is set to your home directory. You can change your current working directory at any time by using the `cd` command.

The format for the `cd` command is:

```
cd directory
```

The `cd` command entered on a command line by itself changes the working directory to your home directory.

To change directories from the `user1` directory to the `dir1` directory, enter the following:

```
$ cd dir1
$ pwd
/export/home/user1/dir1
```

Using Path Name Abbreviations

Use path name abbreviations as a quick method for moving to or referring to directories on the command line. Table 3-2 shows the path name abbreviations.

Table 3-2 Path Name Abbreviations

Symbol	**Path Name**
`.`	Current or working directory
`..`	Parent directory, the directory directly above the current working directory

Directories always contain a link to their parent directory (`..`) and a link to themselves (`.`).

Enter the `pwd` command to confirm the current working directory. To move to the parent directory for `dir1`, enter the `cd ..` command.

```
$ pwd
/export/home/user1/dir1
$ cd ..
$ pwd
/export/home/user1
```

Figure 3-5 shows the `dir1` directory and its parent directory, the `user1` directory.

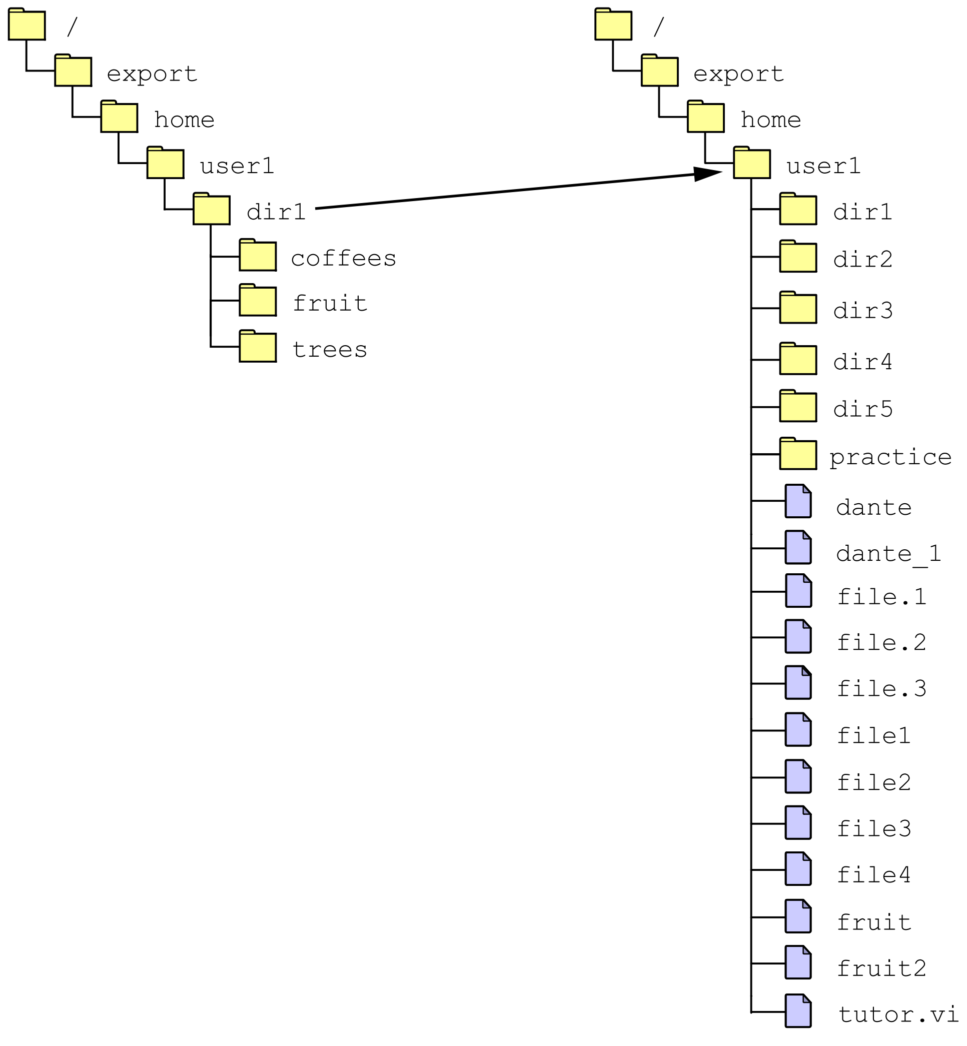


Figure 3-5 The `cd ..` Command

To move to the `/` (root) directory from the `/export/home/user1` directory, enter the `cd ..` command and append a slash (`/`) with the symbol (`..`) for each directory in the current hierarchy above the `/home/user1` directory.

```
$ cd ../../..
$ pwd
/
$
```

Moving Around the Directory Hierarchy

There are two ways to move around the directory hierarchy. You can use an absolute path name or you can use a relative path name. An absolute path name lists all the directories in the path. A relative path name lists the directories in the path relative to the current directory.

```
$ cd
$ cd dir1
$ pwd
/export/home/user1/dir1
$ cd ../dir2
$ pwd
/export/home/user1/dir2
$ cd
```

To change directories using a relative path name, enter the `cd` command with the path name that starts from the current working directory, `user1`.

```
$ cd dir1/coffees
```

To change directories using an absolute path name, enter the `cd` command with the complete path name from the `/` (root) directory.

```
$ cd /export/home/user1/dir1/coffees
```

Returning to Your Home Directory

The home directory of a regular user is where the user is placed after login for creating and storing files.

Note – The directory named `/home` is the default directory used to hold the home directories of regular users.

Often a user's home directory name is the same as the user's login name. For example, if your user login name is user1, your home directory would be `/home/user1`.

You can return to your home directory from any directory by entering the `cd` command without an argument.

```
$ cd
$ pwd
/export/home/user1
```

You can also use the `cd` command with the absolute path name to return to your home directory.

```
$ cd /export/home/user1
```

To move to a user's home directory, enter the `cd` command with a tilde (~) character in front of the user name. The tilde (~) character is an abbreviation of the absolute path name.

```
$ cd ~user1
```

You can also use the tilde (~) to represent your home directory in a relative path. The tilde (~) in the following example represents the `user1` home directory.

```
$ cd ~/dir1/fruit
```

Working With Files

There are different commands available that enable you to view file content in a read-only format or to display information about a file. These include the `cat` command, the `more` command, the `tail` command, and the `wc` command.

Viewing Files Using the cat Command

The `cat` command displays the contents of one or more text files on the screen. The `cat` command displays the entire contents of the file without pausing.

```
cat filename
```

To display the short text file named `dante`, enter the `cat` command.

```
$ cat dante
                    The Life and Times of Dante

                         by Dante Pocai


Mention “Alighieri” and few may know about whom you are talking. Say
“Dante,” instead, and the whole world knows whom you mean. For
Dante Alighieri, like Raphael, Michelangelo, Galileo, etc. is usually
referred to by his first name.
<output omitted>
```

Note – Do not use the `cat` command to read binary files. Using the `cat` command to read binary files can cause a terminal window to freeze. If your terminal window freezes, close the terminal window, and open a new terminal window.

Viewing Files Using the more Command

To view or page through the contents of a long text file, use the `more` command. The `more` command displays the contents of a text file one screen at a time. The following message appears at the bottom of each screen.

```
--More--(n%)
```

The *n%* is the percentage of the file already displayed.

When the `--More--(`*n%*`)` prompt appears, use the keys described in Table 3-3 to scroll through the file.

Table 3-3 Scrolling Keys for the `more` Command

Scrolling Keys	**Action**
Spacebar	Moves forward one screen
Return	Scrolls one line at a time
`b`	Moves back one screen
`h`	Displays a help menu of features
/string	Searches forward for *pattern*
`n`	Finds the next occurrence of *pattern*
`q`	Quits and returns to the shell prompt

When the entire file has been displayed, the shell prompt appears.

Caution – The online man pages use the `more` command to display information. Do not use the `more` command to read binary files. Using the `more` command to read binary files can cause a terminal window to freeze. If your terminal window freezes, close the terminal window and open a new terminal window.

The format for the `more` command is:

```
more filename(s)
```

To display the first screen of the `dante` file, enter the `more` command.

```
$ more dante
                       The Life and Times of Dante

                          by Dante Pocai


Mention "Alighieri" and few may know about whom you are talking.  Say
"Dante," instead, and the whole world knows whom you mean.  For
Dante Alighieri, like Raphael, Michelangelo, Galileo, etc. is usually
referred to by his first name.  There is only one Dante, as we recognize
one Raphael, one Michelangelo, and one Galileo.
```

```
Who is this Dante, whom T.S. Eliot calls "the most universal of poets
in the modern languages?"


YOUTH

Exact details about his youth are few indeed.  He was born in the city
of Florence, Italy, in May of 1265.  His family was of noble origin
by modest means and modest social standing.  His great grandfather died
in the Second Crusade in the Holy Land.  Dante was a thoughtful, quiet,
rather sad young man.  His mother, whose name was Bella (beautiful) died
while he was still a child.  His father remarried a certain Lapa who
didn’t shower too much love and affection on her stepson.

About Dante’s personal appearance, his friend Boccaccio wrote, "Our poet
was of medium height.  He had a long face, an aquiline nose, large jaws,
and his lower lip was more prominent than the upper.  He was slightly
--More--(90%)
```

Viewing File Content Using the head Command

The head command displays the first 10 lines of a file. You can change the number of lines displayed by using the -*n* option. The -*n* option displays *n* lines starting at the beginning of the file.

The format for the head command is:

```
head -n filename(s)
```

To display the first six lines of the /usr/dict/words file, enter the head command with the -*n* option.

```
$ head -6 /usr/dict/words
10th
1st
2nd
3rd
4th
5th
$
```

Viewing File Content Using the `tail` Command

The `tail` command displays the last 10 lines of a file. You can change the number of lines displayed by using the `-`*`n`* or `+`*`n`* options. The `-`*`n`* option displays *`n`* lines from the end of the file. The `+`*`n`* option displays from line *`n`* to the end of the file.

The format for the `tail` command is:

```
tail -n filename
```

or

```
tail +n filename
```

To display the last five lines of the `/usr/dict/words` file, enter the `tail` command with the `-`*`n`* option.

```
$ tail -5 /usr/dict/words
zounds
z’s
zucchini
Zurich
zygote
$
```

To display line 25136 through the end of the `/usr/dict/words` file, enter the `tail` command with the `+`*`n`* option.

```
$ tail +25136 /usr/dict/words
Zorn
Zoroaster
Zoroastrian
zounds
z’s
zucchini
Zurich
zygote
```

Displaying Line, Word, and Character Counts

The `wc` command displays the number of lines, words, and characters contained in a file.

The format for the `wc` command is:

```
wc -option(s) filename(s)
```

Table 3-4 shows the options you can use with the `wc` command.

Table 3-4 Options for the `wc` Command

Option	Description
`-l`	The line count
`-w`	The word count
`-c`	The byte count
`-m`	The character count

Using the `wc` command without options displays the number of lines, words, and characters contained in the file.

To display the number of lines, words, and characters in the `dante` file, enter the `wc` command.

```
$ wc dante
32     223    1319  dante
$
```

To display the number of lines in the `dante` file, enter the `wc` command with the `-l` option.

```
$ wc -l dante
32  dante
$
```

Printing Files

There are different commands available that allow you to submit print requests to a destination, to display the status of all user print requests, and to cancel print requests. These commands include the `lp` command, the `lpstat` command, and the `cancel` command.

Using the `lp` Command

The `lp` command queues text files for printing.

From the command line, you can print ASCII text or PostScript™ technology files. Do not use this method to print data files, such as binary files or files created in programs such as FrameMaker.

The format for the `lp` command is:

```
lp option(s) filename(s)
```

Table 3-5 shows the options you can use with the `lp` command.

Table 3-5 Options for the `lp` Command

Option	**Description**
`-d` *`destination`*	Specifies the desired printer. The `-d` *`destination`* option is not necessary if printing to the default printer.
`-o nobanner`	Specifies that the banner page is not to be printed.
`-n` *`number`*	Prints a specified number of file copies.
`-m`	Sends a mail message to you after a job is complete.

To print the `dante` file located in your home directory on the default printer, enter the `lp` command without options.

```
$ lp ~/dante
request id is printerA-4 (1 file(s))
$
```

To print to the `printerB` printer, which is not the default printer, enter the `lp` command with the `-d` *`destination`* option.

```
$ lp -d printerB ~/dante
request id is printerB-2 (1 file(s))
$
```

Using the `lpstat` Command

The `lpstat` command displays the status of the printer queue.

The format for the `lpstat` command is:

```
lpstat -option(s) printer
```

Table 3-6 shows the options for the `lpstat` command.

Table 3-6 Options for the `lpstat` Command

Option	**Description**
`-p`	Displays the status of all printers
`-o`	Displays the status of all output requests
`-d`	Displays the system default printer
`-t`	Displays complete status information for all printers
`-s`	Displays a status summary for all printers
`-a`	Displays which printers are accepting requests

To display the status of all print requests, enter the `lpstat` command with the `-o` option.

```
$ lpstat -o
printerA-5 user1391 Mar 11 16:30 on printerA
printerB-3user2551Mar 11 16:45 filtered
$
```

Table 3-7 shows the information in the status response.

Table 3-7 Status Information for the `lpstat` Command

Status	**Definition**
`Request-ID`	The name of the printer and job number
`User-ID`	The name of the user accessing the printer
`File Size`	The output size in bytes
`Date/Time`	The current date and time
`Status`	The status of the print request

To display requests in the queue for `printerB`, enter the `lpstat` command with the printer name as the argument.

```
$ lpstat printerB
printerB-3 user2 551 Mar 11 16:45 on printerB
$
```

To determine the status of all configured printers, enter the `lpstat` command with the `-t` option.

```
$ lpstat -t
scheduler is running
system default destination: printerA
system for printerB: host2
system for _default: host1 (as printer printerA)
system for printerA: host1
printerB accepting requests since Mar 7 09:43 2002
_default accepting requests since Mar 8 08:20 2002
printerA accepting requests since Mar 8 08:20 2002
printer printerB is idle. enabled since Mar 7 09:43 2002. available.
printer _default is idle. enabled since Mar 8 08:20 2002. available.
printer printerA is idle. enabled since Mar 8 08:20 2002. available.
$
```

To determine which printers are configured on the system, enter the `lpstat` command with the `-s` option.

```
$ lpstat -s
scheduler is running
system default destination: printerA
system for printerB: host2
system for _default: host1 (as printer printerA)
system for printerA: host1
$
```

To display which printers are accepting requests, enter the `lpstat` command with the `-a` option.

```
$ lpstat -a
printerB accepting requests since Mar 7 09:43 2002
_default accepting requests since Mar 8 08:20 2002
printerA accepting requests since Mar 8 08:20 2002
$
```

Using the `cancel` Command

The `cancel` command enables you to cancel print requests previously sent with the `lp` command. Use the `lpstat` command to identify the *`printer`* associated with your print request.

The format for the `lp cancel` command is:

```
cancel Request-ID
```

or

```
cancel -u username
```

To identify the *`Request-ID`* for your print request, enter the `lpstat` *`Request-ID`* command.

```
$ lpstat printerB
printerB-3 user3 630            Mar 11 16:45
printerB-4 user3 631            Mar 11 16:45
printerB-5 user3 632            Mar 11 16:47
```

To cancel the print request, enter the `cancel` command with the *`Request-ID`* argument.

```
$ cancel printerB-5
printerB-5: cancelled
$
```

To remove all requests made by `user3`, enter the `cancel -u` *`username`* command.

```
$ cancel -u user3
printerB-3: cancelled
printerB-4: cancelled
$
```

As the `root` user, you can cancel all print requests owned by all users.

If you are not a `root` user, you can only remove your own print jobs.

When using the Common Desktop Environment (CDE) Printer Manager, it appears that you can cancel another user's print job, but the job is re-displayed in the Print Manager the next time the Print Manager is refreshed.

Exercise: Accessing Files and Directories

In this exercise, you use the commands described in this module to list and change directories.

Tasks

Complete the following steps. Write the commands that you use to perform each task in the space provided.

1. Display your current working directory.

   ____

2. Change to your home directory.

   ____

3. Display the contents of your current working directory.

   ____

4. Display all files, including any hidden files.

   ____

5. Display a long list of the contents of the current working directory.

   ____

6. Display the file types in your current working directory.

   ____

7. Change to the `dir1` directory.

   ____

8. Change to the `fruit` directory.

   ____

9. Change to the `planets` directory.

   ____

10. Return to your home directory.

    ____

11. Change to the `/etc` directory.

    ____

12. Return to your home directory.

    ____

13. Display the contents of the `fruit` file using the `cat` command.

    ____

14. Display the contents of the `fruit` file and the `fruit2` file using a single command.

    ____

15. Display on the screen the first five lines of the `/usr/dict/words` file.

    ____

16. Display on the screen the last eight lines of the `/usr/dict/words` file.

    ____

17. Determine the total number of lines contained in the `/usr/dict/words` file.

    ____

18. Print the `dante_1` file to the default printer.

    ____

Exercise Summary

Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

Manage the discussion based on the time allowed for this module, which was provided in the "About This Course" module. If you do not have time to spend on discussion, then just highlight the key concepts students should have learned from the lab exercise.

- Experiences

Ask students what their overall experiences with this exercise have been. Go over any trouble spots or especially confusing areas at this time.

- Interpretations

Ask students to interpret what they observed during any aspect of this exercise.

- Conclusions

Have students articulate any conclusions they reached as a result of this exercise experience.

- Applications

Explore with students how they might apply what they learned in this exercise to situations at their workplace.

Exercise Solutions

1. Display your current working directory.

```
$ pwd
```

2. Change to your home directory.

```
$ cd
```

or

```
$ cd ~username
```

3. Display the contents of your current working directory.

```
$ ls
```

or

```
$ ls -R
```

4. Display all files, including any hidden files.

```
$ ls -a
```

5. Display a long list of the contents of the current working directory.

```
$ ls -l
```

6. Display the file types in your current working directory.

```
$ ls -F
```

or

```
$ file *
```

7. Change to the `dir1` directory.

```
$ cd dir1
```

or

```
$ cd ~/dir1
```

8. Change to the `fruit` directory.

```
$ cd fruit
```

or

```
$ cd ~/dir1/fruit
```

9. Change to the `planets` directory.

   ```
   $ cd ../../dir3/planets
   ```

   or

   ```
   $ cd ~/dir3/planets
   ```

10. Return to your home directory.

    ```
    $ cd
    ```

    or

    ```
    $ cd ~username
    ```

11. Change to the `/etc` directory.

    ```
    $ cd /etc
    ```

12. Return to your home directory.

    ```
    $ cd
    ```

    or

    ```
    $ cd ~
    ```

13. Display the contents of the `fruit` file using the `cat` command.

    ```
    $ cat fruit
    ```

14. Display the contents of the `fruit` file and the `fruit2` file using a single command.

    ```
    $ cat fruit fruit2
    ```

15. Display on the screen the first five lines of the `/usr/dict/words` file.

    ```
    $ head -5 /usr/dict/words
    ```

16. Display on the screen the last eight lines of the `/usr/dict/words` file.

    ```
    $ tail -8 /usr/dict/words
    ```

17. Determine the total number of lines contained in the `/usr/dict/words` file.

    ```
    $ wc -l /usr/dict/words
    ```

18. Print the `dante_1` file to the default printer.

    ```
    $ lp dante_1
    ```

Module 4

Changing the Solaris™ Operating Environment Directory Contents

Objectives

Upon completion of this module, you should be able to:

- Copy files and directories
- Move files and directories
- Create files and directories
- Rename files and directories
- Remove files and directories
- Use symbolic links

The following course map shows how this module fits into the current instructional goal.

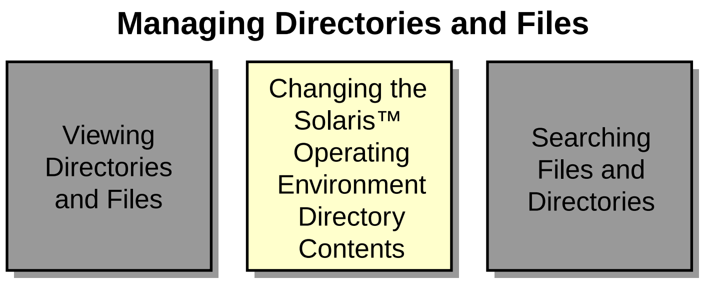

Managing Directories and Files
Viewing Directories and Files
Changing the Solaris™ Operating Environment Directory Contents
Searching Files and Directories

Figure 4-1 Course Map

Copying Files and Directories

You can copy a file or a directory from one place to another using the `cp` command. The `cp` command copies the requested files to the file or directory listed as the last argument.

Copying Files

Use the `cp` command to copy the contents of a file to another file. You can also use the `cp` command to copy multiple files. You can use the `cp` command with options and modify the execution of the command. For example, use the `-i` (interactive) option to prevent overwriting existing files when copying files. When you execute the `cp` command with the `-i` option, it prompts you for confirmation when the copy would overwrite an existing target.

The format for the `cp` command when copying files is:

```
cp -option source(s) target
```

The source is a file. The target is either a file or a directory.

Copying a File Within a Directory

To copy a file to a new file name in the same directory, use the `cp` command with the source file name and the destination file name.

The following example uses the `cp` command to copy the `file3` file to a new file named `feathers`, all within the `user1` directory.

```
$ cd
$ pwd
/export/home/user1
$ ls
dante     dir2      dir5      file.2    file3     fruit2
dante_1   dir3      file.1    file2     file4     practice
dir1      dir4      file1     file.3    fruit     tutor.vi
$ cp file3 feathers
$ ls
dante     dir2      dir5      file1     file.3    fruit     tutor.vi
dante_1   dir3      feathers  file.2    file3     fruit2
dir1      dir4      file.1    file2     file4     practice
```

The following example uses the `cp` command to copy the `feathers` file to a new file named `feathers_6`, all within the `user1` directory.

```
$ cp feathers feathers_6
$ ls
dante       dir3        feathers_6  file2       fruit
dante_1     dir4        file.1      file.3      fruit2
dir1        dir5        file1       file3       practice
dir2        feathers    file.2      file4       tutor.vi
```

Copying Multiple Files

To copy multiple files to a different directory, use the `cp` command with multiple file names for the source and a single directory name for the destination.

The following example uses the `cp` command to copy the `feathers` file and the `feathers_6` file from the `user1` directory into the `dir1` subdirectory.

```
$ pwd
/export/home/user1
$ ls dir1
coffees  fruit    trees
$ cp feathers feathers_6 dir1
$ ls dir1
coffees      feathers      feathers_6     fruit      trees
```

Figure 4-2 shows the `feathers` file and the `feathers_6` file being copied to the `dir1` directory.

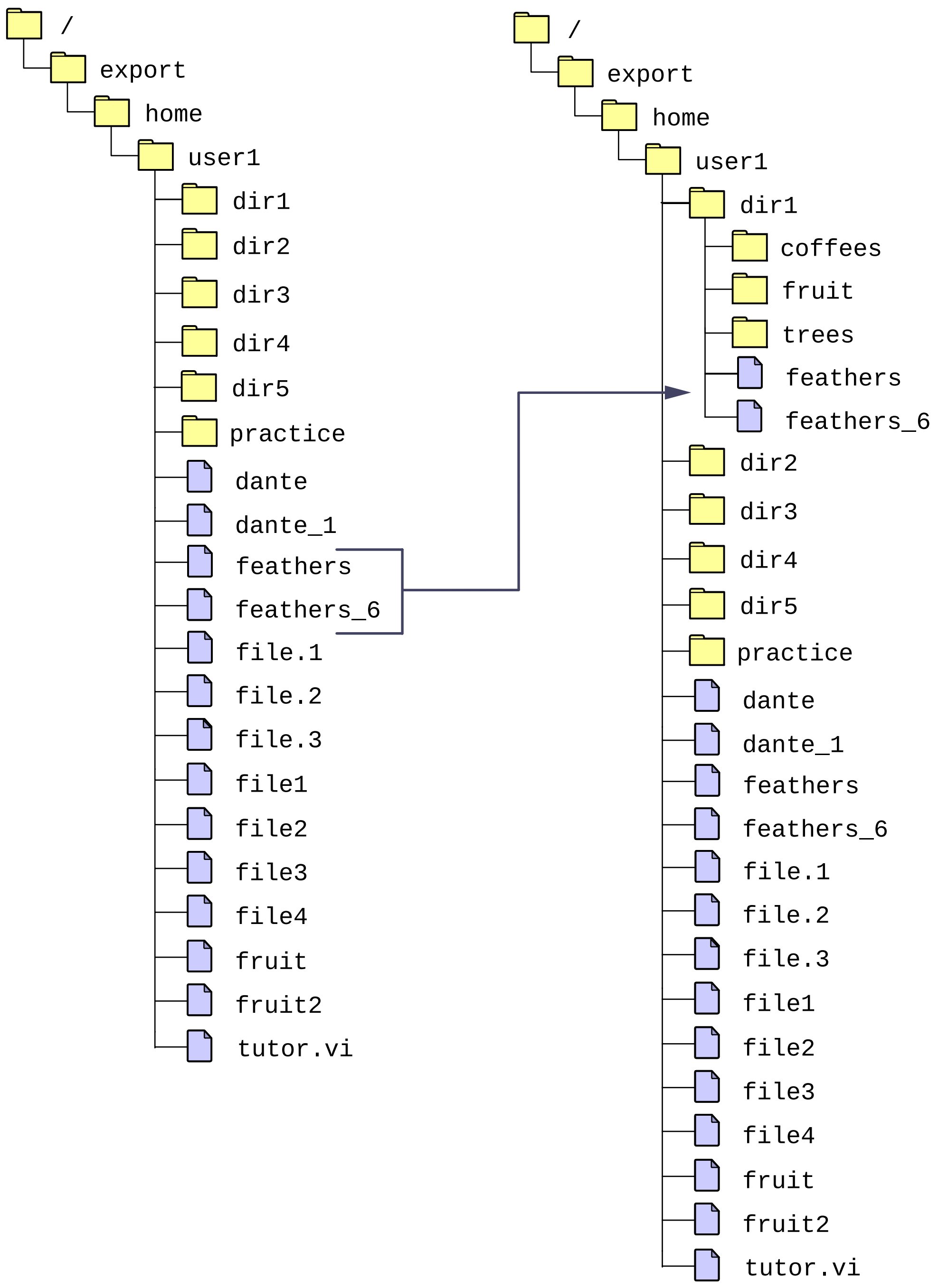


Figure 4-2 Copying Multiple Files

Preventing Overwrites to Existing Files While Copying

Use the `cp` command with the `-i` option to prevent overwriting existing files when copying either a single file or multiple files. When you use the `-i` option, a prompt appears requesting confirmation of the action if copying a file will overwrite an existing file.

- A `yes` response allows the overwrite.
- A `no` response prevents the `cp` command from overwriting the destination.

The following example uses the `cp -i` command to copy the `feathers` file to the `feathers_6` file. The `feathers_6` file already exists so you see the overwrite prompt.

```
$ cp -i feathers feathers_6
cp: overwrite feathers_6 (yes/no)? y
$
```

Copying Directories

Use the `cp` command with the `-r` option to copy a directory and its contents to another directory. If the destination directory does not exist, the `cp` command creates a new directory with that name. If you do not use the `-r` option, you receive the error message:

```
cp: directoryname: is a directory.
```

The format for the `cp` command when copying directories is:

```
cp -option source(s) destination
```

The source is one or more directory names. The destination is a single directory name. Table 4-1 shows the options you can use with the `cp` command when you are copying directories.

Table 4-1 Options for the `cp` Command

Option	**Description**
`-i`	Prevents you from accidentally overwriting existing files or directories
`-r`	Includes the contents of a directory, including the contents of all subdirectories, when you copy a directory

The following example uses the `cp -r` command to copy the `dir3` directory's contents to a new directory named `dir10`. Both directories are in the `user1` directory.

```
$ cd
$ pwd
/export/home/user1
$ ls dir3
planets
$ cp dir3 dir10
cp:  dir3: is a directory
$ cp -r dir3 dir10
$ ls dir10
planets
$ ls dir3
planets
$
```

The following example uses the `cp -r` command to copy the `planets` directory from the `dir3` directory to a new directory called `constellation`, which is not in the current working directory.

```
$ cd
$ pwd
/export/home/user1
$ cd dir3
$ cp -r planets ../dir4/constellation
$ ls ../dir4/constellation
mars   pluto
$
```

Moving Files and Directories

You can move files and directories within the directory hierarchy using the `mv` command. The `mv` command does not affect the contents of the files or directories being moved.

Moving a File to Another Directory

Use the `mv` command to move a file to a different directory. The `-i` option prevents you from accidentally overwriting the existing files.

The format for the `mv` command is:

```
mv -i source target
```

The source is the old file or directory name. The target is the new file or directory name. When you use the `-i` option, you see a prompt that asks you to confirm the action if moving a file would overwrite an existing file or directory.

- A `yes` response allows the overwrite.
- A `no` response prevents the `mv` command from overwriting the target.

The following example uses the `mv` command to move the `brands` file from the `coffees` directory into the `user1` directory.

```
$ cd ~/dir1/coffees
$ pwd
/export/home/user1/dir1/coffees
$ ls
beans   brands  nuts
$ mv brands ~
$ ls
beans   nuts
$ ls ~/brands
/export/home/user1/brands
```

Moving a Directory and Its Contents

Use the `mv` command to move a directory and its contents to a different directory.

The following example uses the `mv` command to move the `practice` directory and its contents to the `letters` directory.

```
$ cd
$ pwd
/export/home/user1
$ ls practice
mailbox     project     projection  research    results
$ mv practice letters
$ ls letters
mailbox     project     projection  research    results
```

When you move a single directory to a target directory that does not exist, that directory is renamed with the target directory name, including the new path.

When you move multiple directories to a target directory that does not exist, the following error message is displayed:

```
mv: target_directory not found.
```

Creating Files and Directories

You can create new files and directories within the directory hierarchy using the touch and mkdir commands. The touch command creates a new empty file, and the mkdir command creates a new directory.

Creating Empty Files

Use the touch command to create an empty file. You can create multiple files on the same command line. If the file name or directory name already exists, the touch command updates the modification time and access time to the current date and time.

The format for the touch command is:

```
touch filename
```

You can use absolute or relative path names on the command line when creating new files.

The following example uses the touch command to create an empty file named space in the dir3 directory.

```
$ cd ~/dir3
$ ls
planets
$ touch space
$ ls
planets  space
```

The following example uses the touch command to create three empty files named moon, sun, and cosmos in the dir3 directory.

```
$ cd ~/dir3
$ ls
planets  space
$ touch moon sun cosmos
$ ls
cosmos   moon     planets  space    sun
$
```

Creating Directories

Use the `mkdir` command with a *`directory_name`* to create a single directory. If the *`directory_name`* includes a path name, use the `mkdir` command with the `-p` option. The command used with the `-p` option creates all of the non-existing parent directories that do not yet exist in the path to the new directory. The formats for the `mkdir` command include:

```
mkdir directory_name

mkdir -p directory_names
```

You can use absolute or relative path names on the command line when creating new directories.

The following example uses the `mkdir` command to create a new directory, named `Reports`, within the `user1` directory.

```
$ cd
$ pwd
/export/home/user1
$ mkdir Reports
$ ls -ld Reports
drwxr-xr-x 2 user1 staff 512 Mar 14 16:24 Reports
```

The following example uses the `mkdir` command with the `-p` option to create a new directory named `empty` located inside a directory named `newdir`, which does not yet exist.

```
$ cd
$ pwd
/export/home/user1
$ ls
brands      dir10       dir5        file1       file3       letters
dante       dir2        feathers    file.2      file4       Reports
dante_1     dir3        feathers_6  file2       fruit       tutor.vi
dir1        dir4        file.1      file.3      fruit2
$ mkdir -p newdir/empty
$ ls newdir
empty
```

Figure 4-3 shows the new directories `Reports` and `newdir` in the `user1` directory.

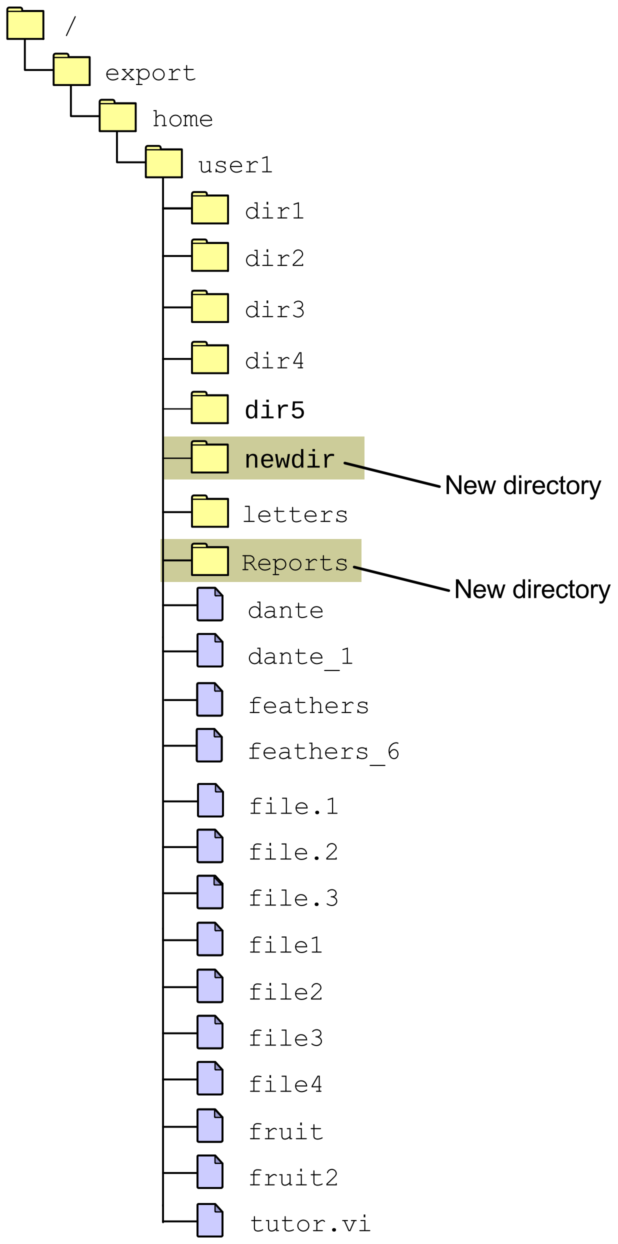


Figure 4-3 Creating a Directory

The following example uses the `mkdir` command to create a `Weekly` directory in the `Reports` directory.

```
$ mkdir Reports/Weekly
$ ls Reports
Weekly
```

The following example uses the mkdir command to create the dir1, dir2, and dir3 directories in the Weekly directory.

```
$ cd Reports/Weekly
$ mkdir dir1 dir2 dir3
$ ls -F
dir1/ dir2/ dir3/
```

The following example uses the mkdir command to create a games directory in the user1 directory.

```
$ mkdir ~/games
$ cd
$ ls -F
brands       dir2/        feathers_6   file.3       games/
dante        dir3/        file.1       file3        letters/
dante_1      dir4/        file1        file4        newdir/
dir1/        dir5/        file.2       fruit        Reports/
dir10/       feathers     file2        fruit2       tutor.vi*
$
```

Renaming Files and Directories

You can rename existing files and directories using the `mv` command. The `mv` command does not affect the contents of the file or directory being renamed.

Renaming a File

Use the `mv` command to rename files.

The following example uses the `mv` command to rename the `nuts` file in the `coffees` directory to `cream`.

```
$ cd ~/dir1/coffees
$ ls
beans           nuts

$ mv nuts cream

$ ls
beans       cream
$
```

The `mv` command creates a new file if the file does not exist; otherwise, it relocates the file.

The following example uses the `mv` command to rename the `project` file in the `letters` directory to `project2`.

```
$ cd
$ pwd
/export/home/user1
$ ls
mailbox     project projection  research    results
$ mv letters/project letters/project2
$ ls letters
mailbox     project2    projection  research    results
```

Renaming a Directory

Use the mv command to rename directories within the current directory.

The following example uses the mv command to rename the maildir directory to monthly.

```
$ cd
$ pwd
/export/home/user1
$ mkdir maildir
$ ls
brands      dir2        feathers_6  file.3      games       tutor.vi
dante       dir3        file.1      file3       letters
dante_1     dir4        file1       file4       maildir
dir1        dir5        file.2      fruit       newdir
dir10       feathers    file2       fruit2      Reports
$
$ mv maildir monthly
$ ls
brands      dir2        feathers_6  file.3      games       tutor.vi
dante       dir3        file.1      file3       letters
dante_1     dir4        file1       file4       monthly
dir1        dir5        file.2      fruit       newdir
dir10       feathers    file2       fruit2      Reports
```

Removing Files and Directories

You can permanently remove files and directories from the directory hierarchy. Use the `rm` command to remove files and directories. Use the `rmdir` command to remove empty directories. Files and directories are removed without asking you for confirmation.

Removing Files

Use the `rm` command to remove a file or multiple files on the same command line.

The format for the `rm` command is:

```
rm -option filename
```

The following example uses the `rm` command to remove the file named `projection` from the `letters` directory.

```
$ cd ~/letters
$ ls
mailbox     project2    projection  research    results
$ rm projection
$ ls
mailbox   project2  research  results
```

The following example uses the `rm` command to remove the `research` file and the `project2` file from the `letters` directory.

```
$ pwd
/export/home/user1/letters
$ ls
mailbox   project2  research  results
$ rm research project2
$ ls
mailbox  results
$
```

The `-i` option prompts you for confirmation before removing any file.

- A `yes` response completes the removal of the file.
- A `no` response prevents the `rm` command from removing the file.

The following example uses the `rm` command to remove the `mailbox` file with the `-i` option.

```
$ rm -i mailbox
rm: remove mailbox: (yes/no) ? y
$ ls
results
$
```

Removing Directories

There are two ways to remove directories. The `rmdir` command removes empty directories only. The `rm` command with the `-r` option is used to remove directories that contain files and subdirectories.

To remove a directory you are currently working in, you must change to the parent directory of that directory.

Removing an Empty Directory

Use the `rmdir` command to remove empty directories.

The format for the `rmdir` command is:

```
rmdir directory(s)
```

If a directory is not empty the `rmdir` command displays the following error message:

```
rmdir: directory "directory_name": Directory not empty
```

The following example uses the `rmdir` command to remove the `empty` directory.

```
$ cd
$ pwd
/export/home/user1
$ cd newdir
$ ls -F
empty/
$ rmdir empty
$ ls
$
```

Removing a Directory With Contents

Use the `rm` command with the `-r` option to remove directories that contain files and subdirectories.

The format for the `rm` command is:

```
rm -option(s) directory(s)
```

If you do not use the `-r` option and the directory is not empty, you receive the following error message:

```
rm: directoryname: is a directory.
```

Table 4-2 shows the options that you can use with the `rm` command when you are removing directories.

Table 4-2 Options for the `rm` Command

Option	Description
`-r`	Includes the contents of a directory and the contents of all subdirectories when you remove a directory
`-i`	Prevents accidental removal of existing files or directories

The `-i` option prompts you for confirmation before removing any file or directory.

- A `yes` response completes the removal.
- A `no` response prevents the removal.

The following example uses the `rm -r` command to remove the `letters` directory and its contents.

```
$ cd
$ pwd
/export/home/user1
$ ls letters
results
$ rm -r letters
$ ls letters
letters: No such file or directory
$
```

The following example uses the `rm -ir` command to interactively remove the `practice` directory and its contents using the `-i` option.

```
$ mkdir -p ~/practice/dir1
$ ls practice
dir1
$ rm -ir ~/practice
rm: examine files in directory /export/home/user1/practice (yes/no)? y
rm: examine files in directory /export/home/user1/practice/dir1 (yes/no)?
y
rm: remove /export/home/user1/practice/dir1: (yes/no)? y
rm: remove /export/home/user1/practice: (yes/no)? y
$
```

Using Symbolic Links

Your files (and directories) might be located on several different file systems. You can use symbolic links to link files that are in different file systems.

Introducing Symbolic Links

A symbolic link is a pointer that contains the path name to another file or directory. The link makes the file or directory easier to access if it has a long path name. A symbolic link file is identified by the letter `l` in the file-type field. This can be viewed with the `ls -l` command output.

Creating Symbolic Links

Use the `ln -s` command to create a symbolic link file. You can use either relative or absolute path names to create a symbolic link file. The file name for the symbolic link appears in the directory in which it was created.

The format for the `ln -s` command is:

```
ln -s source_file target_file
```

The *source_file* variable refers to the file you use to create the link to, while the *target_file* variable refers to the symbolic link name. When creating a symbolic link, the *source_file* might or might not already exist. If the *source_file* does not exist, a symbolic link that points to a non-existing file is created.

The following example uses the `ln -s` command to create a symbolic link file, named `dante_link`, to the `dante` file.

```
$ cd
$ pwd
/export/home/user1
$ ln -s dante dante_link
$ ls -F
brands       dir10/       feathers     file2        fruit2    tutor.vi*
dante        dir2/        feathers_6   file.3       games/
dante_1      dir3/        file.1       file3        monthly/
dante_link@  dir4/        file1        file4        newdir/
dir1/        dir5/        file.2       fruit        Reports/
```

The list of files and directories, displayed using the `ls -F` command, now shows the file `dante_link` with an at sign (@) following it to indicate that `dante_link` is a symbolic link.

If you use the `ls -l` command with a symbolic link file name, you can determine where the symbolic link is pointing.

```
$ ls -l dante_link
lrwxrwxrwx   1 user1    staff          5 Mar 15 09:45 dante_link -> dante
```

Removing Symbolic Links

Use the `rm` command to remove a symbolic link file in the same manner as you would remove a standard file.

The following example uses the `rm` command to remove the `dante_link` symbolic link file.

```
$ cd
$ pwd
/export/home/user1
$ ls -l dante_link
lrwxrwxrwx   1 user1    staff          5 Mar 15 09:45 dante_link -> dante
$ rm dante_link
$ ls dante_link
dante_link: No such file or directory
```

Exercise: Using Directory and File Commands

In this exercise, you use the commands described in this module to copy, move, rename, and remove files and directories.

Tasks

Complete the following steps. Write the commands that you would use to perform each task in the space provided.

1. Return to your home directory (if you need to), and list the contents.

   ____

2. Copy the `dir1/coffees/beans` file into the `dir4` directory, and call it `roses`.

   ____

3. Create a directory called `vegetables` in `dir3`.

   ____

4. Move the `dir1/coffees/beans` file into the `dir2/recipes` directory.

   ____

5. From your home directory, make a directory called `practice1`.

   ____

6. Using one command, copy the files `file.1` and `file.2` to the `practice1` directory.

   ____

7. Copy `dir3/planets/mars` to the `practice1` directory, and name the file `addresses`.

   ____

8. Create a directory called `play` in your `practice1` directory, and move the `practice1/addresses` file to the `play` directory.

   ____

9. Copy the `play` directory in the `practice1` directory to a new directory in the `practice1` directory called `appointments`.

   ____

10. Recursively list the contents of the `practice1` directory.

    ____

11. In your home directory and with one command, create a directory called `house` with a subdirectory of `furniture`.

    ____

12. Create an empty file called `chairs` in the new `furniture` directory.

    ____

13. Using one command, create three directories called `records`, `memos`, and `misc` in your home directory.

    ____

14. Create a new file called `carrot`, and then rename it `celery`.

    ____

15. Using one command, remove the directories called `memos` and `misc` from your home directory.

    ____

16. Try to remove the directory called `house/furniture` with the `rm` (no options) command. What happens?

    ____

17. Identify the command to remove a directory that is not empty. Remove the directory `house/furniture`. List the contents of the `house` directory to verify that the `furniture` directory has been removed.

    ____

18. Create a new directory named `newname`, and then rename it `veggies`.

    ____

19. Create a file named `mycontents` that is a symbolic link to the file `/var/sadm/install/contents`.

    ____

Exercise Summary

Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

Manage the discussion based on the time allowed for this module, which was provided in the "About This Course" module. If you do not have time to spend on discussion, then just highlight the key concepts students should have learned from the lab exercise.

- Experiences

Ask students what their overall experiences with this exercise have been. Go over any trouble spots or especially confusing areas at this time.

- Interpretations

Ask students to interpret what they observed during any aspect of this exercise.

- Conclusions

Have students articulate any conclusions they reached as a result of this exercise experience.

- Applications

Explore with students how they might apply what they learned in this exercise to situations at their workplace.

Exercise Solutions

1. Return to your home directory (if you need to), and list the contents.

   ```
   $ cd; ls
   ```

2. Copy the `dir1/coffees/beans` file into the `dir4` directory, and call it `roses`.

   ```
   $ cp -r dir1/coffees/beans dir4/roses
   ```

3. Create a directory called `vegetables` in `dir3`.

   ```
   $ mkdir dir3/vegetables
   ```

4. Move the `dir1/coffees/beans` file into the `dir2/recipes` directory.

   ```
   $ mv dir1/coffees/beans dir2/recipes
   ```

5. From your home directory, make a directory called `practice1`.

   ```
   $ mkdir practice1
   ```

6. Using one command, copy the files `file.1` and `file.2` to the `practice1` directory.

   ```
   $ cp file.1 file.2 practice1
   ```

7. Copy `dir3/planets/mars` to the `practice1` directory, and name the file `addresses`.

   ```
   $ cp dir3/planets/mars practice1/addresses
   ```

8. Create a directory called `play` in your `practice1` directory, and move the `practice1/addresses` file to the `play` directory.

   ```
   $ mkdir practice1/play
   $ mv practice1/addresses practice1/play
   ```

9. Copy the `play` directory in the `practice1` directory to a new directory in the `practice1` directory called `appointments`.

   ```
   $ cp -r practice1/play practice1/appointments
   ```

10. Recursively list the contents of the `practice1` directory.

    ```
    $ ls -R practice1
    ```

11. In your home directory and with one command, create a directory called `house` with a subdirectory of `furniture`.

    ```
    $ cd; mkdir -p house/furniture
    ```

12. Create an empty file called `chairs` in the new `furniture` directory.

    ```
    $ touch house/furniture/chairs
    ```

13. Using one command, create three directories called `records`, `memos`, and `misc` in your home directory.

    ```
    $ mkdir records memos misc
    ```

14. Create a new file called `carrot`, and then rename it `celery`.

    ```
    $ touch carrot
    $ mv carrot celery
    ```

15. Using one command, remove the directories called `memos` and `misc` from your home directory.

    ```
    $ rmdir memos misc
    ```

16. Try to remove the directory called `house/furniture` with the `rm` (no options) command. What happens?

    ```
    $ rm house/furniture
    rm: house/furniture is a directory
    ```

17. Identify the command to remove a directory that is not empty. Remove the `house/furniture` directory. List the contents of the `house` directory to verify that the `furniture` directory has been removed.

    ```
    $ rm -r house/furniture; ls house
    ```

18. Create a new directory named `newname`, and then rename it `veggies`.

    ```
    $ mkdir newname ; mv newname veggies
    ```

19. Create a file named `mycontents` that is a symbolic link to the file `/var/sadm/install/contents`.

    ```
    $ ln -s /var/sadm/install/contents mycontents
    ```

Module 5

Searching Files and Directories

Objectives

Upon completion of this module, you should be able to:

- Search for content in files
- Search for files and directories

The following course map shows how this module fits into the current instructional goal.

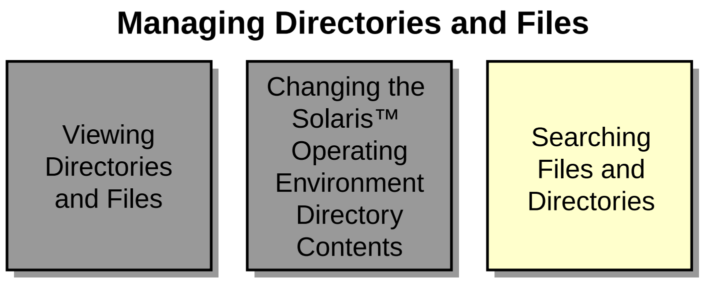

Managing Directories and Files
Viewing Directories and Files
Changing the Solaris™ Operating Environment Directory Contents
Searching Files and Directories

Figure 5-1 Course Map

Searching for Content in Files

You can search the content of files for either patterns or strings of characters using the `grep`, `egrep`, and `fgrep` commands.

The `grep` and `egrep` commands enable you to search the contents of one or more files for a specific character pattern. A pattern can be a single character, a string of characters, a word, or a sentence. The pattern of characters used to match the same characters in a search is called a regular expression (RE).

The `fgrep` command enables you to search the contents of one or more files for a specific string. A string is a literal group of characters. The `fgrep` command does not use regular expressions.

Using the `grep` Command

The `grep` command searches the contents of one or more files for a character pattern. The `grep` command prints every line containing the pattern to the screen. The `grep` command does not change file content.

The format for the `grep` command is:

```
grep option(s) pattern filename(s)
```

The options for the `grep` command modify the search and the output. All but the `-w` option are also compatible with the `egrep` and `fgrep` commands. Table 5-1 shows the options.

Table 5-1 Options for the `grep` Command

Option	**Definition**
`-i`	Ignores case. The search looks for both uppercase and lowercase characters.
`-l`	Lists the names of files with matching lines.
`-n`	Precedes each line with the relative line number in the file.
`-v`	Inverts the search to display lines that do not match the *`pattern`*.
`-c`	Counts the lines that contain the *`pattern`*.
`-w`	Searches for the expression as a word, ignoring those matches that are substrings of larger words.

To search for all lines that contain the pattern `root` in the `/etc/group` file and view line numbers, enter the following:

```
$ grep root /etc/group
root::0:root
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm
adm::4:root,adm,daemon
uucp::5:root,uucp
mail::6:root
tty::7:root,adm
lp::8:root,lp,adm
nuucp::9:root,nuucp
daemon::12:root,daemon
$
```

To search for all lines that do not contain the pattern `root` in the `/etc/group` file, enter the following:

```
$ grep -v root /etc/group
other::1:
staff::10:
sysadmin::14:
nobody::60001:
noaccess::60002:
nogroup::65534:
$
```

To search for the names of the files that contain the pattern `root`, enter the following:

```
$ cd /etc
$ grep -l root group passwd hosts
group
passwd
$
```

Note – For multiple file searches, the results are listed with the file name in which the pattern was found. For single file searches, only the matching entries are displayed.

To count the number of lines containing the pattern `root` in the `/etc/group` file, enter the following:

```
$ grep -c root group
10
$
```

The `grep` command supports several regular expression metacharacters to further define a search pattern. Table 5-2 shows some of the regular expression metacharacters.

Table 5-2 Regular Expression Metacharacters

Metacharacter	Purpose	Example	Result
`^`	Beginning of line anchor	`'^pattern'`	Matches all lines beginning with "pattern"
`$`	End of line anchor	`'pattern$'`	Matches all lines ending with "pattern"
`.`	Matches one character	`'p.....n'`	Matches lines containing a "p," followed by five characters, and followed by an "n"
`*`	Matches the preceding item zero or more times	`'[a-z]*'`	Matches lowercase alphanumeric characters or nothing at all
`[ ]`	Matches one character in the pattern	`'[Pp]attern'`	Matches lines containing "Pattern" or "pattern"
`[^]`	Matches one character not in the pattern	`'[^a-m]attern'`	Matches lines not containing "a" through "m" and followed by "attern"

To print all lines that begin with the letters "no" in the `/etc/passwd` file, enter the following:

```
$ grep '^no' /etc/passwd
nobody:x:60001:60001:Nobody:/:
noaccess:x:60002:60002:No Access User:/:
nobody4:x:65534:65534:SunOS 4.x Nobody:/:
$
```

To print all lines containing an "A" followed by three characters, followed by an "n" in the `/etc/passwd` file, enter the following:

```
$ grep 'A...n' /etc/passwd
etc/passwd
adm:x:4:4:Admin:/var/adm
lp:x:71:8:Line Printer Admin:/usr/spool/lp
uucp:x:5:5:uucp Admin:/usr/lib/uucp
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic
listen:x:37:4:Network Admin:/usr/net/nls
```

To print all lines that end with the word "adm" in the `/etc/group` file, enter the following:

```
$ grep 'adm$' /etc/group
sys::3:root,bin,sys,adm
tty::7:root,tty,adm
lp::8:root,lp,adm
```

Using the egrep Command

The `egrep` command searches the contents of one or more files for a pattern using extended regular expression metacharacters. Extended regular expression metacharacters include the regular expression metacharacters that the `grep` command uses plus some additional metacharacters.

Table 5-3 shows the additional metacharacters.

Table 5-3 Extended Regular Expression Metacharacters

Metacharacter	Purpose	Sample	Result
`+`	Matches one or more of the preceding characters	`'[a-z]+ark'`	Matches one or more lowercase letters followed by "ark" (for example, "airpark," "bark," "dark," "landmark," "shark," "sparkle," "trademark")
x`|`*y*	Matches either *x* or *y*	`'apple|orange'`	Matches for either expression
`( | )`	Groups characters	`'(1|2)+'` `'search(es|ing)+'`	Matches for one or more occurrences (for example, "1" or "2", "searches", or "searching")

The format for the `egrep` command is:

```
egrep -option(s) pattern filename(s)
```

To search for all lines containing the pattern `N` followed by either an `e` or `o` one or more times, enter the following:

```
$ egrep 'N(e|o)+' /etc/passwd
listen:x:37:4:Network Admin:/usr/net/nls:
nobody:x:60001:60001:Nobody:/:
noaccess:x:60002:60002:No Access User:/:
nobody4:x:65534:65534:SunOS 4.x Nobody:/:
```

To search for lines containing the pattern `Network Admin` or `uucp Admin`, enter the following:

```
$ egrep '(Network|uucp) Admin' /etc/passwd
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
listen:x:37:4:Network Admin:/usr/net/nls:
```

Using the `fgrep` Command

The `fgrep` command searches a file for a literal string or a group of characters. The `fgrep` command reads all regular expression characters literally. Regular expression metacharacters have no special meaning to the `fgrep` command, for example a `?` character is interpreted as a question mark and a `$` character is interpreted as a dollar sign.

The format for the `fgrep` command is:

```
fgrep option(s) string filename(s)
```

To search for all lines in the file containing a literal asterisk (*) character, enter the following:

```
$ fgrep '*' /etc/system
*ident  "@(#)system     1.18    97/06/27 SMI" /* SVR4 1.5 */
*
* SYSTEM SPECIFICATION FILE
*
* moddir:
*
*    Set the search path for modules.  This has a format similar to the
*    csh path variable. If the module isn't found in the first directory
*    it tries the second and so on. The default is /kernel /usr/kernel
<output omitted>
```

Searching for Files and Directories

Use the `find` command to locate files or directories in the directory hierarchy. This command searches using criteria such as file name, size, owner, modification time, and type.

Using the `find` Command

The `find` command recursively descends the directory hierarchy in the path name list, seeking those files that match the criteria. As the `find` command locates the files that match those criteria, the path to each file is displayed on the screen.

The format for the `find` command is:

```
find pathname(s) expression(s) action(s)
```

Table 5-4 shows the *pathname*, *expression*, and *action* arguments for the `find` command.

Table 5-4 Arguments for the `find` Command

Argument	**Definition**
pathname	The absolute or relative path in which the search originates.
expression	The search criteria specified by one or more options. Specifying multiple options causes the `find` command to treat the statement as an "AND" request, so all listed expressions must be verified as true.
action	The action to take after the files have been located. The default action is to print all path names matching the criteria to the screen.

Table 5-5 shows some of the expressions that you can use with the `find` command.

Table 5-5 Expressions for the `find` Command

Expression	Definition
`-name` *`filename`*	Finds files matching the specified *`filename`*. Metacharacters are acceptable if placed inside " ".
`-size [+|-]`*`n`*	Finds files that are larger than +*n*, smaller than –*n*, or exactly *n*. The *n* represents 512-byte blocks.
`-atime [+|-]`*`n`*	Finds files that have been accessed more than `+`*`n`* days, less than `-`*`n`* days, or exactly *`n`* days.
`-mtime [+|-]`*`n`*	Finds files that have been modified more than `+`*`n`* days ago, less than `-`*`n`* days ago, or exactly *`n`* days ago.
`-user` *`loginID`*	Finds all files that are owned by the *`loginID`* name.
`-type`	Finds a file type, for example, `f` (file) or `d` (directory).
`-perm`	Finds files that have certain access permission bits.

Table 5-6 shows the action arguments for the `find` command.

Table 5-6 Actions for the `find` Command

Action	Definition
`-exec` *`command`* `{} \;`	Runs the specified *`command`* on each file located. A set of braces, `{}`, delimits where the file name is passed to the command from the preceding expressions. A space, backslash, and semicolon (`\;`) delimits the end of the command. There must be a space before the backslash (\).
`-ok` *`command`* `{} \;`	Requires confirmation before the `find` command applies the *`command`* to each file located. This is the interactive form of the `-exec` command.
`-print`	Instructs the `find` command to print the current path name to the terminal screen. This is the default.
`-ls`	Displays the current path name and associated statistics, such as the inode number, the size in kilobytes, protection mode, the number of hard links, and the user.

To search for a file called `dante` starting in your home directory, enter the following:

```
$ find ~ -name dante
/export/home/user1/dante
$
```

To search for and delete files called `core` starting at your home directory, enter the following:

```
$ find ~ -name core -exec rm {} \;
<no output is returned regardless if any core files are found>
```

To look for all files that have not been modified in the last two days starting at the current directory, enter the following:

```
$ find . -mtime +2
<output will vary on each system>
```

To find files larger than 10 blocks (512-byte blocks) starting at your home directory, enter the following:

```
$ find ~ -size +10
/export/home/user1/.sh_history
/export/home/user1/dir1/coffees/beans
/export/home/user1/tutor.vi
<example output from command, output will vary on each system>
```

To search for files that end with the characters `es` starting at your home directory, enter the following:

```
$ find ~ -name '*es'
/export/home/user1/.dt/sessions/current/dt.resources
/export/home/user1/.dt/sessions/current.old/dt.resources
/export/home/user1/.dt/types
/export/home/user1/.dt/palettes
/export/home/user1/dir1/coffees
/export/home/user1/dir1/trees
/export/home/user1/dir2/notes
/export/home/user1/dir2/recipes
/export/home/user1/games
/export/home/user1/veggies
```

Note – Your output will vary depending on the activity performed in your home directory.

Exercise: Locating Files and Text

In this exercise, you practice searching for files and directories using the `find` command. You also display and manipulate text in files.

Tasks

Complete the following steps. Write the commands that you would use to perform each task in the space provided.

1. Search for the text string `other` in the `/etc/group` file. Display it to the screen.

   ____

2. Using the `grep` command, look for all lines in the `file4` file located in your home directory that do not contain the letter `M`.

   ____

3. Display all lines in the files `dante`, `file1`, and `dante_1` that contain the pattern `he`.

   ____

4. Display all the lines in the file `file4` that contain either the pattern `Sales` or `Finance`.

   ____

5. Starting in your home directory, find all files ending with a `2`.

   ____

6. Use the `find` command to search the `/usr` directory. Display the file names of any length that end with `ln`.

   ____

7. Starting in your home directory, find all files of type `f` for file. Print the full path name of each file located.

   ____

8. In your home directory, find all files of type `d` for directory.

   ____

9. From your home directory, find ordinary files of size 0 (zero) in the `/tmp` directory. Ask if it is permissible to remove them.

   ____

Exercise Summary

Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

Manage the discussion based on the time allowed for this module, which was provided in the "About This Course" module. If you do not have time to spend on discussion, highlight just the key concepts students should have learned from the lab exercise.

- Experiences

Ask students what their overall experiences with this exercise have been. Go over any trouble spots or especially confusing areas at this time.

- Interpretations

Ask students to interpret what they observed during any aspect of this exercise.

- Conclusions

Have students articulate any conclusions they reached as a result of this exercise experience.

- Applications

Explore with students how they might apply what they learned in this exercise to situations at their workplace.

Exercise Solutions

1. Search for the text string `other` in the `/etc/group` file. Display it to the screen.

   ```
   $ grep 'other' /etc/group
   ```

2. Using the `grep` command, look for all lines in the `file4` file located in your home directory that do not contain the letter `M`.

   ```
   $ grep -v 'M' file4
   ```

3. Display all lines in the files `dante`, `file1`, and `dante_1` that contain the pattern `he`.

   ```
   $ grep he dante file1 dante_1
   ```

4. Display all the lines in the file `file4` that contain either the pattern `Sales` or `Finance`.

   ```
   $ egrep 'Sales|Finance' file4
   ```

5. Starting in your home directory, find all files ending with a `2`.

   ```
   $ find ~ -name '*2'
   ```

6. Use the `find` command to search the `/usr` directory. Display the file names of any length that end with `ln`.

   ```
   $ find /usr -name '*ln'
   ```

7. Starting in your home directory, find all files of type `f` for file. Print the full path name of each file located.

   ```
   $ find ~ -type f
   ```

8. In your home directory, find all files of type `d` for directory.

   ```
   $ find ~ -type d
   ```

9. From your home directory, find ordinary files of size 0 (zero) in the `/tmp` directory. Ask if it is permissible to remove them.

   ```
   $ find /tmp -type f -size 0 -ok rm {} \;
   ```

   or

   ```
   $ find /tmp -type f -size 0 -exec rm -i {} \;
   ```

Module 6

Using the `vi` Editor

Objectives

Upon completion of this module, you should be able to:

- Identify the fundamentals of `vi` operation
- Manipulate files within the `vi` editor

The following course map shows how this module fits into the current instructional goal.

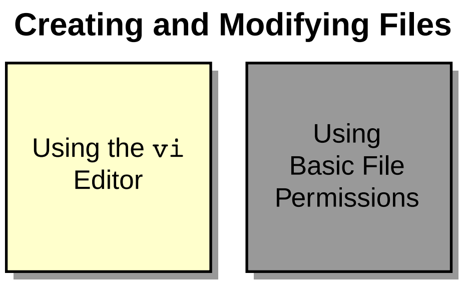


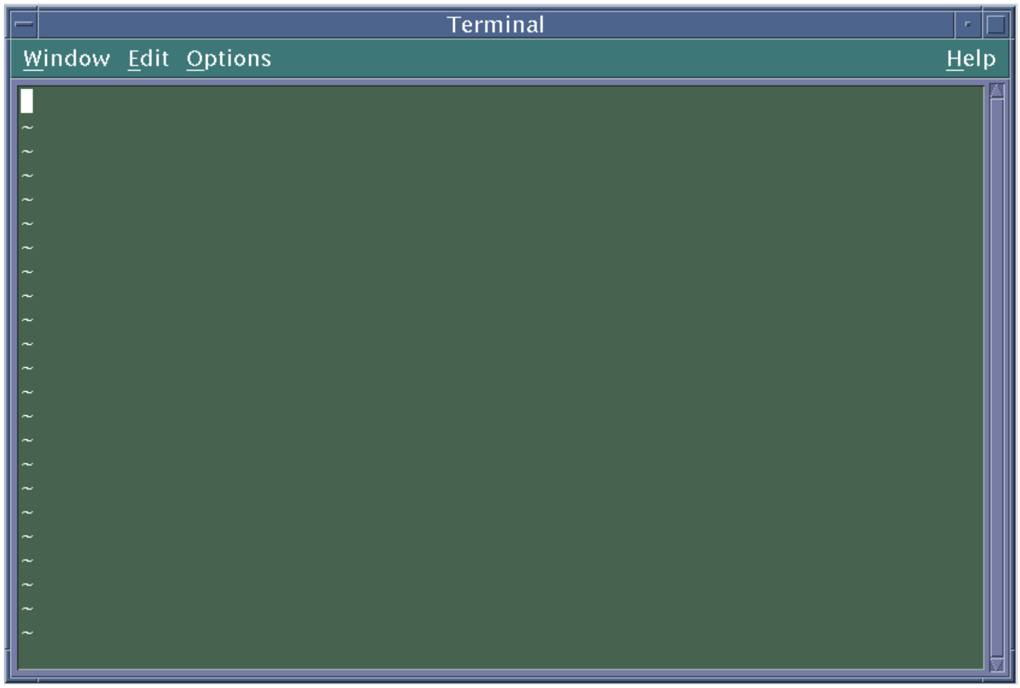
Figure 6-1 Course Map

Identifying the Fundamentals of vi Editor Operation

The visual display, or vi, editor is an interactive editor that you use to create and modify text files. You can use the vi editor when the Common Desktop Environment (CDE) window system is not available. The vi editor is also the only text editor you can use to edit certain system files without changing the permissions of the files.

All text editing with the vi editor takes place in a buffer. Changes can either be written to the disk or be discarded.

Figure 6-2 shows the initial vi display in a terminal window.



```
█
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Figure 6-2 Initial vi Editor Display

Introducing the vi Editor Modes of Operation

The vi editor is a command-line editor that has three basic modes of operation:

- Command mode
- Edit mode
- Last line mode

Introducing Command Mode

Command mode is the default mode for the vi editor. In this mode, you can enter commands to delete, change, copy, and move text. You can also position the cursor, search for text strings, and exit the vi editor.

Introducing Edit Mode

Edit mode enables you to enter text into a file. The vi editor interprets everything you type in edit mode as text. To enter edit mode, type one of the following commands:

- `i` – Inserts text before the cursor
- `o` – Opens a new blank line below the cursor
- `a` – Appends text after the cursor

Introducing Last Line Mode

Last line mode enables you to use advanced editing commands. To access last line mode, enter a colon (`:`) in command mode. The colon places your cursor at the bottom line of the screen.

Switching Between Command and Edit Modes

The default mode for the vi editor is the command mode. When you type an `i`, `o`, or `a` command, the vi editor switches to edit mode. When you are finished editing a file, press Escape to return the vi editor to command mode. In command mode, you can save the file and quit the vi editor.

The following example shows how to switch modes in the `vi` editor:

1. Enter `vi` *`filename`* to create a file. You are automatically in command mode.
2. Type the `i` command to insert text. The `i` command switches the `vi` editor to edit mode.
3. Press Escape to return to command mode.
4. Enter `:wq` to save the file and exit the `vi` editor.

Introducing the `vi` Command

The `vi` command enables you to create, edit, and view files in the `vi` editor.

The format for the `vi` command is:

```
vi
vi filename
vi option(s) filename
```

If you are editing a file and the system crashes, you can use the `-r` option to recover the file that you were editing at the time of the system crash.

To recover a file, enter the following:

```
$ vi -r filename
```

The file is opened so that you can edit it. You can then save the file and exit the `vi` editor.

Manipulating Files Within the vi Editor

You can use the vi editor to view files in read-only mode, or you can edit files in the vi editor using the vi editing commands. When using the vi editor, you can move the cursor using certain key sequences.

Viewing Files in Read-only Mode

The view command enables you to view files in read-only mode. It invokes vi with the read-only option. Most of the vi commands are available, but you cannot save changes to the file.

The format for the view command is:

```
view filename
```

To view the dante file in read-only mode, enter the following:

```
$ cd
$ view dante
```

The dante file appears. Enter :q to exit the file and the vi editor.

Inserting and Appending Text

Table 6-1 shows the commands that insert and append text in the vi editor. These commands switch the system to edit mode. To return to command mode, press Escape.

Table 6-1 Input Commands for the vi Editor

Command	Function
a	Appends text after the cursor
A	Appends text at the end of the line
i	Inserts text before the cursor
I	Inserts text at the beginning of the line
o	Opens a new line below the cursor
O	Opens a new line above the cursor
:r *filename*	Inserts text from another file into the current file

Note – The vi editor is case sensitive. Use the appropriate case for the input commands.

Moving the Cursor Within the vi Editor

Table 6-2 shows the key sequences that move the cursor in the vi editor.

Table 6-2 Key Sequences for the vi Editor

Key Sequence	**Cursor Movement**
h, left arrow, or Backspace	Left one character
j or down arrow	Down one line
k or up arrow	Up one line
l, right arrow, or spacebar	Right (forward) one character
w	Forward one word
b	Back one word
e	To the end of the current word
$	To the end of the line
0 (zero)	To the beginning of the line
^	To the first non-whitespace character on the line
Return	Down to the beginning of the next line
G	Goes to the last line of the file
1G	Goes to the first line of the file
:*n*	Goes to Line *n*
*n*G	Goes to Line *n*

Table 6-2 Key Sequences for the `vi` Editor (Continued)

Key Sequence	**Cursor Movement**
Control-F	Pages forward one screen
Control-D	Scrolls down one-half screen
Control-B	Pages back one screen
Control-U	Scrolls up one-half screen
Control-L	Refreshes the screen

Editing Files Using the `vi` Editing Commands

The `vi` editor offers a number of commands to help you edit your files. The following sections describe basic operations for deleting, changing, replacing, copying, and pasting. Remember that the `vi` editor is case sensitive.

Using the Text Deletion Commands

Table 6-3 shows the commands that delete text in the `vi` editor.

Table 6-3 Text Deletion Commands for the `vi` Editor

Command	**Function**
`R`	Overwrites or replaces characters on the line at and to the right of the cursor until terminated with Escape
`C`	Changes or overwrites characters from cursor to the end of the line
`s`	Substitutes a *`string`* for a character at the cursor
`x`	Deletes a character at the cursor
`dw`	Deletes a word or part of the word to the right of the cursor
`dd`	Deletes the line containing the cursor
`D`	Deletes the line from the cursor to the right end of the line
`:`*`n`*`,`*`n`*`d`	Deletes Lines *`n`* through *`n`* (For example, `:5,10d` deletes Lines `5` through `10`)

Note – The `delete` commands write to a buffer from which text can be retrieved.

Using the Text Changing Commands

Table 6-4 shows the commands that change text, undo a change, and repeat an edit function in the `vi` editor. Many of these commands change the `vi` editor to edit mode. To return to command mode, press Escape.

Table 6-4 Edit Commands for the `vi` Editor

Command	Function
`cw`	Changes or overwrites characters at the cursor location to the end of that word
`r`	Replaces the character at the cursor with one other character
`J`	Joins the current line and the line below
`xp`	Transposes the character at the cursor and the character to the right of the cursor
`~`	Changes the case of the letter, either uppercase or lowercase, at the cursor
`u`	Undoes the previous command
`U`	Undoes all changes to the current line
`.`	Repeats the previous command

Using the Text Replacing Commands

Table 6-5 shows the commands that search for and replace text in the `vi` editor.

Table 6-5 Search and Replace Commands

Command	**Function**
`/`*`string`*	Searches forward for the *`string`*.
`?`*`string`*	Searches backward for the *`string`*.
`n`	Finds the next occurrence of the *`string`*. Use this command after searching for a *`string`*.
`N`	Finds the previous occurrence of the *`string`*. Use this command after searching for a *`string`*.
`:%s/`*`old`*`/`*`new`*`/g`	Searches for the *`old`* string and replaces it with the *`new`* string globally.

Using the Text Copying and Pasting Commands

The `yy` command *yanks* lines of text and holds a copy of the lines in a temporary buffer. The put commands (`p, P`) inserts the lines of text from the temporary buffer and writes the text into the current document at the specified location. The copy (`co`) and move (`m`) commands copy or move specified lines to a requested location within the file.

Table 6-6 shows the commands that yank (`yy`), put (`p, P`), copy (`co`), and move (`m`) text in the `vi` editor.

Table 6-6 Copy and Paste Commands

Command	**Function**
`yy`	Yanks a copy of a line.
`p`	Puts yanked or deleted text under the line containing the cursor.
`P`	Puts yanked or deleted text before the line containing the cursor.

Table 6-6 Copy and Paste Commands (Continued)

Command	**Function**
`:`*n*`,`*n* `co` *n*	Copies Lines *n* though *n* and puts them after Line *n*. For example, `:1,3 co 5` copies Lines `1` through `3` and puts them after Line `5`.
`:`*n*`,`*n* `m` *n*	Moves Lines *n* through *n* to Line *n*. For example, `:4,6 m 8` moves Lines `4` through `6` to Line `8`. Line `6` becomes Line `8`. Line `5` becomes Line `7`. Line `4` becomes Line `6`.

Using the File Save and Quit Commands

Table 6-7 shows the commands that save the file and quit the `vi` editor.

Table 6-7 Save and Quit Commands

Command	**Function**
`:w`	Saves the file with changes by writing to the disk
`:w` *new_filename*	Writes the contents of the buffer to *new_filename*
`:wq`	Saves the file with changes and quits the `vi` editor
`:x`	Saves the file with changes and quits the `vi` editor
`ZZ`	Saves the file with changes and quits the `vi` editor
`:q!`	Quits without saving changes

Customizing a `vi` Session

You can customize a `vi` session by setting variables for the session. When you set a `vi` variable, you enable a feature that is not activated by default. You use the `set` command to enable and disable variables.

Table 6-8 shows some of the variables of the `set` command, including displaying line numbers and invisible characters, such as the Tab and the end-of-line characters.

Table 6-8 Edit Session Customization Commands

Command	**Function**
`:set nu`	Shows line numbers
`:set nonu`	Hides line numbers
`:set ic`	Instructs searches to ignore case
`:set noic`	Instructs searches to be case sensitive
`:set list`	Displays invisible characters, such as `^I` for a Tab and a `$` for end-of-line
`:set nolist`	Turns off the display of invisible characters
`:set showmode`	Displays the current mode of operation
`:set noshowmode`	Turns off the mode of operation display
`:set`	Displays all the `vi` variables that are set
`:set all`	Displays all set `vi` variables and their current values

To create an automatic customization for all of your `vi` sessions, complete the following steps:

1. Create a file in your home directory named `.exrc`.
2. Enter the `set` variables into the `.exrc` file.
3. Enter `set` *`variable`* without the preceding colon.
4. Enter one command on one line.

The `vi` editor reads the `.exrc` file, located in your home directory, each time you open a `vi` session, regardless of your current working directory.

Exercise: Using the vi Editor

In this exercise, you practice performing vi editor commands in the tutor.vi tutorial. Use Figure 6-3 on page 6-13 as a reference to complete the exercise.

Tasks

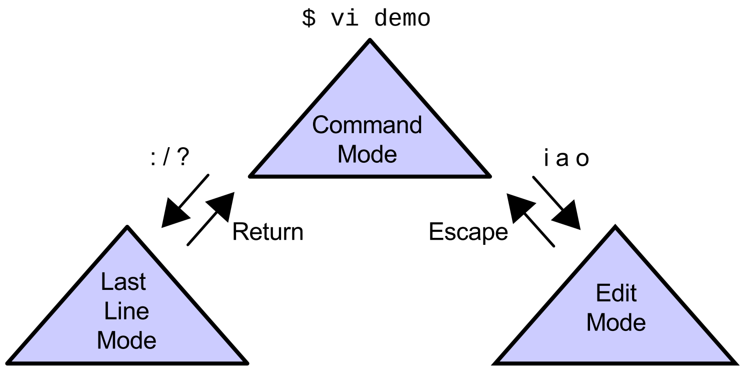
Complete the following steps.

1. Make sure that you are in your home directory. Open the tutor.vi tutorial file with the command:

```
$ vi tutor.vi
```

2. Complete the lessons outlined in this tutorial.

Explain that the following diagram is a useful reference when using the vi editor both in this course and back at the work site.



Search Functions

`/exp`	Go forward to exp
`?exp`	Go backward to exp

Move and Insert Text

`:3,8d`	Delete line 3-8
`:4,9m 12`	Move lines 4-9 to 12
`:2,5t 13`	Copy lines 2-5 to 13
`:5,9w file`	Write lines 5-9 to file

Save Files and Exit

`:w`	Write to disk
`:w newfile`	Write to newfile
`:w! file`	Write absolutely
`:wq`	Write and quit
`:q`	Quit editor
`:q!`	Quit and discard
`:e!`	Re-edit current file, Discard buffer

Control Edit Session

`:set nu`	Display line number
`:set nonu`	Turn off line number
`:set all`	Show all settings
`:set list`	Display invisible Characters
`:set wm=5`	Wrap lines 5 spaces From right margin

Screen/Line Movement

`j`	Move cursor down
`k`	Move cursor up
`h`	Move cursor left
`l`	Move cursor right
`0`	Go to line start (zero)
`$`	Go to line end
`G`	Go to last file line

Word Movement

`w`	Go forward 1 word
`b`	Go backward 1 word

Search Functions

`n`	Repeat previous search
`N`	Reverse previous search

Delete Text

`x`	Delete 1 character
`dw`	Delete 1 word
`dd`	Delete 1 line
`D`	Delete to end of line
`d0`	Delete to start of line
`dG`	Delete to end of file

Cancel Edit Function

`u`	Undo last change
`.`	Do last change again

Copy and Insert Text

`Y`	Yank a copy
`5Y`	Yank a copy of 5 lines
`p`	Put below cursor
`P`	Put above cursor

Add/Append Text

`a`	Append after cursor
`A`	Append at line end
`i`	Insert before cursor
`5i`	Insert text 5 times
`I`	Insert at beginning of line

Add New Lines

`o`	Open a line below cursor
`O`	Open a line above cursor

Search Functions

`n`	Repeat previous search
`N`	Reverse previous search

Change Text

`cw`	Change a word
`3cw`	Change 3 words
`C`	Change line
`r`	Replace one character
`R`	Replace/type over a line

Figure 6-3 Quick Reference Chart for Using the `vi` editor

Exercise Summary

Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

Manage the discussion based on the time allowed for this module, which was provided in the "About This Course" module. If you do not have time to spend on discussion, then just highlight the key concepts students should have learned from the lab exercise.

- Experiences

Ask students what their overall experiences with this exercise have been. Go over any trouble spots or especially confusing areas at this time.

- Interpretations

Ask students to interpret what they observed during any aspect of this exercise.

- Conclusions

Have students articulate any conclusions they reached as a result of this exercise experience.

- Applications

Explore with students how they might apply what they learned in this exercise to situations at their workplace.

Module 7

Using Basic File Permissions

Objectives

Upon completion of this module, you should be able to:

- View file and directory permissions
- Determine file or directory access
- Change the permissions
- Use the default permissions

The following course map shows how this module fits into the current instructional goal.

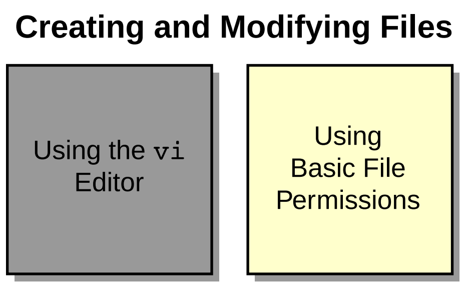


Figure 7-1 Course Map

Viewing File and Directory Permissions

All files and directories in the Solaris™ Operating Environment (Solaris OE) have a standard set of access permissions. The access permissions control who can access files, which provides a fundamental level of security for the system. You can use the `ls -l` command and the `ls -n` command to view the permissions for a given file or directory. You can change the permissions for certain files.

Introducing Security Fundamentals

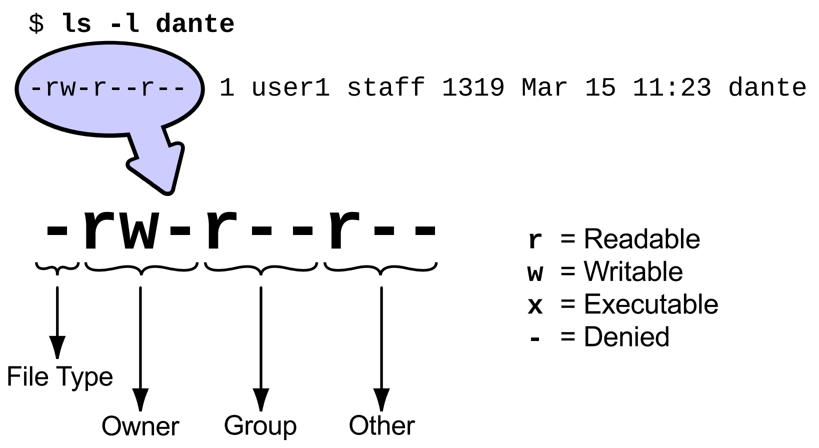
The most important function of a secure system is to deny access to unauthorized users and provide access for authorized users. Although system administrator maintains the primary security of a system, users also play a role in keeping the system secure.

The Solaris OE uses two basic measures to prevent unauthorized access to a system and to protect data. The first measure is to authenticate a user's login by verifying that the user name and password exist in the `/etc/passwd` and `/etc/shadow` files. The second measure is to automatically protect file and directory access. The Solaris OE assigns a standard set of access permissions to files and directories when the files and directories are created.

Note – The Solaris OE also provides a special user account on every system called the `root` user. The `root` user, often referred to as the superuser, has complete access to every user account and all files and directories. The `root` user can override the permissions placed on files and directories.

Viewing Permission Categories

To view the permissions for files and directories, use the `ls -l` command. Figure 7-2 shows the information displayed for the file `dante.`



```
$ ls -l dante
-rw-r--r--   1 user1 staff 1319 Mar 15 11:23 dante
```

Figure 7-2 Permissions Example

The first field of information displayed by the `ls -l` command is the file type. The file type is typically either a file or a directory. A file is represented by a hyphen (`-`). A directory is represented by the letter `d`.

The remaining fields represent three types of user: owner, group, and other.

- Owner – The assigned owner of the file or directory
- Group – A logical group the file or directory and user belong to
- Other – All other users who either do not own the file or directory or are not a member of the file's or the directory's group

Each type of user has three permissions, called a permission set. The first permission set represents the owner permissions. The second permission set represents the group permissions. The last permission set represents the other permissions.

Introducing Permission Types

There are three types of permissions in the owner, group, and other permission sets. These permissions are represented by the characters `r` (read), `w` (write), and `x` (execute). The presence of a character, such as `r` indicates that permission is granted. The dash (`-`) symbol in place of a character indicates that the permission is denied.

The Owner Permissions

The owner permission set determines the type of access the owner has for the file or directory.

The three characters in this set of permissions represent read, write, and execute permissions for the owner.

The Group Permissions

The group permission set determines the type of shared file access for each user who is in the file's group. A group is a collection of users who can access files, owned by the group, based on the permission set.

The three characters in this set of permissions represent read, write, and execute permissions.

Note – The system administrator creates and maintains groups in the `/etc/group` file. The system administrator assigns users to groups according to the need for shared file access.

The Other Permissions

The other permission set determines the type of access for all other users who have access to the system, but who do not own the file or directory, and are not a member of the file's or directory's group.

The three characters in this set of permissions represent read, write, and execute permissions.

Introducing Permission Characters and Sets

The read, write, and execute permissions are interpreted differently when assigned to a file than when assigned to a directory.

Table 7-1 shows the permission definitions.

Table 7-1 Permission Characters

Permission	**Character**	**Access for a File**	**Access for a Directory**
Read	`r`	You can display file contents and copy the file.	You can list the directory contents with the `ls` command.
Write	`w`	You can modify the file contents.	If you also have execute access, you can add and delete files in the directory.
Execute	`x`	You can execute the file if it is an executable. You can execute a shell script if you also have read and execute permissions.	You can use the `cd` command to access the directory. If you also have read access, you can run the `ls -l` command on the directory to list contents.

Note – For a directory to be of general use, enable the read and execute permissions.

Table 7-2 shows examples of different permission sets for files and directories.

Table 7-2 Permission Sets

Permissions	**Description**
`-rwx------`	This file has read, write, and execute permissions set for the file owner only. Permissions for group and other are denied.
`dr-xr-x---`	This directory has read and execute permissions set for the directory owner and the group only.
`-rwxr-xr-x`	This file has read, write, and execute permissions set for the file owner. Read and execute permissions are set for the group and other.

When you create a new file or directory, the Solaris OE automatically assigns initial permissions. The initial permissions for a file are `rw-rw-rw-.`The initial permissions for a directory are `rwxrwxrwx`.

Note – You can assign execute permissions on files with the `chmod` command. The `chmod` command is described later in this module. Execute permissions are not assigned by default when you create a file.

Determining File or Directory Access

The following sections describe how to use the `ls-n` command in the Solaris OE to determine ownership of files and directories.

Using the `ls -n` Command

All files and directories have an associated user identification number (UID) and a group identification number (GID). The UID identifies the user who owns the file or directory. The GID identifies the group of users who own the file or directory. A file or directory can belong to only one group at a time. The Solaris OE uses these numbers to track ownership and group membership of files and directories.

Use the `ls -n` command to view the UIDs and GIDs.

```
$ ls -n
total 108
-rw-r--r--   1 11001    10             0 Feb 22 14:51 brands
-rw-r--r--   1 11001    10          1320 Feb 22 14:51 dante
-rw-r--r--   1 11001    10           368 Feb 22 14:51 dante_1
```

Output from the `ls -n` command contains the following fields from left to right:

- The file type
- The permission sets
- The number of hard links to the file or directory
- The UID of the owner
- The GID of the group
- The size of the file or directory in bytes
- The time and date the file or directory was last modified
- The name of the file or directory

Determining Permissions

When a user attempts to access a file or directory, the Solaris OE compares the UID of the user to the UID of the file or directory. If the UIDs match, the owner's permissions are used to determine if access to the file or directory is granted.

If the UIDs do not match, the Solaris OE compares the user's GID and the GID of the file or directory. If these numbers match, the group permissions apply.

If the GIDs do not match, the Solaris OE uses the other category of permissions to determine file or directory access.

Figure 7-3 shows the decision tree for determining permissions.

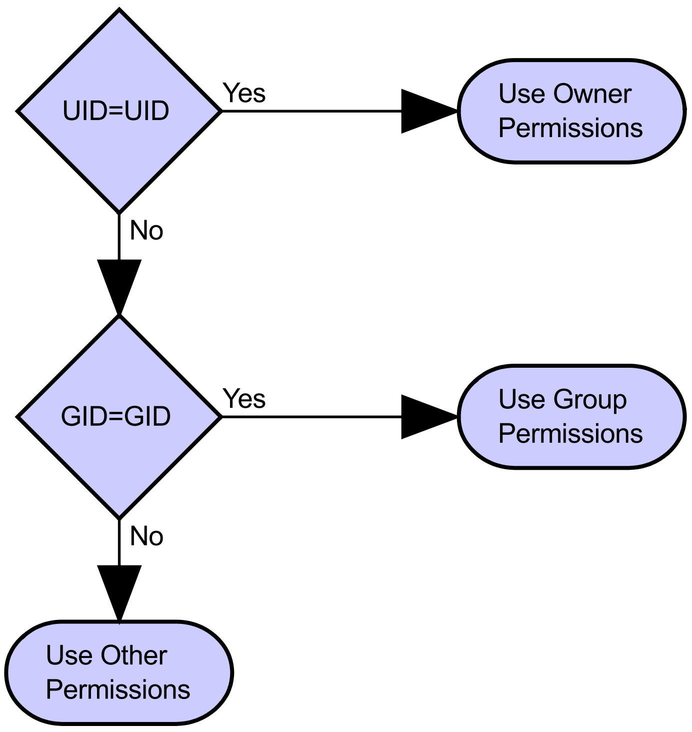


Figure 7-3 Process for Determining Permissions

Changing the Permissions

You can modify the permissions set for files or directories by using the `chmod` command. Either the owner of the file or directory or the `root` user can use the `chmod` command to change permissions.

Understanding Permission Modes

The `chmod` command can modify permissions specified in either symbolic mode or octal mode.

- Symbolic mode uses combinations of letters and symbols to add or remove permissions for each type of user.
- Octal mode uses octal numbers to represent each permission. Octal mode is also referred to as absolute mode.

Changing Permissions in Symbolic Mode

The format for the `chmod` command in symbolic mode is:

```
chmod symbolic_mode filename
```

The *`symbolic_mode`* option consists of three parts: the user category that is affected, the function that is performed, and the permissions that are affected. For example, if the option is `g+x`, the executable permission is added for the group.

Figure 7-4 shows the *symbolic_mode* options.

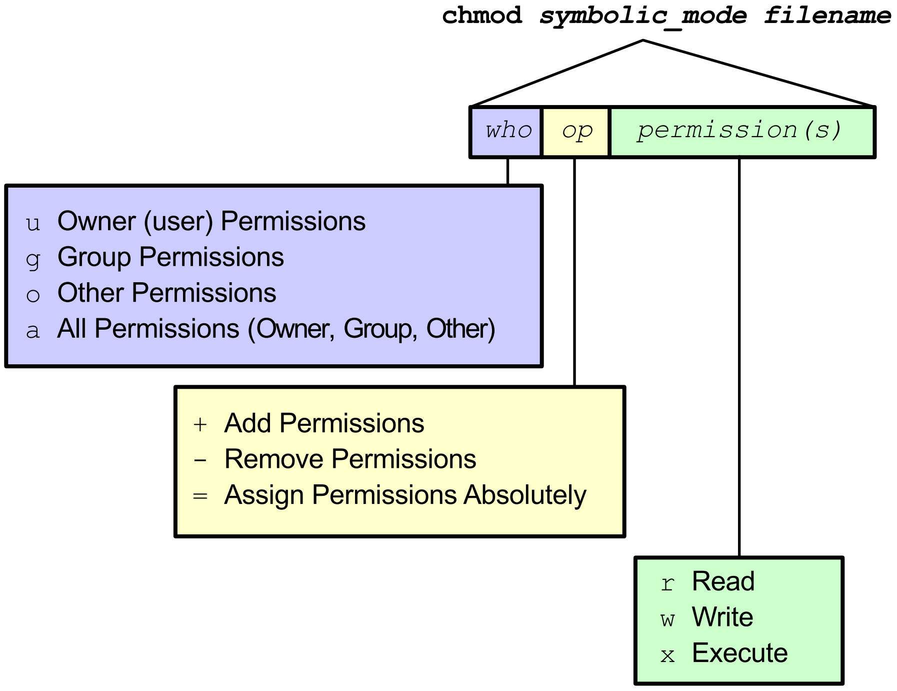

chmod symbolic_mode filename
who
op
permission(s)
u Owner (user) Permissions
g Group Permissions
o Other Permissions
a All Permissions (Owner, Group, Other)
+ Add Permissions
– Remove Permissions
= Assign Permissions Absolutely
r Read
w Write
x Execute

Figure 7-4 Symbolic Mode Command Format

The following examples show how to modify permissions on files and directories using symbolic mode.

To remove the read permission for other, enter the following:

```
$ ls -l dante
-rw-r--r--   1 user1    staff       1319 Mar 22 14:51 dante
$ chmod o-r dante
$ ls -l dante
-rw-r-----   1 user1    staff       1319 Mar 22 14:51 dante
$
```

To remove the read permission for the group, enter the following:

```
$ chmod g-r dante
$ ls -l dante
-rw-------   1 user1    staff       1319 Mar 22 14:51 dante
$
```

To add an execute permission for the owner (user) and a read permission for the group and other, enter the following:

```
$ chmod u+x,go+r dante
$ ls -l dante
-rwxr--r--   1 user1    staff       1319 Mar 22 14:51 dante
$
```

There is no space in `u+x,go+r`.

To assign read and write permissions for owner, group, and other, enter the following:

```
$ chmod a=rw dante
$ ls -l dante
-rw-rw-rw-   1 user1    staff       1319 Mar 22 14:51 dante
$
```

Changing Permissions in Octal Mode

The format for the `chmod` command in octal mode is:

```
chmod octal_mode filename
```

The *`octal_mode`* option consists of three octal numbers, 0 to 7, that represent a combination of permissions for the file or directory.

Table 7-3 shows the octal numbers for each individual permission.

Table 7-3 Assigned Octal Values for Permissions

Octal Value	Permission
4	Read
2	Write
1	Execute

These numbers are combined into one number for each permission set.

Table 7-4 shows the octal numbers that represent a combined set of permissions.

Table 7-4 Octal Digits for Permission Sets

Octal Value	**Permission Sets**
7	`r w x`
6	`r w -`
5	`r - x`
4	`r - -`
3	`- w x`
2	`- w -`
1	`- - x`
0	`- - -`

By combining octal numbers, you can quickly modify the permissions for each category of users. The first octal number defines owner permissions, the second octal number defines group permissions, and the third octal number defines other permissions.

Table 7-5 shows the permission sets for the three-digit octal numbers.

Table 7-5 Combined Values and Permissions

Octal Mode	**Permissions**
644	`rw-r--r--`
751	`rwxr-x--x`
775	`rwxrwxr-x`
777	`rwxrwxrwx`

The `chmod` command automatically fills in any missing digits to the left with zeros.

The following examples show how to modify permissions on files and directories using octal mode.

Note – Each example builds on the resulting permissions from the previous example.

To give owner, group, and other read and execute access only, enter the following:

```
$ ls -l dante
-rw-rw-rw-   1 user1    staff       1319 Mar 22 14:51 dante
$ chmod 555 dante
$ ls -l dante
-r-xr-xr-x   1 user1    staff       1319 Mar 22 14:51 dante
$
```

To change owner and group permissions to include write access, enter the following:

```
$ chmod 775 dante
$ ls -l dante
-rwxrwxr-x   1 user1    staff       1319 Mar 22 14:51 dante
$
```

To change group permission to read and execute only, enter the following:

```
$ chmod 755 dante
$ ls -l dante
-rwxr-xr-x   1 user1    staff       1319 Mar 22 14:51 dante
$
```

Modifying Default Permissions

Every new file or directory created has a set of default permissions assigned to it. You create and modify these permissions with the `umask` utility.

Introducing the umask Utility

The `umask` utility affects the initial permissions for files and directories when the files and directories are created. The `umask` utility is a three-digit octal value that is associated with the read, write, and execute permissions. The first digit determines the default permissions for the owner, the second digit determines the default permissions for the group, and the third digit determines the default permissions for other.

In the Solaris OE, the default `umask` value is 022.

To view the `umask` value, enter the `umask` command.

```
$ umask
022
$
```

Depending on the shell, the umask value appears as either 0022 (sh), 022 (ksh), or 22 (csh).

The Solaris OE automatically assigns initial permission values at file and directory creation.

The initial permission value specified by the system for a file creation is `666` (`rw-rw-rw-`).

The initial permission value specified by the system for a directory creation is `777` (`rwxrwxrwx`).

The `umask` value determines the default permissions for files and directories that the user creates. The `umask` value is subtracted from the Solaris OE initial permission value to set the default permissions.

Applying the umask Utility

You can calculate the default permissions for new files and directories by subtracting the `umask` value from the initial value specified by the system in symbolic mode.

For example, the initial permissions for a new file in symbolic mode are:

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

This corresponds to read-write access for the owner, group, and other. This value is represented in octal mode as:

```
42-42-42- or 666
```

Use the default `umask` value of 022 to remove (or deny) write permission for group and other.

The result in octal mode is:

```
42-4--4-- or 644
```

A dash in the permission fields could be represented by the number 0 (zero).

The result in symbolic mode is derived as shown in Table 7-6.

Table 7-6 Symbolic Mode Permission Fields

Permission Field	**Description**
`rw-rw-rw-`	Initial value specified by the system for a new file
`----w--w-`	Default `umask` utility value to be subtracted
`rw-r--r--`	Default permissions assigned to newly created files

When the access permissions to be denied are masked out from the initial value, the default permissions assigned to the new files remain.

All newly created files are assigned read and write access for the owner, and read access for group and other.

You can apply this same process to determine the default permissions when you create new directories.

For directories, the initial value specified by the system is:

```
rwxrwxrwx
```

This corresponds to read, write, and execute access for the owner, group, and other. This value is represented in octal mode as:

```
421421421 or 777
```

Use the default `umask` value of 022 to remove (or deny) write permission for group and other.

The result in octal mode is:

```
4214-14-1 or 755
```

The result in symbolic mode is derived as shown in Table 7-7.

Table 7-7 Symbolic Mode Permission Fields

Permission Field	**Description**
`rwxrwxrwx`	Initial value specified by the system for a new directory
`----w--w-`	Default `umask` utility value to be subtracted
`rwxr-xr-x`	Default permissions set for newly created directories

When the access permissions to be denied are masked out from the initial value, the default permissions assigned to the new directories remain.

All newly created directories are assigned read, write, execute access for the owner, and read and execute access for group and other.

Changing the umask Value

Some users require a more secure `umask` value of 027, which assigns the following access permissions to newly created files and directories.

- Files have read and write permissions for the owner, read permission for the group, and no permissions for other.

  `rw-r-----`

- Directories have read, write, and execute permissions for the owner, read and execute permissions for the group, and no permissions for other.

  `rwxr-x---`

You can change the `umask` value to a new value on the command line. For example, to change the `umask` value to 027 and verify the new value, enter the following:

```
$ umask 027
$ umask
027
$
```

The new `umask` value affects only those files and directories that are created from this point forward. However, if the user logs out of the system, the new value (027) is replaced by the old value (022) on subsequent logins because the `umask` value was changed using the command line.

Exercise: Changing File Permissions

In this exercise, you practice reading permissions on files and changing permissions using symbolic or octal mode.

Preparation

Set the `umask` value to 022 on your system:

```
$ umask 022
```

Tasks

Complete or answer the following.

1. Execute the following commands:

```
$ mkdir ~/perm
$ cd /etc
$ ls -l group motd shadow vfstab
$ cp group motd shadow vfstab ~/perm
$ ls -l ~/perm
$ cd
$ cp -r /etc/skel perm
$ ls -l perm
```

When trying to copy the `shadow` file the error message: `cp: cannot open shadow: Permission denied` appeared. Why?

____

____

2. Change to the `perm` directory, and list the contents of the directory.

```
$ cd perm
$ ls -l perm
```

In the following table, fill in the permission sets for each file, and write the three-digit octal number that represents the combined set of permissions.

File or Directory	Permissions			Octal Value
	Owner	Group	Other	
`group`	`rw-`			
`passwd`			`r--`	
`vfstab`	`rw-`			
`skel`				755

3. Create a new file and a new directory.
   - What are the default permissions given to the new file?

     ____

   - What are the default permissions given to the new directory?

     ____

4. In a directory with permissions of `drwxr-xr--`, who can perform the following actions with the files shown below? Put an X next to each allowed action.

   `-rw-r--r--`

Owner:	read___	modify___	delete___	execute___
Group:	read___	modify___	delete___	execute___
Other:	read___	modify___	delete___	execute___

   `-rwxrwxr-x`

Owner:	read___	modify___	delete___	execute___
Group:	read___	modify___	delete___	execute___
Other:	read___	modify___	delete___	execute___

5. Using symbolic mode, add write permission for the group to the `motd` file.

   ____

6. Using octal mode, change the permissions on the `motd` file to `-rwxrw----`.

   ____

7. Using octal mode, add write permission for other on the file named `group`.

   ____

8. Create a new file called `memo` in your `dir4` directory.

   ____

9. Remove the read permission for the owner from the `memo` file in the `dir4` directory. Use either symbolic or octal mode.

   ____

   What happens when you try to use the `cat` command to view the `memo` file?

   ____

   What happens when you try to copy the `memo` file?

   ____

   ____

Exercise Summary

Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

Manage the discussion based on the time allowed for this module, which was provided in the “About This Course” module. If you do not have time to spend on discussion, then just highlight the key concepts students should have learned from the lab exercise.

- Experiences

Ask students what their overall experiences with this exercise have been. Go over any trouble spots or especially confusing areas at this time.

- Interpretations

Ask students to interpret what they observed during any aspect of this exercise.

- Conclusions

Have students articulate any conclusions they reached as a result of this exercise experience.

- Applications

Explore with students how they might apply what they learned in this exercise to situations at their workplace.

Exercise Solutions

1. Execute the following commands:

```
$ mkdir ~/perm
$ cd /etc
$ ls -l group motd shadow vfstab
$ cp group motd shadow vfstab ~/perm
$ ls -l ~/perm
$ cd
$ cp -r /etc/skel perm
$ ls -l perm
```

When trying to copy the `shadow` file the error message: `cp: cannot open shadow: Permission denied` appeared. Why?

Because the only user type that has read permission on this file is the owner, who is `root`.

2. Change to the `perm` directory, and list the contents of the directory.

```
$ cd perm
$ ls -l perm
```

In the following table, fill in the permission sets for each file, and write the three-digit octal number that represents the combined set of permissions.

File or Directory	Permissions			Octal Value
	Owner	Group	Other	
`group`	`rw-`	`r--`	`r--`	644
`passwd`	`r--`	`r--`	`r--`	444
`vfstab`	`rw-`	`r--`	`r--`	644
`skel`	`rwx`	`r-x`	`r-x`	755

3. Create a new file and a new directory.
   - What are the default permissions given to the new file?

     `rw-r--r--`
   - What are the default permissions given to the new directory?

     `rwxr-xr-x`

4. In a directory with permissions of `drwxr-xr--`, who can perform the following actions with the files shown below? Put an X next to each allowed action.

   `-rw-r--r--`

Owner:	read **X**	modify **X**	delete **X**	execute___
Group:	read **X**	modify___	delete___	execute___
Other:	read ___	modify___	delete___	execute___

   `-rwxrwxr-x`

Owner:	read **X**	modify **X**	delete **X**	execute **X**
Group:	read **X**	modify **X**	delete ___	execute **X**
Other:	read___	modify___	delete___	execute ___

5. Using symbolic mode, add write permission for the group to the `motd` file.

   ```
   $ chmod g+w motd
   ```

6. Using octal mode, change the permissions on the `motd` file to `-rwxrw----`.

   ```
   $ chmod 760 motd
   ```

7. Using octal mode, add write permission for other on the file named `group`.

   ```
   $ chmod 646 group
   ```

8. Create a new file called `memo` in your `dir4` directory.

   ```
   $ touch ~/dir4/memo
   ```

9. Remove the read permission for the owner from the `memo` file in the `dir4` directory. Use either symbolic or octal mode.

   ```
   $ chmod u-r ~/dir4/memo
   ```

   or

   ```
   $ chmod 244 ~/dir4/memo
   ```

   What happens when you try to use the `cat` command to view the `memo` file?

   You cannot use the `cat` *command because the read permission has been removed for the user. Even though you are part of the group, the permissions are viewed in the order in which they appear.*

   What happens when you try to copy the `memo` file?

   You cannot copy the file because the user has no read permission.

Module 8

Using Commands Within the Korn Shell

Objectives

Upon completion of this module, you should be able to:

- Use the shell metacharacters
- Display the command history
- Describe user initialization files

The following course map shows how this module fits into the current instructional goal.

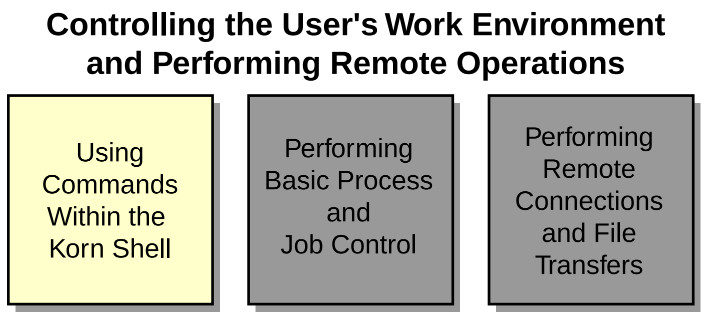

Controlling the User's Work Environment and Performing Remote Operations
Using Commands Within the Korn Shell
Performing Basic Process and Job Control
Performing Remote Connections and File Transfers

Figure 8-1 Course Map

Using the Shell Metacharacters

Korn shell metacharacters are specific characters, generally symbols, that have special meaning to the shell. Three types of metacharacters are path name metacharacters, file name substitution metacharacters, and redirection metacharacters.

Caution – Do not use these metacharacters when creating file and directory names. These characters hold special meaning to the shell.

Using the Path Name Metacharacters

Some of the shell metacharacters are specific to path name functions. These metacharacters simplify location changes within the directory hierarchy. Some examples of path name metacharacters are:

`~ ~`*`username`*` ~+ ~- -`

The Tilde (~) Character

The tilde (~) character represents the home directory of the current user. It is an abbreviation of the absolute path name.

To change directories to `dir1` using the ~ character, enter the following:

```
$ cd ~/dir1
$ pwd
/export/home/user1/dir1/
$
```

Note – The tilde (~) character is available in all shells except the Bourne shell.

The Tilde (~) Character With a User Name

The tilde (~) character followed by a user name represents the home directory of the specified user.

To change directories to the user2 home directory, enter the following:

```
$ cd ~user2
$ pwd
/export/home/user2
$
```

The Tilde and Plus (~+) and Tilde and Dash (~-) Characters

The tilde and plus (~+) characters represent the current working directory. The tilde and dash (~-) characters represent the previous working directory.

To list the contents of the current directory and return to the previous directory, enter the following:

```
$ cd
$ pwd
/export/home/user1
$ cd dir2
$ ls ~+
beans notes recipes
$ cd ~-
$ pwd
/export/home/user1
$ cd ~-
$ pwd
/export/home/user1/dir2
$
```

The Dash (-) Character

The dash (-) character in the shell represents the previous working directory. Use the dash to switch between two specific directories. The shell automatically displays the current directory path.

To switch between the `user1` and `tmp` directories, enter the following:

```
$ cd
$ pwd
/export/home/user1
$ cd /tmp
$ pwd
/tmp
$ cd -
/export/home/user1
$ cd -
/tmp
$
```

Using the File Name Substitution Metacharacters

You can substitute some shell metacharacters for other characters. These metacharacters simplify commands. Some examples of file name substitution metacharacters are:

```
* ? []
```

The Asterisk (*) Character

The asterisk (`*`) character is often referred to as a wild card character that represents any zero or more characters except the leading period (`.`) of a hidden file.

To list all files and directories that start with the letter `f` followed by zero or more other characters, enter the following:

```
$ cd
$ ls f*
feathers    file.1      file.2      file.3      file4       fruit2
feathers_6  file1       file2       file3       fruit
$
```

To list all files and directories that start with the letter d followed by zero or more other characters, enter the following:

```
$ ls d*
dante    dante_1

dir1:
coffees  fruit    trees

dir10:
planets

dir2:
beans    notes    recipes

dir3:
cosmos      moon        planets     space       sun         vegetables

dir4:
constellation  memo          roses

dir5:

$
```

To list all files and directories that end with the number 3, followed by zero or more other characters, enter the following:

```
$ ls *3
file.3  file3

dir3:
cosmos      moon        planets     space       sun         vegetables
$
```

The Question Mark (?) Character

The question mark (?) character represents any single character except the leading period (.) of a hidden file. The question mark character is often referred to as a wild card character.

To list all files and directories that start with the string `dir` and followed by one other character, enter the following:

```
$ ls dir?
dir1:
coffees  fruit    trees

dir2:
beans    notes    recipes

dir3:
cosmos      moon        planets     space       sun         vegetables

dir4:
constellation  memo          roses

dir5
$
```

If no files match an entry with the question mark, an error message appears.

```
$ ls z?
z?: No such file or directory
$
```

The Square Bracket ([]) Characters

The square bracket (`[]`) characters represent a set or range of characters for a single character position.

A set of characters is any number of specific characters, for example, `[acb]`. The characters in a set do not generally need to be in any order. For example, `[abc]` is the same as `[cab]`.

A range of characters is a series of ordered characters. A range lists the first character, a hyphen (`-`), and the last character, for example, `[a–z]` or `[0–9]`. List the characters in a range in the order that you want them to appear in the output. Use `[A–Z]` or `[a-z]` to search for any uppercase or lowercase alphabetic character, respectively.

To list all files and directories that start with the letters a through f, enter the following:

```
$ ls [a-f]*
brands      dante_1     file.1      file2       file4
celery      feathers    file1       file.3      fruit
dante       feathers_6  file.2      file3       fruit2

dir1:
coffees  fruit    trees

dir10:
planets

dir2:
beans    notes    recipes

dir3:
cosmos      moon        planets     space       sun         vegetables

dir4:
constellation  memo          roses

dir5:
$
```

To list all files and directories that start with the letters f or p, enter the following:

```
$ ls [fp]*
feathers    file.1      file.2      file.3      file4       fruit2
feathers_6  file1       file2       file3       fruit

perm:
group   motd    passwd  skel    vfstab

practice1:
appointments  file.1       file.2       play
$
```

Using the Quoting Characters

Quoting is a process that instructs the shell to mask, or ignore, the special meaning of metacharacters. The quoting characters are the single forward quotation marks (′ ′), the double quotation marks (" "), and the backslash (\).

The Quotation Marks and Backslash Characters

Quotation marks around a string of metacharacters prevent the shell from interpreting the special meaning of the metacharacters.

There are two types of quotation marks that mask the special meaning of metacharacters, single forward quotation marks (′ ′) and double quotation marks (" ").

Single forward quotation marks instruct the shell to ignore all enclosed metacharacters.

Double quotation marks instruct the shell to ignore all enclosed metacharacters except the following three characters: `$ ‘ \`

A backslash (\) character in front of a metacharacter prevents the shell from interpreting the next character as a metacharacter.

To ignore the special meaning of the dollar sign metacharacter, enter the following:

```
$ echo ′$SHELL′
$SHELL
```

Note – The echo utility writes arguments to the standard output.

To interpret the special meaning of the dollar sign metacharacter, enter the following:

```
$ echo "$SHELL"
/bin/ksh
```

To ignore the special meaning of the dollar sign metacharacter when you use double quotes, enter the following:

```
$ echo "\$SHELL"
$SHELL
```

Single backward quotation marks instruct the shell to execute and display the output for a UNIX® system command.

To execute and display the output for the `date` command, enter the following:

```
$ echo "Today's date is 'date'"
Today's date is Tue May 2 14:10:05 MDT 2002
$
```

To execute the `pwd` command using parentheses, enter the following:

```
$ echo "The user is currently in the $(pwd) directory."
The user is currently in the /export/home/user1 directory.
$
```

Using the Redirection Metacharacters

You can redirect input to and output from commands by using redirection metacharacters. There are three redirection metacharacters, the greater than (>) character, the less than (<) character, and the pipe (|) character.

The shell typically receives or reads command input from the keyboard and displays or writes command output to the screen. Figure 8-2 shows standard input and output.

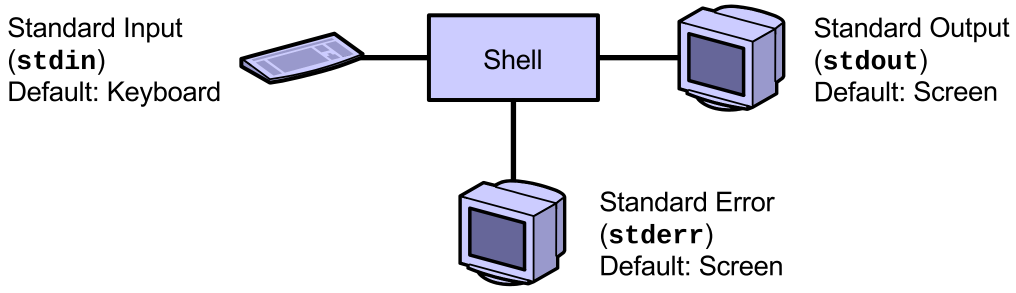


Figure 8-2 Standard Command I/O in the Shell

You can instruct the shell to redirect command input from and command output to files or other commands on the command line or within shell scripts. Input redirection forces a command to read input from a file instead of from the keyboard. Output redirection sends output from a command into a file or to another command instead of sending the output to the screen.

Error messages are sent to standard error as the messages are generated by the command. Usually error messages are sent to the terminal screen.

The File Descriptors

Each process that the shell creates works with file descriptors. File descriptors determine where input to the command originates and where output and error messages are sent. Table 8-1 shows the file descriptors.

Table 8-1 File Descriptors

File Descriptor Number	**File Description Abbreviation**	**Definition**
`0`	`stdin`	Standard command input
`1`	`stdout`	Standard command output
`2`	`stderr`	Standard command error

All commands that process file content read from standard input and write to standard output.

The following example shows standard input (bold text) and standard output (plain text) for the `cat` command. Press Control-D to exit the `cat` command.

```
$ cat(Read from stdin)
First line(Read from stdin)
First line(Write to stdout)
What’s going on?(Read from stdin)
What’s going on?(Write to stdout)
Control-d (Read from stdin)
$
```

The `cat` command takes standard input from the keyboard and sends standard output to the terminal window.

You can modify the default action of standard input, standard output, and standard error within the shell by redirecting `stdin`, `stdout`, and `stderr`.

Redirecting Standard Input

The following format example shows a command using the less than (<) metacharacter to process a file as standard input instead of reading input from the keyboard.

```
command < filename
```

or

```
command 0< filename
```

To use the `dante` file as input for the `mailx` command, enter the following:

```
$ mailx user1 < ~/dante
```

Redirecting Standard Output

The following format example shows a command using the greater than (>) metacharacter to direct standard output to a file instead of printing the output to the screen. If file does not exist, the system creates it. If the file exists, the redirection overwrites the contents of the file.

```
command > filename
```

or

```
command 1> filename
```

To list all the current processes running on the system and redirect that list of processes into a file called `process_list`, enter the following:

```
$ ps -ef > process_list
```

The following example shows a command using two greater than (>>) metacharacters to direct standard output to the end of existing content in a file. If the file does not exist, the system creates it.

```
command >> filename
```

To append `That's my process file` to the end of the `my_file` file, enter the following:

```
$ ps > my_file; cat my_file
   PID TTY      TIME CMD
   663  pts/5    0:01 ksh
$ echo "That's my process file" >> my_file; cat my_file
   PID TTY      TIME CMD
   663  pts/5    0:01 ksh
That's my process file
$
```

Note – The semicolon (`;`) is a shell metacharacter that allows multiple commands on a single command line.

Redirecting Standard Error

The following example shows a command using the file descriptor number (2) and the greater than (>) metacharacter to redirect any standard error messages to the file `/dev/null`. This redirection is useful to suppress error messages so no error messages appear on the screen.

command 2> `/dev/null`

This example of how to direct error messages to the `/dev/null` file lists the errors you would see if not for the redirection of `stderr`.

```
$ find /etc -type f -exec grep PASSREQ {} \; -print
$   find /etc -type f -exec grep PASSREQ {} \; -print
# PASSREQ determines if login requires a password.
PASSREQ=YES
/etc/default/login
grep: can't open /etc/inet/mipagent.conf-sample
grep: can't open /etc/inet/mipagent.conf.fa-sample
grep: can't open /etc/inet/mipagent.conf.ha-sample
grep: can't open /etc/security/dev/audio
grep: can't open /etc/security/dev/fd0
<output truncated>

$ find /etc -type f -exec grep PASSREQ {} \; -print 2> /dev/null
# PASSREQ determines if login requires a password.
PASSREQ=YES
/etc/default/login
$
```

The `-print` expressions cause the current path name to print to the screen.

The following example shows a command redirecting standard output to a file and standard error to the same file. The syntax `2>&1` instructs the shell to redirect `stderr` (2) to the same file that receives `stdout` (1).

command `1>` *filename* `2>&1`

To print standard output and standard error to the `dat` file, enter the following:

```
$ ls /var /test 1> dat 2>&1
$ more dat
/test: No such file or directory        (stderr)
/var:                                   (stdout)
adm                                     (stdout)
audit                                   (stdout)
cron                                    (stdout)
<remaining output omitted>
```

The Pipe Character

The following example shows a command using the pipe (`|`) character to redirect standard output to the standard input of another command.

command `|` *command*

You can insert a pipe between any two commands if the first command writes output to standard output and the second command reads input from standard input.

To use the standard output from the `who` command as standard input for the `wc -l` command, enter the following:

```
$ ls | wc -l
        35
$
```

The output of the `who` command never appears on the terminal screen because it is piped directly into the `wc -l` command.

To view a list of all the subdirectories located in the `/etc` directory, enter the following:

```
$ ls -F /etc | grep "/"
acct/
cron.d/
default/
<output omitted>
```

You can use pipes to connect numerous commands.

To use the output of the `head` command as input for the `tail` command and print the results, enter the following:

```
$ head -10 dante | tail -3 | lp
request id is printerA-177       (Standard input)
```

To use the output of the `ps` command as input for the `tail` command and list the word count, enter the following:

```
$ ps -ef | tail +2 | wc -l
      74
```

Displaying the Command History

The shell keeps a history of recently entered commands. This history mechanism enables you to view, repeat, or modify previously executed commands.

Note – Command history is shared among all Korn shells for a given user.

Using the `history` Command

The `history` command displays previously executed commands. By default, the `history` command displays the last 16 commands to standard output.

The format for the `history` command is:

```
history option
```

To display previously executed commands, enter the following:

```
$ history
...
87    date
88    cd /etc
89    touch dat1 dat2
90    ps -ef
91    history
```

Note – Your output varies based on which commands you recently entered in your terminal window.

The numbers on the left are command numbers. You can use the command numbers to instruct the shell to re-execute a particular command line.

To display the command history without line numbers, enter the following:

```
$ history -n
    ...
    date
    cd /etc
    touch dat1 dat2
    ps -ef
    history
```

To display the current command and the four commands preceding it, enter the following:

```
$ history -4
...
107    date
108    cd /etc
109    touch dat1 dat2
110    ps -ef
111    history
```

To display the history list in reverse order, enter the following:

```
$ history -r
111    history
110    ps -ef
109    touch dat1 dat2
108    cd /etc
107    date
...
```

To display the most recent `cd` command to the most recent `ls` command, enter the following:

```
$ history cd ls
31  cd
32  man lp
33  date
34  ls
<output omitted>
```

Note – The Korn shell stores command history in a file specified by the `HISTFILE` variable. The default file is `$HOME/.sh_history`. You can use the `HISTSIZE` variable to specify the number of commands to store in this buffer. If this variable is not set, then the shell stores the most recent 128 commands.

Using the r Command

The `r` command is an alias built into the Korn shell that enables you to repeat a command.

To repeat the `cal` command using the `r` command, enter the following:

```
$ cal
    May 2002
 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 29 30 31


$ r
cal
    May 2002
 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 29 30 31
```

To re-execute a command by line number using the `r` command, enter the line number, for example, 160:

```
$ history
155 history
156 cat dante
157 history
158 date
159 cal
160 ls
161 cd
r 160
```

You can also use the `r` command to re-execute a command beginning with a particular character, or string of characters.

To rerun the most recent occurrence of a command that begins with the letter "c," enter the following:

```
$ r c
cd /etc
```

To rerun the most recent occurrence of the ps command, enter the following:

```
$ r ps
ps -ef
```

You can use the r command to repeat a previous command, perform a simple edit, and execute the modified command.

For example, to repeat the most recent occurrence of a command beginning with the letter "c," replace dir1 with dir2, and execute the modified command, enter the following:

```
$ history
...
100 cd
101 cat dante
102 ls
103 cd ~/dir1
104 history
$
$ r c
cd ~/dir1
$ r dir1=dir2
cd ~/dir2
```

Editing and Rerunning Previously Executed Commands

You can edit previously executed commands and rerun these commands using a shell in-line editor.

Use the `vi` editor to turn on and enable the shell history editing feature with one of the following commands:

```
$ set -o vi
```

or

```
$ export EDITOR=/bin/vi
```

or

```
$ export VISUAL=/bin/vi
```

Note – All three commands turn on `vi` command-line editing for the shell. To verify this, you can use the `set -o` command.

You can access a command in the history buffer, edit the command with the `vi` editor, and execute the modified command by following these steps:

1. Verify that the built-in `vi` editor is enabled.

```
$ set -o | grep -w vi
vi              on
```

2. Type the `history` command to view the command history list.

```
$ history
```

3. Press Escape to access the command history list.

   Use the following keyboard keys to move the cursor through the command history.

   - `k` – Moves the cursor up one line at a time
   - `j` – Moves the cursor down one line at a time
   - `l` – Moves the cursor to the right
   - `h` – Moves the cursor to the left

Note – You cannot use the arrow keys to move the cursor within the command history.

4. Use the `vi` commands to edit any previously executed command.
5. To execute a modified command, press Return.

Introducing User Initialization Files

You can modify three initialization files in your home directory to customize your working environment.

When you log in, the Korn shell looks for an environmental initialization file called `.profile`. The `.profile` file contains the `ENV` variable.

Note – You can choose and set any name for the `ENV` variable.

The shell executes the commands in the `.profile` file first and the commands in the `ENV` variable next.

In addition, the shell can be directed to a file that has been customized specifically for the Korn shell. This file is typically named `.kshrc`. Then the shell prompt (`$`) appears on the screen, and the Korn shell waits for commands.

The Common Desktop Environment (CDE) reads another initialization file, the `.dtprofile` file, upon execution. You can modify this configuration file to customize the desktop environment.

The `~/.profile` File

The `.profile` file is an initialization file that you define in your home directory. The login shell executes the `.profile` file when you log in. You can customize environment variables and terminal settings in the `.profile` file to modify your working environment. You can also instruct the system to initiate applications in the `.profile` file.

Note – The `/etc/profile` file is a separate systemwide file that the system administrator maintains. This file sets up tasks that the Korn shell executes for every user who logs in.

The `~/.kshrc` File

The `.kshrc` file in your home directory contains shell variables and aliases. The system executes the `.kshrc` file every time it starts a `ksh` subshell.

You typically set the following items in the `.kshrc` file:

- Shell prompt definitions (`PS1`, `PS2`)
- Alias definitions
- Shell functions
- History variables
- Shell options (`set -o` *`option`*)

When you make changes to your individual initialization files, the changes take effect the next time you log in. However, if you want the changes to take effect immediately, you can source the `.profile` file and the `.kshrc` file using the dot (`.`) command.

```
$ .  ~/.profile
$
$ .  ~/.kshrc
```

Note – Sourcing or reading the `.profile` file and the `.kshrc` file only updates the current shell.

The `~/.dtprofile` File

Another initialization file, called `.dtprofile`, resides in your home directory. The `.dtprofile` file determines generic and customized settings for the CDE. Your variable settings in the `.dtprofile` file can overwrite any default CDE settings.

The CDE uses this default file to generate a `.dtprofile` file for your home directory the first time you log in to the CDE.

Note – If the `DTSOURCEPROFILE` variable is set to `true,` then the `dtlogin` causes the shell to read the `.profile` file. If this variable does not exist or it is not set to `true,` then the `.profile` file is not read by the shell.

Each time you log in to the CDE, the shell reads the `.dtprofile` file and then your `.profile` file and your `.kshrc` file. The shell reads the `.profile` and `.kshrc` files again when you open a console window. The shell also reads the `.kshrc` file when you open a terminal window in the CDE.

Exercise: Accessing Files and Directories

In this exercise, answer the questions on the material presented in this module.

Tasks

Complete the following steps. Write your answer in the space provided.

1. Which specific shell characters have special meaning to the shell?

   ____

2. Name some common shell metacharacters.

   ____

3. Which metacharacter is a shell substitute for the home directory of a user?

   ____

4. Which two metacharacters are often referred to as wild card characters?

   ____

5. Which character do you use to match a single character, excluding a leading period?

   ____

6. Which character(s) would you use to match a set or range of characters?

   ____

7. Which character(s) would you use to have the shell ignore the special meaning of metacharacters?

   ____

8. Which symbol(s) do you use to redirect output and append the output to a file?

   ____

9. Which symbol or character do you use to connect two or more commands on a single command line?

   ____

10. Which command do you use to display a list of previously executed commands in the shell?

    ___

11. List the three user initialization files described in this module.

    ___

    ___

    ___

Exercise Summary

Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

Manage the discussion based on the time allowed for this module, which was provided in the "About This Course" module. If you do not have time to spend on discussion, then just highlight the key concepts students should have learned from the lab exercise.

- Experiences

Ask students what their overall experiences with this exercise have been. Go over any trouble spots or especially confusing areas at this time.

- Interpretations

Ask students to interpret what they observed during any aspect of this exercise.

- Conclusions

Have students articulate any conclusions they reached as a result of this exercise experience.

- Applications

Explore with students how they might apply what they learned in this exercise to situations at their workplace.

Exercise Solutions

1. Which specific shell characters have special meaning to the shell?

   metacharacters

2. Name some common shell metacharacters.

   `~ - + * ? []`

3. Which metacharacter is a shell substitute for the home directory of a user?

   `~`

4. Which two metacharacters are often referred to as wild card characters?

   * *and* **`?`**

5. Which character do you use to match a single character, excluding a leading period?

   `?`

6. Which character(s) would you use to match a set or range of characters?

   `[]`

7. Which character(s) would you use to have the shell ignore the special meaning of metacharacters?

   `\ ' "`

8. Which symbol(s) do you use to redirect output and append the output a file?

   `>>`

9. Which symbol or character do you use to connect two or more commands on a single command line?

   `|`

10. Which command do you use to display a list of previously executed commands in the shell?

    The `history` *command*

11. List the three user initialization files described in this module.

    `~/.profile`
    `~/.kshrc`
    `~/.dtprofile`

Module 9

Performing Basic Process and Job Control

Objectives

Upon completion of this module, you should be able to:

- Describe a process
- View a process
- Search for a specific process
- Send a signal to a process
- Manage a job in the Korn shell

The following course map shows how this module fits into the current instructional goal.

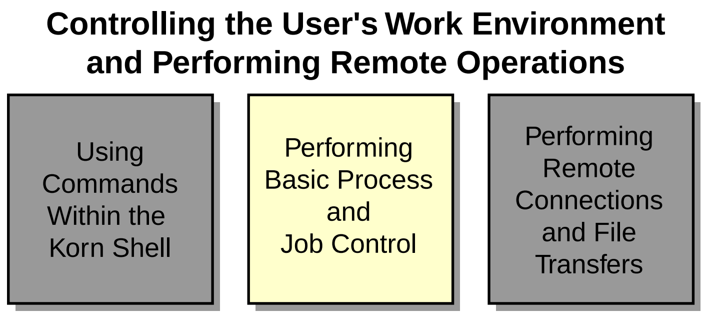

Controlling the User's Work Environment and Performing Remote Operations
Using Commands Within the Korn Shell
Performing Basic Process and Job Control
Performing Remote Connections and File Transfers

Figure 9-1 Course Map

Introducing Solaris™ Operating Environment Processes

Every program you run in the Solaris Operating Environment (Solaris OE) creates a process. When you log in and start the shell, you start a process. When you execute a command or when you open an application, you start a process.

The system starts processes that are called daemons. Daemons are processes that run in the background and provide services. For instance, the desktop login daemon (`dtlogin`) provides a graphical prompt that you use to log in.

Using a PID

Every process is assigned a unique process identification number (PID), which the `kernel` uses to track and manage the process. You can use PID to identify and control processes.

Using Process UID and GID Numbers

A user identification number (UID) and a group identification number (GID) is associated with each process. These numbers indicate who owns a process and determine what a process can do. Generally the UID and GID associated with a process are the same as the UID and GID of the user who started the process.

Understanding the Parent Process

When one process creates another, the first process is considered to be the parent of the new process. The new process is called the child process.

While the child process runs, the parent process waits. When the child finishes its task, it informs the parent process. The parent process then terminates the child process. If the parent process is an interactive shell, a prompt appears, ready for a new command.

Viewing a Process

You use the process status (ps) command to list the processes that are scheduled to run on the system. The ps command has several options you can use to determine which processes to display and how to format the output.

Using the ps Command

The format for the ps command is:

```
ps options
```

For each process, the ps command displays the PID, the terminal identifier (TTY), the cumulative execution time (TIME), and the command name (CMD).

Identifying the ps Options

Table 9-1 shows some of the options you can use with the ps command:

Table 9-1 Options for the ps Command

Option	**Description**
`-e`	Prints information about every process on the system, including the PID, TTY, TIME, and CMD
`-f`	Generates a full (verbose) listing, which adds fields including the UID, parent process identification number (PPID), and process start time (STIME)

Note – Refer to the ps command man page for a complete list of options.

Displaying a Listing of All Processes

You use the `ps -ef` command to see a listing of all the processes currently scheduled to run on the system. The following example uses the `ps -ef` command to display a listing of all processes.

```
$ ps -ef | more
UID PIDPPIDCSTIMETTYTIMECMD
root0  0  0  16:46:41?0:01sched
root1  0  0  16:46:44?0:40/etc/init -
root2  0  0  16:46:44?0:00pageout
root3  0  0  16:46:44?4:33fsflush
root2361  0  16:48:08?0:01/usr/lib/saf/sac
root8441  0  12:12:10?0:00/usr/lib/lpsched
--More--
```

Table 9-2 shows and defines the values in the output of the `ps -ef` command.

Table 9-2 Output for the `ps -ef` Command

Value	**Description**
`UID`	The user name of the owner of the process.
`PID`	The unique process identification number of the process.
`PPID`	The parent process identification number of the process.
`C`	The central processing unit (CPU) utilization for scheduling. This value is obsolete.
`STIME`	The time the process started (*`hh:mm:ss`*).
`TTY`	The controlling terminal for the process. Note that system processes (daemons) display a question mark (`?`), indicating the process started without the use of a terminal.
`TIME`	The cumulative execution time for the process.
`CMD`	The command name, options, and arguments.

Searching for a Specific Process

In the Solaris OE, you need to be able to search for processes that are running on the system. You can use the `ps` and `grep` commands together, or the `pgrep` command alone to search for specific processes.

Using the ps and grep Commands

To search for a specific process, you can combine the `ps` and `grep` commands using the pipe (`|`) character. The pipe character causes the `ps` command to send the list of processes to the `grep` command. The `grep` command then searches the list of processes for the text that you specify.

For example, to find active processes with names that contain the string `lp`, enter the following:

```
$ ps -e | grep lp
217 ?        0:00 lpsched
$
```

Note – The `lpsched` daemon is responsible for controlling print services on the system.

Using the pgrep Command

You can use the `pgrep` command to search for specific processes by name.

By default, the `pgrep` command displays the PID of every process that matches the criteria you specify on the command line.

The format for the `pgrep` command is:

```
pgrep options pattern
```

Table 9-3 shows some of the options you can use with the `pgrep` command.

Table 9-3 Options for the `pgrep` Command

Option	**Description**
`-x`	Displays the PIDs that match the pattern exactly.
`-n`	Displays only the most recently created PID that contains the pattern.
`-U` *`uid`*	Displays only the PIDs that belong to the specified user. This option uses either a user name or a UID.
`-l`	Displays the name of the process along with the PID.
`-t` *`term`*	Displays only those processes that are associated with a terminal in the `term` list.

Note – See the man page for the `pgrep` command to view all the options for the command.

To display the PID of any process with a name that contains the string `lp`, enter the `pgrep` command as follows:

```
$ pgrep lp
217
$
```

To display the PID and name of any process with a name that contains the string `lp`, enter the `pgrep` command with the `-l` option as follows:

```
$ pgrep -l lp
    217   lpsched
```

To display the PID and name of any process with a name that contains the string `mail`, enter the `pgrep` command with the `-l` option as follows:

```
$ pgrep -l mail
     230  sendmail
    13453  dtmail
```

Note – The `dtmail` process only appears if you have the CDE mail process running; it does not run by default.

To display the PID and name of any process associated with a terminal window, enter the `pgrep` command with the `-lt` options. For example:

```
$ pgrep -lt pts/2
  697 ksh
```

You use the `pgrep` command with the `-xl` options to display only the processes with names that exactly match the string you specify. For example:

```
$ pgrep -l mountd
  155 automountd
$ pgrep -xl mountd
$ pgrep -xl automountd
  155 automountd
```

Sending a Signal to a Process

A signal is a message that you can send to a process. Processes respond to signals by performing the action that the signal requests. Signals are identified by a signal number and by a signal name, and each signal has an associated action.

For example, signal 2 is known as the `SIGINT` signal. If you press Control-C, you send the `SIGINT` signal to the process running in your current terminal window. The process responds by exiting. Generally, you use either the `kill` command or the `pkill` command to send signals to processes.

Table 9-4 shows some of the available signals.

Table 9-4 Signal Numbers and Names

Signal Number	Signal Name	Event	Definition	Default Response
1	`SIGHUP`	Hang up	A hang-up signal that drops a telephone line or terminal connection. This signal also causes some programs to re-initialize themselves without terminating.	Exit
2	`SIGINT`	Interrupt	An interrupt signal you generate from your keyboard, usually using Control-C.	Exit
9	`SIGKILL`	Kill	A signal that kills a process. A process cannot ignore this signal.	Exit
15	`SIGTERM`	Terminate	A signal that terminates a process in an orderly manner. Some processes ignore this signal. This is the default signal that `kill` and `pkill` send.	Exit

Note – See the `-s3head signal` man page to view a complete list of signals and their default actions.

Terminating Using the `kill` Command

You use the `kill` command to send a signal to one or more processes. You can use the `kill` command only on processes that you own. The `root` user can use the `kill` command on any process. The `kill` command sends signal 15, the terminate signal, by default. This signal causes the process to terminate in an orderly manner.

The format for the `kill` command is:

```
kill [-signal] PID(s)
```

Before you can terminate a process using the `kill` command, you must know what PID to specify. Use either the `ps` command or the `pgrep` command to locate the PID for the process you want to terminate. You can terminate several processes at the same time by entering multiple PIDs on one command line.

For example, to terminate the `dtmail` process using the `kill` command, enter the following:

```
$ pgrep -l mail
   215  sendmail
 12047  dtmail
$ kill 12047
$ pgrep -l mail
   215  sendmail
```

Note – The `dtmail` process is only present when the CDE mail tool is running. This process is not present by default.

Some processes ignore the default signal that the `kill` command sends. For example, a process waiting for a tape drive to complete an operation might ignore signal 15.

You can force processes that do not respond to signal 15 to terminate by using signal 9 with the `kill` command. To terminate a process using signal 9, enter the following:

```
$ kill -9 PID
```

Caution – Use the `kill -9` command only when necessary. When you use the `kill -9` command on an active process, the process terminates instantly instead of performing an orderly shutdown. Using signal 9 on processes that control databases or programs that update files could cause data corruption.

Terminating Using the `pkill` Command

You use the `pkill` command to send signals to processes. The `pkill` command sends signal 15, the terminate signal, by default. The `pkill` command allows you to use process names to identify the process you want to terminate.

The format for the `pkill` command is:

```
pkill [-options] pattern
```

The options for the `pkill` command are similar to those for the `pgrep` command.

For example, to terminate the mail session process using the `dtmail` command name, enter the following:

```
$ pgrep -l mail
 215 sendmail
 470 dtmail
$ pkill dtmail
$ pgrep -l mail
 215 sendmail
$
```

You can force processes that do not respond to signal 15 to terminate by using signal 9 with the `pkill` command.

To terminate a process using signal 9, enter the following:

```
$ pkill -9 -x process_name
```

Managing a Job in the Korn Shell

This section introduces job management in the Korn shell. The Korn shell lets you run jobs in the background, which frees your terminal to execute other commands. You can also run jobs in the foreground. You can use Korn shell commands to list current jobs and to stop a job.

Introducing Jobs

A job is a process that the shell can manage. Shells start and control jobs. Because jobs are processes, each job has an associated PID. The shell also assigns each job a sequential job ID number.

The shell allows you to run multiple jobs at the same time. Job control commands allow you to manage multiple jobs within a shell. There are three types of jobs that shells manage: foreground jobs, background jobs, and stopped jobs.

When you execute a command in a terminal window, the command occupies that terminal window until it completes. This type of job is called a foreground job.

When you enter an ampersand (&) symbol at the end of a command line, the command runs without occupying the terminal window. The shell prompt is displayed immediately after you press Return. This type of job is called a background job.

If you press Control-Z for a foreground job, or use the `stop` command for a background job, the job stops. This is called a stopped job.

Job control commands allow you to place jobs in the foreground or background, and to start or stop jobs. Table 9-5 shows the commands you use for job control.

Table 9-5 Job Control Commands

Command	**Value**
`jobs`	Lists all jobs that are currently running or are stopped in the background
`bg %`*`n`*	Runs the current or specified job in the background (*`n`* is the job ID)

Table 9-5 Job Control Commands (Continued)

Command	Value
`fg %n`	Brings the current or specified job into the foreground (*n* is the job ID)
Control-Z	Stops the foreground job and places it in the background as a stopped job
`stop %n`	Stops a job that is running in the background (*n* is the job ID)

Note – You can control a job by using these commands only in the shell in which the job started.

Running a Job in the Background

To run a job in the background, enter the command you want to run along with an ampersand (&) symbol at the end of the command line. For example, to run the `sleep` command in the background, enter the following:

```
$ sleep 500 &
[1]     3028
$
```

Note – The `sleep` command suspends execution of a program for *n* seconds.

The shell returns the job ID number it assigned to the command, contained in brackets, and the PID associated with the command. You use the job ID number to manage the job with job control commands. The `kernel` can use the PID number to manage the job.

When a background job has finished and you press Return, the shell displays a message indicating that the job is done.

```
[1] +  Done                    sleep 500 &
$
```

Listing Current Jobs

You use the `jobs` command to display the list of jobs that are running or stopped in the background. For example:

```
$ jobs
[1] +  Running                 sleep 500 &
$
```

Bringing a Background Job Into the Foreground

You use the `fg` command to bring a background job to the foreground. For example:

```
$ fg %1
sleep 500
```

Note – The foreground job occupies the shell until the job is completed, stopped, or stopped and placed into the background.

Sending a Foreground Job to the Background

You use the Control-Z keys and `bg` command to return a job to the background. The Control-Z keys suspend the job, and place it in the background as a stopped job. The `bg` command runs the job in the background. For example:

```
$ sleep 500
^Z[1] + Stopped (SIGTSTP)        sleep 500
$ jobs
[1] + Stopped (SIGTSTP)        sleep 500
$ bg %1
[1]     sleep 500&
$ jobs
[1] +  Running                 sleep 500
$
```

Note – When you place a stopped job in either the foreground or the background the job restarts.

Stopping a Background Job

To stop a background job, use the `stop` command and specify the job ID number of the job you want to stop. For example:

```
$ jobs
[1] +  Running                  sleep 500&
$ stop %1
$ jobs
[1] + Stopped (SIGSTOP)        sleep 500&
$
```

You can terminate a job using the `kill` command. For example, to terminate the job running the `sleep` command, enter the following:

```
$ jobs
[1] + Stopped (SIGSTOP)        sleep 500&
$ kill %1
[1] + Terminated               sleep 500&
$ jobs
$
```

Exercise: Manipulating System Processes

In this exercise, you use the commands described in this module to determine PIDs, kill processes, and control jobs.

This exercise introduces you to the `tty` command, which displays the name of the current terminal window. The name displayed by the `tty` command includes a unique identification number assigned by the Solaris OE to each open terminal window; for example: `/dev/pts/2`. In the tasks illustrating the `tty` command, the unique identification number is displayed as `/dev/pts/`*`n`*, where *`n`* is a numeral.

Tasks

Complete the following steps. Write the commands that you would use to perform each task in the space provided.

1. Issue the following command in the background:
   ```
   $ sleep 500 &
   ```
2. Find the job number of the `sleep` command in Step 1.

   Job number:____________________________________
3. Bring the job to the foreground, and then put it back in the background.

   ____
4. Terminate the job running the `sleep` command.

   ____
5. Use the following `ps` commands to list the processes currently running on your system. What information does each command provide?
   ```
   $ ps
   ```
   ____
   ```
   $ ps -f
   ```
   ____
   ```
   $ ps -e
   ```
   ____
   ```
   $ ps -ef
   ```
   ____

6. In a terminal window, run the `ps -ef` command. Identify the PID of this command.

   ```
   $ ps -ef
   ```

   PID: ___________________________________

7. In a separate terminal window, issue the following commands:

   ```
   $ cd /
   $ ls -lR *
   ```

Note – This command produces a continuously running process for demonstration purposes only.

---

8. Open another terminal window. Use the `ps` command to identify the PID of the `ls` command.

   PID: ___________________________________

9. From this terminal window, terminate the `ls` command using the PID.

   ____

10. From the same window, enter the `tty` command to identify the name of this terminal window. The name appears as `/dev/pts/`*`n`*, where *`n`* is a numeral; for example, `/dev/pts/4`.

    ```
    $ tty
    ```

    `/dev/pts/`____

11. Move back to the original terminal window. Use the `pgrep` command to find the PID associated with the name of the second terminal window.

    ```
    $ pgrep -t pts/n
    ```

    PID: ___________________________________

12. In the current window, use the `kill` command to terminate the second terminal window.

    ```
    $ kill PID
    ```

    Did it work? ___________________________________

13. Use the `kill` command with the `-9` option to terminate the second terminal window.

    ```
    $ kill -9 PID
    ```

    Did it work? ___________________________________

14. Run the following `kill` commands to identify the signal names associated with signal numbers.

    $ **kill -l 9** *(`-l` is the letter l)*

    signal: ______________________________

    $ **kill -l 15** *(`-l` is the letter l)*

    signal: ______________________________

Exercise Summary

Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

Manage the discussion based on the time allowed for this module, which was provided in the "About This Course" module. If you do not have time to spend on discussion, then just highlight the key concepts students should have learned from the lab exercise.

- Experiences

Ask students what their overall experiences with this exercise have been. Go over any trouble spots or especially confusing areas at this time.

- Interpretations

Ask students to interpret what they observed during any aspect of this exercise.

- Conclusions

Have students articulate any conclusions they reached as a result of this exercise experience.

- Applications

Explore with students how they might apply what they learned in this exercise to situations at their workplace.

Exercise Solutions

1. Issue the following command in the background:

```
$ sleep 500 &
```

2. Find the job number of the `sleep` command started in Step 1.

```
$ jobs
```

The job number differs from system to system.

3. Bring the job to the foreground, and then put it back in the background.

```
$ fg %1
^Z
$ jobs
$ bg %1
```

4. Terminate the job running the `sleep` command.

```
$ kill %1
```

5. Use the following `ps` commands to list the processes currently running on your system. What information does each command provide?

```
$ ps
```

This command prints information for the current user and terminal.

```
$ ps -f
```

This command prints a full listing of the `ps` *command.*

```
$ ps -e
```

This command prints information about every process running.

```
$ ps -ef
```

This command prints a full listing of the `ps -e` *`command`.*

6. In a terminal window, run the `ps -ef` command. Identify the PID of this command.

The PID differs from system to system.

7. In a separate terminal window, issue the following commands:

```
$ cd /
$ ls -lR *
```

8. Open another terminal window. Use the `ps` command to identify the PID of the `ls` command.

```
$ ps -ef | grep ls
```

9. From this terminal window, kill the `ls` command using the PID.

   ```
   $ kill PID
   ```

   where `PID` *is the PID of the* `ls` *command.*

10. From the same window, enter the `tty` command to identify the name of this terminal window. The name appears as `/dev/pts/n`, where `n` is a numeral; for example, `/dev/pts/4`.

    ```
    $ tty
    ```

    `/dev/pts/n` *– This name differs from system to system.*

11. Move back to the original terminal window. Use the `pgrep` command to find the PID associated with the name of the second terminal window.

    ```
    $ pgrep -t pts/n
    ```

    The PID differs from system to system.

12. In the current window, use the `kill` command to terminate the second terminal window.

    ```
    $ kill PID
    ```

    Did it work?

    No.

13. Use the `kill` command with the `-9` option to terminate the second terminal window.

    ```
    $ kill -9 PID
    ```

    Did it work?

    `Yes.`

14. Run the following `kill` commands to identify the signal names associated with signal numbers.

    ```
    $ kill -l 9
    ```

    signal: `KILL`

    ```
    $ kill -l 15
    ```

    signal: `TERM`

Module 10

Performing Remote Connections and File Transfers

Objectives

Upon completion of this module, you should be able to:

- Establish a remote login session on another system
- Copy files or directories to and from another system
- Transfer files between systems

The following course map shows how this module fits into the current instructional goal.

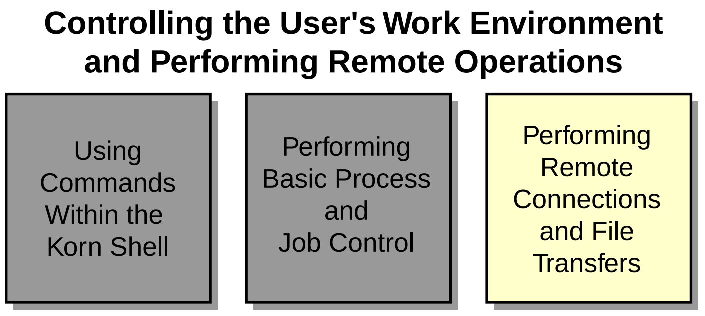

Controlling the User's Work Environment and Performing Remote Operations
Using Commands Within the Korn Shell
Performing Basic Process and Job Control
Performing Remote Connections and File Transfers

Figure 10-1 Course Map

Establishing a Remote Login Session

A remote login session enables you to access a remote host from your current machine and to use the resources of the remote host. This section explains how to log in to your account remotely from another system on a network.

Introducing an Example Network

A network is a connection of computer systems that enables an exchange of information among the systems. A network also enables you to access your user account from another system. Two types of networks are:

- Local area network (LAN) – A network that covers a small area, usually less than a few thousand feet or meters
- Wide area network (WAN) – A network that can span thousands of miles or kilometers

A host is a computer system on a network. The local host is your current working system. A remote host is a different system that you access from your local host.

Note – Servers and clients are two types of hosts in a distributed computing environment. A server is a process or program that provides services to hosts on a network. A client is both a process that uses services provided by the server and a host that runs a client process.

Figure 10-2 shows the relationship between the network and the host.

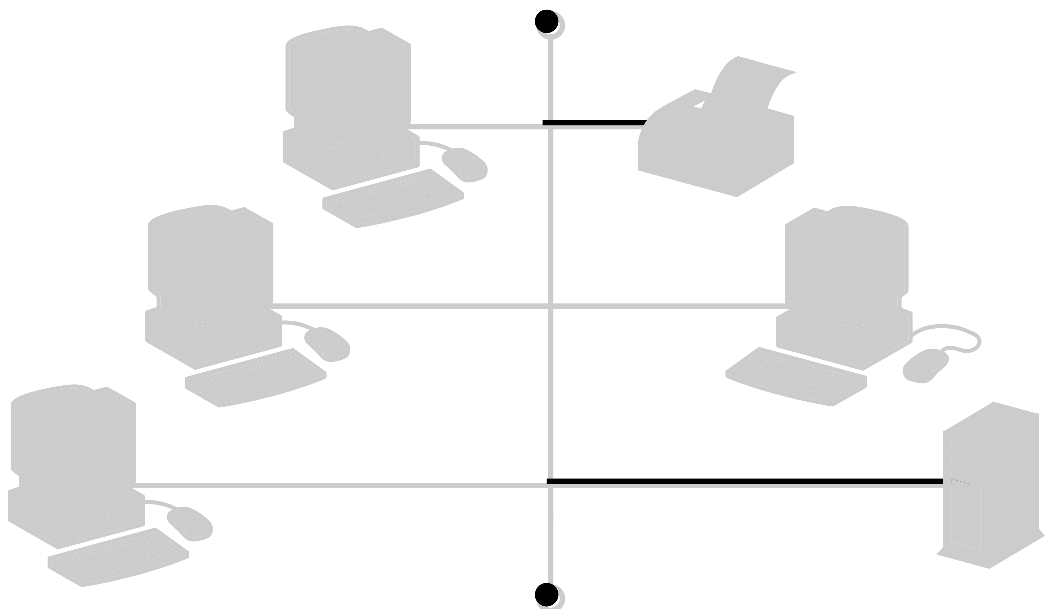


Figure 10-2 Example Network

If you are not running a naming service, the host name and IP address should be in the `/etc/inet/hosts` file.

If you do not have a home directory on the remote system, the `/` (root) directory automatically becomes your home directory when you log in.

Introducing the `~/.rhosts` File

When a remote user requests login access to a local host, the local host searches the local `/etc/passwd` file for an entry for that user. If an entry for the user exists, the user can log in to the local host from a remote system. If a password is associated with the account, the remote user supplies the password to gain system access. If no entry exists in the local `/etc/passwd` file for the remote user, the remote user cannot access the system.

The `~/.rhosts` file provides another authentication procedure to determine if a remote user can access the local host with the identity of a local user. This procedure bypasses the password authentication. The system checks the `~/.rhosts` file in the home directory of the local host user to determine if the remote user can access the system.

Note – The user's `.rhosts` file can contain a plus (+) character, which gives the user the ability to log in from any known system on the network without supplying a password.

Using the `rlogin` Command

You use the `rlogin` command to establish a remote login session on another system.

The format for the `rlogin` command is:

```
rlogin hostname
```

The following example shows you how to use the `rlogin` command to log in to another system remotely. You use the `uname -n` command to verify the name of the remote system.

```
$ rlogin host2
Last login: Mon Mar  6 16:22:12 from host1
Sun Microsystems Inc.   SunOS 5.9       Generic February 2000
$ uname -n
host2
$ pwd
/export/home/user1
$ exit
Connection to host2 closed.
$
```

You can use the `rlogin` command with the `-l` option to specify a different user name for a remote login session.

The format for the `rlogin -l` command is:

```
rlogin -l username hostname
```

To log in as a different user, you use the following information to identify and log in to the account:

- The host name
- The user name
- The password for the user on the remote host

The following example shows how to log in remotely to the `host2` system as another user, `user2`:

```
$ rlogin -l user2 host2
Password:
Last login: Mon Mar  6 16:36:35 from host2
Sun Microsystems Inc.   SunOS 5.9       Generic February 2000
<output omitted>
$ pwd
/export/home/user2
$ uname -n
host1
$ exit
Connection to host2 closed.
$
```

Running a Program on a Remote System

You use the `rsh` command to run a program on a remote system.

The format for the `rsh` command is:

```
rsh hostname command
```

or

```
rsh -l username hostname command
```

Note – The `rsh` command format also allows you to use the system's *`IP-Address`* on the command line instead of *`hostname`*.

The following example shows how to use the `rsh` command as `user1` to run the `ls` command remotely on the `host2` system.

```
$ rsh host2 ls
```

You can use the `-l` *`username`* option with the `rsh` command to specify a different user name. This requires that the `~/.rhosts` file is configured with an entry for this user name. The following example shows how `user1` runs the `rsh -l` command as `user2` to invoke the `ls /var/mail` command on the remote system named `host2`.

```
$ rsh -l user2 host2 ls /var/mail
```

The output from these commands shows information from the `host2` system.

Note – The `rsh` command only works if a `.rhosts` file exists for the user because `rsh` does not prompt for a password to authenticate the new users.

Terminating a Process Remotely by Logging In to Another System

If your system is not responding to your keyboard or to mouse input, the window system might be frozen. You can use the `rlogin` command to access your system remotely from another system. Then you can enter the `pkill` command to terminate the corrupted session.

The following example shows how to use the `rlogin` command to terminate a process remotely on the `host2` system.

```
$ rlogin host2
Password:
Last login: Fri Feb 04 16:50:30 from host1
Sun Microsystems Inc. SunOS 5.9 Generic February 2000
$ pkill shell
```

or

```
$ pkill -9 shell
```

Using the `telnet` Command

The `telnet` command is available in the Solaris 9 OE. You can use a `telnet` connection to log in to a remote system and work in that environment. You can use the `telnet` command to open a session on a remote system.

The format for the `telnet` command is:

```
telnet hostname
```

The following example shows how to use the `telnet` command to connect to a remote system called `host2`.

```
$ telnet host2
Trying host2
Connected to host2
Escape character is '^]'.

SunOS 5.9

login: user2
Password:
Last login: Mon May 6 14:13:40 from host1
Sun Microsystems Inc.   SunOS 5.9 Generic May 2002
$
$ uname -n
host2
$ exit
Connection closed by foreign host.
$
```

Note – The `telnet` command always prompts for the user's password, and does not use the `.rhosts` file.

Copying Files or Directories to and From Another System

You use the `rcp` command to copy files or directories from one host to another. To copy subdirectories and their contents, use the `rcp -r` command.

The format for the `rcp` command is:

```
rcp source_file hostname:destination_file
```

or

```
rcp hostname:source_file destination_file
```

or

```
rcp hostname:source_file hostname:destination_file
```

Note – The *`source_file`* is your original file and the *`destination_file`* is the copy of the original file.

Copying Files From a Local Directory to a Remote Host

You use the `rcp` command to copy files from a local directory to a remote host. The `rcp` command checks the `~/.rhosts` file to determine access.

The following example shows how to copy the `dante` file from the local directory to the `/tmp` directory on a system called `host2`.

```
$ rcp dante host2:/tmp
```

Caution – Do not use the `/tmp` directory for long-term storage of important files. The `/tmp` directory is emptied each time the system is rebooted.

Copying Files From a Remote Host to a Local Directory

You use the `rcp` command to copy files from a remote host to the local `/tmp` directory. The `/tmp` directory is used because it has read, write, and execute permissions for each category of user.

The following example shows how to copy the dante file from a remote host called host2 to the local /tmp directory.

```
$ rcp host2:/tmp/dante /tmp
```

Copying Remote Directories

You use the rcp command with the -r option to copy directories to and from another system. If your current working directory contains the file or directory that you want to copy, use the file or directory name. You do not need to use the absolute path name.

The following example shows you how to copy the perm directory in the local home directory to the /tmp directory on the host system called host2.

```
$ rcp -r ~/perm host2:/tmp
```

Transferring Files Between Systems

This section explains how to transfer files between systems using File Transfer Protocol (FTP). The `ftp` command allows you to copy files from one computer to another over a network.

Using the `ftp` Command

The `ftp` command uses standard FTP to transfer files between systems with similar or dissimilar operating systems. You use FTP to transfer files in American Standard Code for Information Interchange (ASCII) mode or binary mode.

FTP does not use the `.rhosts` file, and unless you are using anonymous FTP, you must supply a password when accessing a remote system using the `ftp` command.

The format for the `ftp` command is:

```
ftp hostname
```

When you access a remote system using the `ftp` command, some file and directory access commands, such as the `ls` and `cd` commands, are available at the `ftp>` prompt.

If the permissions allow you to view the contents of a directory, the `ls` command lists files in a directory. If you do not have access to a directory or file, FTP generates a "Permission denied" error.

The `cd` command changes the current working directory on the remote system.

Note – You can use the `lcd` command at the `ftp>` prompt to change the current working directory on your local system.

To end an `ftp` session, enter `bye` or `quit` at the `ftp>` prompt.

Introducing FTP Transfer Modes

The default mode for an `ftp` connection is binary mode in the Solaris 9 OE. You use the `bin` command to set the FTP transfer mode to binary mode. Binary mode allows you to transfer binary, image, or any non-text files.

The default mode for an `ftp` connection in the Solaris 8 OE or earlier version is ASCII mode. You use this mode to transfer plain files such as text files.

Transferring Files Using ASCII Mode

The following example establishes an FTP connection from the `host1` system to the `host2` system, gets the `feathers` file from the `user2` directory on `host2`, stores the `feathers` file in the `user1` home directory on `host1`, and quits the `ftp` session.

```
$ ftp host2
Connected to host2.
220 host2 FTP server ready.
Name (host1:user2): user2
331 Password required for user2.
Password:
230 User user2 logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> lcd ~user1
Local directory now /export/home/user2
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data for the file list.
brands
celery
dante
dante_1
dir1
dir10
dir2
dir3
dir4
dir5
feathers
feathers_6
```

```
file.1
file1
file.2
file2
file.3
file3
file4
fruit
fruit2
games
house
mbox
monthly
mycontents
newdir
perm
practice1
process_list
records
Reports
tutor.vi
veggies
226 Transfer complete.
113 bytes received in 0.023 seconds (0.30 Kbytes/s)
ftp> get feathers
200 PORT command successful.
150 Opening BINARY mode data connection for feathers (37 bytes).
226 Transfer complete.
local: feathers remote: feathers
37 bytes received in 0.026 seconds (1.41 Kbytes/s)
ftp> bye
221-You have transferred 150 bytes in 2 files.
221-Total traffic for this session was 1674 bytes in 4 transfers.
221-Thank you for using the FTP service on host2.
221 Goodbye.
$
```

Transferring Multiple Files

You use the `mget` command to transfer multiple files from the remote system to the current working directory on the local system.

You use the `mput` command to transfer multiple files from the local system to a directory on the remote system.

You use the `prompt` command to switch interactive prompting on and off. Prompting is on by default. When prompting is on during multiple file transfers, you are prompted to confirm the transfer of each file in the multiple file transfer.

The following example shows how to establish an `ftp` connection from the `host1` system to the `host2` system. The `ftp` connection allows you to transfer multiple files using the `prompt` command, the `mget` command, and the `mput` command.

```
$ ftp host2
Connected to host2.
220 host2 FTP server ready.
Name (host2:user2): user2
331 Password required for user2.
Password:
230 User user2 logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful.
150 Opening ASCII data connection for file list.
brands
celery
dante
dante_1
dir1
dir10
dir2
dir3
dir4
dir5
feathers
feathers_6
file.1
file1
file.2
file2
file.3
file3
file4
fruit
fruit2
games
house
mbox
```

```
monthly
mycontents
newdir
perm
practice1
process_list
records
Reports
tutor.vi
veggies
226 Transfer complete
52 bytes received in 0.028 seconds (1.79 Kbytes/s)
ftp> **prompt**
Interactive mode off
ftp> **mget file.1 file.2**
200 PORT command successful.
150 Opening BINARY data connection for file.1 (208877 bytes).
226 Transfer complete.
local: file.1 remote: file.1
208877 bytes received in 0.16 seconds (1308.48 Kbytes/s)
200 PORT command successful.
150 Opening BINARY data connection for file.2 (208877 bytes).
226 Transfer complete.
local: file.2 remote: file.2
208877 bytes received in 0.12 seconds (1662.93 Kbytes/s)
ftp> **mput file3 file4**
200 PORT command successful.
150 Opening BINARY data connection for file3.
226 Transfer complete.
local: file3 remote: file3
208877 bytes sent in 0.048 seconds (4269.98 Kbytes/s)
200 PORT command successful.
150 Opening BINARY data connection for file4.
226 Transfer complete.
local: file4 remote: file4
208877 bytes received in 0.051 seconds (4006.86 Kbytes/s)
ftp> **prompt**
Interactive mode on.
ftp> **mget file.1 file.2**
mget file.1? **y**
200 PORT command successful.
150 Opening BINARY mode data connection for file.1 (208877 bytes).
226 Transfer complete.
local: file.1 remote: file.1
208877 bytes received in 0.15 seconds (1325.33 Kbytes/s)
mget file.2? **y**
```

```
200 PORT command successful.
150 Opening BINARY data connection for file.2 (208877 bytes).
226 Transfer complete.
local: file.2 remote: file.2
208877 bytes received in 0.11 seconds (1862.74 Kbytes/s)
ftp> bye
221-You have transferred 1256975 bytes in 6 files.
221-Total traffic for this session was 1259325 bytes in 11 transfers.
221-Thank you for using the FTP service on host2.
221 Goodbye.
$
```

Transferring Files Using Binary Mode

The following example shows how to use the `bin` command to set the binary mode for the transfer of a binary file.

Note – The `binary.file` file is an example file for demonstration purposes only. The file is not located on your system.

```
$ cd /tmp
$ ftp host2
Connected to host2.
220 host2 FTP server ready.
Name (host2:user2): user2
331 Password required for user2.
Password:
230 User user2 logged in.
Remote system type is UNIX.
ftp> bin
200 Type set to I.
ftp> get binary.file
200 PORT command successful.
150 Opening BINARY mode data connection for binary.file (19084 bytes).
226 Transfer complete.
local: binary.file remote: binary.file
19084 bytes received in 0.0044 seconds (4212064 Kbytes/s)
ftp> bye
221-You have transferred 19084 bytes in 1 files.
221-Total traffic for this session was 19507 bytes in 1 transfers.
221-Thank you for using the FTP service on host2.
221 Goodbye.
$
```

Exercise: Performing Remote Connection Commands

In this exercise, you use some of the remote connection commands introduced in this module.

You need to provide a remote host name and a guest login name for the students prior to the start of this lab for Step 4.

Tasks

Complete the following steps. Write the commands that you would use to perform each task in the space provided.

1. Use the `rlogin` command to log in to another system in your classroom.

   ____

   Which directory are you accessing on the remote system?

   ____

2. Enter the command that shows you the host name of the current system.

   ____

3. Log out of the remote system.

   ____

   Display the host name of your current system to determine if you are back on your host system.

   ____

4. Use the `rlogin` command and an option to log in to another system as another user, as specified by your instructor.

   ____

5. Log out of the remote system.

   ____

6. Which remote connection command do you use to run a program remotely?

   ____

7. Which command do you use to copy files remotely?

   ____

8. Which remote connection command do you use to transfer files from system to system, including binary files?

____

Exercise Summary

Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

Manage the discussion based on the time allowed for this module, which was provided in the "About This Course" module. If you do not have time to spend on discussion, then just highlight the key concepts students should have learned from the lab exercise.

- Experiences

Ask students what their overall experiences with this exercise have been. Go over any trouble spots or especially confusing areas at this time.

- Interpretations

Ask students to interpret what they observed during any aspect of this exercise.

- Conclusions

Have students articulate any conclusions they reached as a result of this exercise experience.

- Applications

Explore with students how they might apply what they learned in this exercise to situations at their workplace.

Exercise Solutions

1. Use the `rlogin` command to log in to another system in your classroom.

   ```
   $ rlogin hostname
   ```

   Which directory are you accessing on the remote system?

   A home directory on the remote system (either `/home/username` *or* `/export/home/username`*) or the* `/` *(root) directory if no home directory exists.*

2. Enter the command that shows you the host name of the current system.

   ```
   $ uname -n
   ```

3. Log out of the remote system.

   ```
   $ exit
   ```

   Display the host name of your current system to determine if you are back on your host system.

   ```
   $ uname -n
   ```

4. Use the `rlogin` command and an option to log in to another system as another user, as specified by your instructor.

   ```
   $ rlogin hostname -l username
   ```

5. Log out of the remote system.

   ```
   $ exit
   ```

6. Which remote connection command do you use to run a program remotely?

   The `rsh` *command*

7. Which command do you use to copy files remotely?

   The `rcp` *command*

8. Which remote connection command do you use to transfer files from system to system, including binary files?

   The `ftp` *command*

Module 11

Creating Archives

Objectives

Upon completion of this module, you should be able to:

- Archive files
- Compress and archive files

The following course map shows how this module fits into the current instructional goal.

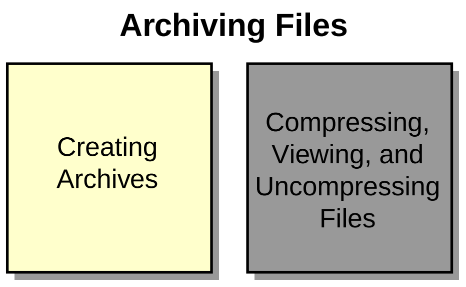


Figure 11-1 Course Map

Archiving Files

To safeguard your files and directories, you need to create a copy, or archive, of the files and directories on a removable medium, such as a cartridge tape. You can use the archived copies to retrieve lost, deleted, or damaged files.

Introducing Archiving Techniques

You can use several commands to store, locate, and retrieve files on a tape device or from an archive file. Two of the commands you can use are:

- The `tar` command, to create and extract files from a file archive or removable media such as a tape or diskette
- The `jar` command, to combine multiple files into a single archive file

Introducing the `tar` Command

The `tar` command archives files to and extracts files from a single file called a `tar` file. The default device for a `tar` file is a magnetic tape device.

The format for the `tar` command is:

```
tar function(s) archivefile filename(s)
```

Note – You should use relative path names to archive files.

Table 11-1 shows the `tar` functions.

Table 11-1 Functions for the `tar` Command

Function	Definition
`c`	Creates a new `tar` file.
`t`	Lists the table of contents of the `tar` file.
`x`	Extracts files from the `tar` file.
`f`	Specifies the archive file or tape device. The default tape device is `/dev/rmt/0`. If the name of the archive file is "`-`", the `tar` command reads from standard input when reading from a `tar` archive, or writes to standard output if creating a `tar` archive.
`v`	Executes in verbose mode, writes to standard output.
`h`	Follows symbolic links as standard files or directories.

Creating an Archive

You can use the `tar` command to create an archive file that contains multiple files or directories. You can then place the file on a tape or a diskette so that other users can share the file or attach it to email messages.

Creating an Archive on a Tape

To create an archive on a tape, first verify that the system has a tape drive available. You can use the `mt` utility with the `status` option to print status information about the tape unit. You use the `mt` utility to send commands to a magnetic tape drive.

Note – For more information on the use of the `mt` command to control a magnetic tape drive, refer to the `mt` man page.

The following example shows you how to use the default tape device to archive your home directory. In this example, `user1` creates a tape archive of the `user1` home directory.

```
$ cd
$ mt -f /dev/rmt/0 status
<output will be your local tape device info>
$ tar cvf /dev/rmt/0 .
a ./ 0 tape blocks
a ./dante 106 tape blocks
a ./dante_1 1 tape blocks
a ./logfile 5 tape blocks
a ./file2 1 tape blocks
<output omitted>
```

You can also use the `tar` command to create an archive file containing multiple files or directories.

The following example shows you how to archive the `file1`, `file2`, and `file3` files in an archive file called `files.tar`.

```
$ cd
$ tar cvf files.tar file1 file2 file3
a file1 2K
a file2 1K
a file3 1K
```

Creating an Archive on a Diskette

Before you can create a file archive on a diskette, you must insert a diskette into the appropriate drive on the system. Then you must run the `volcheck` command. This command informs the Volume Management program to check for media in a drive.

Note – Volume Management provides automatic detection of removable media. However, because of hardware limitations in many floppy drives, Volume Management does not detect the presence of a newly inserted diskette.

The `volcheck` command checks for all removable media managed by the Volume Management program. The `volcheck` command checks for all diskettes as a default. It instructs Volume Management to check each device path name in sequence and determine if a new diskette is in the diskette drive.

The format for the `volcheck` command is:

```
volcheck -v device_pathname
```

If you use the `volcheck -v` command without a device path name, one of the following messages appears:

```
media was found
```

or

```
no media was found
```

Note – The device path name for a diskette is `/dev/diskette`. You can use the device path name to instruct Volume Management to check each device path name in sequence and determine if new media has been inserted in the drive.

Running the `volcheck` command creates the `/floppy` directory and its content when a diskette is present. You can use the `cd` command to access files on the diskette in the `/floppy/floppy0` directory. You can also use the `cp` command to copy an archive into the `/floppy/floppy0` directory on the diskette.

The following example shows you how to use the `cp` command to copy an archive file from your home directory to a diskette.

```
$ cd /floppy/floppy0
$ ls
$
$ cd
$ cp files.tar /floppy/floppy0
$ ls /floppy/floppy0
files.tar
```

Note – You can also access the files on a diskette from the File Manager window by selecting Open Floppy from the File Menu.

To eject a diskette from the diskette drive do one of the following:

- Use the File Manager window.
    - Click File in the File Manager diskette window, and select the Eject option in the File menu.
    - Close the popup Watch Errors window.
- Enter **`cd`** to change from the `/floppy/floppy0` directory to your home directory.

  Enter **`eject floppy`**.

After a few seconds, the diskette ejects from the diskette drive, or a window appears to instruct you to manually eject the diskette.

You cannot eject removable devices while you are in the current working directory for the device. If you see an error message, such as `Device busy`, when trying to eject the diskette, you might still be in the working directory on the diskette. Use the `pwd` command to see if you are in the `/floppy/floppy0` directory. If you are in the `/floppy/floppy0` directory, enter the `cd` command to return to your home directory. Then enter the `eject` command.

Viewing an Archive

You can view the names of all the files that have been written directly to a tape archive or a file archive.

Viewing an Archive From a Tape

To view the contents of the `user1` home directory on a tape, enter the following command:

```
$ tar tvf /dev/rmt/0
./
./brands
./celery
./dante
./dante_1
./feathers
./feathers_6
./file.1
./file1
./file.2
<remainder of file list omitted>
./dir1/
./dir1/coffees/
./dir1/coffees/beans
./dir1/coffees/nuts
./dir1/coffees/brands
<remainder of directory list omitted>
```

Viewing Files in an Archive File

To view the contents of the `files.tar` archive file, enter the following command:

```
$ tar tf files.tar
file1
file2
file3
```

Retrieving tar Archive Data

You can retrieve or extract the contents of an archive that was written directly to a tape device or to a file.

Retrieving a Directory From a Tape

If the contents of your home directory are deleted, you can extract the directory contents from an archive tape.

To retrieve all the files from the tape archive, enter the following commands:

```
$ cd
$ tar xvf /dev/rmt/0 .
x ., 0 bytes, 0 tape blocks
x ./brands 0 tape blocks
a ./celery 0 tape blocks
x ./dante, 54120 bytes, 106 tape blocks
x ./dante_1, 368 bytes, 1 tape blocks
x ./feathers 1 tape blocks
x ./feathers_6 1 tape blocks
x ./file.1 0 tape blocks
x ./file1 4 tape blocks
x ./file2 1 tape blocks
<output omitted>
```

You can extract files from an archive file using the `tar` command. The following example shows how to extract files from the `files.tar` archive file for placement into the current directory.

```
$ tar xvf files.tar
tar: blocksize = 11
x file1, 1610 bytes, 4 tape blocks
x file2, 105 bytes, 1 tape blocks
x file3, 218 bytes, 1 tape blocks
```

Retrieving Files From a Diskette

To retrieve files archived to a diskette, you follow the same steps for archiving to a diskette except that you copy files from the diskette back to your home directory.

The following example shows how to retrieve the archived file `files.tar` from a diskette.

```
$ volcheck -v /dev/diskette
$ cd /floppy/floppy0
$ ls
files.tar
$ cp files.tar /export/home/user1
$ cd
$ ls files.tar
files.tar
$ tar xvf files.tar
```

Compressing and Archiving Files Using the `jar` Command

This section describes how to use the `jar` command to compress and archive files. The `jar` command combines multiple files into a single archive file.

Introducing the `jar` Command

The `jar` command has syntax similar to the `tar` command, but the `jar` command compresses the named files in addition to archiving the files. The `jar` command copies and compresses multiple files into a single archive file.

Note – The `jar` command was created to enable programmers working with Java™ technology to create a single archive that they could download instead of downloading multiple individual files. The `jar` command is a standard feature of the Solaris™ 9 Operating Environment (Solaris 9 OE). The `jar` command is available on any system that uses Java virtual machine (JVM™) software.

The format for the `jar` command is:

```
jar option(s) destination filename(s)
```

Using the Options for the `jar` Command

Table 11-2 shows the options you can use with the `jar` command.

Table 11-2 Options for the `jar` Command

Option	Definition
`c`	Creates a new `jar` file.
`t`	Lists the table of contents of a `jar` file.
`x`	Extracts the specified files from the `jar` file.
`f`	Specifies the `jar` file to process (for example, `/tmp/`*`file`*`.jar`). The `jar` command sends the data to the screen (`stdout`) if the `f` option is not used.
`v`	Executes in verbose mode.

Adding All the Files in a Directory to an Archive

The following example shows how to use the `jar` command to copy and compress multiple files into a single archive file called `bundle.jar`.

```
$ ls
brands         dir4           file.3         mbox           records
celery         dir5           file3          monthly        Reports
dante          feathers       file4          mycontents     tutor.vi
dante_1        feathers_6     files.tar      my_file        veggies
dir1           file.1         fruit          newdir
dir10          file1          fruit2         perm
dir2           file.2         games          practice1
dir3           file2          house          process_list
$
$ jar cvf /tmp/bundle.jar *
added manifest
adding: brands(in = 0) (out= 0)(stored 0%)
adding: celery(in = 0) (out= 0)(stored 0%)
adding: dante(in = 1319) (out= 744)(deflated 43%)
adding: dante_1(in = 368) (out= 242)(deflated 34%)
adding: dir1/(in = 0) (out= 0)(stored 0%)
adding: dir1/coffees/(in = 0) (out= 0)(stored 0%)
adding: dir1/coffees/cream(in = 0) (out= 0)(stored 0%)
adding: dir1/fruit/(in = 0) (out= 0)(stored 0%)
adding: dir1/trees/(in = 0) (out= 0)(stored 0%)
adding: dir10/(in = 0) (out= 0)(stored 0%)
adding: dir10/planets/(in = 0) (out= 0)(stored 0%)
adding: dir10/planets/mars(in = 68) (out= 61)(deflated 10%)
adding: dir10/planets/pluto(in = 42) (out= 44)(deflated -4%)
adding: dir2/(in = 0) (out= 0)(stored 0%)
adding: dir2/beans/(in = 0) (out= 0)(stored 0%)
adding: dir2/notes(in = 0) (out= 0)(stored 0%)
adding: dir2/recipes/(in = 0) (out= 0)(stored 0%)
adding: dir2/recipes/beans(in = 12288) (out= 3161)(deflated 74%)
```

Note – The `jar` command does not back up symbolic links as links. The `jar` command resolves the symbolic link and copies the file's contents.

Exercise: Archiving and Retrieving Files

In this exercise, you practice archiving, viewing, and retrieving files on a tape or a diskette.

Tasks

Complete the following steps. Write your answer or the commands that you would use to perform each task in the space provided.

Note – If you get a `Permission Denied` error message while performing the following exercises, check the write-protect switch on the tape cartridge or diskette. Archive your home directory to a file using the `tar` command.

1. What kind of file does the `tar` command create?

   ____

2. Archive your home directory on a tape.

   ____

3. Which command do you use to enable the system to detect a new diskette?

   ____

4. Which command do you use to copy files to a diskette?

   ____

5. Archive your home directory to a file using the `tar` command.

   ____

6. Use the `tar` command to view the contents of your home directory archive.

   ____

For Step 7, create a new directory in your home directory called `Retrieve.` Use the `cd` command to move to the new directory. You can use the new directory to practice retrieving files from archives.

7. Retrieve the contents of the archive `tar` file that you created on tape.

   ____

8. Ensure that all files in the `~/dir1` directory are readable, then use the `jar` command to archive the `~/dir1` directory.

   ___

9. Use the `tar` command to archive the `~/dir1` directory and the contents of the `dir1` directory.

   ___

10. Compare the current size of the `tar` file and the `jar` file archives.

    ___

11. Is there a difference in the size of the files?

    ___

Exercise Summary

Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

Manage the discussion based on the time allowed for this module, which was provided in the "About This Course" module. If you do not have time to spend on discussion, then just highlight the key concepts students should have learned from the lab exercise.

- Experiences

Ask students what their overall experiences with this exercise have been. Go over any trouble spots or especially confusing areas at this time.

- Interpretations

Ask students to interpret what they observed during any aspect of this exercise.

- Conclusions

Have students articulate any conclusions they reached as a result of this exercise experience.

- Applications

Explore with students how they might apply what they learned in this exercise to situations at their workplace.

Exercise Solutions

1. What kind of file does the `tar` command create?

   A `tar` *file*

2. Archive your home directory on a tape.

   ```
   $ cd
   $ tar cvf /dev/rmt/0 .
   ```

3. Which command do you use to enable the system to detect a new diskette?

   `volcheck -v /dev/diskette`

   or

   `volcheck -v`

4. Which command do you use to copy files to a diskette?

   The `cp` *command*

5. Archive your home directory to a file using the `tar` command.

   ```
   $ cd
   $ tar cvf /tmp/homedir.tar .
   ```

6. Use the `tar` command to view the contents of your home directory archive.

   ```
   $ tar tvf /tmp/homedir.tar
   ```

7. Retrieve the contents of the archive `tar` file that you created on tape.

   ```
   $ cd
   $ tar xvf /dev/rmt/0 .
   ```

8. Ensure that all files in the `~/dir1` directory are readable, then use the `jar` command to archive the `~/dir1` directory.

   ```
   $ cd
   $ jar cvf dir1.jar dir1
   ```

9. Use the `tar` command to archive the `~/dir1` directory and the contents of the `~/dir1` directory.

   ```
   $ cd
   $ tar cvf dir1.tar dir1
   ```

10. Compare the current size of the `tar` file and the `jar` file archives.

    ```
    $ ls -l dir1.tar dir1.jar
    ```

11. Is there a difference in the size of the files?

    Yes

Module 12

Compressing, Viewing, and Uncompressing Files

Objectives

Upon completion of this module, you should be able to:

- Compress files using the `compress` command
- View compressed files using the `zcat` command
- Uncompress files using the `uncompress` command
- Compress a file with the `gzip` command
- View files using the `gzcat` command
- Compress and archive multiple files with the `zip` command

The following course map shows how this module fits into the current instructional goal.

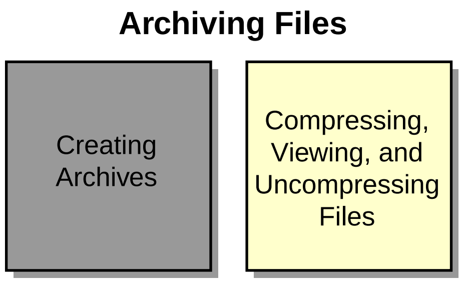


Figure 12-1 Course Map

Compressing Files Using the compress Command

It is often a good idea to compress files to conserve disk space. This section describes how to use the `compress` command to reduce the size of a file.

Introducing the compress Command

You use the `compress` command to reduce the size of a file. You compress large files to reduce the use of disk space and the time required for file transfers over a network. The amount of compression depends on the type of file you compress. Typically, compression reduces a text file by 50 percent to 60 percent.

The format for the `compress` command is:

```
compress [ -v ] filename
```

For example:

```
$ compress dante
```

When you compress a file, the `compress` command replaces the original file with a new file with a `.Z` extension. The ownership and modification time of the original file remain the same, but the contents of the file change.

Compressing a File

The following example shows how to compress a file called `files.tar`, using the `-v` (verbose) option. Using this option with the `compress` command displays a message concerning the percentage reduction or expansion of each file.

```
$ compress -v files.tar
files.tar: Compression: 70.20% -- replaced with files.tar.Z
```

The compressed file, which replaces the `files.tar` file, is called `files.tar.Z`. A file with a `.Z` extension is compressed, and should not be viewed or printed without first uncompressing it.

Caution – Compressing a file that has already been compressed makes the file size larger, not smaller.

If you do not use the `-v` (verbose) option with the `compress` command, no output appears when the command executes correctly. For example:

```
$ compress files.tar
$
```

Viewing Compressed Files Using the zcat Command

You can print the uncompressed form of a compressed file to standard output. This section describes how to view the uncompressed form of a compressed file using the `zcat` command.

Introducing the zcat Command

You use the `zcat` command to view files that were compressed with the `compress` command. The `zcat` command interprets the compressed data and displays the contents of a file as if it were not compressed. The `zcat` command does not change the contents of a compressed file. The compressed file remains on the disk in compressed form.

The format for the `zcat` command is:

```
zcat filename
```

Note – The command `zcat` *`filename`* is functionally identical to the `uncompress -c` *`filename`* command.

Viewing the Contents of a Compressed File

The following example shows how to view the contents of the compressed file called `files.tar.Z`.

```
$ zcat files.tar.Z | more
file1010066000023210000012000000031120744415475300122310ustar00user1staff
00002100000007
        The Achievers

Unconsciously or not, they divide their work totally differently than the
<output omitted>
```

You use the pipe (`|`) character with the `zcat` command to extract the contents of a compressed `tar` file without uncompressing the `tar` file first. For example:

```
$ zcat files.tar.Z | tar xvf -
```

The dash (`-`) at the end of the `tar` command indicates that the `tar` command should read the `tar` file from standard input instead of from a file or tape. In this example the standard input comes from the output of the `zcat` command because of the pipe (`|`).

Uncompressing Files Using the uncompress Command

You can restore a compressed file to its original state after it has been compressed. This section describes how to uncompress files using the `uncompress` command.

Introducing the uncompress Command

The `uncompress` command restores a compressed file back to its original state.

The format for the `uncompress` command is:

```
uncompress options filename
```

Uncompressing a File

The following example shows how to uncompress the `files.tar.Z` file and replace it with the original file named `files.tar`. The `-v` option causes the `uncompress` command to display messages.

```
$ uncompress -v files.tar.Z
files.tar.Z:  -- replaced with files.tar
$
```

Viewing the Contents of a Compressed File

You use the `uncompress` command with the `-c` option to send the contents of a compressed file to the screen (`stdout`), without changing the compressed (`.Z`) file. You use the pipe (`|`) character to send the output of the `uncompress` command to another program. You use the `tar` command to list the contents of the file that the `uncompress` command is reading.

```
$ uncompress -c files.tar.Z | tar tvf -
tar: blocksize = 11
-rw-rw---- 1233/10    1610 May 7 14:12 2002 file1
-rw-rw---- 1233/10     105 May 7 14:12 2002 file2
-rw-rw---- 1233/10     218 May 7 14:12 2002 file3
```

The dash (`-`) at the end of the command line indicates that the `tar` command reads the data from the piped output of the `uncompress` command rather than a `tar` file or a tape.

Compressing a File Using the gzip Command

The gzip command is another command available that reduces the size of files. This section describes how to compress a file using the gzip command.

Introducing the gzip Command

The gzip command reduces the size of files. When the gzip command successfully compresses a file, the file is replaced by a new file with the same name and a .gz extension. The files keep the same ownership modes, access, and modification times. The gzip command compresses regular files.

The format for the gzip command is:

```
gzip [ -v ] filename(s)
```

The following example shows how to compress the file1, file2, file3, and file4 files with the gzip command.

```
$ gzip file1 file2 file3 file4
$ ls *.gz
file1.gz file2.gz file3.gz file4.gz
```

Note – The gzip command performs the same general function as the compress command, but the gzip command generally produces smaller files.

Restoring a gzip File Using the gunzip Command

You use the gunzip command to restore a file that has been compressed with the gzip command.

The following example shows how to uncompresses the file1.gz file.

```
$ gunzip file1.gz
```

Viewing Files Using the `gzcat` Command

You can use the `gzcat` command to look at a compressed file. This section describes how to view files using the `gzcat` command.

Introducing the `gzcat` Command

You use the `gzcat` command to view files that were compressed with either the `gzip` command or the `compress` command.

The `gzcat` command interprets the compressed data and displays the contents of the file as if it were not compressed. The `gzcat` command does not change contents of the compressed file. The compressed file remains on the disk in compressed form.

The format for the `gzcat` command is:

```
gzcat filename
```

The `gzcat` *`filename`* command is functionally identical to the `gunzip -c` *`filename`* command.

Viewing the Contents of a Compressed File

You use the `gzcat` command to view the contents of the compressed `tar` file without uncompressing the file first.

The following example shows how to view the `file.gz` file.

```
$ gzcat file1.gz
```

Compressing and Archiving Multiple Files Using the zip Command

You can compress and archive files with one command. This section describes how to perform this combination of tasks using the `zip` command.

Introducing the zip Command

The `zip` command compresses multiple files into a single archive file. The `zip` command adds the `.zip` extension to the file name of the compressed archive file if you do not assign a new file name with an extension.

Note – You can enter `zip` or `unzip` on the command line to view a list of options used with each command.

The format for the `zip` command is:

```
zip [ -r ] target_filename source_filename(s)
```

The following example shows you how to use the `zip` command to compress the `file1`, `file2`, and `file3` files into the `file.zip` archive file.

```
$ zip file.zip file1 file2 file3
adding: file1 (deflated 48%)
adding: file2 (deflated 16%
adding: file3 (deflated 26%)
$ ls
file.zip
file1
file2
file3
```

The following example shows you how to list the files in a `zip` archive.

```
$ unzip -l zipfile
```

Restoring a zip File Using the unzip Command

You can uncompress the contents of a zip file using the unzip command.

The following example shows you how to uncompress the file.zip archive file.

```
$ unzip file.zip
```

Note – The jar command and the zip command create compatible files. The unzip command can uncompress a jar file, and the jar command can uncompress a zip file.

Exercise: Compressing and Restoring Files

In this exercise, you practice compressing and uncompressing files, and viewing compressed files.

Note – Before you begin this lab, uncompress any file that you compressed in the examples in the module.

Tasks

Complete the following steps. Write the commands that you would use to perform each task in the space provided.

1. In your home directory, compress the `dante` file and the `file1` file using the `compress` command.

2. What are the new names for the compressed versions of the `dante` file and the `file1` file?

3. Which three commands do you use to view the contents of a file that was compressed with the `compress` command?

4. Use the `gzip` command to compress the `file2` file and the `dante_1` file.

5. What are the new names for the compressed versions of the `file2` file and the `dante_1` file?

6. Use the `zip` command to compress the `file3` file, the `fruit2` file, and the `tutor.vi` file to a file called `files.zip`.

7. What is the new name for the packaged and compressed file?

   ____

   Do the original versions of the `file3` file, the `fruit2` file, and the `tutor.vi` file still exist?

   ____

8. Uncompress the `dante` file and the `file1` file. Which command should you use?

   ____

   Do the `dante` file and the `file1` file still have a `.Z` extension on the file names?

   ____

9. Uncompress the `file2` file and the `dante_1` file. Which command should you use?

   ____

   Do the `file2` file and the `dante_1` file still have a `.gz` extension on the file names?

   ____

10. Unarchive the `file3` file, the `fruit2` file, and the `tutor.vi` file from the `zip` file created in Step 6. Which command should you use?

    ____

    Does the `files.zip` file still exist in the directory?

    ____

Exercise Summary

Discussion – Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

Manage the discussion based on the time allowed for this module, which was provided in the "About This Course" module. If you do not have time to spend on discussion, then just highlight the key concepts students should have learned from the lab exercise.

- Experiences

Ask students what their overall experiences with this exercise have been. Go over any trouble spots or especially confusing areas at this time.

- Interpretations

Ask students to interpret what they observed during any aspect of this exercise.

- Conclusions

Have students articulate any conclusions they reached as a result of this exercise experience.

- Applications

Explore with students how they might apply what they learned in this exercise to situations at their workplace.

Exercise Solutions

1. In your home directory, compress the `dante` file and the `file1` file using the `compress` command.

```
$ compress dante
$ compress file1
```

2. What are the new names for the compressed versions of the `dante` file and the `file1` file?

```
dante.Z
file1.Z
```

3. Which three commands do you use to view the contents of a file that was compressed with the `compress` command?

```
$ uncompress -c filename
```

or

```
$ zcat filename
```

or

```
$ gzcat filename
```

4. Use the `gzip` command to compress the `file2` file and the `dante_1` file.

```
$ gzip file2 dante_1
```

5. What are the new names for the compressed versions of the `file2` file and the `dante_1` file?

```
file2.gz
dante_1.gz
```

6. Use the `zip` command to compress the `file3` file, the `fruit2` file, and the `tutor.vi` file to a file called `files.zip`.

```
$ zip files.zip file3 fruit2 tutor.vi
```

7. What is the new name for the packaged and compressed file?

```
files.zip
```

Do the original versions of the `file3` file, the `fruit2` file, and the `tutor.vi` file still exist?

Yes

8. Uncompress the `dante` file and the `file1` file. Which command should you use?

   ```
   $ uncompress dante
   $ uncompress file1
   ```

   Do the `dante` file and the `file1` file still have a `.Z` extension on the file names?

   No

9. Uncompress the `file2` file and the `dante_1` file. Which command should you use?

   ```
   $ gunzip file2 dante_1
   ```

   Do the `file2` file and the `dante_1` file still have a `.gz` extension on the file names?

   No

10. Unarchive the `file3` file, the `fruit2` file, and the `tutor.vi` file from the `zip` file created in Step 6. Which command should you use?

    ```
    $ unzip files.zip
    ```

    Does the `files.zip` file still exist in the directory?

    Yes

Index