Viewing Directories and Files

Objectives

This module introduces files and directories and describes how to display directories and directory contents. The module also describes how to view and print files.

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

- Work with directories
- Work with files
- Print files

Viewing Directories

You can use various commands to display the current directory, view the contents of a directory, and change directories.

Directory Terms

The following sections describe basic terms used in conjunction with directories.

Directory

A *directory* is a list of references to objects, which can include files, sub-directories, and symbolic links. Each reference consists of two components: a name and a number. The *name* of the object is used to identify and access the object. The *number* specifies the inode in which information about the object is stored.

Inode

An *inode* is a list of information relating to a particular object, such as a file, directory, or symbolic link. The information held by the inode includes the type of object about which the inode holds information, permissions, ownership information, and the locations in which data is stored.

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/student
$
```

Displaying the Directory Contents

You can use the `ls` command to display the contents of a directory. To list the files and directories within the specified directory, enter the `ls` command without arguments.

The syntax for the `ls` command is:

```
ls -options filename
```

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

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

You can view a hierarchy of the files and directories in the `/export/home/student` directory, as Figure 3-2 shows.

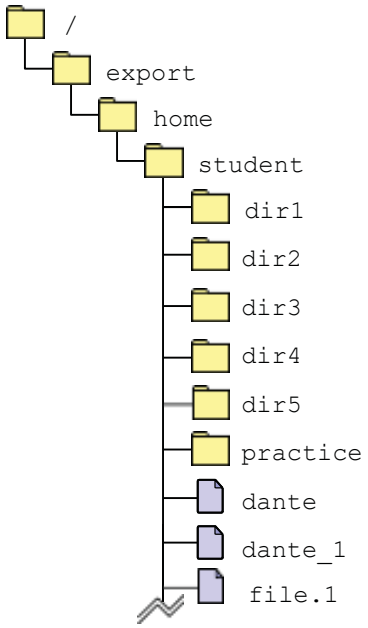


Figure 3-2 Directory List

To display the contents of a specific directory within the current working directory, enter the `ls` command followed by 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 the complete path to that directory.

```
$ ls /export/home/student/dir2
beans   notes   recipes
$
```

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. You can 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
.                .gnome2_private   dante            file2
..               .gtkrc-1.2-gnome2 dante_1          file3
.ICEauthority    .metacity         dir1             file4
.Xauthority      .mozilla          dir2             fruit
.bash_history    .nautilus         dir3             fruit2
.dt              .netbeans         dir4             greetings
.dtprofile       .recently-used    dir5             myvars
.gconf           .rhosts           file.1           practice
.gconfd          .sh_history       file.2           tutor.vi
.gnome           .softwareupdate   file.3
.gnome2          .sunw             file1
$
```

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

Displaying a Long List

You can use the `ls -l` command to view detailed information about the contents of a directory. The output of the `ls -l` command displays a long listing of file information as Figure 3-3 shows.

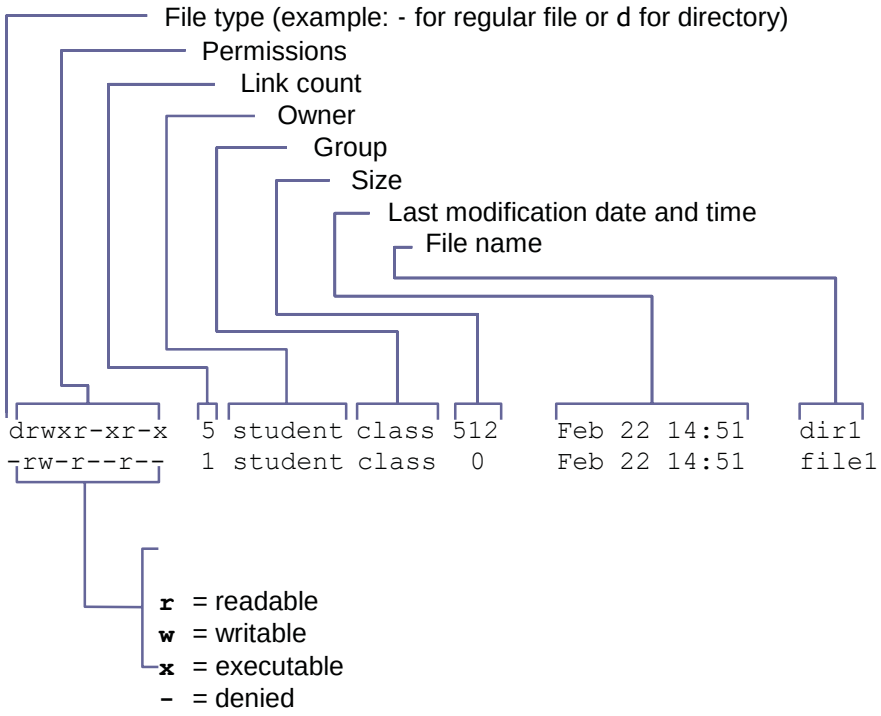


Figure 3-3 Long List File Information

For a long listing of the contents of the `student` directory, enter the `ls -l` command, from the `student` directory.

```
$ ls -l
-rw-r--r--  1 student  class      1319 Feb 6 09:25 dante
-rw-r--r--  1 student  class       368 Feb 6 09:25 dante_1
drwxr-xr-x  5 student  class       512 Feb 6 09:25 dir1
drwxr-xr-x  4 student  class       512 Feb 6 09:32 dir2
drwxr-xr-x  3 student  class       512 Feb 6 09:25 dir3
drwxr-xr-x  2 student  class       512 Feb 6 09:25 dir4
drwxr-xr-x  2 student  class       512 Feb 6 09:25 dir5
-rw-r--r--  1 student  class         0 Feb 6 09:25 file.1
-rw-r--r--  1 student  class         0 Feb 6 09:25 file.2
-rw-r--r--  1 student  class         0 Feb 6 09:26 file.3
-rw-r--r--  1 student  class      1610 Feb 6 09:26 file1
-rw-r--r--  1 student  class       105 Feb 6 09:26 file2
-rw-r--r--  1 student  class       218 Feb 6 09:26 file3
-rw-r--r--  1 student  class       137 Feb 6 09:26 file4
-rw-r--r--  1 student  class        57 Feb 6 09:26 fruit
-rw-r--r--  1 student  class        57 Feb 6 09:26 fruit2
-rw-r--r--  1 student  class        59 Feb 6 09:26 greetings
-rw-r--r--  1 student  class        67 Feb 6 09:26 myvars
drwxr-xr-x  2 student  class       512 Feb 6 09:25 practice
-rw-r--r--  1 student  class     28738 Feb 6 09:26 tutor.vi
$
```

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

```
$ ls -l dir1
total 6

drwxr-xr-x  3 student  class        512 Feb 6 09:30 coffees
drwxr-xr-x  2 student  class        512 Feb 6 09:30 fruit
drwxr-xr-x  2 student  class        512 Feb 6 09:30 trees
$
```

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

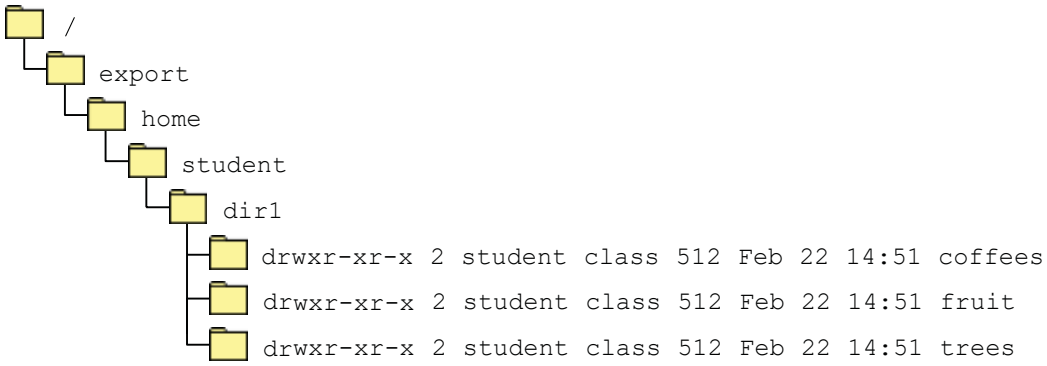


Figure 3-4 Long List of Files and Directories

Displaying Individual Directories

You can use the `ls -ld` command to view detailed information about a directory without viewing its contents.

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

```
$ ls -ld dir1

drwxr-xr-x   5 student  class        512 Feb 6 09:30 dir1
$
```

Displaying a Recursive List

You can 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 the `ls -R dir1` command.

```
$ ls -R dir1
dir1:

coffees  fruit   trees

dir1/coffees:
beans   brands   nuts

dir1/coffees/beans:
beans

dir1/fruit:

dir1/trees:
$
```

Displaying File Types

You can use either the `ls -F` command or the `file` command to display file types.

Using the `ls -F` Command

Table 3-1 shows the symbols used in the `ls -F` command output.

Table 3-1 File Type Symbols

Symbol	File Type
/	Directory
*	Executable
(None)	Plain text file or American Standard Code for Information Interchange (ASCII)
@	Symbolic link

The following example shows the output of the `ls -F` command.

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

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

Using the `file` Command

You can use the `file` command to determine certain file types. If you know the file type, you can decide which command or program to use to read the file.

Note – Normally, in the Solaris environment file suffixes do not indicate the file type unless it was generated by an application, such as StarOffice (`.sxw`) or Framemaker (`.fm`).

The output from the `file` command can be 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 syntax for the `file` command is:

```
file filenames
```

To view the file type for the `dante` file, enter the `file` command and specify the name of the file.

```
$ file dante

dante:          English text
$
```

Changing Directories

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

The syntax for the `cd` command is:

```
cd directory
```

When you use the `cd` command without options or arguments, the current working directory changes to your home directory.

To change directories from the `student` directory to the `dir1` directory, use the `cd` command:

```
$ pwd

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

Using Pathname Abbreviations

You can use pathname abbreviations to easily navigate or refer to directories on the command-line. Table 3-2 shows the pathname abbreviations.

Table 3-2 Pathname Abbreviations

Symbol	Pathname
.	Current or working directory
..	Parent directory, the directory directly above the current working directory

To move to the parent directory for `dir1`, enter the `cd ..` command.

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

Confirm the current working directory using the `pwd` command.

```
$ pwd
/export/home/student
$
```

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

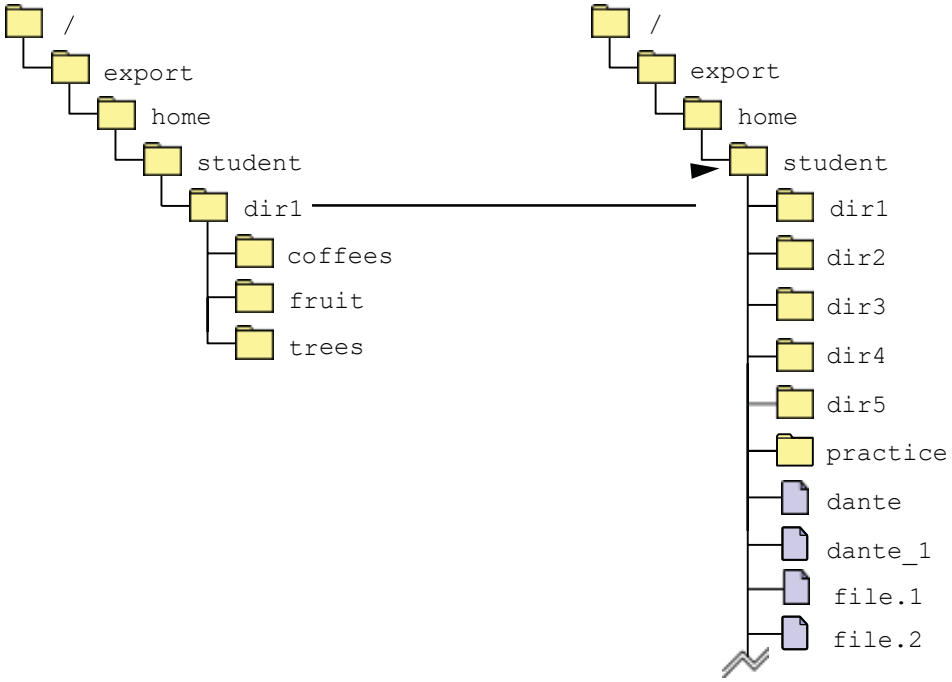


Figure 3-5 Navigating the Directory Tree

You can move up multiple levels of the directory hierarchy using the `cd ..` command followed by a slash (/).

```
$ pwd

/export/home/student
$ cd ../../..
$ pwd
/
$
```

Moving Around the Directory Hierarchy

You can either use a relative pathname or an absolute pathname to move around the directory hierarchy.

A relative pathname lists the directories in the path relative to the current working directory. An absolute pathname lists all the directories in the path, starting with the root (/) directory.

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

```
$ cd
$ cd dir1
$ pwd

/export/home/student/dir1
$ cd ../dir2
$ pwd
/export/home/student/dir2
$ cd
$ cd dir1/coffees
$ pwd
/export/home/student/dir1/coffees
$
```

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

```
$ cd

$ cd /export/home/student/dir1/coffees
$ pwd
/export/home/student/dir1/coffees
$
```

Returning to Your Home Directory

The home directory of a regular user is where the user is placed after logging in. The user can create and store files in the home directory.

Note – The directory named `/export/home` is the default directory that contains the home directories of regular users. However, you can configure systems to use the `/home` directory, instead of the `/export/home` directory, as the default directory that holds the home directories of regular users.

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

You can return to your home directory using two methods:

- Perform the `cd` command without arguments.

```
$ cd
$ pwd

/export/home/student
$
```

- Perform the `cd` command with the absolute pathname to your home directory.

```
$ cd /export/home/student
$
```

To navigate 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 that equates to the absolute pathname of the user.

Note – The tilde (`~`) character is a shell facility and is not available in all shells.

```
$ cd ~student
$ pwd
/export/home/student
$
```

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

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

You can also use the tilde (~) character to navigate to another user's home directory.

```
$ cd ~user2
$ pwd

/export/home/user2
$ cd
$ pwd
/export/home/student
$
```

Viewing 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 commands include the `cat` command, the `more` command, the `tail` command, the `head` 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 contents of the entire file without pausing.

Note – Before you attempt to open a file with the `cat` command, it is recommended that you first run the `file` command to determine the file's type.

```
$ 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 truncated)
```

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%)
```

where *n%* is the percentage of the file that has been displayed.

When the `--More--(n%)` prompt appears, you can 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
Space bar	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.

The syntax of the `more` command is:

```
more filename
```

To display the first screen of the `dante` file, use 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 using the *-n* option. The *-n* option displays the number of lines (*n*) starting at the beginning of the file.

The syntax for the `head` command is:

```
head -n filename
```

To display the first six lines of the `/usr/dict/words` file, enter the `head` command with the *-n* option set to `6`.

```
$ 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 using the *-n* or *+n* options. The *-n* option displays *n* lines from the end of the file. The *+n* option displays the file from line *n* to the end of the file.

The syntax 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 set to `5`.

```
$ 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 set to `25136`.

```
$ 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 syntax for the `wc` command is:

```
wc -options filenames
```

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

Table 3-4 Options for the `wc` Command

Option	Description
`-l`	Line count
`-w`	Word count
`-c`	Byte count
`-m`	Character count

When you use the `wc` command without options, the output 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, use 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

You can use the `lp` command, the `lpstat` command, and the `cancel` command to submit print requests to a destination, to display the status of all user print requests, and to cancel print requests.

Using the `lp` Command

The `lp` command is located in the `/usr/bin` directory. The `lp` command submits a print job to the default printer or to another printer by specifying the printer name. Perform one of the following commands:

```
$ /usr/bin/lp filename
```

or

```
$ /usr/bin/lp -d printername filename
```

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.

Table 3-5 describes some of the options that 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, use the `lp` command without options.

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

Note – The example above assumes that you have a default printer set up that is called `printerA`. To check the name of your default printer, use the `lpstat -d` command.

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

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

Using the `lpstat` Command

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

The syntax for the `lpstat` command is:

```
lpstat -options printer
```

Table 3-6 describes some of 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   student  391        Feb 6  16:30    on printerA
printerB-3   user2    551        Feb 6  16:45     filtered
$
```

Table 3-7 describes the information in the status response of the `lpstat` command.

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

Status	Definition
Request-ID	Name of the printer and job number
User-ID	Name of the user accessing the printer
File Size	Output size in bytes
Date/Time	Current date and time
Status	Status of the print request

To display requests in the queue for `printerB`, enter the `lpstat` command followed by the printer name.

```
$ lpstat printerB
printerB-3     user2    551         Feb 6  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 Feb 7 09:43 2009
_default accepting requests since Feb 8 08:20 2009
printerA accepting requests since Feb 8 08:20 2009
printer printerB is idle. enabled since Feb 7 09:43 2009. available.
printer _default is idle. enabled since Feb 8 08:20 2009. available.
printer printerA is idle. enabled since Feb 8 08:20 2009. 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 Feb 7 09:43 2009
_default accepting requests since Feb 8 08:20 2009
printerA accepting requests since Feb 8 08:20 2009
$
```

Using the `cancel` Command

The `cancel` command enables you to cancel previously sent print requests, using the `lp` command. You can use the `lpstat` command to identify the *`printer`* associated with your print request.

The syntax for the `lp cancel` command is:

```
cancel Request-ID
```

or

```
cancel -u username
```

Note – When you use the desktop environment 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. To identify the *`Request-ID`* for your print request, use the `lpstat` *`printername`* command.

```
$ lpstat printerB
printerB-3 user3 630            Feb 6 16:45
printerB-4 user3 631            Feb 6 16:45
printerB-5 user3 632            Feb 6 16:47
$
```

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

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

To remove all requests made by `user3`, enter the 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.

Exercise: Accessing Files and Directories

In this exercise, you use the commands described in this module to list and change directories. All the directories are situated within your home directory.

Preparation

Ensure that a default printer is configured for your system. You can use the `lpstat -d` command to view the default printer information. Ask your instructor for any help required.

RLDC

In addition to being able to use local classroom equipment, this lab was designed to also use equipment located in a remote lab data center. Directions for accessing and using this resource can be found at:

`http://remotelabs.sun.com/`

Ask your instructor for the particular SSH configuration file that you should use to access the appropriate remote equipment for this exercise.

Tasks

To access files and directories, 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. Display a long list of the contents of the current working directory.

9. Change to the `fruit` directory.

10. Change to the `planets` directory available under `$HOME/dir3` directory using the relative pathname.

11. Return to your home directory.

12. Change directories to the `dir1` directory using an absolute pathname.

13. Return to your home directory.

14. Change to the `/etc` directory using a relative pathname.

15. Return to your home directory.

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

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

18. Under what circumstances must you refrain from using the `cat` command to view the contents of a file?

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

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

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

22. Distinguish between the `head` and `tail` commands.

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

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

25. What action occurs when you enter the `lp -m` command for the default printer?

26. What does the ~ symbol represent?

Exercise Summary

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

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise Solutions: Accessing Files and Directories

To access files and directories, complete the following steps. Write the commands that you use to perform each task in the space provided.

1. Display your current working directory.

```
$ pwd
```

2. Change to your home directory.

```
$ cd
$ pwd
```

3. Display the contents of your current working directory.

```
$ ls
```

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
```

7. Change to the `dir1` directory.

```
$ cd dir1
$ pwd
```

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

```
$ ls -l
```

9. Change to the `fruit` directory.

```
$ cd fruit
$ pwd
```

10. Change to the `planets` directory available under `$HOME/dir3` directory using the relative pathname.

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

11. Return to your home directory.

```
$ cd
$ pwd
```

12. Change to the `dir1` directory available under `$HOME` directory using an absolute pathname.

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

or

```
$ cd ~/dir1
```

13. Return to your home directory.

```
$ cd;pwd
```

14. Change to the `/etc` directory using a relative pathname.

```
$ pwd

/export/home/student
$ cd ../../../etc
$ pwd
/etc
```

15. Return to your home directory.

```
$ cd;pwd
```

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

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

```
$ cat fruit
```

18. Under what circumstances must you refrain from using the cat command to view the contents of a file?

Do not use the `cat` *command to view binary files.*

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

```
$ cat fruit fruit2
```

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

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

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

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

22. Distinguish between the `head` and `tail` commands.

The `head` *command displays the first 10 lines of a file, whereas the* `tail` *command displays the last 10 lines of a file.*

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

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

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

```
$ lp dante_1
```

25. What action occurs when you enter the `lp -m` command for the default printer?

    The `lp -m` *command sends a mail message to you after the print job is complete.*

26. What does the ~ symbol represent?

    The ~ symbol represents the user's home directory.

Notes:

Working with Files and Directories

Objectives

This module introduces commands that enable you to change the contents of directories in the Solaris OS, by using commands to create, copy, rename, and remove files and directories.

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

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

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 files to a specified target file or directory. The target file or directory is the last argument in the command.

Caution – The `cp` command is a destructive command if not used with the correct option.

Copying Files

You can 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 functions of the command. For example, using the `-i` (interactive) option prevents overwriting existing files when copying files. When you use the `cp` command with the `-i` option, it prompts you for confirmation before the copy overwrites an existing target.

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

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

The *`source`* option is a file. The *`target`* option can be a file or a directory.

Table 4-1 describes some options you can use with the `cp` command when you are copying files and 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

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 name of the source file and the target file.

To copy the file named `file3` to a new file named `feathers`, within the `student` directory, enter the following commands:

```
$ cd
$ pwd

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

To copy the `feathers` file to a new file named `feathers_6`, within the `student` directory, enter the `cp` command.

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

Copying Multiple Files

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

To copy the `feathers` file and the `feathers_6` file from the `student` directory into the `dir1` subdirectory, enter the following commands:

```
$ pwd

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

Figure 4-2 shows how you can copy the `feathers` file and the `feathers_6` file to the `dir1` directory.

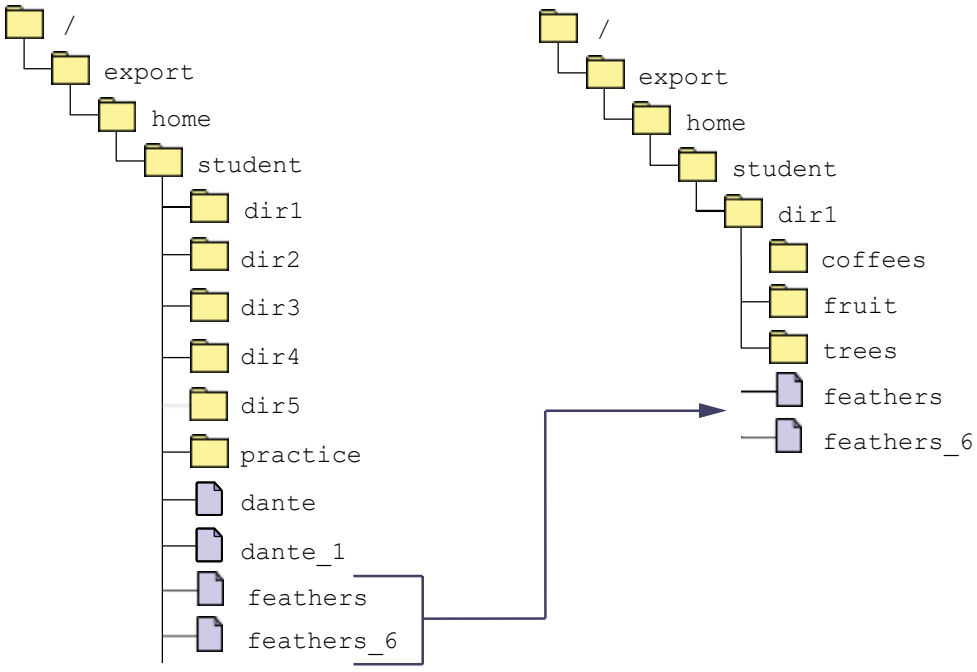


Figure 4-2 Copying Multiple Files

Preventing Overwrites to Existing Files While Copying

To prevent overwriting existing files when copying new files, you can enter the `cp` command with the `-i` option. When you use the `-i` option, the system prompts you for a confirmation before overwriting existing files with new ones.

- A `yes` response permits the overwrite.
- A `no` response prevents the `cp` command from overwriting the target file.

To copy the `feathers` file to the `feathers_6` file, enter the `cp -i` command. Because the `feathers_6` file already exists, the overwrite prompt appears.

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

Copying Directories

You can use the `cp` command with the `-r` option to copy a directory recursively. If the target directory does not exist, the `cp -r` command creates a new directory with that name. If the target directory exists already, the `cp -r` command creates a new sub-directory with that name, below the destination directory.

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

```
cp -option sources target
```

The *source* option is one or more directory names. The *target* option is a single directory name. To copy the contents of the `dir3` directory to a new directory named `dir10`, use the `cp -r` command. Both directories are in the `student` directory.

```
$ cd
$ pwd

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

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

```
$ cd
$ pwd

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

Moving and Renaming Files and Directories

You can use the `mv` command to:

- Move files and directories within the directory hierarchy
- Rename existing files and directories

The `mv` command does not affect the contents of the files or directories being moved or renamed.

Caution – The `mv` command is a destructive command if not used with the correct option.

Moving and Renaming a File

You can use the `mv` command to rename a single file within the same directory.

For example, use the `mv` command to rename the `dante` file to `dantenew` and then back again.

```
$ pwd

/export/home/student
$ mv dante dantenew
$ ls
dante_1     dir2        feathers    file.3      file4       myvars
dantenew    dir3        feathers_6  file1       fruit       practice
dir1        dir4        file.1      file2       fruit2      tutor.vi
dir10       dir5        file.2      file3       greetings
$ mv dantenew dante
$ ls
dante       dir2        feathers    file.3      file4       myvars
dante_1     dir3        feathers_6  file1       fruit       practice
dir1        dir4        file.1      file2       fruit2      tutor.vi
dir10       dir5        file.2      file3       greetings
$
```

Moving a File to Another Directory

You can use the `mv` command to move a file to a different directory. The syntax for the `mv` command is:

mv *source target*

The *source* option is the old file or directory name. The *target* option is the new file or directory name.

To move the `brands` file from the `coffees` directory into the `student` directory, enter the following commands.

```
$ cd ~/dir1/coffees
$ pwd
/export/home/student/dir1/coffees
$ ls
beans  brands nuts
$ mv brands ~
$ ls
beans  nuts
$ cd
$ pwd
/export/home/student
$ ls -l brands
-rw-r--r--  1 student  class          0 Feb 6  2009 brands
$
```

Note – Unlike the `copy` command, the `mv` command moves your file or directory, and the original will no longer exist.

Prevent a File Overwrite With the `-i` Option

The `-i` option prompts you for confirmation to prevent you from overwriting existing file(s) by the new file(s).

mv -i *source target*

- A `yes` response permits the `mv` command to overwrite an existing file(s).
- A `no` response prevents the `mv` command from overwriting the existing file(s).

Moving a Directory and its Contents

You can use the `mv` command to move a directory and its contents to a different directory.

To move the `practice` directory and its contents into a brand new empty directory named `letters`, enter the following commands:

```
$ cd
$ pwd

/export/home/student
$ ls -l practice
-rw-r--r--  1 student  class         0 Feb 6  2009  mailbox
-rw-r--r--  1 student  class         0 Feb 6  2009  project
-rw-r--r--  1 student  class         0 Feb 6  2009  projection
-rw-r--r--  1 student  class         0 Feb 6  2009  research
-rw-r--r--  1 student  class         0 Feb 6  2009  results
$ mkdir letters
$ ls -l letters
total 0
$ mv practice letters
$ ls -l letters
drwxr-xr-x  2 student class       512 Feb 6 14:11 practice
$ ls -lR letters
letters:
total 2
drwxr-xr-x  2 student  class      512 Feb 6 14:12 practice

letters/practice:
total 0
-rw-r--r--  1 student  class         0 Feb 6  2009 mailbox
-rw-r--r--  1 student  class         0 Feb 6  2009 project
-rw-r--r--  1 student  class         0 Feb 6  2009 projection
-rw-r--r--  1 student  class         0 Feb 6  2009 research
-rw-r--r--  1 student  class         0 Feb 6  2009 results
$
```

When you move a single directory to a target directory that *does not exist*, you are actually renaming the current directory and its path.

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

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

Renaming a Directory

You can use the `mv` command to rename directories within the current directory.

Enter the following commands to rename the `dir5` directory to `dir5new`

```
$ cd
$ pwd
/export/home/student
$ mv dir5 dir5new
$ ls
brands     dir10      dir5new    file.2     file3      greetings
dante      dir2       feathers   file.3     file4      letters
dante_1    dir3       feathers_6 file1      fruit      myvars
dir1       dir4       file.1     file2      fruit2     tutor.vi
$ mv dir5new dir5
$ ls
brands     dir10      dir5       file.2     file3      greetings
dante      dir2       feathers   file.3     file4      letters
dante_1    dir3       feathers_6 file1      fruit      myvars
dir1       dir4       file.1     file2      fruit2     tutor.vi
$
```

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

You can 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 syntax for the `touch` command is:

```
touch filename
```

You can use absolute or relative pathnames on the command-line when creating new files.

To create an empty file named `space` in the `dir3` directory, enter the following commands:

```
$ pwd

/export/home/student
$ cd dir3
$ ls
planets
$ touch space
$ ls
planets space
$
```

To create three empty files named `moon`, `sun`, and `cosmos` in the `dir3` directory, use the `touch` command.

```
$ touch moon sun cosmos
$ ls
cosmos   moon    planets space    sun
$
```

Creating Directories

You can use the `mkdir` command with a *`directory_name`* to create directories. If the *`directory_name`* includes a pathname, 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 syntax for the `mkdir` command includes:

```
mkdir directory_name
```

and

```
mkdir -p directory_names
```

You can use absolute or relative pathnames on the command-line when creating new directories.

To create a new directory, named `Reports`, within the `student` directory, use the `mkdir` command.

```
$ cd
$ pwd
/export/home/student
$ mkdir Reports
$ ls -ld Reports
drwxr-xr-x   2 student  class        512 Feb 6 19:02 Reports
$
```

To create a new directory named `empty_directory` located inside a directory named `newdir`, use the `mkdir` command with the `-p` option. The `newdir` directory does not yet exist.

```
$ cd
$ pwd

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

Figure 4-3 shows two new directories named `Reports` and `newdir` in the `student` directory.

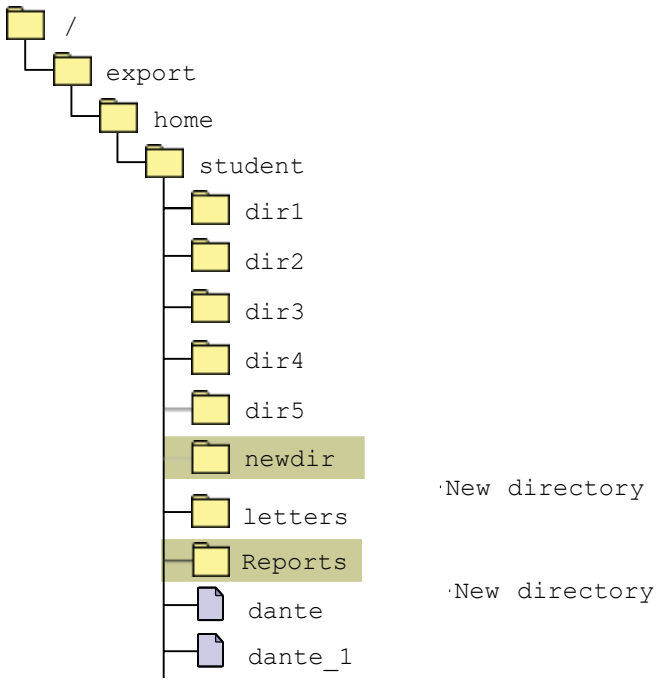


Figure 4-3 Creating a Directory

To create a `Weekly` directory in the `Reports` directory, enter the `mkdir` command.

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

To create the `dir1`, `dir2`, and `dir3` directories in the `Weekly` directory, enter the `mkdir` command.

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

Removing Files and Directories

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

Caution – The `rm` command is a destructive command if not used with the correct option.

Removing Files

You can use the `rm` command to remove a file or multiple files on the same command-line.

The syntax for the `rm` command is:

```
rm -option filename
```

To remove the file named `projection` from the `letters` directory, enter the following commands:

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

To remove the `research` file and the `project` file from the `letters` directory, enter the following commands:

```
$ pwd

/export/home/student/letters
$ ls
mailbox  project research results
$ rm research project
$ 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.

To remove the contents of a directory, enter the `rm` command with the `-i` option.

```
$ cd
$ rm -i file*

rm: remove file1: (yes/no) ? Y
rm: remove file2: (yes/no) ?  Y
rm: remove file3: (yes/no) ? Y
rm: remove file4: (yes/no) ? Y
$ ls
$
```

Note – The asterisk (`*`) symbol used in the previous example is a wild card character, and it is described in more detail in Module 6, "Using Commands Within the Shell."

Removing Directories

There are two ways to remove directories. You can use the `rmdir` command to remove empty directories. You can use the `rm` command with the `-r` option to remove directories that contain files and subdirectories.

To remove a directory in which you are currently working, you must change to its parent directory.

Removing an Empty Directory

You can use the `rmdir` command to remove empty directories.

The syntax for the `rmdir` command is:

```
rmdir directories
```

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

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

To remove a directory that is empty (contains no files or subdirectories), enter the following commands:

```
$ cd
$ pwd
/export/home/student
$ cd newdir
$ pwd
$ ls -F
empty_directory/
$ rmdir empty_directory
$ ls
$
```

Removing a Directory With Contents

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

The syntax for the `rm` command is:

```
rm -options directories
```

If you do not use the `-r` option, the following error message appears:

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

Table 4-2 describes 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 the accidental removal of existing files or directories

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

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

To remove the `letters` directory and its contents, use the `rm -r` command.

```
$ cd
$ pwd

/export/home/student
$ ls letters
mailbox results
$ rm -r letters
$ ls letters
letters: No such file or directory
$
```

To interactively remove a directory and its contents, use the `rm -r` command with the `-i` option. Create a new directory called `rmtest` in your `/export/home/student` directory.

```
$ pwd

/export/home/student
$ mkdir rmtest
$ ls
Reports    dir10      feathers   file1      fruit2     tutor.vi
brands     dir2       feathers_6 file2      greetings
dante      dir3       file.1     file3      myvars
dante_1    dir4       file.2     file4      newdir
dir1       dir5       file.3     fruit      rmtest
$

$ cd rmtest
$ touch testfile
$ cd
$ rm -ir rmtest
rm: examine files in directory rmtest (yes/no)? y
rm: remove rmtest/testfile (yes/no)? y
rm: remove rmtest: (yes/no)? y
$ ls
Reports    dir10      feathers   file1      fruit2
brands     dir2       feathers_6 file2      greetings
dante      dir3       file.1     file3      myvars
dante_1    dir4       file.2     file4      newdir
dir1       dir5       file.3     fruit      tutor.vi
$
```

Using Symbolic Links

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.

There are two main reasons you might choose to use symbolic links:

- To move files to a new location – This includes moving a directory on a different disk (partition) but leaving a link so that other users do not need to know about the move.
- To provide a convenient name for a file rather than the original name, which might be complicated or unknown – When accessing a USB flash drive, a user can type `ls /rmdisk/rmdisk0` without having to find out what the USB flash drive is called.

A file system is a collection of certain types of files, organized for administrative convenience. The organization of these files enables you to store files that need to be shared on one machine. These shared files can be accessed by many machines by using a remote file access mechanism.

A symbolic link is a pointer that contains the pathname to another file or directory. The link makes the file or directory easier to access if it has a long pathname. A symbolic link file is identified by the letter `l` in the file-type field. To view symbolic link files, use the `ls -l` command.

Creating Symbolic Links

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

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

`ln -s` *`source_file target_file`*

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

To create a symbolic link file, named `dante_link`, to the `dante` file, enter the `ln -s` command.

```
$ cd
$ pwd
/export/home/student
$ mv dante /var/tmp
$ ln -s /var/tmp/dante dante_link
$
```

You can display a list of files and directories, by entering the `ls -F` command.

```
$ ls -F
Reports/    dir10/      feathers    file1*      fruit2
brands      dir2/       feathers_6  file2*      greetings
dante_1     dir3/       file.1*     file3*      myvars
dante_link@ dir4/       file.2*     file4*      newdir/
dir1/       dir5/       file.3*     fruit       tutor.vi
$
```

The output of the `ls -F` command shows the file `dante_link` with the `@` symbol following it to indicate that `dante_link` is a symbolic link.

```
$ cat dante_link
                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 truncated)
```

To see the pathname to which a symbolic link is pointing, enter the `ls -l` command with the symbolic link file name.

```
$ ls -l dante_link
lrwxrwxrwx  1 student  class        14 Feb 6 14:17 dante_link ->
/var/tmp/dante
$
```

Removing Symbolic Links

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

To remove the `dante_link` symbolic link file, enter the following commands:

```
$ cd
$ pwd

/export/home/student
$ ls -l dante_link
lrwxrwxrwx  1 student  class         14 Feb 6 14:17 dante_link ->
/var/tmp/dante
$ rm dante_link
$ cat dante
No such file or directory
$ mv /var/tmp/dante ~/dante
$ ls -l dante dante_link
dante_link: No such file or directory
-rw-r--r--  1 student  class     1319 Feb 6 14:18 dante
$
```

Exercise: Using Directory and File Commands

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

Preparation

No special preparation is required for this exercise.

RLDC

In addition to being able to use local classroom equipment, this lab was designed to also use equipment located in a remote lab data center. Directions for accessing and using this resource can be found at:

`http://remotelabs.sun.com/`

Ask your instructor for the particular SSH configuration file that you should use to access the appropriate remote equipment for this exercise.

Tasks

To use directory and file commands, 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/beans` file into the `dir4` directory, and call it `roses`.

   _____

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

   _____

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

   _____

5. Complete the missing options and their descriptions in Table 4-3:

Table 4-3 Directory Options

Option	Description
`cp -i`	
	Includes the contents of a directory, including the contents of all subdirectories, when you copy a directory

6. From your home directory, create a directory called `practice1`.

_____

7. Using a single command, copy the files `file.1` and `file.2` to the `practice1` directory.

_____

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

_____

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

_____

10. Using a single command with options, copy the `play` directory in the `practice1` directory to a new directory in the `practice1` directory called `appointments`.

_____

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

_____

12. In your home directory, create a directory called `house` with a subdirectory of `furniture` using a single command.

_____

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

_____

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

_____

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

16. Using a single command, remove the directories called `memos` and `misc` from your home directory.

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

18. 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.

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

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

21. Verify that the symbolic link works.

22. Type `q` to quit the `mycontents` file view.

23. Remove the symbolic link that you created in Step 20.

Exercise Summary

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

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise Solutions: Using Directory and File Commands

To use directory and file commands, complete the following steps:

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

```
$ cd; ls
```

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

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

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

```
$ mkdir dir3/vegetables
```

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

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

5. Complete the missing options and their descriptions in Table 4-4:

Table 4-4 Directory Options

Option	*Description*
`cp -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

6. From your home directory, create a directory called `practice1`.

```
$ mkdir practice1
```

7. Using a single command, copy the files `file.1` and `file.2` to the `practice1` directory.

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

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

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

9. 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
```

10. Using a single command, copy the `play` directory in the `practice1` directory to a new directory in the `practice1` directory called `appointments`.

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

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

```
$ ls -R practice1
```

12. In your home directory, create a directory called `house` with a subdirectory of `furniture` using a single command.

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

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

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

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

```
$ mkdir records memos misc
```

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

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

16. Using a single command, remove the directories called `memos` and `misc` from your home directory.

```
$ rmdir memos misc
```

or

```
$ rm -r memos misc
```

17. 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
```

18. 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
$
```

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

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

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

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

21. Verify that the symbolic link works.

```
$ more mycontents
```

22. Type `q` to quit the `mycontents` file view.

23. Remove the symbolic link that you created in Step 20.

```
$ rm mycontents
$ ls mycontents
mycontents: No such file or directory
```

Notes:

Using the `vi` Editor

Objectives

This module introduces the `vi` editor and describes the `vi` commands. These commands include input commands, positioning commands, and editing commands.

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

- Describe the fundamentals of the `vi` editor
- Modify files using the `vi` editor

Fundamentals of the vi Editor

The visual display or vi editor is an interactive editor that you can use to create and modify text files. You can use the vi editor when the desktop environment window system is not available. The vi editor is also the only text editor that 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. You can either write the changes to the disk, or discard them.

Figure 5-2 shows the initial vi display in a terminal window.

Figure 5-2 Initial vi Editor Display

The vi Editor Modes of Operation

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

- Command mode
- Input mode
- Last line mode

Command Mode

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

Input Mode

You can insert text into a file in the input mode. The vi editor interprets everything you type in the input mode as text. To invoke input mode, press one of the following lower-case keys:

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

You can also invoke the input mode to insert text into a file by pressing one of the following upper-case keys:

- I – Inserts text at the beginning of the line
- O – Opens a new blank line above the cursor
- A – Appends text at the end of the line

Last Line Mode

You can use advanced editing commands in the last line mode. To access the last line mode, enter a colon (:) while in the command mode. Entering the colon (:) character places the cursor at the bottom line of the screen.

Switching Between the Command and Input Modes

The default mode for the vi editor is the command mode. When you enter an i, o, or a command, the vi editor switches to the input mode. After editing a file, press Esc key to return the vi editor to the command mode. When in the 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 the vi *filename* command to create a file. You are automatically in the command mode.
2. Type the i command to insert text. The i command switches the vi editor to the input mode.
3. Press Esc key to return to the command mode.
4. Enter the :wq command to save the file, exit the vi editor and return to the shell prompt.

Using the vi Command

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

The syntax for the vi command includes the following three choices:

```
vi
```

or

```
vi filename
```

or

```
vi options filename
```

If the system crashes while you are editing a file, you can use the `-r` option to recover the file.

To recover a file, enter the command:

```
$ vi -r filename
```

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

```
$ vi -R filename
```

The file opens in read-only mode to prevent accidental overwriting of the contents of the file.

Modifying Files With the vi Editor

You can use the vi editor to view files in the 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 the Read-Only Mode

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

The syntax for the view command is:

```
view filename
```

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

```
$ cd
$ view dante
```

The dante file appears. Enter the :q command to exit the file, the vi editor, and return to the shell prompt.

Inserting and Appending Text

Table 5-1 describes the commands that you can use to insert and append text to a new or existing file using the vi editor. Each of these commands switch the vi editor to the input mode. To return to the command mode, press the Esc key.

Table 5-1 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. Also, most of the input commands and cursor movements can be preceded by a number to repeat the command that number of times.

Moving the Cursor Within the vi Editor

Table 5-2 shows the key sequences that move the cursor in the vi editor.

Table 5-2 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 space bar	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-white space 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*
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
Control-G	Displays current buffer information

Editing Files Using the vi Editing Commands

You can use numerous commands to edit files using the vi editor. 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 5-3 shows the commands that delete text in the vi editor.

Table 5-3 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. To terminate this operation, press Esc key.
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*–*n* (for example, :5,10d deletes Lines 5–10).

Note – Output from the delete command writes to a buffer from which text can be retrieved.

Using the Text-Changing Commands

Table 5-4 describes the commands that you can use to change text, undo a change, and repeat an edit function in the vi editor. Many of these commands change the vi editor to input mode. To return to command mode, press the Esc key.

Table 5-4 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	Undo the previous command
U	Undo all changes to the current line
.	Repeats the previous command

Using the Text-Replacing Commands

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

Table 5-5 Search and Replace Commands

Command	Function
/*string*	Searches forward for the *string*.
?*string*	Searches backward for the *string*.
n	Searches for the next occurrence of the *string*. Use this command after searching for a *string*.
N	Searches for 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 Text-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 those lines of text, held in the temporary buffer, into the current document at the specified location.

The copy (co) and move (m) commands copy or move specified lines to the selected location within the file.

Table 5-6 shows the commands that yank (yy) and put (p, P) text in the vi editor.

Table 5-6 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 5-7 shows the commands that copy (co) and move (m) text in the vi editor.

Table 5-7 Additional Copy and Paste Commands

Command	Function
:*n*,*n* co *n*	Copies lines *n* –*n* and puts them after line *n*. For example, :1,3 co 5 copies lines 1–3 and puts them after line 5.
:*n*,*n* m *n*	Moves lines *n*–*n* to line *n*. For example, :4,6 m 8 moves lines 4–6 to line 8, line 6 becomes line 8, line 5 becomes line 7, and line 4 becomes line 6.

Using the File Save and Quit Commands

Table 5-8 describes the commands that save the text file, quits the vi editor and returns to the shell prompt.

Table 5-8 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 can use the set command to enable and disable variables.

Table 5-9 describes some of the variables of the set command, including displaying line numbers and invisible characters, such as the Tab character and end-of-line characters.

Table 5-9 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 $ for end-of-line characters
: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 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 any of the set variables into the .exrc file.
3. Enter each set *variable* without the preceding colon, (as shown in Table 5-9).
4. Enter each 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 5-3 on page 5-15 as a reference to complete the exercise.

Preparation

No special preparation is required for this exercise.

RLDC

In addition to being able to use local classroom equipment, this lab was designed to also use equipment located in a remote lab data center. Directions for accessing and using this resource can be found at:

http://remotelabs.sun.com/

Ask your instructor for the particular SSH configuration file that you should use to access the appropriate remote equipment for this exercise.

Tasks

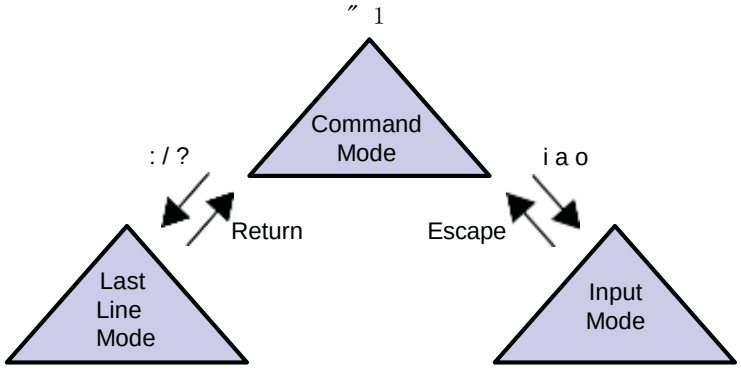
To learn how to use the vi editor, complete the following steps.

1. Make sure that you are in your home directory. To open the tutor.vi tutorial file, enter the following command:

```
$ vi tutor.vi
```

2. Complete the lessons outlined in this tutorial.

Figure 5-3 shows a quick reference chart for the vi editor.



Search Functions

Go forward to exp
Go backward to exp

Move and Insert Text

Delete line 3-8
Move lines 4-9 to 12
Copy lines 2-5 to 13
Write lines 5-9 to file

Save Files and Exit

Write to disk
Write to newfile
Write absolutely
Write and quit
Quit editor
Quit and discard
Re-edit current file,
Discard buffer

Control Edit Session

Display line number
Turn off line number
Show all settings
Display invisible
Characters
Wrap lines 5 spaces
From right margin

Screen/Line Movement

Move cursor down
Move cursor up
Move cursor left
Move cursor right
! Go to line start (zero)
Go to line end
Go to last file line

Word Movement

Go forward 1 word
$ Go backward 1 word

Search Functions

Repeat previous search
% Reverse previous search

Delete Text

Delete 1 character
Delete 1 word
Delete 1 line
& Delete to end of line
! Delete to start of line
Delete to end of file

Cancel Edit Function

Undo last change
, Do last change again

Copy and Insert Text

(Yank a copy
(Yank a copy of 5 lines
Put below cursor
) Put above cursor

Add/Append Text

Append after cursor
* Append at line end
Insert before cursor
Insert text 5 times
+ Insert at beginning of line

Add New Lines

Open a line below cursor
, Open a line above cursor

Search Functions

Repeat previous search
% Reverse previous search

Change Text

Change a word
Change 3 words
. Change line
/ Replace one character
0 Replace/type over a line

Figure 5-3 Quick Reference Chart for the vi Editor

Exercise Summary

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

- Experiences
- Interpretations
- Conclusions
- Applications