A Guide to Unix Using Linux Fourth Edition

Chapter 2

Exploring the UNIX/Linux File Systems and File Security

Objectives

- Navigate the file system
- Create and remove directories
- Copy and delete files
- Configure file permissions

Using Paths, Pathnames, and Prompts

- Files are stored in directories in the file system, starting from the root file system directory
- To specify a file or directory, use its **pathname**
  - Follows the branches of the file system to the desired file
- A forward slash (/) separates each directory name
  - Example: /home/jean/source/phones.502

Using and Configuring Your Command-Line Prompt

~ is shorthand for the home directory

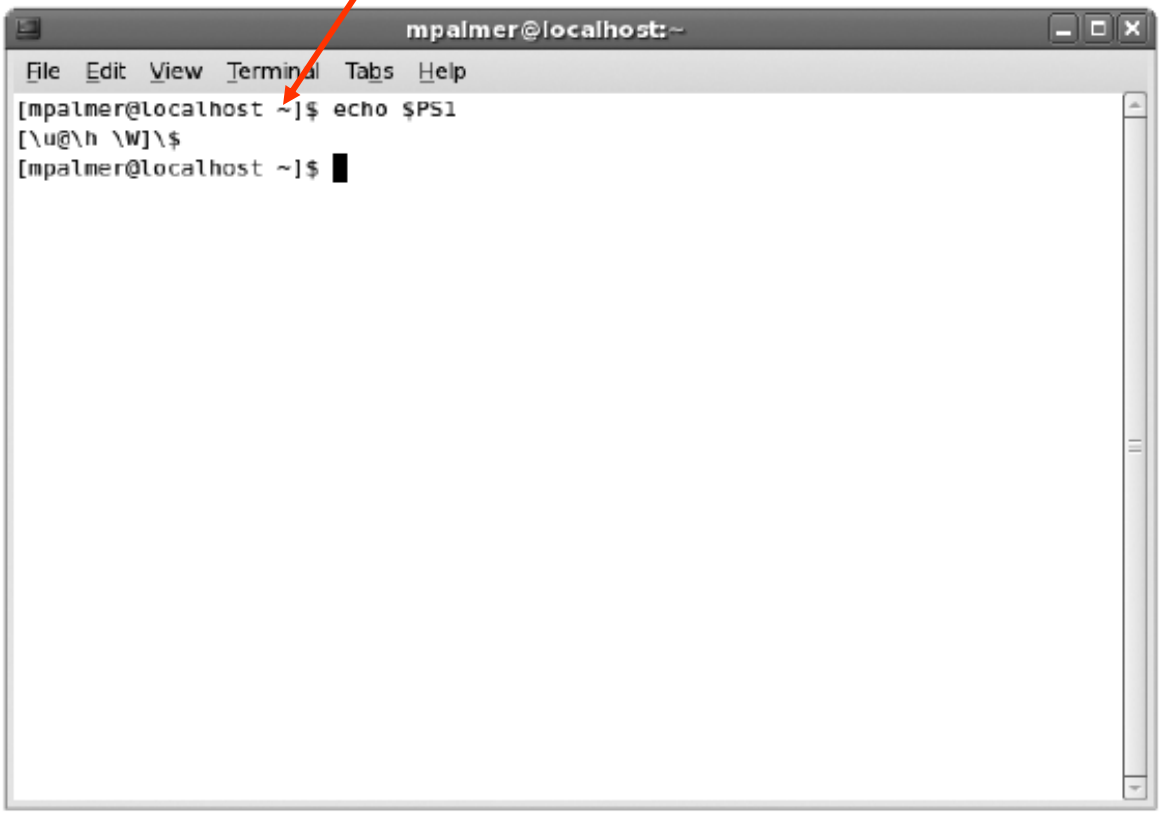

Figure 2-4 Viewing the contents of the PS1 variable

The pwd Command

- *pwd* prints the working directory

Syntax **pwd**

Dissection

- Use *pwd* to determine your current working directory.
- Typically, there are no options with this command.

- Useful for regular users, system administrators, and in scripts

Navigating the File System

- *cd* stands for change directory

Syntax **cd** [*directory*]

Dissection

- *directory* is the name of the directory to which you want to change. The directory name is expressed as a path to the destination, with slashes (/) separating subdirectory names.

- Provide an absolute or relative path to the directory
  - **Absolute path**: begins at the root level and lists all subdirectories to the destination file
    - Example: `cd /home/jean/source`
  - **Relative path**: takes a shorter journey
    - Example: `cd source or cd`

Using Dot and Dot Dot Addressing Techniques

- A single dot character means the current working directory
- Dot dot means the parent directory
- These addressing mechanisms are useful when navigating the file system
  - Example: `cd ../tricia/source`

Listing Directory Contents

- Use the *ls* (list) command to display a directory's contents, including files and other directories

Syntax **ls** [-option] [*directory or filename*]

Dissection

- Common arguments include a directory name (including the path to the directory) or a file name.
- Useful options include:
  - *-l* to view detailed information about files and directories
  - *-S* to sort by size of the file or directory
  - *-X* to sort by extension
  - *-r* to sort in reverse order
  - *-t* to sort by the time when the file or directory was last modified
  - *-a* to show hidden files ← Appear with a dot at the beginning
  - *-i* to view the inode value associated with a directory or file

Listing Directory Contents (continued)

Group (root)

Size (4096 bytes)

Owner (root)

Date and time of last modification

Number of links (2)

File or directory name (bin)

File type and access permissions: **drwxr-xr-x**

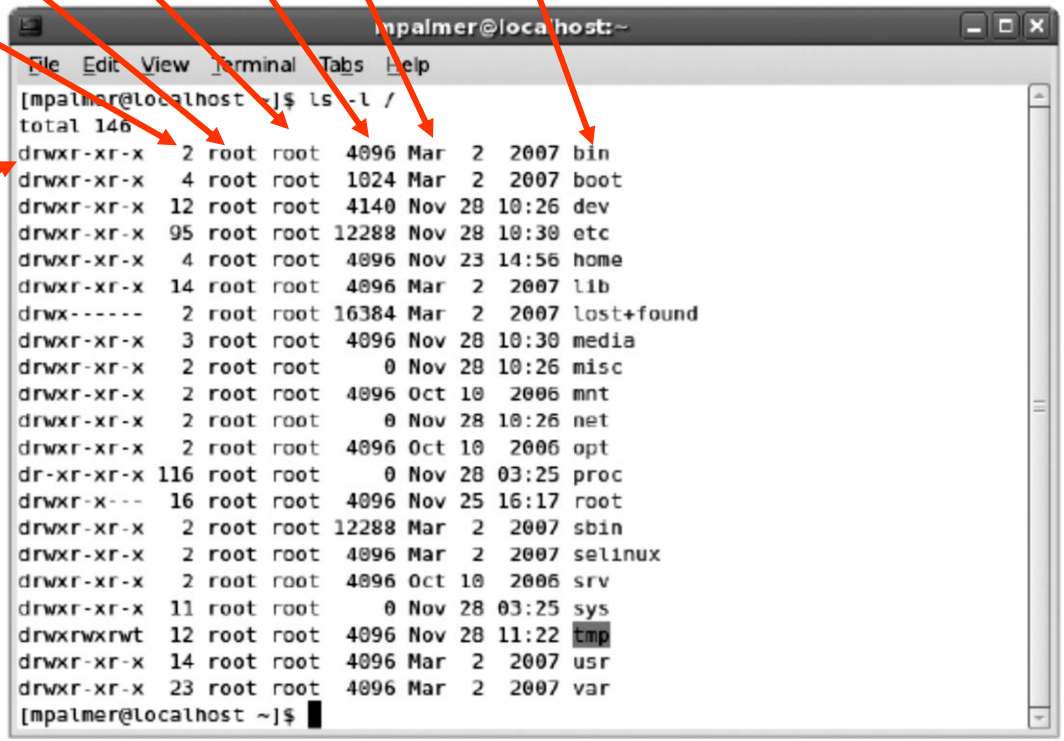

Figure 2-5 Using *ls -l* to view the root file system directory contents

Using Wildcards

- **Wildcard:** special character that can stand for any other character or a group of characters
  - * represents any group of characters in a file name
    - Example: *ls *.txt*

      ```
      instructions.txt minutes.txt
      ```
  - ? takes the place of only a single character
    - Example: *ls list?*

      ```
      list1 list2
      ```

Creating and Removing Directories

- *mkdir* is used to create a new directory

Syntax **mkdir** [-option] *directory*

Dissection

- The argument used with *mkdir* is a new directory name.
- There are only a few options used with *mkdir*. One option is to use *-v* to display a message that verifies the directory has been made.

- Delete empty directories using *rmdir*

Syntax **rmdir** [-option] *directory*

Dissection

- The argument used with *rmdir* is a directory.
- As is true for *mkdir*, *rmdir* has only a few options. Consider using the *-v* option to display a message that verifies the directory has been removed.

  - Use *rm -r* to delete a directory that is not empty

Copying and Deleting Files

- Use *cp* to copy files and *rm* to delete them

Syntax **cp** [-option] *source destination*

Dissection

- The argument consists of the source and destination directories and files, such as *cp /home/myaccount/myfile /home/youraccount*.
- Common options include:
  - *-b* makes a backup of the destination file if the copy will overwrite a file
  - *-i* provides a warning when you are about to overwrite a file
  - *-u* specifies to only overwrite if the file you are copying is newer than the one you are overwriting

Syntax **rm** [-option] *filename*

Dissection

- The argument consists of the name of the file to delete.
- The *-i* option causes the operating system to prompt to make certain you want to delete the file before it is actually deleted.

Configuring File Permissions for Security

- Users can set **permissions** for files/directories they own so as to establish security
  - System administrators also set permissions to protect system and shared files
- Permissions manage who can read, write, or execute files
- Original file owner of a file is the account that created it
  - File ownership can be transferred to another account

Configuring File Permissions for Security (continued)

File type	Meaning
-	Normal file
d	Subdirectory
l	Symbolic link
b	Block device file
c	Character device file

```
Excerpt from ls -l /etc

drwxr-xr-x   16 root    root         4096   Jan  17    9:29   X11
-rw-r--r--    1 root    root           46   Jan  15   19:11   adjtime
drwxr-xr-x    1 root    root         1024   Feb  27    2007   cron.daily
```

```
Excerpt from ls -l /home/jean/source

rw-rw-r--     1 jean    jean          387   Dec  12   23:11   phones.502
```

Figure 2-6 File types described in directory listings

Configuring File Permissions for Security (continued)

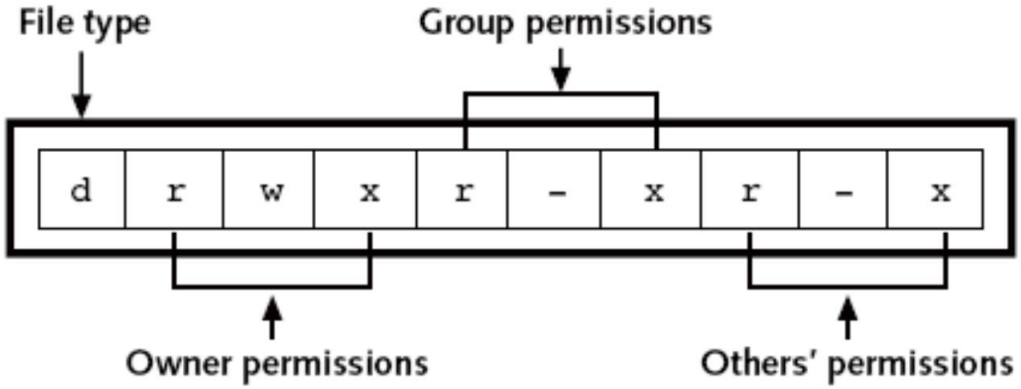

Figure 2-7 Example of the file type and the file permissions for a file

Syntax **chmod** [-option] *mode filename*

Dissection

- The argument can include the mode (permissions) and must include the file name. You can also use a wildcard to set the permissions on multiple files.
- Permissions are applied to owner (u), group (g), and others (o). The permissions are read (r), write (w), and execute (x). Use a plus sign (+) before the permissions to allow them or a hyphen (-) to disallow permissions. Octal permissions are assigned by a numeric value for each owner, group, and others.

Configuring File Permissions for Security (continued)

- The system administrator assigns group ids when he or she adds a new user account
  - A **group id (GID)** gives a group of users equal access to files that they all share
- Using *chmod* to change permissions of a file:

```
chmod ugo+rwx myfile
chmod go-wx account_info
```

  - Or, use the octal permission format

```
chmod 711 data
chmod 642 data
```

Table 2-5 Suggestions for setting permissions

Type of File or Directory	Permissions Suggestion
System directories such as /bin, /boot, /dev, /etc, /sbin, /sys, and /usr	Give all permissions to root (the owner), rx to group and others—*chmod 755*.
/root directory for the root account	Give all permissions to root (the owner), rx to group, and no permissions to others—*chmod 750*.
Your home directory	Give all permissions to owner (your account), x or no permissions to group, and no permissions to others—*chmod 710* or *chmod 700*. (If you are a student and need to give your instructor access to your home directory, consider using *chmod 705* so your instructor has rx permissions.)
A subdirectory under your home directory that you want to share with others so they can access and create files	Give all permissions to owner (your account), group, and others—*chmod 777*.
A file in your home directory that you want people to be able to view, but not change	Give all permissions to owner (your account), rx to group, and rx to others—*chmod 755*.
A file that should only be accessed by you	Give all permissions to owner and no permissions to group and others—*chmod 700*.
An archived file in your home directory that should not be changed (just preserved) and that only you should be able to view	Give rx permissions to owner and no permissions to group and others—*chmod 500*.

Summary

- In UNIX/Linux, a file is the basic component for data storage
- A file system is the UNIX/Linux systems' way of organizing files on storage devices
- The standard tree structure starts with the root (/) file system directory
- The section of the disk that holds a file system is called a partition
- A path, as defined in UNIX/Linux, serves as a map to access any file on the system

Summary (continued)

- You can customize your command prompt to display useful information
- The *ls* command displays the names of files and directories contained in a directory
- Wildcard characters can be used in a command and take the place of other characters in a file name
- Use *mkdir* to create a new directory
- Use *cp* to copy a source file to a destination file
- Use *chmod* to set permissions for files that you own

Command Summary

Command	Purpose	Options Covered in This Chapter
cd	Changes directories (with no options, *cd* goes to your home directory)	. Changes to the current working directory. .. Changes to the parent directory.
chmod	Sets file permissions for specified files	+ assigns permissions. -removes permissions.
cp	Copies files from one directory to another	**-b** makes a backup of the destination file, if an original one already exists (so you have a backup if overwriting a file). **-i** prevents overwriting of the destination file without warning. **-u** overwrites an existing file only if the source is newer than the file in the current destination.
ls	Displays a directory's contents, including its files and subdirectories	**-a** lists the hidden files. **-l** (lowercase L) generates a long listing of the directory. **-r** sorts the listing in reverse order. **-S** sorts the listing by file size. **-t** sorts by the time when the file or directory was last modified. **-X** sorts by extension.

Command Summary (continued)

Command	Purpose	Options Covered in This Chapter
mkdir	Makes a new directory	**-v** verifies that the directory is made.
mount	Connects the file system partitions to the directory tree when the system starts, and mounts additional devices, such as the CD/DVD drive	**-t** specifies the type of file system to mount.
rm	Removes a file	**-i** prompts before you delete the file.
rmdir	Removes an empty directory	**-v** provides a message to verify the directory is removed.
umask	Sets file permissions for multiple files	
umount	Disconnects the file system partitions from the directory tree	