Summary Of LPIC-1 5th Edition

**Chapter 1) Exploring Linux Command-Line Tools**

**Objectives**

✓✓ 103.1 Work on the command line

✓✓ 103.2 Process text streams using filters

✓✓ 103.4 Use streams, pipes, and redirects

✓✓ 103.7 Search text files using regular expressions

✓✓ 103.8 Basic file editing

**From Book:**

Reaching a Shell:

Ctrl+Alt+F2 key or Ctrl+Alt+T. combination (which gets you to the tty2 terminal).

Exploring Your Linux Shell Options:

to reach a command-line prompt, the program providing that prompt is a

shell. While the Bash shell program is the most popular and commonly used by the various Linux distributions, there are a few others you need to know:

**Bash**: a replacement for the standard Unix operating system shell, called the Bourne shell (named for its creator). It is also available for Windows 10, macOS, and Solaris operating systems.

**Dash**: The Debian shell, does provide faster shell program (also called a script) execution.

**KornShell:** It is a programming shell compatible with the Bourne shell but supports advanced programming features, such as those available in the C programming languages.

**Tcsh**: is an upgraded version of the C Shell

**Z shell:** This advanced shell incorporates features from Bash, tcsh, and KornShell. Advanced programming features, shared history files, and themed prompts are a few of the extended Bourne shell components it provides.

**Note**: On Linux systems, the /bin/sh file is now a symbolic link to a shell.

Listing 1.1: /bin/sh points on a CentOS distribution

$ readlink /bin/sh

bash

Listing 1.2: /bin/sh points on an Ubuntu distribution

$ readlink /bin/sh

Dash

Listing 1.3: Displaying the current shell on a CentOS distribution

$ echo $SHELL

/bin/bash

$

$ echo $BASH\_VERSION

4.2.46(2)-release

Quoting Metacharacters:

Within the Bash shell are several characters that have special meanings and functions:

\* ? [ ] ' " \ $ ; & ( ) | ^ < >

For example, the dollar sign ($) often indicates that the characters following it are a variable name. Shell quoting allows you to use metacharacters as regular characters. To shell quote a single character, use the backslash (\) as shown in Listing 1.7.

Listing 1.7: Using the echo command and shell quoting a single metacharacter

$ echo It cost \$1.00

It cost $1.00

Navigating the Directory Structure:

When you log into the Linux system, your process’s current working directory is your account’s home directory, think of the current working directory as the room you are currently in within your home. You can also employ the single dot (.) directory reference, which refers to the current working directory. it is commonly employed for tasks such as copying or moving files.

For example, to change your current working directory to your user account’s home directory,

use one of the following:

cd

cd ~

cd $HOME

Listing 1.12: Using type to determine whether a command is external or internal

$ type echo

echo is a shell builtin

$ type pwd

pwd is a shell builtin

$ type uname

uname is /usr/bin/uname

Notice in Listing 1.12 that both the echo and pwd commands are internal (built-in) programs.However, the uname command is an external program.

Using Environment Variables:

Table 1.1 Commonly used environment variables

Name Description

**BASH\_VERSION** Current Bash shell instance’s version number (Chapter 1)

**EDITOR** Default editor used by some shell commands (Chapter 1)

**GROUPS** User account’s group memberships (Chapter 7)

**HISTFILE** Name of the user’s shell command history file (Chapter 1)

**HISTSIZE** Maximum number of commands stored in history file (Chapter 1)

**HOME** Current user’s home directory name (Chapter 1)

**HOSTNAME** Current system’s host name (Chapter 8)

**LANG** Locale category for the shell (Chapter 6)

**LC\_\*** Various locale settings that override LANG (Chapter 6)

**LC\_ALL** Locale category for the shell that overrides LANG (Chapter 6)

**LD\_LIBRARY\_PATH** Colon-separated list of library directories to search prior to looking through the standard library directories (Chapter 2)

**PATH** Colon-separated list of directories to search for commands

(Chapter 1)

**PS1** Primary shell command-line interface prompt string (Chapter 1)

**PS2** Secondary shell command-line interface prompt string

**PWD** User account’s current working directory (Chapter 1)

**SHLVL** Current shell level (Chapter 1)

**TZ** User’s time zone, if different from system’s time zone (Chapter 6)

**UID** User account’s user identification number (Chapter 7)

**VISUAL** Default screen-based editor used by some shell commands

(Chapter 1)

Listing 1.13: Using set to display active environment variables

$ set

[…]

BASH=/bin/bash

[…]

HISTFILE=/home/Christine/.bash\_history

[…]

HISTSIZE=1000

you can also employ the env and printenv commands to display variables. The env

and printenv utilities allow you to see locally defined variables, such as those created in a shell script.

Listing 1.15: Executing a program outside the PATH directories

$ /home/Christine/Hello.sh

The which utility It searches through the PATH directories to find the program.

Listing 1.16: Using the which utility

$ which Hello.sh

/usr/bin/which: no Hello.sh in (/usr/local/bin:/usr/bin:

/usr/local/sbin:/usr/sbin:/home/Christine/.local/bin:/home/Christine/bin)

Listing 1.17: Using different references to run a command

$ /usr/bin/echo Hello World

Hello World

You can determine whether your process is currently in a subshell by looking at the data stored in the SHLVL environment variable. A 1 indicates you are not in a subshell, because subshells have higher numbers. Thus, if SHLVL contains a number higher than 1, this indicates you’re in a subshell. The bash command automatically creates a subshell.

To preserve an environment variable’s setting, You can either use export when typing in the original variable definition, or use it after the variable is defined, by typing export

At variable-name the command-line prompt. you can simply reverse any modifi cations you make to the variable by using the unset command.

**Getting Help:**

A handy feature of the man utility is the ability to search for keywords in the documentation.

Listing 1.22: Using the man -k command to search for keywords

$ man -k passwd

[…]

passwd (1) - update user's authentication tokens

[…]

passwd (5) - password file

[…]

smbpasswd (5) - The Samba encrypted password file

Listing 1.24: Re executing commands in the command history

$ !920

man -k passwd

[…]

passwd (1) - update user's authentication tokens

[…]

passwd (5) - password file

To reexecute your most recent command, enter !! at the command line

Keep in mind that the history fi le will not have commands you have used during your

current login session. These commands are stored only in the history list.

If you desire to update the history fi le or the current history list,

-a appends the current history list commands to the end of the history file.

-r overwrites the current history list commands with the commands stored in the

history file.

Note: If you want to remove your command-line history, it is fairly easy to do.

First, clear your current history list by typing history -c at the command

line. After that, wipe the history file by issuing the history -w command,

which copies the now blank history list to the .bash\_history file, overwriting its contents.

Editing Text Files

Three popular Linux command-line text editors are

emacs

nano

vim

Change your standard editor to your desired editor by typing, for example, export EDITOR=nano at the command line.

emacs editor, you need to learn the various

shortcut keystrokes. Here are a few examples:

Press the Ctrl+X and then the Ctrl+S key combinations to save the editor buffers contents to the file.

Press the Ctrl+X and then the Ctrl+C key combinations to leave the editor.

Press the Ctrl+H key combination and then the T key to reach the e macs tutorial.

CentOS distribution has aliased the vi command to point to the vim command. Thus, for this distribution both the vi and vim commands will start the vim editor. Some distributions, such as Ubuntu, do not have the vim editor installed by default. Type type vi and press Enter, and if you get an error or an alias, then enter type vim . After you receive the program’s directory and filename, type the command readlink -f and follow it up with the directory and filename—for example, readlink -f /usr/bin/vi. If you see

/usr/bin/vi.tiny, you need to either switch to a different distribution to practice the

Vim commands or install the vim package.

Table 1.2 Commonly used vim command mode moving commands

Keystroke(s) Description

**h** Move cursor left one character.

**l** Move cursor right one character.

**j** Move cursor down one line (the next line in the text).

k Move cursor up one line (the previous line in the text).

Editing Text Files 25

Keystroke(s) Description

w Move cursor forward one word to front of next word.

e Move cursor to end of current word.

b Move cursor backward one word.

^ Move cursor to beginning of line.

$ Move cursor to end of line.

gg Move cursor to the file’s first line.

G Move cursor to the file’s last line.

n G Move cursor to file line number n .

Ctrl+B Scroll up almost one full screen.

Ctrl+F Scroll down almost one full screen.

Ctrl+U Scroll up half of a screen.

Ctrl+D Scroll down half of a screen.

Ctrl+Y Scroll up one line.

Ctrl+E Scroll down one line.

Table 1. 3 Commonly used vim command mode editing commands

Keystroke(s) Description

a Insert text after cursor.

A Insert text at end of text line.

dd Delete current line.

dw Delete current word.

i Insert text before cursor.

I Insert text before beginning of text line.

o Open a new text line below cursor, and move to insert mode.

O Open a new text line above cursor, and move to insert mode.

p Paste copied text after cursor.

P Paste copied (yanked) text before cursor.

yw Yank (copy) current word.

yy Yank (copy) current line.

In command mode, you can take the editing commands a step further by using their full

syntax, which is as follows:

COMMAND [ NUMBER-OF-TIMES ] ITEM

Table 1.4 Commonly used vim Ex mode commands

Keystrokes Description

:! command Execute shell command and display results, but don’t quit editor.

:r! command Execute shell command and include the results in editor buffer area.

:r file Read file contents and include them in editor buffer area.

Table 1.5 Saving changes in the vim text editor

Mode Keystrokes Description

Ex :x Write buffer to file and quit editor.

Ex :wq Write buffer to file and quit editor.

Ex :wq! Write buffer to file and quit editor (overrides protection).

Ex :w Write buffer to file and stay in editor.

Ex :w! Write buffer to file and stay in editor (overrides protection).

Ex :q Quit editor without writing buffer to file.

Ex :q! Quit editor without writing buffer to file (overrides

protection).

Command ZZ Write buffer to file and quit editor.

File-Combining Commands:

Putting together short text fi les for viewing on your screen and comparing them is useful. The file-combining commands covered here will do just that.

There is a handy new clone of the cat command called bat. Its developer calls it “

Cat with wings,” because of the bat utility’s many additional features. You can read about its features at github.com/sharkdp/bat.

Listing 1.28: Using the cat command to concatenate files

$ cat numbers.txt random.txt

TabLE 1. 6 The cat command’s commonly used options

Short Long Description

-A --show-all Equivalent to using the option -vET combination.

-E --show-ends Display a $ when a newline linefeed is encountered.

-n --number Number all text file lines and display that number in the output.

-s --squeeze-blank Do not display repeated blank empty text file lines.

-T --show-tabs Display a ^I when a tab character is encountered.

-v --show-nonprinting Display nonprinting characters when encountered using

either ^ and/or M- notation.

There are interesting variants of the cat command— bzcat , xzcat , and

zcat . These utilities are used to display the contents of compressed files.

If you want to display two files side-by-side and you do not care how sloppy the output

is, you can use the paste command.

$ paste random.txt numbers.txt

42 42

Flat Land 2A

Schrodinger's Cat 52

0010 1010 0010 1010

**File-Transforming Commands:**

Uncovering with od:

The od utility can help, because it allows you to display a fi le’s contents in octal (base 8), hexadecimal (base 16), decimal (base 10), and ASCII. Its basic syntax is as follows:

od [ OPTION ]... [ FILE ]...

Listing 1.31: Using the od command to display a file’s text in octal

$ cat fourtytwo.txt

42

fourty two

od fourtytwo.txt

0000000 031064 063012 072557 072162 020171 073564 005157 072561

Listing 1.32: Using the od -cb command to display additional information

$ od -cb fourtytwo.txt

0000000 4 2 \n f o u r t y t w o \n q u

064 062 012 146 157 165 162 164 171 040 164 167 157 012 161 165

There is a proposal on the table to add a -u option to the od command.

This option would allow the display of all Unicode characters, besides just

the ASCII character subset now available.

Separating with split:

One nice command to use is split . This utility allows you to divide a large fi le into smaller chunks, which is handy when you want to quickly create a smaller text fi le for testing purposes.

The basic syntax for the split command is as follows:

split [ OPTION ]... [ INPUT [ PREFIX ]]

You can divide up a fi le using size, bytes, lines, and so on.

Listing 1.33: Using the split -l command to split a file by line count

$ split -l 3 fourtytwo.txt split42

$

$ ls split42\*

split42aa split42ab

Notice that to split a file by its line count, you need to employ the -l (lowercase L)

option and provide the number of text file lines to attempt to put into each new fi le. In

the example, the original fi le has five text lines, so one new file ( split42aa ) gets the first three lines of the original fi le, and the second new file ( split42ab ) has the last two lines. Be aware that even though you specify the new files’ name ( PREFIX ), the split utility tacks additional characters, such as aa and ab , onto the names.

**File-Formatting Commands:**

Listing 1.34: Employing the sort command:

$ sort alphabet.txt

To obtain proper numeric order, add the -n option to the command If you’d like to save the output from the sort command to a file, all it takes is adding the -o switch. For example, sort -o newfile.txt alphabet .txt will sort the alphabet.txt file and store its sorted contents in the newfile.txt file.

Numbering with nl:

Another useful fi le-formatting command is the nl utility (number line utility). This little

command allows you to number lines in a text fi le in powerful ways. It even allows you to use regular expressions (covered later in this chapter) to designate which lines to number.

The nl command’s syntax is fairly simple:

nl [ OPTION ]... [ FILE ]...

If you do not use any options.

Listing 1.36: Using the nl command to add numbers to non-blank lines

$ nl ContainsBlankLines.txt

1 Alpha

2 Tango

3 Bravo

4 Echo

5 Foxtrot

Listing 1.37: Using the nl -ba command to number all text file lines

$ nl -ba ContainsBlankLines.txt

1 Alpha

2 Tango

3

4 Bravo

5 Echo

6

7

8 Foxtrot

**File-Viewing Commands:**

The less utility: You can search for a particular word within the fi le by pressing the ? key, typing in the word you want to fi nd, and pressing Enter to search backward.

Replace the ? key with the / key and you can search forward. Like the more pager, you do need to use the Q key to exit.

Listing 1.39: Using the head command to display fewer lines

$ head -n 2 /etc/passwd

root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:/sbin/nologin

Viewing Files with tail

If you want to display a file’s last lines instead of its first lines, employ the tail utility.

One of the most useful tail utility features is its ability to watch log fi les. Log fi les typically have new messages appended to the fi le’s bottom. Watching new messages as they are added is very handy. Use the -f (or --follow ) switch on the tail command and provide the log fi lename to watch as the command’s argument. You will see a few recent log fi le entries immediately. As you keep watching, additional messages will display as they are being added to the log fi le.

To watch messages being added to the journal file, use the journalctl –follow command.

Listing 1.41: Watching a log file with the tail command

$ sudo tail -f /var/log/auth.log

[sudo] password for Christine:

Aug 27 10:15:14 Ubuntu1804 sshd[15662]: Accepted password […]

Aug 27 10:15:14 Ubuntu1804 sshd[15662]: pam\_unix(sshd:sess[…]

Aug 27 10:15:14 Ubuntu1804 systemd-logind[588]: New sessio[…]

TabLE 1. 7 The wc command’s commonly used options

Short Long Description

-c --bytes Display the file’s byte count.

-L --max-line-length Display the byte count of the file’s longest line.

-l --lines Display the file’s line count.

-m --chars Display the file’s character count.

-w --words Display the file’s word count.

Listing 1.43: Using the wc command to check line length

$ wc -L /etc/nsswitch.conf

72 /etc/nsswitch.conf ((lines))

**Pulling Out Portions with cut:**

To sift through the data in a large text file, it helps to quickly extract small data sections.

The cut utility is a handy tool for doing this. It will allow you to view particular fields

within a file’s records. The command’s basic syntax is as follows:

cut OPTION... [FILE]...

TabLE 1. 8 The cut command’s commonly used options

Short Long Description

-c nlist --characters nlist Display only the record characters in the nlist (e.g.,

1–5).

-b blist --bytes blist Display only the record bytes in the blist (e.g., 1–2).

-d d --delimiter d Designate the record’s field delimiter as d . This overrides

the Tab default delimiter. Put d within quotation

marks to avoid unexpected results.

-f flist --fields flist Display only the record’s fields denoted by flist

(e.g., 1,3).

-s --only-delimited Display only records that contain the designated delimiter.

-z --zero-terminated Designate the record end-of-line character as the ASCII

character NUL.

Listing 1.44: Employing the cut command

$ head -2 /etc/passwd

root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:/sbin/nologin

$

$ cut -d ":" -f 1,7 /etc/passwd

root:/bin/bash

bin:/sbin/nologin

[…]

$

In Listing 1.44, the head command displays the password fi le’s fi rst two lines. This

text fi le employs colons ( : ) to delimit the fi elds within each record. The fi rst use of the cut command designates the colon delimiter using the -d option. Notice the colon is encased in quotation marks to avoid unexpected results. The -f option specifi es that only fi elds 1 (username) and 7 (shell) should be displayed.

Discovering Repeated Lines with uniq

A quick way to find repeated lines in a text file is with the uniq utility. Just type uniq and

follow it with the filename whose contents you want to check.

Listing 1.45: Using the uniq command

$ cat NonUniqueLines.txt

A

**C**

**C**

A

$

$ uniq NonUniqueLines.txt

**A**

C

**A**

the uniq utility recognizes

only repeated lines that are one after the other in a text file, only one of the C text

lines are removed from the display.

**Digesting an MD5 Algorithm:**

The md5sum utility is based on the MD5 message-digest algorithm.

Listing 1.46: Using md5sum to check the original file

$ md5sum fourtytwo.txt

0ddaa12f06a2b7dcd469ad779b7c2a33 fourtytwo.txt

The md5sum produces a 128-bit hash value.

A malicious attacker can create two files that have the same MD5 hash

value. However, at this point in time, a file that is not under the attacker’s

control cannot have its MD5 hash value modified. An even better solution is to use a stronger hash algorithm.

Securing Hash Algorithms

The Secure Hash Algorithms (SHA) is a family of various hash functions. Though typically used for cryptography purposes, they can also be used to verify a fi le’s integrity after it is copied or moved to another location.

Listing 1.47: Looking at the SHA utility names

$ ls -1 /usr/bin/sha???sum

/usr/bin/sha224sum

/usr/bin/sha256sum

/usr/bin/sha384sum

/usr/bin/sha512sum

The sha512sum utility uses the SHA-512 algorithm, which is the best to use for

security purposes and is typically employed to hash salted passwords in the /etc/shadow fi le on Linux.

**Using Regular Expressions**

Many commands use regular expressions. A regular expression is a pattern template you define for a utility such as grep, which then uses the pattern to filter text. Employing regular expressions along with text-filtering commands expands your mastery of the Linux command line.

Using grep:

will help with filtering text file

Table 1.9 The grep command’s commonly used options

Short Long Description

-c --count Display a count of text file records that contain a PATTERN match.

-d action --directories=action When a file is a directory, if action is set to read,

read the directory as if it were a regular text file; if action is set to skip, ignore the directory; and if action is set to recurse, act as if the - R, -r, or --recursive option was used.

-E --extended-regexp Designate the PATTERN as an extended regular expression.

-i --ignore-case Ignore the case in the PATTERN as well as in any text file records.

-R, -r --recursive Search a directory’s contents, and for any subdirectory within the original directory tree, consecutively search its contents as well (recursively).

-v --invert-match Display only text files records that do not contain a PATTERN match.

Listing 1.49: Using a simple grep command to search a file

$ grep root /etc/passwd

root:x:0:0:root:/root:/bin/bash

operator:x:11:0:operator:/root:/sbin/nologin

Listing 1.50: Using the grep command to search for patterns stored in a text file

$ fgrep -f accounts.txt /etc/passwd

sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin

Christine:x:1001:1001::/home/Christine:/bin/bash

nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin

$

$ grep -F -f accounts.txt /etc/passwd

sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin

Christine:x:1001:1001::/home/Christine:/bin/bash

nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin

$

The patterns are stored in the accounts.txt file, which is first displayed using the cat

command. Next, the fgrep command is employed, along with the -f option to indicate

the file that holds the patterns. The /etc/passwd file is searched for all the patterns stored within the accounts.txt file, and the results are displayed.

Also notice in Listing 1.49 that the third command is the grep -F command. The

grep -F command is equivalent to using the fgrep command, which is why the two commands produce identical results.

**Understanding Basic Regular Expressions**

Basic regular expressions ( .\*) to represent multiple characters and a single dot ( . ) to represent one character. use brackets to represent multiple characters, such as [a,e,i,o,u]. a range of characters, such as [A-z].

To fi nd text fi le records that begin with particular characters, you can precede them

with a caret (^) symbol. For fi nding text fi le records where particular characters are at

the record’s end, append them with a dollar sign ( $ ) symbol.

Listing 1.51: Using the grep command with a BRE pattern

$ grep daemon.\*nologin /etc/passwd

daemon:x:2:2:daemon:/sbin:/sbin/nologin

[…]

daemon:/dev/null:/sbin/nologin

[…]

$

$ grep root /etc/passwd

root:x:0:0:root:/root:/bin/bash

operator:x:11:0:operator:/root:/sbin/nologin

$

$ grep ^root /etc/passwd

root:x:0:0:root:/root:/bin/bash

$

In the fi rst snipped grep example within Listing 1.51, the grep command employs a pattern using the BRE .\* characters. In this case, the grep utility will search the password fi le for any instances of the word daemon within a record and display that record if it also contains the word nologin after the word daemon . The next two grep examples in Listing 1.51 are searching for instances of the word root within the password fi le. Notice that the one command displays two lines from the fi le. The

next command employs the BRE ^ character and places it before the word root . This regular expression pattern causes grep to display only lines in the password fi le that begin with root .

If you would like to get a better handle on regular expressions, there are several

good resources. Our favorite is Chapter 20 in the book Linux Command

Line and Shell Scripting Bible by Blum and Bresnahan (Wiley, 2015).

You can also look at the man pages, section 7, on regular expressions

(called regex(7) in the certification objectives). View this information by

typing man 7 regex or man -S 7 regex at the command line.

The -v option is useful, It produces a list of text fi le records that do not contain the pattern.

Listing 1.52: Using the grep command to audit the password file

$ grep -v nologin$ /etc/passwd

root:x:0:0:root:/root:/bin/bash

sync:x:5:0:sync:/sbin:/bin/sync

[…]

Christine:x:1001:1001::/home/Christine:/bin/bash

$

If you need to filter out all the blank lines in a file (display only lines with text), use grep with the -v option to invert the matching pattern. Then employ the ^ and $ anchor characters like grep -v ^$ filename at the command line.

A special group of bracket expressions are character classes . These bracket expressions

have predefi ned names and could be considered bracket expression shortcuts.

TabLE 1.10 Commonly used character classes

Class Description

[:alnum:] Matches any alphanumeric characters (any case), and is equal to using

the [0-9A-Za-z] bracket expression

[:alpha:] Matches any alphabetic characters (any case), and is equal to using the

[A-Za-z] bracket expression

[:blank:] Matches any blank characters, such as tab and space

[:digit:] Matches any numeric characters, and is equal to using the [0-9] bracket

expression

[:lower:] Matches any lowercase alphabetic characters, and is equal to using the

[a-z] bracket expression

[:punct:] Matches punctuation characters, such as !, #, $, and @

[:space:] Matches space characters, such as tab, form feed, and space

[:upper:] Matches any uppercase alphabetic characters, and is equal to using the

[A-Z] bracket expression

Listing 1.53: Using the grep command and a character class:

$ cat random.txt

42

Flat Land

Schrodinger's Cat

0010 1010

0000 0010

$ grep [[:digit:]] random.txt

42

0010 1010

0000 0010

If you need to search for a character in a file that has special meaning inan

expression or at the command line, such as the $ anchor character, precede

it with a backslash (\). This lets the grep utility know you are searching

for that character and not using it in an expression.

**Understanding Extended Regular Expressions**

(EREs) allow more complex patterns. a vertical bar symbol (|) allows you to specify two possible words or character sets to match.

Listing 1.54: Using the grep command with an ERE pattern

$ grep -E "^root|^dbus" /etc/passwd

root:x:0:0:root:/root:/bin/bash

dbus:x:81:81:System message bus:/:/sbin/nologin

$

$ egrep "(daemon|s).\*nologin" /etc/passwd

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

[…]

$

In the first example, the grep command uses the -E option to indicate the pattern is an

extended regular expression.

Using Streams, Redirection, and Pipes:

These structures allow you to build commands from other commands,

use a program’s output as input to another program, put together utilities to perform custom operations, and so on.

**Redirecting Input and Output**

you may need to combine multiple refinement steps to obtain the information you need.

Handling Standard Output:

By default, STDOUT directs output to your current terminal. Your process’s current terminal is represented by the /dev/tty file. A simple command to use when discussing standard output is the echo command.

Listing 1.56: Employing a STDOUT redirection operator

$ grep nologin$ /etc/passwd > NologinAccts.txt

$

$ less NologinAccts.txt

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

If you use the > redirection operator and send the output to a file that already exists, hat file’s current data will be deleted, To append data to a preexisting fi le, you need to use a slightly different redirection operator. The >> operator will append data to a preexisting fi le.

$ echo "Nov 16, 2019" > AccountAudit.txt

$ wc -l /etc/passwd >> AccountAudit.txt

$ cat AccountAudit.txt

Nov 16, 2019

44 /etc/passwd

Redirecting Standard Error

The fi le descriptor that identifies a command or script fi le error is 2(STDERR). is by default sent to your terminal (/dev/tty ).

The basic redirection operator to send STDERR to a fi le is the 2> operator.

Listing 1.58: Employing a STDERR redirection operator

$ grep -d skip hosts: /etc/\*

grep: /etc/anacrontab: Permission denied

grep: /etc/audisp: Permission denied

$ grep -d skip hosts: /etc/\* 2> err.txt

/etc/nsswitch.conf:#hosts: db files nisplus nis dns

/etc/nsswitch.conf:hosts: files dns myhostname

$

$ cat err.txt

grep: /etc/anacrontab: Permission denied

grep: /etc/audisp: Permission denied

$

The fi rst command in Listing 1.58 was issued to fi nd any fi les with the /etc/ directory

that contain the hosts: directive. Unfortunately, since the user does not have super user

privileges, several permissions denied error messages are generated. This clutters up the output

and makes it difficult to see what fi les contain this directive. To declutter the output, the second command in Listing 1.58 redirects STDERR to the err.txt fi le using the 2> redirection operator. This makes it much easier to see what fi les contain the hosts: directive. If needed, the error messages can be reviewed because they reside now in the err.txt fi le. Sometimes you want to send standard error and standard output to the same file. In these cases, use the &> redirection operator to accomplish your goal.

Listing 1.59: Using a STDERR redirection operator to remove error messages

$ grep -d skip hosts: /etc/\* 2> /dev/null

/etc/nsswitch.conf:#hosts: db files nisplus nis dns

/etc/nsswitch.conf:hosts: files dns myhostname

[…]

$

The /dev/null fi le is sometimes called the black hole.

Regulating Standard Input:

The fi le descriptor that identifi es an input into a command or script fi le is 0(STDIN) he basic redirection operator is the < symbol. The tr command is one of the few utilities that require you to redirect standard input.

Listing 1.60: Employing an STDIN redirection operator

$ cat Grades.txt

89 76 100 92 68 84 73

$

$ tr " " "," < Grades.txt

89,76,100,92,68,84,73

$

In Listing 1.60, the file Grades.txt contains various integers separated by a space. The

second command utilizes the tr utility to change each space into a comma (,).

Table 1.11 Commonly used redirection operators

Operator Description

> Redirect STDOUT to specified file. If file exists, overwrite it. If it does not

exist, create it.

>> Redirect STDOUT to specified file. If file exists, append to it. If it does not

exist, create it.

2> Redirect STDERR to specified file. If file exists, overwrite it. If it does not

exist, create it.

2>> Redirect STDERR to specified file. If file exists, append to it. If it does not

exist, create it.

&> Redirect STDOUT and STDERR to specified file. If file exists, overwrite it. If it

does not exist, create it.

&>> Redirect STDOUT and STDERR to specified file. If file exists, append to it. If it

does not exist, create it.

< Redirect STDIN from specified file into command.

<> Redirect STDIN from specified file into command and redirect STDOUT to

specified file.

Piping Data between Programs:

( |), which is called the vertical bar, vertical slash, or vertical line.

Be aware that some keyboards and text display the vertical bar not as a single vertical line. Instead, it looks like a vertical double dash.

The basic syntax for redirection with the pipe symbol is as follows:

COMMAND1 | COMMAND2 [| COMMANDN]…

The syntax for pipe redirection shows that the fi rst command, COMMAND1 , is executed. Its STDOUT is redirected as STDIN into the second command, COMMAND2.

Listing 1.62: Employing pipe redirection for several commands

$ grep /sbin/nologin$ /etc/passwd | cut -d ":" -f 1 | sort | less

abrt

adm

avahi

bin

chrony

[…]

In Listing 1.62, the output from the grep command is fed as input into the cut command.

The cut utility removes only the first field from each password record, which is the

account username. The output of the cut command is used as input into the sort command, which alphabetically sorts the usernames. Finally, the sort utility’s output is piped as input into the less command for leisurely perusing through the account usernames. In cases where you want to keep a copy of the command pipeline’s output as well as view it, the tee command will help. Similar to a tee pipe fitting in plumbing, where the water flow is sent in multiple directions, the tee command allows you to both save the output to a file and display it to STDOUT. Listing 1.63 contains an example of this handy command.

Listing 1.63: Employing the tee command

$ grep /bin/bash$ /etc/passwd | tee BashUsers.txt

root:x:0:0:root:/root:/bin/bash

user1:x:1000:1000:Student User One:/home/user1:/bin/bash

Christine:x:1001:1001::/home/Christine:/bin/bash

$ cat BashUsers.txt

root:x:0:0:root:/root:/bin/bash

user1:x:1000:1000:Student User One:/home/user1:/bin/bash

Christine:x:1001:1001::/home/Christine:/bin/bash

Using sed

There are times where you will want to edit text without having to pull out a full-fledged text editor. The sed utility edits a stream of text data based on a set of commands you supply ahead of time. The sed editor changes data based on commands either entered into the command line. or stored in a text file. The process the editor goes through is as follows:

1. Reads one text line at a time from the input stream

2. Matches that text with the supplied editor commands

3. Modifies the text as specified in the commands

4. Displays the modified text

Before looking at some sed examples, it is important to understand the command’s basic syntax. It is as follows:

sed [OPTIONS] [SCRIPT]… [FILENAME]

Listing 1.64: Using sed to modify STDIN text

$ echo "I like cake." | sed 's/cake/donuts/'

I like donuts.

The sed utility’s s command (substitute) specifies that if the first text string, cake,

is found, it is changed to donuts in the output. Note that the entire command after sed is

considered to be the SCRIPT, and it is encased in single quotation marks. Also notice that the text words are delimited from the s command, the quotation marks, and each other via the forward slashes (/).

Listing 1.65: Using sed to globally modify STDIN text

$ echo "I love cake and more cake." | sed 's/cake/donuts/'

I love donuts and more cake.

$

$ echo "I love cake and more cake." | sed 's/cake/donuts/g'

I love donuts and more donuts.

$

In the first command in Listing 1.65, only the first occurrence of the word cake was

modified. However, in the second command a g, which stands for global, was added to the sed script’s end. This caused all occurrences of cake to change to donuts.

It may be tempting to think that the sed utility is operating on the text file as a whole, but it is not. The stream editor applies its commands to each text file line individually. Thus, in our previous example, if the word cake was found multiple times within a single text file line, you’d need to use the g global command to change all instances.

Listing 1.67: Using sed to delete file text

$ sed '/Christine/d' cake.txt

Rich likes lemon cake.

Tim only likes yellow cake.

Samantha does not like cake.

You can also change an entire line of text. To accomplish this, you use the syntax of ' ADDRESS c NEWTEXT ' for the sed command’s SCRIPT . The ADDRESS refers to the file’s line number, and the NEWTEXT is the different text line you want displayed. An example of this method is shown in Listing 1.68.

Listing 1.68: Using sed to change an entire file line

$ sed '4cI am a new line' cake.txt

Christine likes chocolate cake.

Rich likes lemon cake.

Tim only likes yellow cake.

I am a new line

$

The stream editor has some rather useful command options.

Table 1.12 The sed command’s commonly used options

Short Long Description

-e script --expression=script Add commands in script to text processing. The

script is written as part of the sed command.

-f script --file=script Add commands in script to text processing. The

script is a file.

-r --regexp-extended Use extended regular expressions in script.

A handy option to use is the -e option. This allows you to employ multiple scripts in the

sed command. An example is shown in Listing 1.69.

Listing 1.69: Using sed -e to use multiple scripts

$ sed -e 's/cake/donuts/ ; s/like/love/' cake.txt

Christine loves chocolate donuts.

Rich loves lemon donuts.

Tim only loves yellow donuts.

Samantha does not love donuts.

Pay close attention to the syntax change in Listing 1.69. Not only is the -e option

employed, but the script is slightly different too. Now the script contains a semicolon (;)

between the two script commands. This allows both commands to be processed on the

text stream.

**Generating Command Lines**

By piping STDOUT from other commands into the xargs utility, you can build

command-line commands on the fly. Listing 1.70 shows an example of doing this.

Listing 1.70: Employing the xargs command

$ touch EmptyFile1.txt EmptyFile2.txt EmptyFile3.txt

$

$ ls EmptyFile?.txt

EmptyFile1.txt EmptyFile2.txt EmptyFile3.txt

$

$ ls -1 EmptyFile?.txt | xargs -p /usr/bin/rm

/usr/bin/rm EmptyFile1.txt EmptyFile2.txt EmptyFile3.txt ?...n

$

The xargs command uses the -p option. This option causes the xargs utility to stop and ask permission before enacting the constructed command-line command.

This is sometimes needed when employing xargs, depending on your distribution, The created command, in Listing 1.70, attempts to remove all three empty files with one

rm command.

Listing 1.71: Using the $() method to create commands

$ rm -i $(ls EmptyFile?.txt)

rm: remove regular empty file ‘EmptyFile1.txt’? y

rm: remove regular empty file ‘EmptyFile2.txt’? y

rm: remove regular empty file ‘EmptyFile3.txt’? y

In Listing 1.71, the ls command is again used to list any files that have the name

EmptyFilen.txt. Because the command is encased by the $() symbols, it does not display

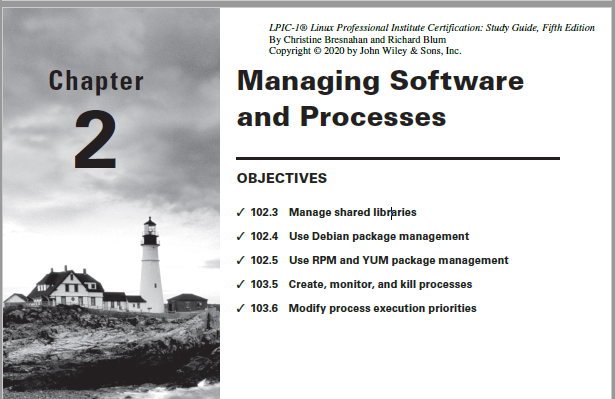
to STDOUT. Instead, the filenames are passed to the rm -i command, which inquires as

whether or not to delete each found file. This method allows you to get very creative when building commands on the fly.

**Summary**

Understanding fundamental shell concepts and being able to effectively and swiftly use the right commands at the shell command line is important for your daily job. It allows you to gather information, peruse text files, filter data, and so on. This chapter’s purpose was to improve your Linux command-line tool belt. Not only will this help you in your day-to-day work life, but it will also help you successfully pass the LPI certification exam.

End of chapter 1



Looking at Package Concepts

Most Linux users want to download an application and use it. Thus, Linux distributions

have created a system for bundling already compiled applications for distribution. This

bundle is called a *package*, and it consists of most of the files required to run a single application. You can then install, remove, and manage the entire application as a single package rather than as a group of disjointed files. Tracking software packages on a Linux system is called *package management*. Linux implements package management by using a database to track the installed packages on the system. The package management database keeps track of not only what packages re installed but also the exact files and file locations required for each application. Determining what pplications are installed on your system is as easy as querying the package management database. As you would expect, different Linux distributions have created different package management systems. However, over the years, two of these systems have risen to the top and become standards:

* Red Hat package management (RPM)
* Debian package management (Apt)

Each package management system uses a different method of tracking application packages

and files, but they both track similar information:

* Application files: The package database tracks each individual file as well as the folder where it’s located.

Library dependencies: The package database tracks what library files are required for

each application and can warn you if a dependent library file is not present when you

install a package.

* Application version: The package database tracks version numbers of applications so that you know when an updated version of the application is available. The sections that follow discuss the tools for using each of these package managementsystems.

Using RPM

RPM package fi les have an .rpm fi le extension and follow this naming format:

PACKAGE-NAME - VERSION - RELEASE . ARCHITECTURE .rpm

**PACKAGE-NAME** The PACKAGE-NAME is as you would expect—the name of the software package.

**VERSION** The VERSION is the program’s version number and represents software modifi cations that are more recent than older version numbers.

**RELEASE** The RELEASE is also called the *build number* .

**ARCHITECTURE** This is a designation of the CPU architecture for which the software

package was optimized. Typically you’ll see x86\_64 listed for 64-bit processors. Sometimes noarch is used, which indicates the package is architecturally neutral. Older CPU architecture designations include i386 (x86), ppc (PowerPC), and i586 and i686 (Pentium).

Note: There are two types of RPM packages: source and binary. Most of the time, you’ll want the binary package, because it contains the program bundle needed to successfully run the software. A source RPM contains the program’s source code, which can be useful for analysis (or for incorporating your own package customizations). You can tell the difference between these two package file types because a source RPM has src as its

ARCHITECTURE in the RPM filename.

**Listing 2.1**: Viewing RPM package files on a CentOS distribution

# **ls -1 \*.rpm**

docker-1.13.1-94.gitb2f74b2.el7.centos.x86\_64.rpm

emacs-24.3-22.el7.x86\_64.rpm

openssh-7.4p1-16.el7.x86\_64.rpm

zsh-5.0.2-31.el7.x86\_64.rpm

#

Note: If you want to obtain copies of RPM files on a Red Hat–based distro such as CentOS or Fedora, employ the yumdownloader utility. For example, use super user privileges and type **yumdownloader emacs** at the command line to download the emacs RPM file to your current working directory. On openSUSE, you’ll need to employ the zypper install -d package-name command, using super user privileges. This will download the RPM package file(s) to a /var/cache/zypp/packages/ subdirectory.

**The *rpm* Command Set**

rpm ACTION [ OPTION ] PACKAGE-FILE

tabLe 2.1 The rpm command actions

**Short Long Description**

-e --erase :Removes the specified package

-F --freshen :Upgrades a package only if an earlier version already exists

-i --install :Installs the specified package

-q --query :Queries whether the specified package is installed

-U --upgrade :Installs or upgrades the specified package

-V --verify :Verifies whether the package files are present and the package’s integrity

**Installing and Updating RPM Packages**

it’s more common to use the –U action, which installs the new package or upgrades the package if it’s already installed.

Listing 2.2: Installing/upgrading an RPM package file

# **rpm -Uvh zsh-5.0.2-31.el7.x86\_64.rpm**

Preparing... ################################# [100%]

Updating / installing...

1:zsh-5.0.2-31.el7 ################################# [100%]

#

**Querying RPM Packages**

Use the -q action to perform a simple query on the package management database for

installed packages.

Listing 2.3: Performing a simple query on an RPM package

# **rpm -q zsh**

zsh-5.0.2-31.el7.x86\_64

#

# **rpm -q docker**

package docker is not installed

#

tabLe 2. 2 The rpm command query action options

**Short option Long option Description**

-c --configfiles :Lists the names and absolute directory references of package configuration files

-i --info :Provides detailed information, including version, installation date, and signatures

N/A --provides :Shows what facilities the package provides

-R --requires :Displays various package requirements (dependencies)

-s --state :Provides states of the different files in a package, such as normal (installed), not installed, or replaced

N/A --what-provides :Shows to what package a file belongs

((The -**qi** options provide a great deal of information on the package))

Listing 2.4: Performing a detailed query on an RPM package

# **rpm -qi zsh**

Name : zsh

**Version : 5.0.2**

Release : 31.el7

Architecture: x86\_64

**Install Date: Tue 09 Apr 2019 02:51:26 PM EDT**

Group : System Environment/Shells

Size : 5854390

License : MIT

**Signature : RSA/SHA256, Mon 12 Nov 2018 09:49:55 AM EST, Key ID 24c6a[…]**

Source RPM : zsh-5.0.2-31.el7.src.rpm

Build Date : Tue 30 Oct 2018 12:48:17 PM EDT

Build Host : x86-01.bsys.centos.org

Relocations : (not relocatable)

Packager : CentOS BuildSystem <http://bugs.centos.org>

Vendor : CentOS

URL : http://zsh.sourceforge.net/

Summary : Powerful interactive shell

Description :

The zsh shell is a command interpreter usable as an interactive login

[…]

#

Tip: To display a list of all the installed packages on your system that use RPM package management, type **rpm -qa** at the command line. Interestingly, you get the same detailed information on a specific package if you enter **rpm -qa PACKAGE-NAME** as you would using the -qi options.

Listing 2.5: Determining an RPM package’s dependencies

# **rpm -qR zsh**

[…]

libc.so.6()(64bit)

libc.so.6(GLIBC\_2.11)(64bit)

[…]

libncursesw.so.5()(64bit)

librt.so.1()(64bit)

librt.so.1(GLIBC\_2.2.5)(64bit)

libtinfo.so.5()(64bit)

[…]

#

Listing 2.6: Determining configuration filenames that belong to an RPM package

# **rpm -qc zsh**

/etc/skel/.zshrc

/etc/zlogin

/etc/zlogout

/etc/zprofile

/etc/zshenv

/etc/zshrc

#

)At some point in time, you may want to determine information such as an RPM package’s signature or license from an uninstalled package file type **rpm -qRp zsh-5.0.2-31.el7.x86\_64.rpm** at the command line.(

Listing 2.7: Determining to what RPM package a file belongs

# **rpm -q --whatprovides /usr/bin/zsh**

zsh-5.0.2-31.el7.x86\_64

#

**Verifying RPM Packages**

Keeping a watchful eye on your system’s packages is an important security measure. For these operations, the rpm utility’s verify action is helpful. If you receive nothing or a single dot (.) from the rpm -V command, that’s a good thing.

Table 2.3 Verify action response codes for the rpm command

**Code Description**

? Unable to perform verification tests

5 Digest number has changed

c File is a configuration file for the package

D Device number (major or minor) has changed

G Group ownership has changed

L Link path has changed

missing Missing file

M Mode (permission or file type) has changed

P Capabilities have changed

S Size of file has changed

T Time stamp (modification) has changed

U User ownership has changed

Listing 2.8: Checking an RPM package’s integrity

# **rpm -V zsh**

.....UGT. /bin/zsh

.......T. c /etc/zlogin

missing c /etc/zprofile

#

In this example, response codes appear for the integrity check in Listing 2.8. Each fi le

that has a discrepancy is listed. Using the code interpretations from Table 2.3 , you can

determine that the /bin/zsh fi le has had both its owner and group changed, and the modification time stamp differs from the one in the package database. The /etc/zlogin fi le is a zsh package confi guration fi le, and its modifi cation time stamp has also been changed. Notice too that the /etc/zprofile confi guration fi le is missing.

**Removing RPM Packages**

Listing 2.9: Removing an RPM package

# **rpm -e zsh**

warning: file /etc/zprofile: remove failed: No such file or directory

#

# **rpm -q zsh**

package zsh is not installed

#

**Extracting Data from RPMs**

Listing 2.10: Creating a cpio archive from an RPM package

$ **rpm2cpio emacs-24.3-22.el7.x86\_64.rpm > emacs.cpio**

$

The next step is to move the files from the cpio archive into directories. This is accomplished

via the cpio command using the -id options. The -i switch employs copy-in mode,

which allows files to be copied in from an archive file. The -d switch creates subdirectories

in the current working directory whose names match the directory names in the archive,

with the exception of adding a preceding dot (.) to each name. A snipped example is shown

in Listing 2.11. Notice we added the verbose option (-v) to display what the command was

doing as it created the needed subdirectories and extracted the files.

Listing 2.11: Extracting the files from a cpio archive

$ **cpio -idv < emacs.cpio**

./usr/bin/emacs-24.3

./usr/share/applications/emacs.desktop

./usr/share/applications/emacsclient.desktop

./usr/share/icons/hicolor/128x128/apps/emacs.png

./usr/share/icons/hicolor/16x16/apps/emacs.png

./usr/share/icons/hicolor/24x24/apps/emacs.png

./usr/share/icons/hicolor/32x32/apps/emacs.png

./usr/share/icons/hicolor/48x48/apps/emacs.png

./usr/share/icons/hicolor/scalable/apps/emacs.svg

./usr/share/icons/hicolor/scalable/mimetypes/emacs-document.svg

28996 blocks

$

$ **ls ./usr/bin/emacs-24.3**

./usr/bin/emacs-24.3

After the files are finally extracted from the RPM package file and the subsequent cpio

archive, you can explore them as needed.

**Using YUM**

each Linux distribution has its own central clearinghouse of packages, called a *repository*. The repository contains software packages that have been tested and known to install and work correctly in the distribution environment. By placing all known packages into a single repository, the Linux distribution can create a one-stop shopping location for installing all applications.

Many third-party package repositories have also sprung up on the Internet that contain

specialized or custom software packages not distributed as part of the official Linux distribution repository.

The yum command uses the /etc/yum.repos.d/ directory to hold files that list the different

repositories it checks for packages. For a default CentOS system, that directory contains

several repository files, as shown in Listing 2.12.

Listing 2.12: Viewing the /etc/yum.repos.d/ repository files on a CentOS distro

$ **ls /etc/yum.repos.d/**

CentOS-Base.repo CentOS-CR.repo

CentOS-Debuginfo.repo CentOS-fasttrack.repo

CentOS-Media.repo CentOS-Sources.repo

CentOS-Vault.repo

$

Each file in the yum.repos.d folder contains information on a repository.

The basic yum command syntax is

yum [OPTIONS] [COMMAND] [PACKAGE…]

Table 2.4 The yum commands

**Command Description**

check-update :Checks the repository for updates to installed packages

clean :Removes temporary files downloaded during installs

deplist :Displays dependencies for the specified package

groupinstall :Installs the specified package group

info :Displays information about the specified package

install :Installs the specified package

list :Displays information about installed packages

localinstall :Installs a package from a specified RPM file

localupdate :Updates the system from specified RPM files

provides :Shows to what package a file belongs

reinstall :Reinstalls the specified package

remove :Removes a package from the system

resolvedep :Displays packages matching the specified dependency

search :Searches repository package names and descriptions for specified keyword

shell :Enters yum command-line mode

update :Updates the specified package(s) to the latest version in the repository

upgrade :Updates specified package(s) but removes obsolete packages

Listing 2.13: Installing software with yum on a CentOS distro

# **yum install emacs**

[…]

Resolving Dependencies

--> Running transaction check

[…]

--> Running transaction check

[…]

--> Finished Dependency Resolution

Dependencies Resolved

Package Arch Version Repository Size

Installing:

emacs x86\_64 1:24.3-22.el7 base 2.9 M

Installing for dependencies:

ImageMagick x86\_64 6.7.8.9-16.el7\_6 updates 2.1 M

[…]

Install 1 Package (+8 Dependent packages)

Is this ok [y/d/N]: **y**

Total 1.2 MB/s | 26 MB 00:22

Installed:

emacs.x86\_64 1:24.3-22.el7

Dependency Installed:

ImageMagick.x86\_64 0:6.7.8.9-16.el7\_6 […]

emacs-common.x86\_64 1:24.3-22.el7 […]

libXaw.x86\_64 0:1.0.13-4.el7 […]

libotf.x86\_64 0:0.9.13-4.el7 […]

Complete!

#

One nice feature of yum is the ability to group packages together for distribution.

Employ the yum grouplist command to see a list of the various package groups available, and use yum groupinstall group-package-name for an even easier way to get packages installed on your system.

dnf provides some advanced features that yum is missing. One such feature is speeding up resolving dependency searches with library files.

Listing 2.14: Reinstalling software with the yum utility

# **rpm -V emacs**

missing /usr/bin/emacs-24.3

#

# **yum reinstall emacs**

[…]

---> Package emacs.x86\_64 1:24.3-22.el7 will be reinstalled

[…]

Is this ok [y/d/N]: **y**

Installed:

Complete!

#

# **rpm -V emacs**

#(Notice in Listing 2.14 that the rpm -V emacs command discovers a missing file in the

package. Using the yum reinstall feature quickly fixes the issue.)

Listing 2.15: Removing software with the yum utility

# **yum remove emacs**

Note: Typically there is no need to modify the primary YUM configuration that is stored in the /etc/yum.conf file. This file contains settings (also called directives) that determine things such as where to record YUM log data. Although you can add third-party repositories by editing the primary configuration file or creating a /etc/yum.repos.d/ repository file manually, it is not recommended. The desired method is to install new repositories via RPM or YUM.

**Using ZYpp**

The openSUSE Linux distribution uses the RPM package management system and distributes software in .rpm fi les but doesn’t use the yum or dnf tool.

tabLe 2. 5 The zypper commands

**Command Description**

Help: Displays overall general help information or help on a specified command

install: Installs the specified package

info: Displays information about the specified package

list-updates: Displays all available package updates for installed packages from the

repository

lr: Displays repository information

packages: Lists all available packages or lists available packages from a specified

repository

what-provides: Shows to what package a file belongs

refresh: Refreshes a repository’s information

remove: Removes a package from the system

search: Searches for the specified package(s)

update: Updates the specified package(s) or if no package is specified, updates

all currently installed packages to the latest version(s) in the repository

verify: Verifies that installed packages have their needed dependencies satisfied

Listing 2.16: Installing software with the zypper utility

$ **sudo zypper install emacs**

Listing 2.17: Displaying package information with the zypper info command

$ **zypper info emacs**

The zypper utility is user-friendly and continually provides helpful messages to guide

your package management process.

Listing 2.18: Determining to which package a file belongs

$ **which emacs**

/usr/bin/emacs

$

$ **zypper what-provides /usr/bin/emacs**

**Command 'what-provides' is replaced by 'search --provides --match-exact'.**

See 'help search' for all available options.

Loading repository data...

Reading installed packages...

S | Name | Summary | Type

---+-------+------------------------+--------

i+ | emacs | GNU Emacs Base Package | package

$

You can easily obtain help on the zypper tool through its man pages and interactively

using the zypper help for general help or zypper help command for specific assistance.

In addition, the zypper utility allows you to shorten some of its commands. For example,

you can shorten install to in, remove to re, and search to se, as shown in Listing 2.19.

Listing 2.19: Searching for a package with the zypper search command

$ **zypper se nmap**

Loading repository data...

Reading installed packages...

S | Name | Summary | Type

--+---------+--------------------------------+--------

| nmap | Portscanner | package

| nmapsi4 | A Graphical Front-End for Nmap | package

| zenmap | A Graphical Front-End for Nmap | package

$

Removing packages with zypper is simple as well. An example of the command, process,

and utility’s helpful messages is shown snipped in Listing 2.20.

Listing 2.20: Removing a package with the zypper remove command

$ **sudo zypper remove emacs**

**Using Debian Packages**

As you can probably guess, the Debian package management system is mostly used on

Debian-based Linux distros, such as Ubuntu. With this system you can install, modify,

upgrade, and remove software packages.

**Debian Package File Conventions**

uses the following fi lename format:

PACKAGE-NAME -V ERSION - RELEASE A RCHITECTURE .deb

This fi lenaming convention for .deb packages is very similar to the .rpm fi le format. However, in the ARCHITECTURE , you typically fi nd amd64 , denoting it was optimized for the AMD64/Intel64 CPU architecture. Sometimes all is used, indicating the package is architecturally neutral. A few .deb package fi les are shown in Listing 2.21.

Listing 2.21: Software packages with the .deb filenaming conventions

$ **ls -1 \*.deb**

docker\_1.5-1build1\_amd64.deb

emacs\_47.0\_all.deb

openssh-client\_1%3a7.6p1-4ubuntu0.3\_amd64.deb

vim\_2%3a8.0.1453-1ubuntu1\_amd64.deb

zsh\_5.4.2-3ubuntu3.1\_amd64.deb

$

Note: If you want to obtain copies of Debian package files on a Debian-based distro,

such as Ubuntu, employ the apt-get download command.

**The *dpkg* Command Set**

The core tool to use for handling .deb fi les is the *dpkg* program, which is a command-line utility that has options for installing, updating, and removing .deb package fi les on your Linux system. The basic format for the dpkg command is as follows:

dpkg [ OPTIONS] ACTION PACKAGE-FILE

The ACTION parameter defi nes the action to be taken on the fi le. Table 2.6 lists the more common actions you’ll need to use.

tabLe 2. 6 The dpkg command actions

**Short Long Description**

-c --contents :Displays the contents of a package file

-C --audit :Searches for broken installed packages and suggests how to fix them

N/A –configure :Reconfigures an installed package

N/A --get-selections :Displays currently installed packages

-i --install :Installs the package; if package is already installed,upgrades it

-I --info :Displays information about an uninstalled package file

-l --list :Lists all installed packages matching a specified pattern

-L --listfiles :Lists the installed files associated with a package

-p --print-avail :Displays information about an installed package

-P --purge :Removes an installed package, including configuration files

-r --remove :Removes an installed package but leaves the configuration files

-s --status :Displays the status of the specified package

-S --search :Locates the package that owns the specified files

Note: The Debian distribution also provides a central clearinghouse for Debian

packages at www.debian.org/distrib/packages .

Listing 2.22: Looking at an uninstalled .deb package with the dpkg -I command

$ **dpkg -I zsh\_5.4.2-3ubuntu3.1\_amd64.deb 🡪 Inform**

If you want to see the package file’s contents, replace the -I option with the –contents switch.

Listing 2.23: Installing a .deb package with the dpkg -i command 🡪 install

$ sudo dpkg -i zsh\_5.4.2-3ubuntu3.1\_amd64.deb

dpkg: dependency problems prevent configuration of zsh:

zsh depends on zsh-common (= 5.4.2-3ubuntu3.1); however:

Package zsh-common is not installed. In order to check the problem we do 2.24

Listing 2.24: Displaying an installed package status with the dpkg -s command🡪status

$ dpkg -s zsh

Package: zsh

Status: install ok unpacked

Listing 2.25: Displaying all installed packages with the dpkg -l command 🡪 installed or LISTS

$ dpkg -l

ii adduser 3.116ubuntu1 all add and remove users

iU zsh 5.4.2-3ubunt amd64 shell with lots of features

The possible package status codes are shown in the fi rst few lines as output by the dpkg

command. For example, the last line that shows the zsh package displays the iU code. This means that while the package is installed ( i ), it is unpacked ( U ), but not configured, which is a problem. (well are you looking for no dependency problem? Go for snappy package installer)

If you need to **remove** a package, you have two options. The -r action removes the package but keeps any confi guration and data fi les associated with the package installed. If you really do want to remove the entire package, use the -P option, which purges the entire package, including confi guration fi les and data fi les from the system

Listing 2.27: Purging an installed package with the dpkg -P command 🡪 Purge (The

-p option lists the packages,)

$ sudo dpkg -P zsh

**Looking at the APT Suite (The Advanced Package Tool (APT))**

the apt-get program that does the work of installing, updating, and removing packages. The APT suite of tools relies on the /etc/apt/sources.list fi le to identify the locations of where to look for repositories.

**Using apt-cache can be used with these commands:**

■ depends: Displays the dependencies required for the package

■ pkgnames: Shows all the packages installed on the system

■ search: Displays the name of packages matching the specified item

■ showpkg: Lists information about the specified package

■ stats: Displays package statistics for the system

■ unmet: Shows any unmet dependencies for all installed packages or the specified

installed package

Typically you can issue the apt-cache commands without employing super user privileges. (shows every thing!!)

Displaying all installed packages with the apt-cache pkgnames command

$ apt-cache pkgnames | grep ^nano

Listing 2.29: Searching for a package with the apt-cache search command

$ apt-cache search zsh

zsh - shell with lots of features

Using apt-get

Table 2.7 The apt-get program action commands

Action Description

autoclean Removes information about packages that are no longer in the

repository

check Checks the package management database for inconsistencies

clean Cleans up the database and any temporary download files

dist-upgrade Upgrades all packages, but monitors for package dependencies

dselect-upgrade Completes any package changes left undone

install Installs or updates a package and updates the package management

database

remove Removes a package from the package management database

source Retrieves the source code package for the specified package

update Retrieves updated information about packages in the repository

upgrade Upgrades all installed packages to newest versions

**Reconfiguring Packages**

If the package required configuration when it was installed, you can employ the handy dpkg-reconfigure tool ($ sudo dpkg-reconfigure app)

**Library Principles**

A system library is a collection of items, such as program functions. Functions are self contained code modules that perform a specific task within an application, such as opening and reading a data fi le.The benefi t of splitting functions into separate library fi les is that multiple applications that use the same functions can share the same library fi les. These fi les full of functions make it easier to distribute applications, but also make it more complicated to keep track of what library fi les are installed with which applications. Linux supports two different fl avors of libraries. One is static libraries (also called statically linked libraries ) that are copied into an application when it is compiled. The other fl avor is shared libraries (also called dynamic libraries ) where the library functions are copied into memory and bound to the application when the program is launched. This is called loading a library.

**Locating Library Files**

looking in directories stored within the

1. LD\_LIBRARY\_PATH environment variable

2. Program’s PATH environment variable

3. /etc/ld.so.conf.d/ folder

4. /etc/ld.so.conf file

5. /lib\*/ and /usr/lib\*/ folders

It is important to know that the /lib\*/ folders, such as /lib/ and /lib64/, are for libraries needed by system utilities that reside in the /bin/ and /sbin/ directories.

**Loading Dynamically**

When a program is started, the dynamic linker (also called the dynamic linker/loader) is responsible for finding the program’s needed library functions.

Listing 2.36: Locating the dynamic linker executable on CentOS

$ locate ld-linux

/usr/lib64/ld-linux-x86-64.so.2

/usr/share/man/man8/ld-linux.8.gz

/usr/share/man/man8/ld-linux.so.8.gz

$

Listing 2.37: Loading and running the echo command with the dynamic linker utility (manually)

$ /usr/lib64/ld-linux-x86-64.so.2 /usr/bin/echo "Hello World"

Hello World

$

**Managing the Library Cache**

you’ll have to manually run the ldconfig command for any applications you are developing yourself.

**Developing New Libraries**

To accommodate testing of the newly created program library, you’ll need to modify

the LD\_LIBRARY\_PATH environment variable by including the program in its definition

as such:

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/home/devops/library/

After testing and refinement of the new function library is completed, move the library

file to its production folder (most likely somewhere in the /usr/lib\*/ directory tree).

And then create a library configuration file within the /etc/ld.so.conf.d/ directory that

points to the library file’s location.

When those items are completed, you’ll need to manually update the library cache.

Using super user privileges, issue the ldconfig command to load the new library into

the catalog.

**Listing 2.38**: Listing files in the library cache via the /ldconfig -v command

$ ldconfig -v 2> /dev/null | grep libmysqlclient

libmysqlclient.so.18 -> libmysqlclient.so.18.0.0

$

The ldd utility can come in handy if you need to track down missing library fi les for an

Application

Listing 2.39: Using the ldd command to view an application’s libraries

$ ldd /usr/bin/echo

linux-vdso.so.1 => (0x00007ffd3bd64000)

libc.so.6 => /lib64/libc.so.6 (0x00007f7c39eff000)

/lib64/ld-linux-x86-64.so.2 (0x00007f7c3a2cc000)

$

\*\* Sometimes a library is dependent on another library.\*\*

**PS: (Process Status)**

**Listing 2.40**: Viewing your processes with the ps command

$ ps

PID TTY TIME CMD

1615 pts/0 00:00:00 bash

1765 pts/0 00:00:00 ps

$

By default, the ps program shows only the processes that are running in the current

user shell. (TTY means terminal and pts means slaved terminal that is doing the job)

Linux supports three different styles of command-line options:

■■ Unix-style options, which are preceded by a dash

■■ Berkley Software Distribution (BSD)–style options, which are not preceded by a dash

■■ GNU long options, which are preceded by a double dash

So in order to ask for all processes in unix we add -rf (PPID: The process ID of the parent process (if the process was started by another process)

**Selecting Processes with ps**

Table 2.8 Some ps program selection options

Option(s) Description

a Display every process on the system associated with a tty terminal

-A, -e Display every process on the system

-C CommandList Only display processes running a command in the CommandList

-g GIDList, or -group GIDList Only display processes whose effective group is in GIDList

-G GIDList, or -Group GIDList Only display processes whose real group is in GIDList

-N Display every process except selected processes

p PIDList, -p PIDList or --pid PIDList Only display PIDList processes

-r Only display selected processes that are in a state of running

-t ttyList, or --tty ttyList List every process associated with the ttyList terminals

-T List every process associated with the current tty terminal

-u UserList, or --user UserList Only display processes whose effective user

(username or UID) is in UserList

-U UserList, or --User UserList Only display processes whose real user (username

or UID) is in UserList

x Remove restriction of “associated with a tty terminal”;

typically used with the a option

Listing 2.42: Viewing effective and real username processes with the ps command

$ ps -u Christine -U Christine

PID TTY TIME CMD

7802 ? 00:00:00 systemd

7803 ? 00:00:00 (sd-pam)

7876 ? 00:00:00 sshd

7877 pts/0 00:00:00 bash

7888 pts/0 00:00:00 ps

$

For a quick look at memory usage, employ the free command:

$ free -h

total used free shared buff/cache available

Mem: 3.9G 1.0G 2.2G 30M 710M 2.6G

Swap: 472M 0B 472M

$

A handy little utility for monitoring process information is the watch command. To use it, you enter watch and follow it by a command you’d like to enact over and over again. By default watch will reissue the command every two seconds. For example, you can type watch uptime to only monitor the system load.

**Multiplexing with tmux**

tabLe 2.11 The tmux utility prefix shortcut Ctrl+B commands (Press the Ctrl+B key combination and then the ? key to view a complete list of all the various key bindings and more.)

Key/Key Combination Description

& Kill the current window

% Split current screen window vertically into two panes

" Split current screen window horizontally into two panes

D Detach from current window

L Move to previous window

N Move to next window

O Move to next pane

Ctrl+O Rotate panes forward in current window

**Understanding Foreground and Background Processes**

Listing 2.47: Sending a command to the background via the & symbol

$ sleep 3000 & /\*this will be 3 secs\*/

[1] 1539

$

$ jobs

[1]+ Running sleep 3000 &

$

$ jobs -l

[1]+ 1539 Running sleep 3000 &

**Sending Multiple Jobs to the Background**

Listing 2.48: Showing multiple background jobs with the jobs command

$ bash CriticalBackups.sh &

[2] 1540

$

$ jobs -l

[1]- 1539 Running sleep 3000 &

[2]+ 1540 Running bash CriticalBackups.sh &

$

**Bringing Jobs to the Foreground**

$ jobs -l

[1]- 1539 Running sleep 3000 &

[2]+ 1540 Running bash CriticalBackups.sh &

$

$ fg %2

bash CriticalBackups.sh

**Sending a Running Program to the Background**

Listing 2.50: Sending a paused job to the background with the bg command

$ bash CriticalBackups.sh

^Z

[2]+ Stopped bash CriticalBackups.sh

$

$ bg %2

[2]+ bash CriticalBackups.sh &

$

$ jobs -l

[1]- 1539 Running sleep 3000 &

[2]+ 1540 Running bash CriticalBackups.sh &

$

**Stopping a Job**

Listing 2.51: Stopping a background job with the kill command

$ jobs -l

[1]- 1539 Running sleep 3000 &

[2]+ 1540 Running bash CriticalBackups.sh &

$

$ kill %1

[1]- Terminated sleep 3000

**Keeping a Job Running after Logout**

**Listing 2.52:** Keeping a background job running after log out with the nohup command

$ nohup bash CriticalBackups.sh &

[1] 2090

$ nohup: ignoring input and appending output to 'nohup.out' \*/ By default STDOUT and STDERR are redirected to the $HOME/nohup.out fi le\*/

**Managing Process Priorities**

**Listing 2.53**: Modifying an program’s niceness level with the nice command

$ nice -n 10 bash CriticalBackups.sh

**Listing 2.55**: Changing a running program’s niceness level with the renice command

$ renice 15 -p 1949

1949 (process ID) old priority 10, new priority 15

$

**Sending Signals to Processes**

tabLe 2.12 Linux process signals

Number Name Description

1 HUP Hangs up

2 INT Interrupts

3 QUIT Stops running

9 KILL Unconditionally terminates

11 SEGV Segments violation

15 TERM Terminates if possible

17 STOP Stops unconditionally, but doesn’t terminate

18 TSTP Stops or pauses, but continues to run in background

19 CONT Resumes execution after STOP or TSTP

**Listing 2.58**: Stopping a process with the kill command and a higher signal (Notice that the process was unaffected by the default TERM signal and the HUP signal. Thus, kill signal number 9 ( KILL ) had to be employed to stop the process. It’s usually a good idea to run the **lsof** command first to see a list of the open files and their processes.)

$ ps 2305

PID TTY STAT TIME COMMAND

2305 pts/0 T 0:00 vi

$

$ kill 2305

$

$ ps 2305

PID TTY STAT TIME COMMAND

2305 pts/0 T 0:00 vi

$

$ kill -s HUP 2305

$

$ ps 2305

PID TTY STAT TIME COMMAND

2305 pts/0 T 0:00 vi

$

$ kill -9 2305

[1]+ Killed vi

$

$ ps 2305

PID TTY STAT TIME COMMAND

$

**Sending Signals with the killall Command**

**Listing 2.59**: Stopping a group of processes with the killall command

$ killall stress-ng

**Sending Signals with the pkill Command**

**Listing 2.60**: Stopping a group of processes with the pkill command

$ pgrep -t tty3 /\*shows all stuff tty3 is doing. the -t option is used on the pgrep utility to see all the

PIDs attached to the tty3 terminal.\*/

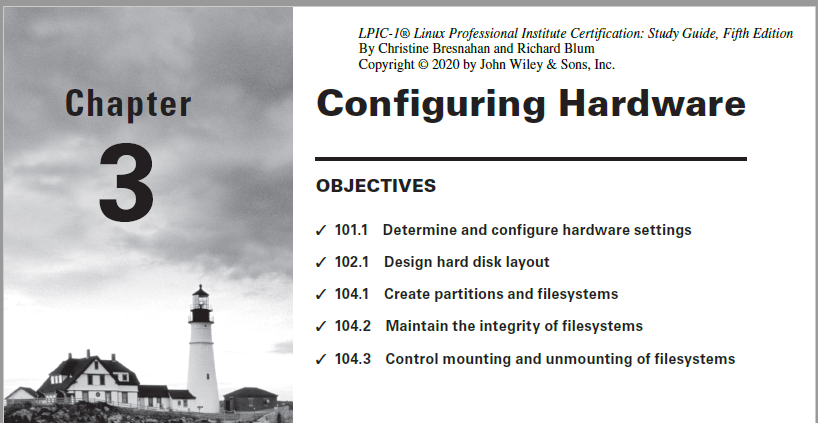
1716

1804

1828

$ sudo pkill -t tty3

$



**The /dev Directory**

After the Linux kernel communicates with a device on an interface, it must be able to transfer data to and from the device. This is done using device fi les .

**Listing 3.1**: Partial output from the /dev directory

$ ls -al sd\* tty\*

brw-rw---- 1 root disk 8, 0 Feb 16 17:49 sda

brw-rw---- 1 root disk 8, 1 Feb 16 17:49 sda1

crw-rw-rw- 1 root tty 5, 0 Feb 16 17:49 tty

crw--w---- 1 root tty 4, 0 Feb 16 17:49 tty0

crw--w---- 1 gdm tty 4, 1 Feb 16 17:49 tty1

Linux also provides a system called the device mapper . The device mapper function is performed by the Linux kernel. It maps physical block devices to virtual block devices. These virtual block devices allow the system to intercept the data written to or read from the physical device and perform some type of operation on them. Mapped devices are used by the Logical Volume Manager (LVM) for creating logical drives and by the Linux Unifi ed Key Setup (LUKS) for encrypting data on hard drives when those features

are installed on the Linux system

The device mapper creates virtual devices in the /dev/mapper directory.

These files are links to the physical block device files in the /dev directory.

**The /proc Directory**

The /proc directory is one of the most important tools you can use when roubleshooting

hardware issues on a Linux system.(its virtual directory that the kernel dynamically populates to provide access to information about the system hardware settings and status.)

when you have issue with the hardware it shows it like next question:

**Listing 3.2**: Listing system interrupts from the /proc directory

$ cat /proc/interrupts

0: 36 IO-APIC 2-edge timer

1: 297 IO-APIC 1-edge i8042

NMI: 0 Non-maskable interrupts

LOC: 93356 Local timer interrupts

SPU: 0 Spurious interrupts

PMI: 0 Performance monitoring interrupts

IWI: 0 IRQ work interrupts

RTR: 0 APIC ICR read retries

RES: 0 Rescheduling interrupts

TRM: 0 Thermal event interrupts

**Listing 3.3**: Displaying the I/O ports on a system

$ sudo cat /proc/ioports

As with I/O ports, each hardware device that uses DMA(Direct Memory Access) must be assigned a unique channel number. To view the DMA channels currently in use on the system, just display the /proc/dma file:

$ cat /proc/dma

4: cascade

$

This output indicates that only DMA channel 4 is in use on the Linux system.

**The /sys Directory**

The /sys directory is another virtual directory, Many different information files are available within the /sys directory. They are broken down into subdirectories based on the device and function in the system.

Listing 3.4: The contents of the /sys directory (al shows hidden files)

$ sudo ls -al /sys

**Finding Devices**

the Linux system. Fortunately, there are a few command-line tools to help out with that.

The **lsdev** command-line command displays information about the hardware devices installed on the Linux system. It retrieves information from the /proc/interrupts, proc/

ioports, and /proc/dma virtual files and combines them together in one output and the **lsblk -s** command displays all block devices.All these are lovely when u start using theyr ports:

**Listing 3.5**: Output from the lsdev command

$ sudo lsdev

Device DMA IRQ I/O Ports

...

acpi 9

ACPI 4000-4003 4004-4005 4008-400b 4020-4021

ahci d240-d247 d248-d24b d250-d257 d258-d25b

**Working with PCI Cards**

Table 3.1 The lspci command-line options

Option Description

-A Define the method to access the PCI information

-b Display connection information from the card point-of-view

-k Display the kernel driver modules for each installed PCI card

-m Display information in machine-readable format

-n Display vendor and device information as numbers instead of text

-q Query the centralized PCI database for information about the installed PCI cards

-t Display a tree diagram that shows the connections between cards and buses

-v Display additional information (verbose) about the cards

-x Display a hexadecimal output dump of the card information

**Working with USB Devices**

Table 3.2 The lsusb command options

Option Description

-d Display only devices from the specified vendor ID

-D Display information only from devices with the specified device file

-s Display information only from devices using the specified bus

-t Display information in a tree format, showing related devices

-v Display additional information about the devices (verbose mode)

-V Display the version of the lsusb program

**Listing Installed Modules**

A host of command-line commands can help you troubleshoot and fix kernel module issues. This section walks through the different module commands **available to help with any module issues you might run into.**

**Listing 3.9**: The lsmod command output

$ lsmod

Module Size Used by

vboxsf 39706 1

snd\_intel8x0 38153 2

snd\_ac97\_codec 130285 1 snd\_intel8x0

**Getting Module Information**

**Listing 3.10**: The modinfo command output

$ modinfo bluetooth

filename: /lib/modules/3.13.0-63-generic/kernel/net/bluetooth/bluetooth.ko

alias: net-pf-31

**Installing New Modules (.ko files)**

If you need to manually install a new module, there are two commands to help with that:

■■ insmod

■■ modprobe

The insmod command is the most basic, requiring you to specify the exact module file to

load. As you’ve seen, the kernel module files are stored in the /lib/modules directory structure,

with each kernel version having its own directory. If you look in that directory on

your Linux system, you’ll see a directory tree structure for the different types of hardware.

The downside to using the insmod program is that you may run into modules that

depend on other modules, and the insmod program will fail if those other modules aren’t already installed. To make the process easier, the modprobe command helps resolve module dependencies for you.

Another nice feature of the modprobe command is that it understands module names,

and it will search the module library for the module file that provides the driver for the

module name.

Because of this versatility, there are many options available for the modprobe command.

Table 3.3 shows the command-line options that you can use.

Option Description

-a Insert all modules listed on the command line

-b Apply any blacklist commands specified in the configuration file

-C Specify a different configuration file other than the default

-c Display the current configuration used.

-d Specify the root directory to use for installing modules. The default is /.

-f Force the module installation even if there are version issues.

-i Ignore the install and remove commands specified in the configuration file for

the module.

-n Perform a dry run of the module install to see if it will work, without actually

installing it.

-q Quiet mode—doesn’t display any error messages if the module installation or

removal fails.

-r Remove the module listed.

-s Send any error messages to the syslog facility on the system.

-V Display the program version and exit.

-v Provide additional information (verbose) as the module is processed.

Example: (v is for verbose)

$ sudo modprobe -iv btusb

insmod /lib/modules/3.13.0-63-generic/kernel/drivers/bluetooth/btusb.ko

$

**Removing Modules**

Normally it does no harm to install a module in the system if the hardware device is not

present. The kernel just ignores unused modules. However, some Linux administrators prefer to keep the kernel as lightweight as possible, so the Linux developers created a method for removing unnecessary modules: the rmmod command. The rmmod command removes a module by specifying the module name. However, our friend the modprobe command can also remove modules for us, so you don’t need to memorize another command. Instead, just use the –r option with the modprobe command:

$ sudo modprobe -rv btusb

rmmod btusb

$

The modprobe –r command invokes the rmmod command automatically, removing the

module by name. You can verify that the module has been removed by using the lsmod

command.

**Automatic Drive Detection**

The udev program runs in the background at all times and automatically detects new hardware connected to the running Linux system. As you connect new drives, USB devices, or optical drives (such as CD and DVD devices), udev will detect them and assign each one a unique device filename in the /dev directory.

**Logical Volume Manager**

The Linux Logical Volume Manager (LMV) also utilizes the /dev/mapper dynamic device directory to allow you to create virtual drive devices. You can aggregate multiple physical drive partitions into virtual volumes, which you then treat as a single partition on your system. The benefit of LVM is that you can add and remove physical partitions as needed to a logical volume, expanding and shrinking the logical volume as needed. For each physical partition, you must mark the partition type as the Linux LVM filesystem type in fdisk or gdisk. Then, you must use several LVM tools to create and manage the logical volumes:

■■ pvcreate: Creates a physical volume

■■ vgcreate: Groups physical volumes into a volume group

■■ lvcreate: Creates a logical volume from partitions in each physical volume

**Partitioning Tools**

Working with **fdisk** (works with **mbr**)

Table 3.4 Common fdisk commands

Command Description

a Toggle a bootable flag

b Edit BSD disk label

c Toggle the DOS compatibility flag

d Delete a partition

g Create a new empty GPT partition table

G Create an IRIX (SGI) partition table

l List known partition types

m Print this menu

n Add a new partition

o Create a new empty DOS partition table

p Print the partition table

q Quit without saving changes

s Create a new empty Sun disk label

t Change a partition’s system ID

u Change display/entry units

v Verify the partition table

w Write table to disk and exit

x Extra functionality (experts only)

**Working with gdisk (works with GPT)**

Command Description

b Back up GPT data to a file

c Change a partition’s name

d Delete a partition

i Show detailed information on a partition

l List known partition types

n Add a new partition

o Create a new empty GUID partition table (GPT)

p Print the partition table

q Quit without saving changes

r Recovery and transformation options (experts only)

s Sort partitions

t Change a partition’s type code

v Verify disk

w Write table to disk and exit

x Extra functionality (experts only)

? Print this menu

**As Always I prefer gparted cause its graphical.**

**filesystem hierarchy standard (FHS).**

table 3. 6 Common Linux FHS directories

Directory Description

/boot Contains boot loader files used to boot the system

/etc Contains system and application configuration files

/home Contains user data files

/media Used as a mount point for removable devices

/mnt Also used as a mount point for removable devices

/opt Contains data for optional third-party programs

/tmp Contains temporary files created by system users

/usr Contains data for standard Linux programs

/usr/bin Contains local user programs and data

/usr/local Contains data for programs unique to the local installation

/usr/sbin Contains data for system programs and data

/var Contains variable data files, including system and application logs

**Linux Filesystems**

When you create a filesystem specifically for use on a Linux system, you can choose fromfour main filesystems:

■■ btrfs: A newer, high-performance filesystem that supports files up to 16 exbibytes

(EiB) in size, and a total filesystem size of 16 EiB. It also can perform its own form of

Redundant Array of Inexpensive Disks (RAID) as well as logical volume management

(LVM) and subvolumes. It includes additional advanced features such as built-in snapshots for backup, improved fault tolerance, and data compression on the fly.

■■ ecryptfs: The Enterprise Cryptographic Filesystem (eCryptfs) applies a Portable Operating System Interface (POSIX)–compliant encryption protocol to data before storing it on the device. This provides a layer of protection for data stored on the device. Only the operating system that created the filesystem can read data from it.

■■ ext3: Also called ext3fs, this is a descendant of the original Linux ext filesystem. It

supports files up to 2 tebibytes (TiB), with a total filesystem size of 16 TiB. It supports

journaling, as well as faster startup and recovery.

■■ ext4: Also called ext4fs, it’s the current version of the original Linux filesystem. It supports files up to 16 TiB, with a total filesystem size of 1 EiB. It also supports journaling and utilizes improved performance features.

■■ reiserFS: Created before the Linux ext3fs filesystem and commonly used on older

Linux systems, it provides features now found in ext3fs and ext4fs. Linux has dropped

support for the most recent version, reiser4fs.

■■ swap: The swap filesystem allows you to create virtual memory for your system using space on a physical drive. The system can then swap data out of normal memory into the swap space, providing a method of adding additional memory to your system. This is not intended for storing persistent data.

The default filesystem used by most Linux distributions these days is ext4fs. The ext4fs

filesystem provides journaling, which is a method of tracking data not yet written to the

drive in a log file, called the journal. If the system fails before the data can be written to the drive, the journal data can be recovered and stored upon the next system boot.

Many Linux administrators have taken a liking to the newer btrfs filesystem. The btrfs

filesystem provides many advanced features, such as the ability to create a filesystem across multiple devices, automatic data compression, and the ability to create subvolumes.

**Creating Filesystems**

$ sudo mkfs -t ext4 /dev/sdb1

mke2fs 1.44.1 (24-Mar-2018)

Creating filesystem with 2621440 4k blocks and 655360 inodes

**Manually Mounting Devices**

mount -f fstype device mountpoint

Use the -f command-line option to specify the fi lesystem type of the device:

$ sudo mount -t ext4 /dev/sdb1 /media/usb1

$

If you specify the mount command with no parameters, it displays all devices currently

mounted on the Linux system. Be prepared for a long output, though, as most Linux distributions mount lots of virtual devices in the virtual directory to provide information about system resources. Listing 3.11 shows a partial output from a mount command.

**Listing 3.11**: Output from the mount command

$ mount

**Automatically Mounting Devices**

For permanent storage devices, Linux maintains the /etc/fstab fi le to indicate which drive

devices should be mounted to the virtual directory at boot time. The /etc/fstab fi le is a

table that indicates the drive device fi le (either the raw fi le or one of its permanent udev fi lenames),

the mount point location, the fi lesystem type, and any additional options required

to mount the drive. Listing 3.12 shows the /etc/fstab fi le from an Ubuntu workstation.

Listing 3.12: The /etc/fstab file

$ cat /etc/fstab

# /etc/fstab: static file system information.

This /etc/fstab fi le references the devices by their Universally Unique Identifi er

(UUID) value, ensuring the correct drive partition is accessed no matter what order it

appears in the raw device table. The fi rst partition is mounted at the /boot/efi mount

point in the virtual directory. The second partition is mounted at the root ( / ) of the virtual directory, and the third partition is mounted as a swap area for virtual memory.

You can manually add devices to the /etc/fstab fi le so that they are mounted

automatically when the Linux system boots. However, if they don’t exist at boot time,

that will generate a boot error.

**Managing Filesystems**

**Retrieving Filesystem Stats**

As you use your Linux system, there’s no doubt that at some point you’ll need to monitor

disk performance and usage. A few different tools are available to help you do that:

■ df : Displays disk usage by partition

■ du : Displays disk usage by directory; good for finding users or applications that are

taking up the most disk space

■ iostat : Displays a real-time chart of disk statistics by partition

■ lsblk : Displays current partition sizes and mount points

**Filesystem Tools**

Linux uses the e2fsprogs package of tools to provide utilities for working with ext fi lesystems

(such as ext3 and ext4). The most popular tools in the e2fsprogs package are

■ blkid : Display information about block devices, such as storage drives

■ chattr : Change file attributes on the filesystem

■ debugfs : Manually view and modify the filesystem structure, such as undeleting a file

or extracting a corrupted file

■ dumpe2fs : Display block and superblock group information

■ e2label : Change the label on the filesystem

■ resize2fs : Expand or shrink a filesystem

■ tune2fs : Modify filesystem parameters

These tools help you fine-tune parameters on an ext filesystem, but if corruption occurs

on the filesystem, you’ll need the fsck program.

The XFS filesystem also has a set of tools available for tuning the filesystem. You’ll most

likely use the following:

■■ xfs\_admin: Display or change filesystem parameters such as the label or UUID

assigned

■■ xfs\_db: Examine and debug an XFS filesystem

■■ xfs\_fsr: Improve the organization of mounted filesystems

■■ xfs\_info: Display information about a mounted filesystem, including the block sizes

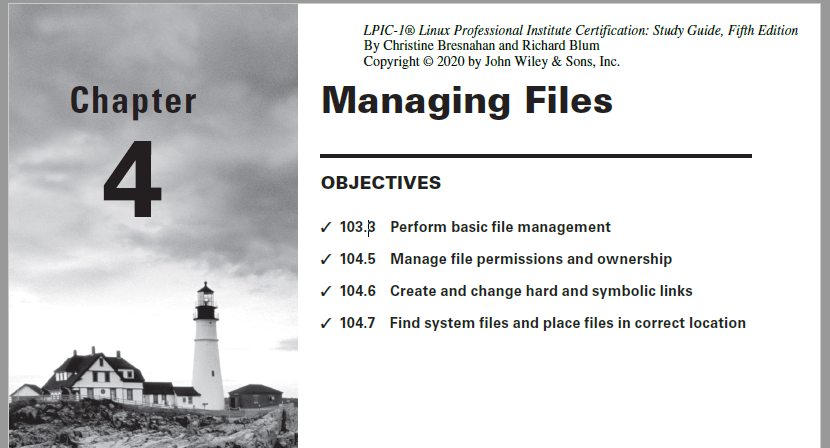
and sector sizes, as well as label and UUID information

■■ xfs\_repair: Repair corrupted or damaged XFS filesystems

Although these ext and XFS tools are useful, they can’t help fix things if the filesystem

itself has errors. For that, the fsck program is the tool to use:

$ sudo fsck -f /dev/sdb1



Listing 4.1: Using the ls and pwd commands

$ ls

Desktop Downloads Pictures Public Videos

Documents Music Project47.txt Templates

$

$ pwd

/home/Christine

$

tabLe 4.1 The ls command’s commonly used options

Short Long Description

-1 N/A List one file or subdirectory name per line

-a --all Display all file and subdirectory names, including hidden files’

names

-d --directory Show a directory’s own metadata instead of its contents

-F --classify Classify each file’s type using an indicator code ( \* , / , = , > , @ , or | )

-i --inode Display all file and subdirectory names along with their associated index number

-l N/A Display file and subdirectory metadata, which includes file type, file access permissions, hard link count, file owner, file’s group, modification date and time, and filename

-R N/A Show a directory’s contents, and for any subdirectory within the original directory tree, consecutively show its contents as well (recursively)

**Listing 4.4**: Using the touch command

$ touch Project43.txt

$

$ ls

Everything Life Project42.txt Project43.txt Universe

**Listing 4.5:** Using the file command (On Linux, everything is a file, ok so what kind of file is this?)

$ file Project42.txt

Project42.txt: ASCII text

$

$ file Everything

Everything: directory

$

**WILD CARDS**

**Listing 4.6**: Using an asterisk wildcard with the ls command

$ ls

cake.txt carmelCake.sh carmelPie.txt carrotCake.txt

$

$ ls c\*.txt

cake.txt carmelPie.txt carrotCake.txt

$

**Listing 4.7:** Using a question mark wildcard with the ls command

$ ls

bard bat beat bed bet bird bit bot bunt

$

$ ls b?t

bat bet bit bot

$

$ ls b??d

bard bird

$

**Listing 4.8**: Using a bracketed wildcard with the ls command

$ ls

bard bat beat bed bet bEt bird bit bot bunt

$

$ ls b[eio]t

bet bit bot

$

**Listing 4.9**: Using multiple bracketed wildcards with the ls command

$ ls

bard bat beat bed bet bEt bird bit bot bunt

$

$ ls b[eu][an]t

beat bunt

$

**Listing 4.10**: Using a bracketed range wildcard with the ls command

$ ls b[a-z]t

bat bet bEt bit bot

$

**Listing 4.11**: Using a negated bracketed wildcard with the ls command

$ ls

bard bat beat bed bet bEt bird bit bot bunt

$

$ ls b[^eio]t

bat bEt

$

**Listing 4.12**: Exploring the mkdir command

$ ls -F

Everything/ Project42.txt Project44.txt Project46.txt

Life/ Project43.txt Project45.txt Universe/

$

$ mkdir Galaxy

$

$ ls -F

Everything/ Life/ Project43.txt Project45.txt Universe/Galaxy/

Project42.txt Project44.txt Project46.txt

**Listing 4.13:** Avoiding problems with the mkdir command

$ mkdir -p Projects/42/

$

$ ls -F

Everything/ Life/ Project43.txt Project45.txt Projects/

Galaxy/ Project42.txt Project44.txt Project46.txt Universe/

$

$ ls -F Projects

42/

$

**Listing 4.14**: Using the cp command

$ cp melodrama.txt space-warfare.txt

$

$ ls

melodrama.txt space-warfare.txt

$

Table 4.2 The cp command’s commonly used options

Short Long Description

-a --archive Perform a recursive copy and keep all the files’ original

attributes, such as permissions, ownership, and timestamps.

-f --force Overwrite any preexisting destination files with same name

as DEST.

-i --interactive Ask before overwriting any preexisting destination files with

same name as DEST.

-n --no-clobber Do not overwrite any preexisting destination files with same

name as DEST

-R , -r --recursive Copy a directory’s contents, and also copy the contents

of any subdirectory within the original directory tree

(recursive).

-u --update Only overwrite preexisting destination files with the same

name as DEST , if the source file is newer.

-v --verbose Provide detailed command action information as command

executes.

**Listing 4.15**: Performing a recursive copy with the cp command

$ pwd

/home/Christine/SpaceOpera

$

$ ls -F

Emphasis/

$

$ cp Emphasis Story-Line

cp: omitting directory 'Emphasis'

$

$ ls -F

Emphasis/

$

$ cp -R Emphasis Story-Line

$

$ ls -F

Emphasis/ Story-Line/

$

$ ls -R Emphasis

Emphasis:

chivalric-romance.txt melodrama.txt

interplanatary-battles.txt space-warfare.txt

$

$ ls -R Story-Line/

Story-Line/:

chivalric-romance.txt melodrama.txt

interplanatary-battles.txt space-warfare.txt

$

**Moving/Renaming Files and Directories**

Ta b le 4.3 The mv command’s commonly used options

Short Long Description

-f --force Overwrite any preexisting destination files with the same

name as DEST.

-i --interactive Ask before overwriting any preexisting destination files with

the same name as DEST.

-n --no-clobber Do not overwrite any preexisting destination files with the same name as DEST.

-u --update Only overwrite preexisting destination files with the same

name as DEST if the source file is newer.

-v --verbose Provide detailed command action information as the

command executes.

**Listing 4.17**: Renaming a directory using the mv command

$ pwd

/home/Christine/SpaceOpera

$

$ ls -F

Emphasis/ Story-Line/

$

$ mv -i Story-Line Story-Topics

$

$ ls -F

Emphasis/ Story-Topics/

$

Ta b le 4.4 The rm command’s commonly used options

Short Long Description

-d --dir Delete any empty directories.

-f --force Continue on with the deletion process, even if some files

designated by the command for removal do not exist, and do

not ask prior to deleting any existing files.

-i --interactive Ask before deleting any existing files.

-I N/A Ask before deleting more than three files or when using the

-r option.

-R, -r --recursive Delete a directory’s contents, and also delete the contents

of any subdirectory within the original directory tree

(recursive).

-v --verbose Provide detailed command action information as command

Executes

**Listing 4.20**: Deleting a directory containing files using the rm command

$ cd SpaceOpera/

$

$ ls -F

Emphasis/ Story-Topics/

$

$ rm -i Emphasis/

rm: cannot remove 'Emphasis/': Is a directory

$

$ rm -ir Emphasis

rm: descend into directory 'Emphasis'? y

rm: remove regular empty file 'Emphasis/melodrama.txt'? y

rm: remove regular empty file 'Emphasis/interplanatary-battles.txt'? y

rm: remove regular empty file 'Emphasis/chivalric-romance.txt'? y

rm: remove directory 'Emphasis'? y

$

$ ls -F

Story-Topics/

$

**Listing 4.22**: Using the rmdir command

$ mkdir -v EmptyDir

mkdir: created directory 'EmptyDir'

$

$ rmdir -v EmptyDir/

rmdir: removing directory, 'EmptyDir/'

$

The -p (or --parents) switch is required along with providing the entire directory tree name as an argument

**Listing 4.23**: Using the rmdir command to delete an empty directory tree

$ mkdir -vp EmptyDir/EmptySubDir

mkdir: created directory 'EmptyDir'

mkdir: created directory 'EmptyDir/EmptySubDir'

$

$ rmdir -vp EmptyDir/EmptySubDir

rmdir: removing directory, 'EmptyDir/EmptySubDir'

rmdir: removing directory, 'EmptyDir'

$

**Compress party with amazing xz compressor**

Listing 4.26: Using the xcat command to view an xz compressed file’s contents

$ xz alphabet.txt

$

$ ls alphabet\*

alphabet.txt.xz

$

$ xzcat alphabet.txt.xz

Alpha

Tango

Bravo

Echo

Foxtrot

$

$ ls alphabet\*

alphabet.txt.xz

$

tabLe 4.5 The compressed file display commands

Command Equivalent Description

bzcat bzip2 -dc Used to display bzip2 compressed files.

xzcat xz --decompress --stdout Displays the contents of xz compressed files.

zcat gunzip -c Used to display gzip compressed files. Some Unix-like systems have a gzcat command instead.

**Archiving File Commands**

There are several programs you can employ for managing backups

**Copying with cpio “copy in and out.”**

Ta b le 4.6 The cpio command’s commonly used options

Short Long Description

-I N/A Designates an archive file to use.

-i --extract Copies files from an archive or displays the files within the archive, depending upon the other options employed. Called copy-in mode.

--no-absolute-filenames Designates that only relative path names are to be used. (The default is to use absolute path names.)

-o --create Creates an archive by copying files into it. Called copy-out mode.

-t --list Displays a list of files within the archive. This list is called a table of contents.

-v --verbose Displays each file’s name as each file is processed

**Listing 4.27**: Employing cpio to create an archive

$ ls Project4?.txt

Project42.txt Project43.txt Project44.txt

Project45.txt Project46.txt

$

$ ls Project4?.txt | cpio -ov > Project4x.cpio

Project42.txt

Project43.txt

Project44.txt

Project45.txt

Project46.txt

59 blocks

$

**Listing 4.28**: Using cpio to list an archive’s contents

$ cpio -itvI Project4x.cpio

-rw-r--r-- 1 Christin Christin 29900 Aug 19 17:37 Project42.txt

-rw-rw-r-- 1 Christin Christin 0 Aug 19 18:07 Project43.txt

**Listing 4.29**: Using cpio to restore files to a different directory location

$ ls -dF Projects

Projects/

$

$ mv Project4x.cpio Projects/

$

**Archiving with tar**

Ta b le 4.7 The tar command’s commonly used tarball creation options

Short Long Description

-c --create Creates a tar archive file. The backup can be a full or incremental backup, depending on the other selected options.

-u --update Appends files to an existing tar archive file, but copies only those files that were modified since the original archive file was created.

-g --listed-incremental Creates an incremental or full archive based on metadata stored in the provided file.

-z --gzip Compresses tar archive file into a tarball using gzip .

-j --bzip2 Compresses tar archive file into a tarball using bzip2 .

-J --xz Compresses tar archive file into a tarball using xz .

-v --verbose Displays each file’s name as each file is processed.

**Listing 4.30**: Using tar to create an archive file

$ ls Project4?.txt

Project42.txt Project43.txt Project44.txt

Project45.txt Project46.txt

$

$ tar -cvf Project4x.tar Project4?.txtIf you are backing up lots of fi les or large amounts of data, it is a good idea to employ a

compression utility. This is easily accomplished by adding an additional switch to your tar command options. An example is shown in Listing 4.31, which uses gzip compression to create a tarball.

Listing 4.31: Using tar to create a tarball

$ tar -zcvf Project4x.tar.gz Project4?.txt

Project42.txt

Project43.txt

Project44.txt

Project45.txt

Project46.txt

$

$ ls Project4x.tar.gz

Project4x.tar.gz

$

Notice in Listing 4.31 that the tarball fi lename has the .tar.gz fi le extension. It is considered good form to use the .tar extension and tack on an indicator showing the compression method that was used. However, you can shorten it to .tgz if desired. There are many compression methods. However, when you use a compression utility along with an archive and restore program for data backups, it is vital that you use a lossless compression method. A lossless compression is just as it sounds; no data is lost. The gzip , bzip2 , xz , and zip utilities provide lossless compression. Obviously it is important not to lose data when doing backups.

There is a useful variation of this command to create both full and incremental backups.

A simple example helps to explain this concept. The process for creating a full backup is

shown in Listing 4.32.

**Listing 4.32**: Using tar to create a full backup

$ tar -g FullArchive.snar -Jcvf Project42.txz Project4?.txt

Project42.txt

Project43.txt

Project44.txt

Project45.txt

Project46.txt Ta b le 4.8 The tar command’s commonly used archive verification options

Short Long Description

-d --compare

--diff

Compares a tar archive file’s members with external files and lists the differences.

-t --list Displays a tar archive file’s contents.

-W --verify Verifies each file as the file is processed. This option cannot be used with the compression options.

**Listing 4.34**: Using tar to list a tarball’s contents

$ tar -tf Project4x.tar.gz

Project42.txt

Project43.txt

Project44.txt

Project45.txt

Project46.txt

$

Ta b le 4.9 The tar command’s commonly used file restore options

Short Long Description

-x --extract

--get

Extracts files from a tarball or archive file and places

them in the current working directory

-z --gunzip Decompresses files in a tarball using gunzip

-j --bunzip2 Decompresses files in a tarball using bunzip2

-J --unxz Decompresses files in a tarball using unxz

**Listing 4.37**: Using tar to extract files from a tarball

$ tar -zxvf Project4x.tar.gz

Project42.txt

Project43.txt

Project44.txt

Project45.txt

Project46.txt

**Duplicating with dd**

The dd utility allows you to back up nearly everything on a disk

dd if= INPUT\_DEVICE of= OUTPUT-DEVICE [ OPERANDS ]

Ta b le 4.10 The dd command’s commonly used operands

Operand Description

bs=BYTES Sets the maximum block size (number of BYTES) to read and write at

a time. The default is 512 bytes.

count=N Sets the number (N) of input blocks to copy.

status=LEVEL Sets the amount (LEVEL) of information to display to STDERR.

The status=LEVEL operand needs a little more explanation. LEVEL can be set to one of

the following:

■ none only displays error messages.

■ noxfer does not display final transfer statistics.

■ progress displays periodic transfer statistics.

It is usually easier to understand the dd utility through examples. A snipped example of

performing a bit-by-bit copy of one entire disk to another disk is shown in Listing 4.38.

**Listing 4.38**: Using dd to copy an entire disk

# dd if=/dev/sdb of=/dev/sdc status=progress

8192+0 records in

8192+0 records out

4194304 bytes (4.2 MB) copied, 0.232975 s, 18.0 MB/s

#

You can also create a system image backup using a dd command similar to the one

shown in Listing 4.38, with a few needed modifications. The basic steps are as follows:

1. Shut down your Linux system.

2. Attach the necessary spare drives. You’ll need one drive the same size or larger for each

system drive.

3. Boot the system using a live CD, DVD, or USB so that you can either keep the system’s

drives unmounted or unmount them prior to the backup operation.

4. For each system drive, issue a dd command, specifying the drive to back up with the if

operand and the spare drive with the of operand.

5. Shut down the system, and remove the spare drives containing the system image.

6. Reboot your Linux system.

If you have a disk you are getting rid of, you can also use the dd command to zero out

the disk. An example is shown in Listing 4.39.

**Listing 4.39**: Using dd to zero an entire disk

# dd if=/dev/zero of=/dev/sdc status=progress

1061724672 bytes (1.1 GB) copied, 33.196299 s, 32.0 MB/s

dd: writing to '/dev/sdc': No space left on device

2097153+0 records in

2097152+0 records out

1073741824 bytes (1.1 GB) copied, 34.6304 s, 31.0 MB/s

# You need to perform this operation at least 10 times or more to thoroughly wipe the disk.

**Establishing a Hard Link**

A hard link is a file or directory that has one index (inode) number but at least two different filenames. Having a single inode number means that it is a single data file on the filesystem. Having two or more names means the file can be accessed in multiple ways. A hard link allows you to have a pseudo-copy of a file without truly copying its data. This is often used in file backups where not enough filesystem space exists to back up the file’s data. If someone deletes one of the file’s names, you still have another filename that links to its data.

**Listing 4.40**: Using the ln command to create a hard link

$ touch OriginalFile.txt

$

$ ls

OriginalFile.txt

$

$ ln OriginalFile.txt HardLinkFile.txt

$

$ ls

HardLinkFile.txt OriginalFile.txt

$

$ ls -i

2101459 HardLinkFile.txt 2101459 OriginalFile.txt

$

$ touch NewFile.txt

$

$ ls -og

total 0

-rw-rw-r--. 2 0 Aug 24 18:09 HardLinkFile.txt

-rw-rw-r--. 1 0 Aug 24 18:17 NewFile.txt

-rw-rw-r--. 2 0 Aug 24 18:09 OriginalFile.txt

Notice that both OriginalFile.txt and HardLinkFile.txt have a link count of 2 . This is because they are both hard-linked to one other fi le. NewFile.txt has a link count of 1 because it is not hard-linked to another fi le. If you want to remove a linked file but not the original file, use the unlink command.

When you create and use hard links, there are a few important items to remember:

■ The original file must exist before you issue the ln command.

■ The second filename listed in the ln command must not exist prior to issuing the

command.

■ An original file and its hard links share the same inode number.

■ An original file and its hard links share the same data.

■ An original file and any of its hard links can exist in different directories.

■ An original file and its hard links must exist on the same filesystem.

**Constructing a Soft Link**

Typically, a soft link fi le provides a pointer to a fi le that may reside on another fi lesystem. The two fi les do not share inode numbers because they do not point to the same data.

To create a symbolic link, the ln command is used with the -s or --symbolic option.

An example is shown in Listing 4.41.

Listing 4.41: Using the ln command to create a soft link

$ touch OriginalSFile.txt

$

$ ls

OriginalSFile.txt

$

$ ln -s OriginalSFile.txt SoftLinkFile.txt

$

$ ls -i

2101456 OriginalSFile.txt 2101468 SoftLinkFile.txt

$

$ ls -og

total 0

-rw-rw-r--. 1 0 Aug 24 19:04 OriginalSFile.txt

lrwxrwxrwx. 1 17 Aug 24 19:04 SoftLinkFile.txt -> OriginalSFile.txt

$

Sometimes you have a soft-linked file that points to another soft-linked file. If you want to quickly find the final file, use the readlink -f command and pass one of the soft-linked filenames as an argument to it. The readlink utility will display the final file’s name and directory location.

When creating and using soft links, keep in mind a few important items:

■ The original file must exist before you issue the ln -s command.

■ The second filename listed in the ln -s command must not exist prior to issuing the

command.

■ An original file and its soft links do not share the same inode number.

■ An original file and its soft links do not share the same data.

■ An original file and any of its soft links can exist in different directories.

■ An original file and its soft links can exist in different filesystems.

Stale links can be a serious security problem. A stale link, sometimes called a dead link, is when a soft link points to a file that was deleted or moved. The soft-linked file itself is not removed or updated. If a file with the original file’s name and location is created, the soft link now points to that new file. If a malicious file is put in the original file’s place, your server’s security could be compromised. Use symbolic links with caution and

employ the unlink command if you need to remove a linked file.

**WHERE ARE THE USAGE U WONDER? LOOK DOWN BELOW:**

**Version Links** When you use a program launcher, such as python or java , it’s convenient if you don’t have to know the currently installed version. Soft links help with this,

as shown here:

$ **which** java

/usr/bin/java

$

**$ readlink -f** /usr/bin/java

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.201.b09-2.el7\_6.x86\_64/jre/bin/java

**Backups** Hard links are useful as a pseudo-backup. This is handy when you have a working shell script (covered in Chapter 9), program, or data file in your home directory. You can simply hard-link it to another filename in a subdirectory to protect you from yourself:

$ ln ImportantFile.txt SpaceOpera/ImportantFile.txt

$

$ ls -i ImportantFile.txt SpaceOpera/ImportantFile.txt

17671201 ImportantFile.txt 17671201 SpaceOpera/ImportantFile.txt

Now, if you accidentally delete ImportantFile.txt, you’ve got a backup copy filename that

connects to the original data on the disk. That’s convenient!

**Command Substitution**(dastor janeshin) As time goes on, program names change. To maintain backward compatibility to previous command names, often links are employed. In addition, a program may be called by multiple commands; thus, links save the day here, too. One example is the make filesystem (mkfs) command for formatting ext2, 3, and 4 filesystems (covered in Chapter 3). These commands all share the same inode number. Thus, they are hard-linked, and though they have three names, are a single program, as shown here:

$ ls -i /sbin/mkfs.ext[234]

228513 /sbin/mkfs.ext2 228513 /sbin/mkfs.ext3 228513 /sbin/mkfs.ext4

Sometimes, for command substitution, you’ll discover soft links are employed instead. This is the case for the /sbin/mkfs.msdos and mkfs.vfat filesystem formatting commands:

$ ls -l /sbin/mkfs.\* | grep ^l

lrwxrwxrwx. 1 root root 8 Mar 19 17:10 /sbin/mkfs.msdos -> mkfs.fat

lrwxrwxrwx. 1 root root 8 Mar 19 17:10 /sbin/mkfs.vfat -> mkfs.fat

This is nice, because now a system admin doesn’t have to remember that the command is actually mkfs.fat and can use mkfs.msdos or mkfs.vfat instead.

**Managing File Ownership**

**Assessing File Ownership**

Linux uses a three-tiered approach to protecting fi les and directories:

■ Owner: Within the Linux system, each file and directory is assigned to a single owner.

■ Group: The Linux system also assigns each file and directory to a single group of users. The administrator can assign that group specific privileges to the file or directory that differ from the owner privileges.

■ Others: This category of permissions is assigned accounts that are neither the file

owner nor in the assigned user group.

You can view the assigned owner and group for a fi le or directory by adding the -l

option to the ls command

Viewing file owner and group settings

$ ls -l

total 12

-rw-rw-r-- 1 Rich sales 1521 Jan 19 15:38 customers.txt

-rw-r--r-- 1 Christine sales 479 Jan 19 15:37 research.txt

-rw-r--r-- 1 Christine sales 696 Jan 19 15:37 salesdata.txt

$ (The third column shows the user account assigned as the owner of the fi le

( Rich or Christine ). The fourth column shows the group assigned to the fi le ( sales ).)

When a user creates a fi le or directory, by default the Linux system automatically assigns that user as the owner. It also uses the primary group of the user as the **group** designation for the fi le or directory.

**Changing a File’s Owner**

chown [ OPTIONS ] NewOwner FILENAMES

**Listing 4.43**: Changing a file’s owner with the chown command

$ sudo chown Christine customers.txt(the -R option, which recursively changes the owner of all files under the specified directory)

**Changing a File’s Group**

chgrp [OPTIONS] NEWGROUP FILENAMES

If you’re the owner of the file, you can only change the file’s group to a group in which you have membership.

**Listing 4.44:** Changing a file’s group with the chgrp command

$ sudo chgrp marketing customers.txt (The chgrp command also uses the -R option to recursively change the group assigned to all files and directories under the specified directory)

chown NEWOWNER:NEWGROUP FILENAMES

This is often preferred over using the separate chgrp command. You can also avoid the chgrp command altogether by using

chown :NEWGROUP FILENAMES

You can check your current group’s name by issuing the **id** command. If you have membership in another group and need to make that group your current group, type newgrp groupname at the command line. Keep in mind that after you log out,

your current group will be set back to the group listed in your user account record.

**Controlling Access to Files**

Understanding Permissions

Listing 4.44: Viewing a file’s long listing

$ ls -l cake.txt

-rw-rw-r--. 1 Christine Bakers 42 Apr 24 10:45 cake.txt

$

Here is a brief description of the different items along with their value in Listing 4.44:

■ File type code ( - )

■ Permission string ( rw-rw-r-- )

■ Hard link count ( 1 )

■ File owner ( Christine )

■ File group ( Bakers )

■ File size (42 bytes)

■ Last modification date (April 24 10:45)

■ Filename (cake.txt)Ta b le 4.11 File type codes

Code Description

- The file is a binary file, a readable file (such as a text file), an image file, or a

compressed file.

d The file is a directory.

l The file is a symbolic (soft) link to another file or directory.

p The file is a named pipe or regular pipe used for communication between two or more processes.

s The file is a socket file, which operates similar to a pipe but allows more styles of communication, such as bidirectional or over a network.

b The file is a block device, such as a disk drive.

c The file is a character device, such as a point-of-sale device.

Ta b le 4.12 File vs. directory permissions

Permission File Directory

Read 🡪 Provides the ability to read/view the data stored within the file

Allows a user to list files contained within directory

Write 🡪 Allows a user to modify the data stored in the file

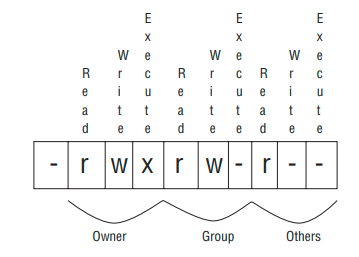
Lets the user create, move (rename), modify attributes of, and delete files within the directory

Execute 🡪 Provides the ability to run the file as a script or binary on the system.

Allows a user to change their present working directory to this location as long as this permission is set on all its parent directories as well

You can assign each tier of protection (owner, group, and other) different read, write,

and execute permissions. This creates a set of nine different permissions that are assigned to each file and directory on the Linux system. Figure 4.3 shows the order in which the permissions are displayed in the ls -l output :



**Changing a File’s Mode**

The format of the chmod command can be somewhat confusing. It uses two different

modes for denoting the read, write, and execute permission settings for the owner, group, and other: symbol and octal mode.

**Using chmod with Symbolic Mode**

In symbolic mode, you denote permissions by using a letter code for the levels shown in

Table 4.13 along with another letter code for the read (r), write (w), or execute (x) permission.

Ta b le 4.13 Symbolic mode levels

Level Description

u owner

g group

o others

a all tiers

The two codes are separated with a plus sign (+) if you want to add the permission, a

minus sign (-) to remove the permission, or an equal sign (=) to set the permission as the only permission. Listing 4.45 shows an example of this.

**Listing 4.45:** Changing the file owner

$ chmod g-w customers.txt

$

$ ls -l

total 12

-rw-r--r-- 1 Christine marketing 1521 Jan 19 15:38 customers.txt

**In Listing 4.45**, the g-w code in the chmod command indicates to remove the write permission for the group from the customers.txt file.

You can combine letter codes for both to make multiple changes in a single chmod command, as shown in Listing 4.46.

**Listing 4.46**: Combining permission changes

$ chmod ug=rwx research.txt

$

$ ls -l

total 12

-rw-r--r-- 1 Christine marketing 1521 Jan 19 15:38 customers.txt

-rwxrwxr-- 1 Christine sales 479 Jan 19 15:37 research.txt

The ug code assigns the change to both the owner and the group, and the rwx code

assigns the read, write, and execute permissions. The equal sign indicates to set those

permissions.

**Using chmod with Octal Mode**

tabLe 4.14 Octal mode permissions

Octal value Permission Meaning

0 --- no permissions

1 --x execute only

2 -w- write only

3 -wx write and execute

4 r-- read only

5 r-x read and execute

6 rw- read and write

7 rwx read, write, and execute

Listing 4.47: Using octal mode to assign permissions

$ chmod 664 research.txt

-rw-rw-r-- 1 Christine sales 479 Jan 19 15:37 research.txt

So what’s my default mask? (default permission given to files)

The user mask value is set with the umask command. You can view your current umask

setting by entering the command by itself on the command line as shown in Listing 4.48.

Listing 4.48: Viewing the current user mask setting via the umask command

$ umask

0022

$

The output of the umask command shows four octal values. The first octal value represents the mask for the SUID (4), SGID (2), and sticky (1) bits assigned to files and directories you create (covered later in this chapter). You’ll notice, however, that in Listing 4.48,

the value is set to 0, which means these bits are ignored. The next three octal values mask

the owner, group, and other permission settings.

The mask is a bitwise (works with individual bits) operation applied to the permission

bits on the file or directory. Any bit that’s set in the mask is removed from the permissions

for the file or directory. If a bit isn’t set, the mask doesn’t change the setting. Table 4.15

demonstrates how the umask values work in practice when creating files and directories on

your Linux system. Note that we are ignoring the first octal value (which applies to SUID,

SGID, and the sticky bit) for now and focusing on the last three.

Ta b le 4.15 Results from common umask values for files and directories

umask Created files Created directories

000 666 (rw-rw-rw-) 777 (rwxrwxrwx)

002 664 (rw-rw-r--) 775 (rwxrwxr-x)

022 644 (rw-r--r--) 755 (rwxr-xr-x)

027 640 (rw-r-----) 750 (rwxr-x---)

077 600 (rw-------) 700 (rwx------)

277 400 (r--------) 500 (r-x------)

You can test this by determining your current user mask value, creating a new fi le and

directory on your Linux system, and then reviewing the resulting permissions. An example

is shown in Listing 4.49.

**Listing 4.49**: Viewing the effect of the current user mask setting on permissions

$ umask

0022

$

$ mkdir test1

$ touch test2

$ ls -l

[…]

drwxr-xr-x 2 rich rich 4096 Jan 19 17:08 test1

-rw-r--r-- 1 rich rich 0 Jan 19 17:08 test2

$

The umask value of 0022 created the default fi le permissions of rw-r--r-- , or octal 644 ,

on the test2 fi le, and rwx-r-xr-x , or octal 755 , on the test1 directory, as expected.

You can change the default umask setting for your user account by using the umask command from the command line. An example of doing this is shown in Listing 4.50.

**Listing 4.50:** Changing the user mask setting and viewing the change’s effect

$ umask 027

$

$ touch test3

$ ls -l test3

-rw-r----- 1 rich rich 0 Jan 19 17:12 test3

$

The umask value is normally set in a script that the Linux system runs at login time, such as in the /etc/profile file.

**Changing Special Access Modes**

There are three special permission bits that Linux uses for controlling advanced behavior of files and directories: SUID, SGID, and the sticky bit.

**Looking at SUID**

**The Set User ID (SUID)** bit is used with executable files. It tells the Linux kernel to run the program with the permissions of the file owner and not the user account actually running the file. This feature is most commonly used in server applications that must run as the root user account to have access to all files on the system, but the Linux system starts them as a standard user account.

The SUID bit is indicated by an s in place of the execute permission letter for the file

owner**: rwsr-xr-x**. The execute permission is assumed for the system to run the file. If the SUID bit is set on a file that doesn’t have execute permission for the owner, it’s indicated by a capital S. A practical example of SUID on Linux is the passwd utility. The passwd utility allows you to change your password, which is stored in the /etc/shadow file (covered in Chapter 7). Because the shadow file only allows the root user (the file’s owner) to write to it, you must temporarily gain the root user’s permission status. This is done via the SUID permission set on the passwd program’s file as shown in Listing 4.51.

**Listing 4.51**: Viewing the passwd utility’s and /etc/shadow file’s permissions

$ ls -l /etc/shadow

-rw-r----- 1 root shadow 1425 Mar 21 17:51 /etc/shadow

$

$ which passwd

/usr/bin/passwd

$

$ ls -l /usr/bin/passwd

-rwsr-xr-x 1 root root 59640 Jan 25 2018 /usr/bin/passwd

$

To set the SUID bit for a file, in symbolic mode add s to the owner permissions, or in

octal mode include a 4 at the start of the octal mode setting.

# chmod u+s myapp

# chmod 4750 myapp

**Looking at SGID**

**The Set Group ID (SGID)** bit works differently in files and directories. For files, it tells

Linux to run the program file with the file’s group permissions. It’s indicated by an s in the group execute position**: rwxrwsr--.** Like SUID, if the execute permission is not granted, the setting is benign and shown as a capital S in the group execute position.

For directories, the SGID bit helps us create an environment where multiple users can

share files. When a directory has the SGID bit set, any files users create in the directory

are assigned the group of the directory and not that of the user. That way, all users in that group can have the same permissions to all of the files in the shared directory.

To set the SGID bit, in symbolic mode add s to the group permissions, or in octal mode

include a 2 at the start of the octal mode setting:

# chmod g+s /sales

# chmod 2660 /sales

**Looking at the Sticky Bit**

Finally, the sticky bit is used on directories to protect one of its files from being deleted by those who don’t own the file, even if they belong to the group that has write permissions to the file. The sticky bit is denoted by a t in the execute bit position for others**: rwxrw-r-t**. The sticky bit is often used on directories shared by groups. The group members have read and write access to the data files contained in the directory, but only the file owners can remove files from the shared directory. Typically the /tmp directory has the sticky bit set as shown in Listing 4.52.

**Listing 4.52**: Viewing the /tmp directory’s sticky bit permission

$ ls -ld /tmp

drwxrwxrwt 12 root root 4096 Apr 25 13:50 /tmp

$

To set the sticky bit, in symbolic mode add t to the owner permissions, or in octal mode

include a 1 at the start of the octal mode setting:

# chmod o+t /sales

# chmod 1777 /sales

**Locating Files**

**Getting to Know the FHS (the Linux filesystem hierarchy standard)**

tabLe 4.16 Common Linux FHS folders

Folder Description

/ The root filesystem

/boot Contains bootloader files used to boot the system

/dev Holds device files

/home Contains user data files

/lib Holds shared libraries and kernel modules

/media Traditionally used as a mount point for removable devices

/mnt Used as the current mount point for removable devices

/opt Contains data for optional third-party programs

/tmp Contains temporary files created by system users

/usr Contains data for standard Linux programs

/usr/bin Contains local user programs and data

/usr/lib Holds libraries for programming and software packages

/usr/local Contains data for programs unique to the local installation

/usr/sbin Contains data for system programs and data

**Using the which Command**

The which command shows you **the full path name of a shell command** passed as an argument. Examples of using this utility are shown in Listing 4.53.

Listing 4.53: Using the which command

$ which passwd

/usr/bin/passwd

$

$ which shutdown

/usr/sbin/shutdown

$

**Listing 4.54**: Using the which command to see a command alias

$ which ls

alias ls='ls --color=auto'

/usr/bin/ls

$

$ unalias ls

$

$ which ls

/usr/bin/ls

$

When the which utility is used on the ls command in Listing 4.54, it shows that currently the ls command has an alias. Thus, when you type ls, it is as if you have typed the

ls --color=auto command. After employing the unalias command on ls, the which utility shows only the ls program’s location.

**Using the whereis Command**

Another command for locating files is the whereis utility. This utility allows you to locate any command’s program binaries and locate source code files as well as any manual pages.

Examples of using the whereis utility are shown in Listing 4.55.

Listing 4.55: Employing the whereis command

$ whereis diff

diff: /usr/bin/diff /usr/share/man/man1/diff.1.gz

/usr/share/man/man1p/diff.1p.gz

**Using the locate Command**

A very convenient and simple utility to use in finding files is the locate program. This utility searches a database, mlocate.db, which is located in the /var/lib/mlocate/ directory, to determine if a particular file exists on the local system.

locate [OPTION]... PATTERN...

Ta b le 4.17 The locate command’s commonly used options

Short Long Description

-A --all Display filenames that match all the patterns, instead of displaying files that match only one pattern in the pattern list.

-b --basename Display only filenames that match the pattern and do not include any directory names that match the pattern.

-c --count Display only the number of files whose name matches the pattern instead of displaying filenames.

-i --ignore-case Ignore case in the pattern for matching filenames.

-q --quiet Do not display any error messages, such as permission denied, when processing.

-r --regexp R Use the regular expression, R, instead of the pattern list to match filenames.

-w --wholename Display filenames that match the pattern and include any directory names that match the pattern. This is default behavior.

**Listing 4.56**: Using the locate command to find a file

$ locate Project42.txt

/home/Christine/Answers/Project42.txt

If the file is on your system and you have permission to view it, the locate utility will display the file’s directory path and name.

**Listing 4.57:** Using the locate command with no file globbing (don’t forget the problem of wild cards)

$ locate -b passwd

/etc/passwd

/etc/passwd-

/etc/pam.d/passwd

/etc/security/opasswd

/usr/bin/gpasswd $ locate -b '\passwd'

/etc/passwd

/etc/pam.d/passwd

/usr/bin/passwd

/usr/share/bash-completion/completions/passwd

$

Another problem you can run into deals with newly created or downloaded fi les. The

locate utility is really searching the mlocate.db database as opposed to searching the

virtual directory structure. This database is typically updated only one time per day via a cron job. Therefore, if the fi le is newly created, locate won’t fi nd it.

The mlocate.db database is updated via the updatedb utility. You can run it manually

using super user privileges if you need to fi nd a newly created or downloaded fi le. Be aware that it may take a while to run.

you can prevent certain directory locations from being scanned via the updatedb utility. Simply modify the /etc/updatedb.conf configuration file, and modify one of the PRUNEFS , PRUNENAMES , or PRUNEPATHS directives in order to designate directories to skip during an updated scan. Type man updatedb.conf at the command line for more details.

**Using the find Command**

The fi nd command is very fl exible. It allows you to locate fi les based on data, such as who owns the fi le, when the fi le was last modifi ed, permissions set on the fi le, and so on. The basic command syntax is as follows:

find [ PATH ...] [ OPTION ] [ EXPRESSION)

tabLe 4.18 The find command’s commonly used options and expressions

Option Expression Description

-cmin n Display names of files whose status changed n minutes ago.

-empty N/A Display names of files that are empty and are a regular text file or a directory.

-gid n Display names of files whose group ID is equal to n.

-group name Display names of files whose group is name.

-inum n Display names of files whose inode number is equal to n.

-maxdepth n When searching for files, traverse down into the starting point directory’s tree only n levels.

-mmin n Display names of files whose data changed n minutes ago.

-name pattern Display names of files whose name matches pattern. Many regular expression arguments may be used in the pattern and need to be enclosed in quotation marks to avoid unpredictable results. Replace -name with -iname to ignore case.

-nogroup N/A Display names of files where no group name exists for the file’s group ID.

-nouser N/A Display names of files where no username exists for the file’s user ID.

-perm mode Display names of files whose permissions matches mode. Either octal or symbolic modes may be used.

-size n Display names of files whose size matches n. Suffixes can be used to make the size more human readable, such as G for gigabytes.

-user name Display names of files whose owner is name.

**Listing 4.59:** Employing the find command

$ find . -name "\*.txt"

./Project47.txt

./Answers/Project42.txt

The find command is very handy for auditing your system on a regular basis as well as

when you are concerned that your server has been hacked. The -perm option is useful for one of these audit types, and an example is shown in Listing 4.60.

Listing 4.60: Using the find command to audit a server

$ find /usr/bin -perm /4000

/usr/bin/newgrp

/usr/bin/chsh

/usr/bin/arping

/usr/bin/gpasswd

/usr/bin/chfn

/usr/bin/traceroute6.iputils

/usr/bin/pkexec

/usr/bin/passwd

/usr/bin/sudo

$

In Listing 4.60, the /usr/bin directory is being audited for the potentially dangerous

SUID permission by using the find utility and its -perm option. The expression used is

/4000, which will ask the find utility to search for SUID settings (octal code 4) and, due to the forward slash ( / ) in front of the number, ignore the other fi le permissions (octal

codes 000 ). The resulting fi lenames all legitimately use SUID, and thus, nothing suspicious is going on here.

**Using the type Command**

So you found the fi le, but you don’t know what kind of fi le it is. You can employ the file

command for some fi les, but another useful utility is the type program. (by the way the whatis command tells what exactly a command does for you)

The type utility will display how a fi le is interpreted by the Bash shell if it is entered

at the command line. Three categories it returns are alias, shell built-in, and external

command (displaying its absolute directory reference). A few examples are shown in

Listing 4.61.

Listing 4.61: Using the type command to determine a command’s interpretation

$ type ls

ls is aliased to 'ls --color=auto'

$

$ type cd

cd is a shell builtin

$

$ type find

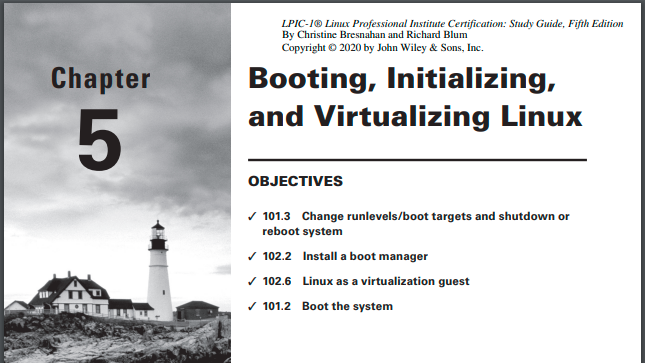
find is /usr/bin/find

$

Notice that the ls command on this system is an alias. The cd program is built into the

Bash shell (covered in Chapter 1). The find command is an external command, because

the type utility provides its absolute directory location within the Linux virtual directory structure.



If you need to troubleshoot boot problems, you can review the boot-time messages

using the dmesg command.

**Listing 5.1**: Using the dmesg command to display the kernel ring buffer’s contents

$ dmesg

[ 0.000000] Initializing cgroup subsys cpuset

[ 0.000000] Initializing cgroup subsys cpu

[ 0.000000] Initializing cgroup subsys cpuacct

[ 0.000000] Linux version 3.10.0-957.10.1.el7.x86\_64

**Listing 5.2**: Using the journalctl utility to display the kernel ring buffer’s contents

-- Logs begin at Tue 2019-04-30 11:21:34 EDT, end at Tue 2019-04-30 12:53:41 EDT

Apr 30 11:21:34 localhost.localdomain kernel: Initializing cgroup subsys cpuset

Apr 30 11:21:34 localhost.localdomain kernel: Initializing cgroup subsys cpu

Apr 30 11:21:34 localhost.localdomain kernel: Initializing cgroup subsys cpuacct

Apr 30 11:21:34 localhost.localdomain kernel: Linux version 3.10.0-957.10.1.el7.

Some Linux distributions also store the boot messages in a log fi le, usually in the

/var/log directory. For Debian-based systems, the fi le is usually /var/log/boot , and

for Red Hat-based systems, the fi le is /var/log/boot

When troubleshooting boot problems, instead of slogging through the

dmesg utility’s output or one of the boot message log file’s output, pipe

their output into the grep command or the less utility (both covered in

Chapter 1). That way, you can search for specific devices (such as /dev/sda ),

key phrases (such as disabled ), and/or specific items (such as the loaded

BOOT\_IMAGE ) easily.

The MBR Startup

When booting from a hard drive, you must designate which hard drive, and from which

partition on the hard drive the BIOS should load the boot loader program. This is done by defining a master boot record (MBR). The MBR is the first sector on the first hard drive partition on the system. There is only one MBR for the computer system. The BIOS looks for the MBR and reads the program stored there into memory. Since the boot loader program must fit in one sector, it must be very small, so it can’t do too much. The boot loader program mainly points to the location of the actual operating system kernel fi le, which is stored in a boot sector of a separate partition on the system. There are no size limitations on the kernel boot fi le. The boot loader program isn’t required to point directly to an operating system kernel file—it can point to any type of program, including another boot loader program. You can create a primary boot loader program that points to a secondary boot loader program, which provides options to load multiple operating systems. This process is called chain loading.

**The UEFI Startup**

Instead of relying on a single boot sector on a hard drive to hold the boot loader program,

UEFI specifi es a special disk partition called the EFI System Partition (ESP) to store

boot loader programs. This allows for any size of boot loader program, plus the ability to

store multiple boot loader programs for multiple operating systems. The ESP setup utilizes the old Microsoft File Allocation Table (FAT) fi lesystem to store the boot loader programs. On Linux systems, the ESP is typically mounted in the /boot/efi/ directory, and the boot loader fi les are commonly stored using the .efi filename extension. The UEFI fi rmware utilizes a built-in mini boot loader (sometimes referred to as a boot manager ), which allows you to confi gure the specifi c boot loader program fi le to launch. With UEFI, you need to register each individual boot loader fi le that you want to appear at boot time in the boot manager interface menu. You can then select the boot loader to run each time you boot the system.

If you’re not sure whether your system is using UEFI, you can easily check. After the Linux system boots, issue the **command ls /sys/firmware/efi** .

**GRand Unifi ed Bootloader (GRUB)**

GRUB2 was created in 2005 as a total rewrite of the GRUB Legacy system. It supports advanced features, such as the ability to load hardware driver modules and using logic statements to alter the boot menu options dynamically, depending on conditions detected on the system (such as if an external hard drive is connected).

**Using GRUB Legacy as the Boot Loader**

The GRUB Legacy boot loader was designed to simplify the process of creating boot menus and passing options to kernels. GRUB Legacy allows you to select multiple kernels and/or operating systems using both a menu interface as well as an interactive shell.

**Configuring GRUB Legacy**

The GRUB Legacy system stores the menu commands in a standard text confi guration

fi le called menu.lst ; it is stored in the /boot/grub/ directory. Red Hat–derived Linux distributions use grub.conf instead of menu.lst, you may find that the menu.lst file is symbolically linked to the grub.conf file.

The GRUB Legacy configuration file consists of two sections:

■■ Global definitions

■■ Operating system boot definitions

Tabl e 5.1 GRUB Legacy global commands

Setting Description

Color Specifies the foreground and background colors to use in the boot menu

Default Defines the default menu option to select

Fallback A secondary menu selection to use if the default menu option fails

Hiddenmenu Don’t display the menu selection options

Splashimage Points to an image file to use as the background for the boot menu

Timeout Specifies the amount of time to wait for a menu selection before using

the default

For GRUB Legacy, to define a value for a command, you just list the value as a command-line parameter:

default 0

timeout 10

color white/blue yellow/blue

**and define the operating system:**

■ Title: The first line for each boot definition section, this is what appears in the

boot menu.

■ Root: Defines the disk and partition where the GRUB /boot folder partition is located on the system.

■ Kernel : Defines the kernel image file stored in the / boot folder to load.

■ Initrd : Defines the initial RAM disk file or filesystem, which contains drivers necessary for the kernel to interact with the system hardware.

■ Rootnoverify : Defines non-Linux boot partitions, such as Windows.

The root command defi nes the hard drive and partition that contains the /boot folder for GRUB Legacy. GRUB Legacy uses a somewhat odd way of referencing those values**:(hddrive , partition).** to reference the fi rst partition on the fi rst hard drive of the system, you’d use (hd0,0) . To reference the second partition on the fi rst hard drive, you’d use (hd0,1) . The initrd command is another important feature in GRUB Legacy. It helps to solve a problem that arises when using specialized hardware or fi lesystems as the root drive. The initrd command defi nes a fi le that’s mounted by the kernel at boot time as a RAM disk or fi lesystem. The kernel can then load modules from the RAM disk or fi lesystem, which then allows it to access hardware or fi lesystems not compiled into the kernel itself. Before Linux kernel v2.6, the initial RAM disk was used to hold kernel

modules needed at boot time. Since that time, it has been replaced with the initial RAM filesystem ( initramfs ). It can be a little confusing, because the initrd command within the GRUB configuration file denotes either an initial RAM disk or an initial RAM filesystem.

**Listing 5.3**: Sample GRUB Legacy configuration file

default 0

timeout 10

color white/blue yellow/blue

title CentOS Linux

root (hd1,0)

kernel (hd1,0)/boot/vmlinuz

initrd /boot/initrd (The Linux boot selection specifi es both the kernel fi le to load as well as the initrd image fi le to load into memory.)

title Windows

rootnoverify (hd0,0)

After you build the GRUB Legacy configuration fi le, you must install the GRUB Legacy

program in the MBR. The command to do this is grub-install. You can specify the partition using either the Linux or the GRUB Legacy format. For example, to use the Linux format, you’d type

# grub-install /dev/sda

To install GRUB on the MBR of the fi rst hard drive and use the GRUB Legacy format,

you must enclose the hard drive format in quotes:

# grub-install '(hd0)'

If you’re using the chainloading method and prefer to install a copy of GRUB Legacy on the boot sector of a partition instead of to the MBR of a hard drive, you must specify the partition, again using either the Linux or the GRUB format:

# grub-install /dev/sda1

# grub-install 'hd(0,0)'

**Interacting with GRUB Legacy**

1. Use the arrow key to move to the boot option you want to modify, and then press the

E key.

2. Use the arrow key to move the cursor to the line that you need to modify, and then

press the E key to edit it.

3. Press the B key to boot the system using the new values.

You can also press the C key at any time to enter an interactive shell mode (also called

GRUB command line), allowing you to submit GRUB commands on the fly.

**Using GRUB 2 as the Boot Loader**

he GRUB2 system changes the configuration file name to grub.cfg. Where the file is stored depends on your system’s firmware:

■ BIOS: The grub.cfg file is stored in the /boot/grub/ or /boot/grub2/ directory. (This allows you to have both GRUB Legacy and GRUB2 installed at the same time.)

■ UEFI: The grub.cfg file is stored in the /boot/efi/EFI/distro-name/ directory.

**Configuring GRUB2**

There are also a few changes to the commands used in GRUB2.

**Listing 5.5**: Simplified sample GRUB2 configuration file

[…]

menuentry "CentOS Linux" {

[…]

set root=(hd1,1)

linux16 /vmlinuz[…]

initrd /initramfs[…]

}

menuentry "Windows" {

set root=(hd0,1)

[…]

GRUB2 changes the numbering system for partitions. Although it still uses 0 for the first hard drive, the first partition is set to 1. So to define the /boot directory on the first partition of the first hard drive, you now need to use **set root=(hd0,1**).

**For GRUB2, the commands to remember are as follows:**

■ Menuentry : The first line for each boot definition section; this is what appears in the

boot menu.

■ set root : Defines the disk and partition where the GRUB2 /boot directory partition is

located on the system.

■ linux , linux16 : For BIOS systems, defines the kernel image file stored in the / boot directory to load.

■ Linuxefi : For UEFI systems, defines the kernel image file stored in the / boot directory to load.

■ Initrd : For BIOS systems, defines the initial RAM filesystem, which contains drivers necessary for the kernel to interact with the system hardware.

■ Initrdefi : For UEFI systems, defines the initial RAM filesystem, which contains drivers necessary for the kernel to interact with the system hardware.

Notice in Listing 5.5 that you must enclose each individual boot section with braces immediately following the menuentry command. In addition, the rootnoverify command

is not used in GRUB2. Non-Linux boot options are now defi ned the same as Linux boot options using the root environment variable. The confi guration process for GRUB2 is also somewhat different. Although GRUB2 uses the /boot/grub/grub.cfg fi le as the confi guration fi le, you should never modify that fi le. Instead, there are separate confi guration fi les stored in the /etc/grub.d folder. This allows you (or the system) to create individual confi guration fi les for each boot option installed on your system (for example, one confi guration fi le for booting Linux and another for booting Windows). For global commands, the /etc/default/grub confi guration fi le is used. Typically, you should not modify this fi le either, but again use the confi guration fi les in the /etc/grub.d/ directory. The format for some of the global commands has changed from the GRUB Legacy commands, such as GRUB\_TIMEOUT instead of just timeout .

Most Linux distributions generate a new grub.cfg confi guration fi le automatically after certain events, such as when upgrading the kernel. Usually, the distribution will keep a boot menu option pointing to the old kernel fi le, which is handy just in case the new one fails.

**Installing GRUB2**

Unlike with GRUB Legacy, you don’t need to install GRUB2. All you need to do is to rebuild the main installation fi le. This is done by running either the grub-mkconfig or grub2-mkconfig program.

Some graphical desktops (such as Ubuntu) hide the GRUB boot menu behind a graphical interface. Usually, if you hold down the Shift key when the system first boots, this will display the GRUB boot menu.

taBLe 5. 2 GRUB2 menu interface keystrokes

Key(s) Description

Arrow Select boot menu option

C Starts the GRUB command-line interface

E Enters editing mode for currently selected menu option

Enter Boots currently selected menu option

P Used to enter password, if required

Tabl e 5.3 GRUB2 entry editor keystrokes

Key(s) Description

Arrow Highlight the entry line to modify

Enter Adds a new line (best to use at a line’s end)

ESC Discards any changes and return to menu interface

Ctrl+C Starts the GRUB command-line interface

Ctrl+X Boot system with edited entry

Tabl e 5.4 Kernel parameters

Parameter Description

console= Set the console device

debug Enable kernel debugging

init= Execute the specified program, such as a Bash shell

(/bin/bash) instead of /sbin/init

initrd= Change the location of the initial RAM filesystem

mem Set the total amount of available system memory

ro Mount root filesystem as read-only

root= Change the root filesystem

rootflags= Set root filesystem’s mount options

rw Mount root filesystem as read-write

selinux Disable SELinux at boot time

single , Single , 1 , or S Boot a SysVinit system to single-user mode

systemd.unit = Boot a systemd system to specified target

A legacy boot loader that predates GRUB Legacy is LILO

The systemd-boot loader program is starting to gain popularity in Linux distributions that use the systemd initialization method (covered later in this chapter). This boot loader generates a menu of boot image options and can load any UEFI boot image. The U-Boot boot loader (also called Das U-Boot ) can boot from any type of disk. It can load any type of boot image. The SYSLINUX project includes fi ve separate boot loader programs that have special uses in Linux:

■ SYSLINUX: A boot loader for systems that use the Microsoft FAT filesystem (popular for booting from USB flash drives)

■ EXTLINUX: A mini-boot loader for booting from an ext2, ext3, ext4, or btrfs filesystem

■ ISOLINUX: A boot loader for booting from a LiveCD or LiveDVD

■ PXELINUX: A boot loader for booting from a network server

■ MEMDISK: A utility for booting older DOS operating systems from the other SYSLINUX project boot loaders

Since UEFI can load any size of boot loader program, it’s now possible to load a Linux operating system kernel directly without a special boot loader program. This feature was incorporated in the Linux kernel starting with version 3.3.0.

**The Initialization Process (daemon yani roh)**

The initialization daemon ( init ) determines which services are started and in what order.

This daemon also allows you to stop and manage the various system services. There are two initialization daemons with which you should be familiar:

**SysVinit** The SysVinit (SysV) was based on the Unix System V initialization daemon. Though it is not used by many major Linux distributions anymore, you still may fi nd it lurking around that older Linux server at your company.

**systemd** The systemd initialization method is the new kid on the block. Started around 2010, it is now the most popular system service initialization and management mechanism. This daemon reduces initialization time by starting services in a parallel manner.

**INIT**

service startups are handled by the init program. This program can be located in the /etc/ , the /bin/ , or the /sbin/ directory. Also, it typically has a process ID (PID) of 1.

**Listing 5.7**: Checking the init program for links (so you know what is your system and service manager)

# readlink -f /sbin/init

/usr/lib/systemd/**systemd**

#

**Listing 5.8**: Checking PID 1

# ps -p 1

PID TTY TIME CMD

1 ? 00:00:06 systemd

#

The init program or systemd is the parent process for every service on a Linux system.

The following brief list shows a few Linux distribution versions that used Upstart

■ Fedora v9–v14

■ openSUSE v11.3–v12.2

■ RHEL v6

■ Ubuntu v6.10–v15.04

**Exploring Unit Files The easiest way to start exploring systemd is through the systemd units.**

A unit defines a

service, a group of services, or an action. Each unit consists of a name, a type, and a configuration file. There are currently 12 different systemd unit types:

■■ automount

■■ device

■■ mount

■■ path

■■ scope

■■ service

■■ slice

■■ snapshot

■■ socket

■■ swap

■■ target

■■ timer

The systemctl utility is the main gateway to managing systemd and system services. Its

basic syntax is as follows:

systemctl [OPTIONS...] COMMAND [NAME...]

Listing 5.9: Looking at systemd units

$ systemctl list-units

UNIT LOAD ACTIVE SUB DESCRIPTION

[…]

smartd.service loaded active running Self Monitor[…]

sshd.service loaded active running OpenSSH serv[…]

sysstat.service loaded active exited Resets Syste[…]

[…]

graphical.target loaded active active Graphical I[…]

[…]

$

Groups of services are started via target unit fi les. At system startup, the default.target unit ensures that all required and desired services are launched at system initialization. The systemctl get-default command displays the target fi le, as shown in Listing 5.10 on a

CentOS distribution.

**Listing 5.10**: Looking at the **default.target link**

$ systemctl get-default

graphical.target

$

Table 5.5 shows the commonly used system boot target unit fi les.

taBLe 5. 5 Commonly used system boot target unit files

Name Description

graphical.target 🡪Provides multiple users access to the system via local terminals and/or through the network. Graphical user interface (GUI) access is offered.

multi-user.target 🡪 Provides multiple users access to the system via local terminals and/or through the network. No GUI access is offered.

runlevel n .target 🡪 Provides backward compatibility to SysVinit systems, where n is set to 1–5 for the desired SysV runlevel equivalence.

The master systemd configuration file is /etc/systemd/system.conf . In this file you will find all the default configuration settings commented out via a hash mark ( # ). Viewing this file is a quick way to see the current systemd configuration. If you need to modify the configuration, just edit the file. However, it would be wise to peruse the file’s man page first by typing man systemd-system.conf at the command line.

**Focusing on Service Unit Files**

Service unit files contain information such as which environment file to use, when a service must be started, what targets want this service started, and so on. These configuration files are located in different directories. The following list shows the directory locations in ascending priority order:

1. /etc/systemd/system/

2. /run/systemd/system/

3. /usr/lib/systemd/system/

**Listing 5.11**: Looking at systemd unit files

$ systemctl list-unit-files

UNIT FILE STATE

[…]

dev-hugepages.mount static

dev-mqueue.mount static

proc-fs-nfsd.mount static

[…]

nfs.service disabled

nfslock.service static

ntpd.service disabled

ntpdate.service disabled

[…]

ctrl-alt-del.target disabled

default.target static

emergency.target static

[…]

$

There are at least 12 different enablement states, but you’ll commonly see these 3:

■■ enabled: Service starts at system boot.

■■ disabled: Service does not start at system boot.

■■ static: Service starts if another unit depends on it. Can also be manually started.

**Listing 5.12: Finding and displaying a systemd unit file**

$ systemctl cat ntpd.service

# /usr/lib/systemd/system/ntpd.service

[Unit]

Description=Network Time Service

After=syslog.target ntpdate.service sntp.service

[Service]

Type=forking

EnvironmentFile=-/etc/sysconfig/ntpd

ExecStart=/usr/sbin/ntpd -u ntp:ntp $OPTIONS

PrivateTmp=true

[Install]

WantedBy=multi-user.target

$

Tabl e 5.6 Commonly used service unit file [Unit] section directives

Directive Description

**After** Sets this unit to start after the designated units.

**Before** Sets this unit to start before the designated units.

**Description** Describes the unit.

**Documentation** Sets a list of uniform resource identifiers (URIs) that point to documentation sources. The URIs can be web locations, system files, info pages, and man pages.

**Conflicts** Sets this unit to not start with the designated units. If any of the designated units start, this unit is not started. (Opposite of Requires.)

**Requires** Sets this unit to start together with the designated units. If any of the designated units do not start, this unit is not started. (Opposite of Conflicts .)

**Wants** Sets this unit to start together with the designated units. If any of the designated units do not start, this unit is still started.

There is a great deal of useful information in the man pages for systemd and unit configuration files. Just type man -k systemd to find several items you can explore. The [Service] directives within a unit fi le set confi guration items that are specifi c to that service. The commonly used [Service] section directives are described in Table 5.7.

taBLe 5. 7 Commonly used service unit file [Service] section directives

Directive Description

**ExecReload** Indicates scripts or commands (and options) to run when unit is reloaded.

**ExecStart** Indicates scripts or commands (and options) to run when unit is started.

**ExecStop** Indicates scripts or commands (and options) to run when unit is stopped.

**Environment** Sets environment variable substitutes, separated by a space.

**Environment File** Indicates a file that contains environment variable substitutes.

**RemainAfterExit** Set to either no (default) or yes . If set to yes , the service is left active

even when the process started by ExecStart terminates. If set to no, then ExecStop is called when the process started by ExecStart terminates.

**Restart** Service is restarted when the process started by ExecStart terminates. Ignored if a systemctl restart or systemctl stop command is issued. Set to no (default), on-success , on-failure , on-abnormal , on-watchdog , on-abort , or always .

**Type** Sets the startup type.

You will only find a unit file [Service] section in a service unit file. This middle section is different for each unit type. For example, in auto-mount unit files, you would find an [Automount] section as the middle unit file section.

The [Install] directives within a unit fi le determine what happens to a service if it is enabled or disabled. An enabled service starts at system boot. A disabled service does not start at system boot. The commonly used [Install] section directives are described in Table 5.8.

taBLe 5. 8 Commonly used service unit file [Install] section directives

Directive Description

Alias🡪 Sets additional names that can denote the service in systemctl commands.

Also🡪 Sets additional units that must be enabled or disabled for this service. Often the additional units are socket type units.

RequiredBy🡪 Designates other units that require this service.

WantedBy🡪 Designates which target unit manages this service.

**Focusing on Target Unit Files**

The primary purpose of target unit fi les is to group together various services to start at

system boot time.

**Listing 5.13**: Finding and displaying the systemd target unit file

$ systemctl get-default

graphical.target

$

**Modifying systemd Configuration Files**

You should not modify any unit files in the /lib/systemd/system/ or /usr/lib/systemd/system/ directory. To modify a unit configuration file, copy the file to the /etc/systemd/system/ directory and modify it there. This modified file will take precedence over the original unit file left in the original directory. After you make these modifications, you must complete a few more steps. Find and compare any unit file that overrides another unit file by issuing the systemd-delta command. It will display any unit files that are duplicated, extended, redirected, and so on. To have your changes take effect, issue the systemctl daemon-reload command.

**Listing 5.14**: Viewing a service unit’s status via systemctl

$ systemctl status ntpd

• ntpd.service - Network Time Service

Loaded: loaded (/usr/lib/systemd/system/ntpd.service;

disabled; vendor preset: disabled)

Active: inactive (dead)

Tabl e 5.9 Commonly used systemctl service management commands

Command Description

**daemon-reload** Load the unit configuration file of the running designated unit(s) to make unit file configuration changes without stopping the service. Note that this is different from the reload command.

**disable** Mark the designated unit(s) to not be started automatically at system boot time.

**enable** Mark the designated unit(s) to be started automatically at system boot time.

**mask** Prevent the designated unit(s) from starting. The service cannot be started using the start command or at system boot. Use the –now option to immediately stop any running instances as well. Use the --running option to mask the service only until the next reboot or unmask is used.

**restart** Stop and immediately restart the designated unit(s). If a designated unit is not already started, this will simply start it.

**start** Start the designated unit(s).

**status** Display the designated unit’s current status.

**stop** Stop the designated unit(s).

**reload** Load the service configuration file of the running designated unit(s) to make service configuration changes without stopping the service. Note that this is different from the daemon-reload command.

**unmask** Undo the effects of the mask command on the designated unit(s).

Notice the difference in Table 5.9 between the daemon-reload and the reload command.

This is an important difference. Use the daemon-reload command if you need to load systemd unit file configuration changes for a running service. Use the reload command to

load a service’s modified configuration file.

Listing 5.15: Determining if a service is running via systemctl

# systemctl stop sshd

#

# systemctl is-active sshd

inactive

#

# systemctl start sshd

#

# systemctl is-active sshd

active

#

Tabl e 5.10 Convenient systemctl service status commands

Command Description

**is-active** Displays active for running services and failed for any service that has reached a failed state.

**is-enabled** Displays enabled for any service that is configured to start at system boot and disabled for any service that is not configured to start at system boot.

**is-failed** Displays failed for any service that has reached a failed state and active for running services.

Services can fail for many reasons: for hardware issues, a missing dependency set in the unit configuration file, an incorrect permission setting, and so on.

**Listing 5.16**: Determining if a service has failed via systemctl

$ systemctl is-failed NetworkManager-wait-online.service

failed

$

**Examining Special systemd Commands**

**The systemctl utility has several commands that go beyond service management**

**Listing 5.17**: Determining a system’s operational status

$ systemctl is-system-running

running

$

Tabl e 5.11 Operational statuses provided by systemctl is-system-running

Status Description

running System is fully in working order.

degraded System has one or more failed units.

maintenance System is in emergency or recovery mode.

initializing System is starting to boot.

starting System is still booting.

stopping System is starting to shut down.

If you receive degraded status, however, you should review your units to see which ones have failed and take appropriate action.

**Listing 5.18**: Finding failed units

$ systemctl is-system-running

degraded

$

$ systemctl --failed

UNIT LOAD ACTIVE SUB DESCRIPTION

• rngd.service loaded failed failed Hardware RNG Entropy Gatherer Daemon

[…]

$

Other useful systemctl utility commands deal with obtaining, setting, and jumping between the system’s target. They are as follows:

■ get-default

■ set-default

■ isolate

As you may have guessed, you can set the system’s default target with super user privileges via the systemctl set-target command.

The isolate command is handy for jumping between system targets. When this command

is used along with a target name for an argument, all services and processes not enabled in the listed target are stopped. Any services and processes enabled and not running in the listed target are started. A snipped example is shown in Listing 5.19.

**Listing 5.19**: Jumping to a different target unit

# systemctl get-default

graphical.target

#

# systemctl isolate multi-user.target

#

# systemctl status graphical.target

[…]

Active: inactive (dead) since Thu 2018-09-13 16:57:00 EDT; 4min 24s ago

Docs: man:systemd.special(7)

Sep 13 16:54:41 localhost.localdomain systemd[1]: Reached target Graphical In...

Sep 13 16:54:41 localhost.localdomain systemd[1]: Starting Graphical Interface.

Sep 13 16:57:00 localhost.localdomain systemd[1]: Stopped target Graphical In...

Sep 13 16:57:00 localhost.localdomain systemd[1]: Stopping Graphical Interface.

[…]

#

In Listing 5.19, using super user privileges, the systemctl isolate command caused the system to jump from the default system target to the multi-user target. Unfortunately, there is no simple command to show your system’s current target in this case. However, the systemctl status command is useful. If you employ the command and give it the previous target’s name ( graphical.target in this case), you should see that it is no longer active and thus not the current system target. Notice that a short history of the graphical target’s starts and stops is also shown in the status display.(BTW The systemctl isolate command can be used only with certain targets. The target’s unit file must have the AllowIsolate=yes directive set.)

Two extra special targets are rescue and emergency. These targets, sometimes called modes, are described here:

**Rescue Target** When you jump your system to the rescue target, the system mounts all the local fi lesystems, only the root user is allowed to log into the system, networking services are turned off, and only a few other services are started. The systemctl is-system- running command will return the maintenance status. Running disk utilities to fix corrupted disks is a useful task in this particular target.

**Emergency Target** When your system goes into emergency mode, the system mounts only the root fi lesystem, and it mounts it as read-only. Similar to rescue mode, it only allows the root user to log into the system, networking services are turned off, and only a few other services are started. The systemctl is-system-running command will return the maintenance status. If your system goes into emergency mode by itself, there are serious problems. This target is used for situations where even rescue mode cannot be reached.

Listing 5.20: Jumping to the emergency target unit

# systemctl isolate emergency

Welcome to emergency mode!

#

# systemctl is-system-running

maintenance

#

# systemctl list-units --type=target 🡪 I like this one!!!

UNIT LOAD ACTIVE SUB DESCRIPTION

emergency.target loaded active active Emergency Mode

[…]

#

# systemctl default

#

**Using the SysV Initialization Process**

**Understanding Runlevels**

taBLe 5.12 Red Hat–based distribution SysVinit runlevels

Runlevel Description

0 Shut down the system.

1, s, or S Single-user mode used for system maintenance. (Similar to systemd

rescue target.)

2 Multi-user mode without networking services enabled.

3 Multi-user mode with networking services enabled.

4 Custom.

5 Multi-user mode with GUI available.

6 Reboot the system.

taBLe 5.13 Debian-based distribution SysVinit runlevels

Runlevel Description

0 Shut down the system.

1 Single-user mode used for system maintenance. (Similar to systemd

rescue target.)

2 Multi-user mode with GUI available.

6 Reboot the system.

SysVinit systems employ a configuration file, /etc/inittab .

**Listing 5.22**: The /etc/inittab file line that sets the default runlevel

# grep :initdefault: /etc/inittab

id:5:initdefault:

#

**Listing 5.23**: Listing script files in the /etc/init.d/ directory

# ls -1F /etc/init.d/

anacron\*

atd\*

[…]

yum-updatesd\*

#

These initialization scripts are responsible for starting, stopping, restarting, reloading, and displaying the status of various system services. The program that calls these initialization scripts is the rc script, and it can reside in either the /etc/init.d/ or the /etc/rc.d/ directory. The rc script runs the scripts in a particular directory. The directory picked depends on the desired runlevel. Each runlevel has its own subdirectory in the /etc/rc.d/ directory, as shown in Listing 5.24.

**Listing 5.24**: Runlevel subdirectories in the /etc/rc.d/ directory

# ls /etc/rc.d/

init.d rc0.d rc2.d rc4.d rc6.d rc.sysinit

rc rc1.d rc3.d rc5.d rc.local

#

**Listing 5.25**: Files in the /etc/rc.d/rc3.d directory

# ls -1F /etc/rc.d/rc3.d/

K01smolt@

K02avahi-dnsconfd@

K02NetworkManager@

[…]

K99readahead\_later@

S00microcode\_ctl@

S04readahead\_early@

Notice in Listing 5.25 that the script names start with either a K or an S , are followed by a number, and then their service name. The K stands for kill (stop), and the S stands for start. The number indicates the order in which this service should be stopped or started for that runlevel.

If you need to enact certain commands or run any scripts as soon as system initialization is completed, there is a file for that purpose. The /etc/ rc.local script allows you to add additional scripts and or commands. Just keep in mind that this script is not run until all the other SysVinit scripts have been executed.

To jump between runlevels on a SysVinit system, the basic syntax is as follows:

init Destination-Runlevel

telinit Destination-Runlevel

Listing 5.27: Jumping from runlevel 5 to runlevel 3

# runlevel

N 5

#

# init 3

#

# runlevel

5 3 (Note that the runlevel command is employed to show the previous and current runlevels.)

#

To view a SysVinit managed service’s status and control whether it is currently running,

use the service utility. This utility has the following basic syntax:

service SCRIPT COMMAND [ OPTIONS ]

The SCRIPT in the service utility refers to a particular service script within the /etc/init.d/ directory. The service utility executes the script, passing it the designated COMMAND.

taBLe 5.14 Commonly used service utility commands

Command Description

**restart** Stop and immediately restart the designated service. Note that if a designated service is not already started, a FAILED status will be generated on the stop attempt, and then the service will be started.

**start** Start the designated service.

**status** Display the designated service’s current status.

**stop** Stop the designated service. Note if a designated service is already stopped, a FAILED status will be generated on the stop attempt.

**reload** Load the service configuration file of the running designated service. This allows you to make service configuration changes without stopping the service. Note that if you attempt the reload command on a stopped service, a FAILED status will be generated.

**Stopping the System**

**halt** : Stops all processes and shuts down the CPU.

**poweroff** : Stops all processes, shuts down the CPU, and sends signals to the hardware to power down.

**reboot** : Stops all processes, shuts down the CPU, and then restarts the system.

**shutdown** : Stops all processes, shuts down the CPU, and sends signals to the hardware

to power down. (shutdown [ OPTIONS ...] TIME [ WALL-MESSAGE ])( After you’ve started a shutdown process, you can typically cancel it using the shutdown -c command)

For operating systems using Advanced Configuration and Power Interface (ACPI)–compliant chipsets, these are ACPI signals.

**Notifying the Users**

a Linux system offers the following additional utilities and fi les to help with communication:

■ /etc/issue : Contains text to be displayed on the tty terminal login screens (prior to logging into the system).

■ /etc/issue.net : Contains logon screen messages for remote logins.

■ /etc/motd : Called the Message of the Day file, contains text that is displayed after a user has logged into a tty terminal.

■ /bin/notify-send (or /usr/bin/notify-send ): Sends messages to a user employing the GUI but who is not logged into a tty terminal or does not have a GUI terminal emulator open.

■ /bin/wall (or /usr/bin/wall ): Sends messages (called wall messages ) to users logged

into a tty terminal or who have a GUI terminal emulator open and have their message status set to “yes.”

To prevent a wall message from being sent while using systemctl , include the --no-wall option in its command line.

**Listing 5.29**: Viewing your message status with the mesg command

$ mesg (the mesg command shows the current message status. You can issue the mesg y command to turn on messaging and mesg n to turn it off.)

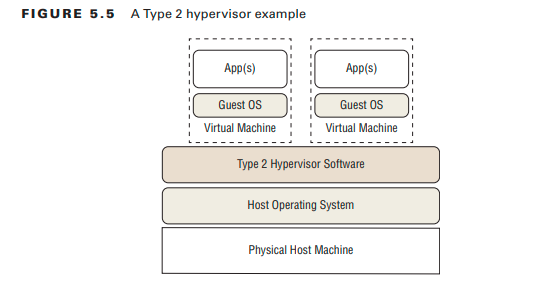
is y

$

**Virtualizing Linux**

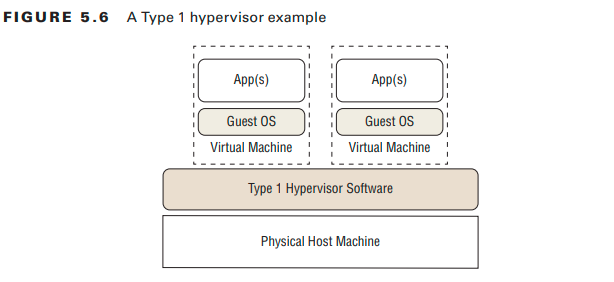
Managing VMs

The primary software tool used to create and manage VMs is a hypervisor. Hypervisors come in two basic flavors: Type 1 and Type 2. A Type 2 hypervisor is a software application that operates between its created virtual machine (guest) and the physical system (host) on which the hypervisor is running.



**A Type 2** hypervisor acts as a typical software application in that it interacts with the host’s operating system. However, its distinction lies in the fact that it provides one or more virtualized environments or virtual machines. These VMs each have their own operating system (guest OS) and can have various applications running on them. The host OS on the physical system can be completely different than the VM’s guest OS.

**A Type 1** hypervisor eliminates the need for the physical host’s OS. This software runs directly on the physical system and due to this is sometimes called a bare-metal hypervisor. A diagrammed example of a Type 1 hypervisor is shown in Figure 5.6.



A few options include KVM, Xen, and Hyper-V. An interesting feature with KVM and Hyper-V is that they can both be started while the host OS is running. These hypervisors then take over for the host OS and run as a Type 1 hypervisor. This is a case where the VMMs don’t neatly fi t into the Type 1 category.

**Creating a Virtual Machine**

A virtual machine is made up of either one file or a series of files that reside on the host machine. The file (or files) contains configuration information, such as how much RAM is needed, as well as the VM’s data, such as the guest OS and any installed application binaries Which methods you use depend on your organization’s needs as well as the number of VMs you must deploy.

**Clone** A clone is essentially a copy of another guest VM. The fi les that make up the original VM are copied to a new filesystem location, and the VM is given a new name. The following is a brief list of items that may need to be modifi ed for a Linux clone:

■ Host name

■ NIC MAC address

■ NIC IP address, if using a static IP

■ Machine ID

■ Any items employing a universally unique identifier (UUID)

■ Configuration settings on the clone that employ any item in this list

Your system’s machine ID is a unique hexadecimal 32-character identifier. The ID is stored in the /etc/machine-id file. D-Bus (covered in Chapter 3) will use this ID, if its own machine ID file, /var/lib/dbus/machine-id , does not exist. Typically on modern distributions, the D-Bus machine ID file will not exist or will be symbolically linked to the /etc/machine-id file. Problems can ensue if you clone a machine and boot it so that the two machines share the same ID. These problems may include not being able to get an IP address if your network manager is configured to use the machine’s ID instead of a NIC’s MAC address for DHCP services. To prevent this problem, after you clone a VM, you’ll need to address the duplicate machine ID. Typically, you can do this on the clone by performing the following steps:

1. Delete the machine ID fi le: rm /etc/machine-id

2. Delete the D-Bus ID fi le: rm /var/lib/dbus/machine-id

3. Regenerate the ID: dbus-uuidgen --ensure

Keep in mind that your distribution may require additional steps, such as linking the /var/lib/dbus/machine-id file to /etc/machine-id (soft links were covered in Chapter 4). Be sure to peruse your distro’s documentation prior to changing a machine’s identity.

**Open Virtualization Format (OVF)**

This standard allows the hypervisor to export a VM’s fi les into the OVF fi le format for use in other hypervisors It’s like cloning a machine between two different VMM software applications. While the OVF fi le standard creates multiple fi les, some hypervisors recognize a single compressed archive fi le of OVF fi les, called an Open Virtualization Archive (OVA). This is useful if you need to transfer a VM’s fi les across a network to a different host system.

**Template**

In virtualization, a VM template is a master copy. It is similar to a VM clone, except you cannot boot it. Virtual machines are created using these templates as their base. To create a template, you need a system image (sometimes called a VM image ). This image contains the guest OS, any installed applications, as well as confi guration and data fi les. The system image is created from a VM you have confi gured as your base system. Keep in mind that for a template-created VM you may need to modify items prior to booting it. The same list covered in the “Clone” section applies here.

If you have hypervisor software installed, most likely you can employ the virsh shell utility (not typically installed by default) to manage your VMs using shell scripts, which is convenient. You’ll need the libvirt library installed as well to support this utility.

**Integrating via Linux Extensions**

s, it’s important to check that your Linux host system will support virtualization and the hypervisor product you have chosen. First, you should determine if your system’s CPU has these hardware extensions

available. You can research this via the /proc/cpuinfo fi le’s fl ag information. Type grep ^flags /proc/cpuinfo to view the various enabled features of your server’s CPU. If enabled, you should see one of the following:

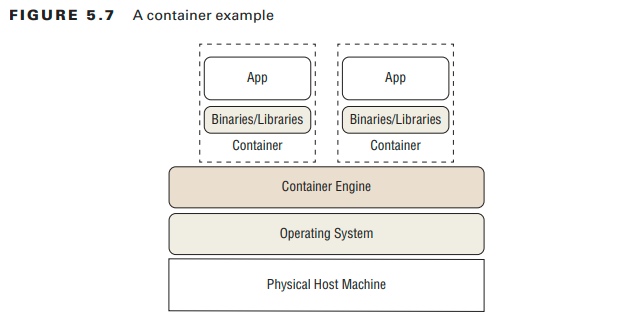
For Intel CPUs: vmx

For AMD CPUs: svm

You can check to see which hypervisor is being employed via the virt-what utility, which may or may not be installed on your Linux distro by default.

**Understanding Containers**

Whereas a VM provides an entire guest operating system, a container’s focus is typically on a single app, application stack, or environment. Instead of a hypervisor, a container is managed by a container engine. A diagrammed example of a container is 4shown in Figure 5.7 .



A container’s focus depends on its purpose in life. Two container focal points are:

**Application Containers**

These containers focus on a single application, or an application stack, such as a web server. Using containers in this way eliminates production and development environment differences and provides little to no downtime for app users.

**Operating System Containers.**

While containers are useful for developers, system admins can love them too. You can use a container that provides a fully functioning Linux OS space and is isolated from your host machine. You can also employ VMs for these different evaluations, but containers provide a faster-to-deploy and more lightweight test area. Docker is a very popular container engine for applications. In this particular software as well as other container engine programs (such as LXC, which is useful for testing workloads), you can employ system images to deploy several containers based on that image.

**Looking at Infrastructure as a Service**

For virtualization, companies turn to Infrastructure as a Service (IaaS) providers. With IaaS, the provider furnishes not only the hypervisor but the data center(s) as well servers, disks, and networking infrastructure. Your virtualized environment is reached over the Internet. A few of the more popular IaaS providers are Amazon Web Services (AWS), Google Cloud Platform’s Google Compute Engine, Digital Ocean, Rackspace, and Microsoft’s Azure. Each not only grants IaaS but also offers additional utilities allowing you to monitor, manage, and protect your virtualized environment. The name for providers that offer one or more of these services is typically cloud service provider (CSP). When using a cloud-based virtualized environment, you should know a few additional terms that will assist in selecting a CSP. They are as follows:

**Computing Instance** A computing instance , sometimes called a cloud instance , is a single virtual machine or container running on a cloud platform. When an instance is started, this is called provisioning an instance.

**Cloud Type** A CSP will offer public, private, and hybrid clouds. Public clouds are just as they sound—open to the public via the Internet or application programming interfaces (APIs). Access to their instances is controlled via virtual firewalls, gateways, and network

routers. These clouds reside solely on the CSP’s infrastructure.

Hybrid clouds are interesting in that they are typically a combination of instances on the

CSP’s infrastructure as well as instances or physical computers at a company’s local data

center. Hybrid clouds are popular with organizations who do not need all the features (or

the price) of a CSP’s cloud infrastructure.

**Elasticity** Elasticity allows an instance to automatically scale up or scale down, depending on current demand.

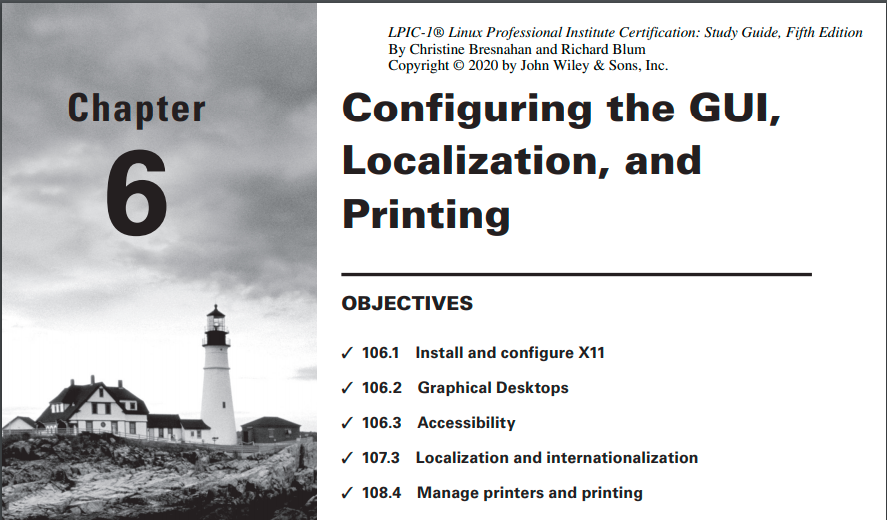
**Load Balancing** Load balancing occurs when a virtualized environment has multiple instances.

**Management Console or Portal** To set up your instances and choose the various CSP features, many providers offer a web browser–based graphical utility called a management console or portal.

**Block and Object Storage** Block storage is familiar to most system admins. Typically the underlying hardware is configured as disk drives in RAID configurations.

**Remote Instance** Access While the console or portal gives you the high-level view of your

various cloud instances, there are times you need to log directly into an instance. A preferred method employs using OpenSSH to access your instance’s IP address.



**Understanding the X11 Architecture**

The X Window System (X for short) is the display server used for Linux systems. Within the past few years, a new X display server package called Wayland has made headway in the Linux world.

**Examining X.Org**

The primary configuration file is /etc/X11/xorg.conf, though the file is sometimes stored in the /etc/ directory. Typically, however, this file is no longer used. Instead, individual applications or devices store their own X11 settings in separate files stored in the /etc/X11/xorg.conf.d directory. When the X11 server boots, it reads the configuration settings stored in those files to customize how it interacts with different display cards, monitors, keyboards, mice, and other input or output devices. These days, the X.Org software can detect most common hardware devices, so no manual configuration is required. However, in some cases, auto-detection might not work properly, and you need to make X11 configuration changes. In this case, you can manually create the configuration file. To do this, shut down the X Server by going to a command prompt, using the command sudo telinit 3 (this usually works on both SysVinit and systemd systems), and using super user privileges to generate the file via the Xorg -configure command. The file, named xorg.conf.new, will be in your local directory. Make any necessary tweaks, rename the file, move the file to its proper location, and restart the X server. The xorg.conf file has several sections. Each section contains important configuration information:

■ Input Device: Configures the session’s keyboard and mouse

■ Monitor: Sets the session’s monitor configuration

■ Modes: Defines video modes

■ Device : Configures the session’s video card(s)

■ Screen : Sets the session’s screen resolution and color depth

■ Module : Denotes any modules that need to be loaded

■ Files : Sets file path names, if needed, for fonts, modules, and keyboard layout files

■ Server Flags : Configures global X server options

■ Server Layout : Links together all the session’s input and output devices

In addition, two utilities are available that can help: xdpyinfo and xwininfo . The xdpyinfo command provides information about the X.Org server, including the different screen types available, the default communicate parameter values, protocol extension information, and so on. The xwininfo utility provides window in format on. If no options are given, an interactive utility asks you to click on the window for which you desire statistics. The displayed stats include location information, the window’s dimensions (width and height), color map ID, and so on. If your X.Org session hangs for any reason, you can reset it by pressing the Ctrl+Alt+Backspace key combination. This restarts the X server, which will attempt to auto-detect the hardware again and generate the configuration file.

The Wayland protocol was initially released back in 2009, and it is now used by many current Linux desktop environments such as GNOME Shell and KDE Plasma.

**Checking Your Display server**

1. Log in to your system’s GUI. This will start a GUI session for you.

2. Open a terminal emulator application.

3. Type echo $WAYLAND\_DISPLAY

4. You need to get the GUI session number, so type loginctl and press Enter. Note the session number.

5. Type the command loginctl show-session session-number -p Type at the command line, where session-number is the number you obtained in the previous step.

**Managing the GUI**

With some operating systems, your GUI is fairly rigid. You may be able to move or add a

few icons, change a background picture, or tweak a few settings.

**Standard GUI Features**

On Linux a GUI is a series of components that work together to provide the graphical setting for the UI. One of these components is the desktop environment. It is typically broken up into the following graphical sections and functions:

**Desktop Settings** Desktop settings consist of programs that allow you to make configuration changes to the desktop environment.

**Display Manager** The desktop environment’s login screen is where you choose a username and enter a password to gain system access.

**File Manager** This program allows you to perform file maintenance activities graphically.

Often a folder icon is shown for directories within the manager program.

**Icons** An icon is a picture representation of a file or program.

**Favorites Bar** This window area contains popular icons, which are typically used more

frequently.

**Launch** This program allows you to search for applications and files.

**Menus** These window areas are typically accessed via an icon.

**Panels** Panels are slim and typically rectangular areas located at the very top or bottom of a desktop environment’s main window. They can also be at the desktop’s far left or right.

**System Tray** A system tray is a special menu, commonly attached to a panel. It provides

access to programs that allow users to log out, lock their screen, manage audio settings,

view notifications, shut down or reboot the system, and so on.

**Widgets** Widgets are divided into applets, screenlets, desklets, and so on. They are programs that provide the user information or functionality on the desktop.

**Windows Manager** These client programs determine how the windows (also called

frames) are presented on the desktop.

**The X GUI Login System**

Every Linux display manager package uses the X Display Manager Control Protocol (XDMCP) to handle the graphical login process. The X Display Manager (XDM) package is the basic display manager software available for Linux. It presents a generic user ID and password login screen, passing the login attempt off to the Linux system for verification.

The main configuration file is /etc/X11/xdm/xdm-config. In most situations, you’ll never need to modify any of these settings. Here are a few of the more popular display managers

you’ll see:

■ KDM: The default display manager used by the KDE desktop environment

■ GDM: The default display manager used by the GNOME desktop environment

■ LightDM: A bare-bones display manager used in lightweight desktop environments

such as Xfce

**Getting to Know GNOME**

tabLE 6.1 GNOME Shell desktop environment default components

Name Program name and/or description

Display Manager GNOME Display Manager (GDM).

File Manager GNOME Files (sometimes just called Files). Formerly called Nautilus.

Favorites Bar GNOME Shell Dash (sometimes called the Dock).

Panels A single panel located at GNOME Shell frame’s top.

System Tray Located on the right-hand side of the single panel.

Windows Manager Mutter.

**Probing KDE Plasma**

tabLE 6.2 KDE Plasma desktop environment default components

Name Program name and/or description

Display Manager SDDM (Simple Desktop Display Manager)

File Manager Dolphin

Favorites Bar Displayed inside Application menu

Panels A single panel located at the Plasma frame’s bottom

System Tray Located on the right side of the single panel

Widgets Called Plasmoids

Windows Manager Kwin

**Considering Cinnamon**

sudo dnf groupinstall -y "Cinnamon Desktop"

**Making Acquaintance with MATE**

sudo apt-get install tasksel

sudo tasksel install ubuntu-mate-desktop

**Providing Accessibility**

It’s important to know the desktop environment configurations for these accommodations so that you can help to provide access for all.

**Using X11 for Remote Access**

Remote X11 Connections

The simplest way to run an X11 desktop remotely is to just forward the standard X11 desktop protocol stream across the network to the remote client. For example, assume the host workstation1 is located in a remote area but you need to run a graphical program from it using your local host, called workstation2 . Follow these steps to accomplish that:

1. Log in to the workstation2 host using the standard graphical desktop environment.

2. Open a terminal session to obtain a command prompt.

3. Type xhost +workstation1 , where workstation1 is either the hostname of the remote workstation1 host or its IP address. The xhost command allows the client workstation to receive data from the sending remote workstation.

4. Log in to the remote host workstation2 using a secure shell (SSH) connection. This will

provide a standard text command prompt interface.

5. At the command prompt, type export DISPLAY=workstation2 :0.0 , where orkstation2 is the hostname or IP address of your local workstation. This command redirects any graphical output generated on the remote workstation1 to the local workstation2 X server.

6. Launch a graphical application from the command prompt on workstation1. The graphical desktop will appear as a new window in your local workstation2 desktop.

7. When you’re done, close the launched program and type xhost -workstation1 on your local workstation2 host to remove the permissions to receive data from the remote host.

The X11 desktop uses TCP ports in the range of 6000 to 6063.

**Tunneling Your X11 Connection**

Another method that provides remote GUI interactions within a secure tunnel is X11 forwarding. This method is enacted using the openSSH service. First you need to check to see if X11 forwarding is permitted. This setting is in the openSSH confi guration fi le, /etc/ssh/sshd\_config . The directive X11Forwarding should be set to yes in the remote system’s confi guration fi le.

**Listing 6.1:** Checking the AllowTCPForwarding directive

# grep "X11Forwarding yes" /etc/ssh/sshd\_config

X11Forwarding yes

#

the command to use is ssh -X user@remote-host

In this section we will take a look at some of common remote desktop implementations

for Linux. They include VNC, Xrdp, NX, and SPICE.

**Viewing VNC Virtual network computing**

The VNC software is multiplatform and employs the Remote Frame Buffer (RFB) protocol. This protocol allows a user on the client side to send GUI commands, such as mouse clicks, to the server. The VNC server offers a GUI service at TCP port 5900 + n , where n equals the display number, usually 1 (port 5901). On the command line you point the VNC client (called a viewer) to the VNC server’s hostname and TCP port. The VNC server is flexible in that you can also use a Java-enabled web browser to access it. It provides that service at TCP port 5800 + n . HTML5 client web browsers are supported as well.

**Grasping Xrdp**

Xrdp is an alternative to VNC. It supports the Remote Desktop Protocol (RDP) and uses X11rdp or Xvnc to manage the GUI session. It allows access from several RDP client implementations, such as rdesktop, FreeFDP, and Microsoft Remote Desktop Connection.

Xrdp comes systemd-ready, so you can simply install, enable, and start the server using the systemctl commands. The package name on Linux is xrdp. After installing and starting the Xrdp server, adjust the firewall so that traffic can access the standard RDP port (TCP 3389). Now direct your RDP client choice to the server via its hostname or IP address, and if necessary, provide the client the RDP port number.

**Understanding Localization**

Localization is the ability to adapt a Linux system to a specific locale.

**Character Sets**

The most common ones you’ll run into (and the ones you’ll see on the LPIC-1 exam) are as follows:

ASCII: The American Standard Code for Information Interchange (ASCII) uses 7 bits to store characters found in the English language.

ISO-8859: The International Organization for Standardization (ISO) worked with the International Electrotechnical Commission (IEC) to produce a series of standard codes for handling international characters.

Unicode: The Unicode Consortium, composed of many computing industries company, created an international standard that uses a 3-byte code and can represent every character known to be in use in all countries of the world.

UTF: The Unicode Transformation Format (UTF) transforms the long Unicode values into either 1-byte (UTF-8) or 2-byte (UTF-16) simplified codes. For work in English-speaking countries, the UTF-8 character set is replacing ASCII as the standard.

**Environment Variables**

Linux stores locale information in a special set of environment variables (see Chapter 9).

Listing 6.2: The Linux locale environment variables

$ locale

LANG=en\_US.UTF-8

**Changing Your Locale**

Two methods are available that let you do that. You can manually set the LC\_ environment variables, or you can use the localectl command.

$ export LC\_MONETARY=en\_GB.UTF-8

Or

Instead of having to change all of the LC\_ environment variables individually, the LANG environment variable controls all of them at one place:

$ export LANG=en\_GB.UTF-8

**The localectl Command**

By default, the localectl command just displays the current

localization settings:

$ localectl

System Locale: LANG=en\_US.UTF-8

to set it:

$ localectl set-locale LANG=en\_GB.utf8

**Working with Time Zones**

Most Debian-based Linux systems defi ne the local time zone in the /etc/timezone fi le, whereas most Red Hat–based Linux systems use /etc/localtime.

To determine the current time zone setting for your Linux system, use the date command, with no options:

$ date

Fri Aug 2 05:52:29 EDT 2019

To change the time zone for a Linux system, copy or link the appropriate time zone template fi le from the /usr/share/zoneinfo folder to the /etc/timezone or /etc/ localtime location. The /usr/share/zoneinfo folder is divided into subfolders based on location. Each location folder may also be subdivided into more detailed location folders. Eventually, you’ll see a time zone template fi le associated with your specifi c time zone, such as /usr/share/zoneinfo/US/Eastern. If you don’t know the formal name of your time zone, run the tzselect command from the command prompt. It determines your timezone value based on answers to several location questions. Before you can copy the new time zone fi le, you’ll need to remove the original timezone or localtime file:

$ sudo mv /etc/localtime /etc/localtime.bak

$ sudo ln -s /usr/share/zoneinfo/US/Pacific /etc/localtime

$ date

Fri Aug 2 02:55:28 PDT 2019

$

The new time zone appears in the output from the date command.

**Setting the Time and Date**

$ date +"%A, %B %d %Y"

Friday, August 02 2019

$

Table 6.8 shows the various command sequences available in the date command.

Ta b le 6.8 The date format command sequences

%a The abbreviated weekday name

%A The full weekday name

%b The abbreviated month name

%B The full month name

%c The date and time

%C The century (e.g., 20)

%d The numeric day of month

%D The full numeric date

%e The day of month, space padded

%F The full date in SQL format (YYYY-MM-dd)

%g The last two digits of year of the ISO week number

%G The year of the ISO week number

%h An alias for %b

%H The hour in 24-hour format

%I The hour in 12-hour format%j The numeric day of year

%k The hour in 24-hour format, space padded

%l The hour in 12-hour format, space padded

%m The numeric month

%M The minute

%n A newline character

%N The nanoseconds

%p AM or PM

%P Lowercase am or pm

%r The full 12-hour clock time

%R The full 24-hour hour and minute

%s The seconds since 1970-01-01 00:00:00 UTC

%S The second

%t A tab character

%T The full time in hour:minute:second format

%u The numeric day of week; 1 is Monday

%U The numeric week number of year, starting on Sunday

%V The ISO week number

%w The numeric day of week; 0 is Sunday

%W The week number of year, starting on Monday

%x The locale’s date representation as month/day/year or day/month/year

%X The locale’s full time representation

%y The last two digits of the year

%Y The full year

%z The time zone in +hhmm format

%:z The time zone in +hh:mm format

%::z The time zone in +hh:mm:ss fotmat

%:::z The numeric time zone with: to necessary precision

%Z The alphabetic time zone abbreviation

$ timedatectl

Local time: Fri 2019-08-02 06:00:20 EDT

Universal time: Fri 2019-08-02 10:00:20 UTC

RTC time: Fri 2019-08-02 10:00:19

Time zone: US/Eastern (EDT, -0400)

System clock synchronized: no

systemd-timesyncd.service active: yes

RTC in local TZ: no

$

The timedatectl command provides one-stop shopping for all of the time information, including the hardware clock, called RTC; the date information; and the time zone information.

You can also use the timedatectl command to modify any of those settings as well by using the set-time option:

$ sudo timedatectl set-time "2019-08-02 06:15:00"

**Configuring Printing**

Fortunately, the Common Unix Printing System (CUPS) solves many of those problems for us. CUPS provide a common interface for working with any type of printer on your Linux system. The CUPS software uses the Ghostscript program to convert the PostScript document into a format understood by the different printers. The confi guration fi les are

stored in the /etc/cups directory. To defi ne a new printer on your Linux system, you can use the CUPS web interface. Open your browser and navigate to http://localhost:631/. Figure 6.13 shows the web interface used by CUPS. The CUPS web interface allows you to defi ne new printers, modify existing printers, and check on the status of print jobs sent to each printer. Not only does CUPS recognize directly connected printers, but you can also confi gure network printers using several standard network printing protocols, such as the Internet Printing Protocol (IPP) or the Microsoft Server Message Block (SMB) protocol. a few command-line tools are available that you can use for interacting with the printers and print queues:

■ cancel: Cancels a print request

■ cupsaccept: Enables queuing of print requests

■ cupsdisable: Disables the specified printer

■ cupsenable: Enables the specified printer

■ cupsreject: Rejects queuing of print requests

Besides the standard CUPS command-line commands, CUPS also accepts commands from the legacy BSD command-line printing utility:

■ lpc: Start, stop, or pause the print queue

■ lpq: Display the print queue status, along with any print jobs waiting in the queue

■ lpr: Submit a new print job to a print queue

■ lprm: Remove a specific print job from the print queue

**Listing 6.4**: Printing from the command line in Linux

$ lpq -P EPSON\_ET\_3750\_Series

EPSON\_ET\_3750\_Series is ready

no entries

$ lpr -P EPSON\_ET\_3750\_Series test.txt

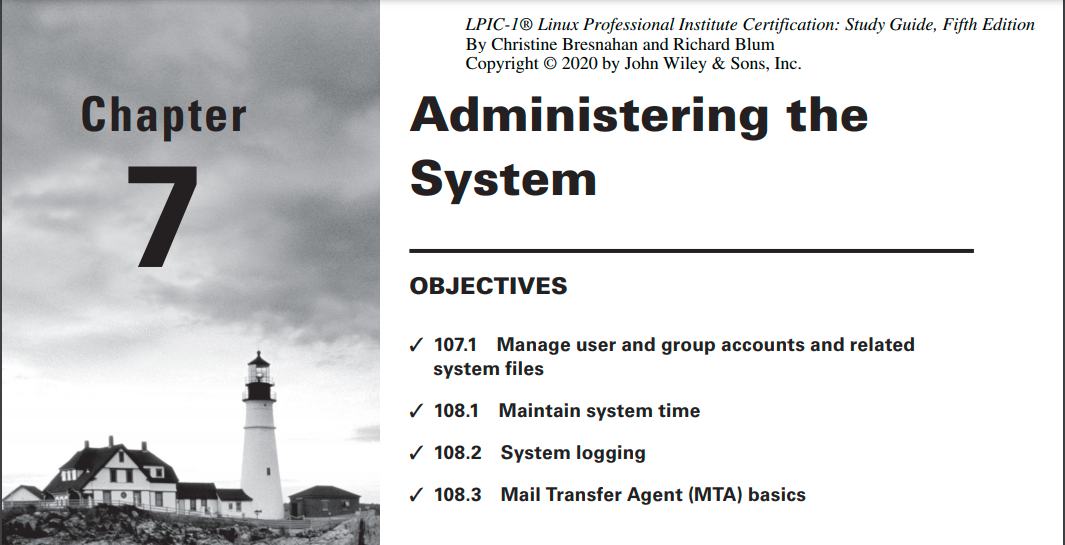
$ lpq -P EPSON\_ET\_3750\_Series

EPSON\_ET\_3750\_Series is ready and printing

Rank Owner Job File(s) Total Size

active rich 1 test.txt 1024 bytes

$



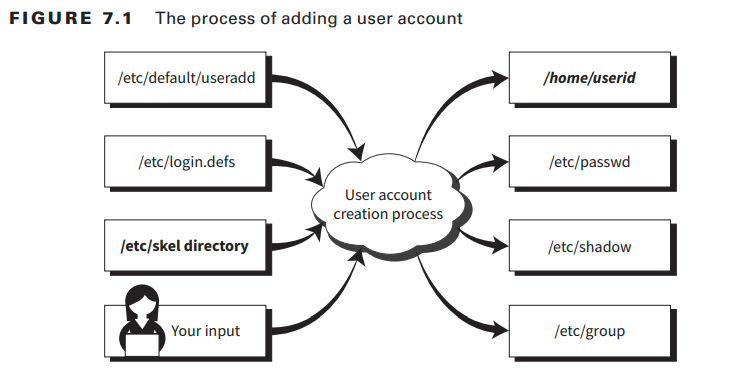
**Managing Users and Groupsauthentication**: is formerly defined as determining if a person or program is who they claim to be. Besides user authentication, you need to know how to check a user’s access to files, manage group memberships, and change passwords.

**Understanding Users and Groups**

DAC (discretionary access control) is the traditional Linux security control, where access to a file, or any object, is based on the user’s identity and current group membership.

**Configuring User Accounts**

To add a new user account on the system, the useradd utility is typically used.



**The /etc/login.defs File**

It contains directives for use in various shadow password suite commands. Shadow password suite is an umbrella term for commands dealing with account credentials, such as the useradd, userdel, and passwd commands.

**Listing 7.1:** Active directives in the /etc/login.defs configuration file

$ grep -v ^$ /etc/login.defs | grep -v ^\#

MAIL\_DIR /var/spool/mail

PASS\_MAX\_DAYS 99999

PASS\_MIN\_DAYS 0

PASS\_MIN\_LEN 5

PASS\_WARN\_AGE 7

UID\_MIN 1000

UID\_MAX 60000

SYS\_UID\_MIN 201

SYS\_UID\_MAX 999

GID\_MIN 1000

GID\_MAX 60000

SYS\_GID\_MIN 201

SYS\_GID\_MAX 999

CREATE\_HOME yes

UMASK 077

USERGROUPS\_ENAB yes

ENCRYPT\_METHOD SHA512

$

Ta ble 7.1 A few vital /etc/login.defs directives

Name Description

PASS\_MAX\_DAYS Number of days until a password change is required. This is

the password’s expiration date.

PASS\_MIN\_DAYS Number of days after a password is changed until the

password may be changed again.

PASS\_MIN\_LENGTH Minimum number of characters required in password.

PASS\_WARN\_AGE Number of days a warning is issued to the user prior to a

password’s expiration.

CREATE\_HOME Default is no. If set to yes, a user account home directory is

created.

ENCRYPT\_METHOD The method used to hash account passwords.

**The /etc/default/useradd File**

is another configuration file that directs the process of creating accounts.

**Listing 7.2**: The /etc/default/useradd configuration file

$ cat /etc/default/useradd

# useradd defaults file

GROUP=100

HOME=/home

INACTIVE=-1

EXPIRE=

SHELL=/bin/bash

SKEL=/etc/skel

CREATE\_MAIL\_SPOOL=yes

$

tAbLe 7. 2 A few vital /etc/default/useradd directives

Name Description

HOME Base directory for user account directories.

INACTIVE Number of days after a password has expired and has not been changed

until the account will be deactivated. See PASS\_MAX\_DAYS in Table 7.1 .

SKEL The skeleton directory.

SHELL User account default shell program.

**The /etc/skel/ Directory**

The /etc/skel directory, or the skeleton directory (see Table 7.2 ) as it is commonly called, holds fi les. If a home directory is created for a user, these fi les are to be copied to the user account’s home directory, when the account is created Therefore, if you make changes to the files later, you’ll have to migrate those changed files to current user accounts either by hand or by shell scripts.

**Listing 7.3:** Files in the /etc/skel directory

$ ls -a /etc/skel

. .. .bash\_logout .bash\_profile .bashrc .mozilla

$

**The /etc/passwd File**

Account information is stored in the /etc/passwd fi le. Each account’s data occupies a single line in the fi le.

tAbLe 7. 3 The /etc/passwd file’s record fields

Field No. Description

1 User account’s username.

2 Password field. Typically this file is no longer used to store passwords. An x in this field indicates passwords are stored in the /etc/shadow file.

3 User account’s user identification number (UID).

4 User account’s group identification number (GID).

5 Comment field. This field is optional. Traditionally it contains the user’s full name.

6 User account’s home directory.

7 User account’s default shell. If set to /sbin/nologin or /bin/false , then the user cannot interactively log into the system.

noticed that in an /etc/password record, fi eld #7 may contain either the /sbin/nologin or the /bin/false default shell. This is to prevent an account from interactively logging into the system. /sbin/nologin is typically set for system service account records. System services (daemons) do need to have system accounts, but they do not interactively log in. Instead, they run in the background under their own account name. If a malicious person attempted to interactively log in using the account (and they made it past other blockades, which you’ll learn about shortly), they are politely kicked off the system. Basically, /sbin/nologin displays a brief message and logs you off before you reach a command prompt. If desired, you can modify the message shown by creating the fi le /etc/ nologin.txt and adding the desired text. The /bin/false shell is a little more brutal. If this is set as a user account’s default shell, no messages are shown, and the user is just logged out of the system.