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.

-----103.1. Work on the command line

Description: Candidates should be able to interact with shells and commands using thd command line. The objective assumes the bash shell

Summery:

Escape sequence Function

\a Alert (bell)

\b Backspace

\c Suppress trailing newline (same as -n option)

\f Form feed (clear the screen on a video display)

\n New line

\r Carriage return

\t Horizontal tab

\*\*you can use \ to break a command in many line\*\*

\*\*if you run a command inside parentheses that command will be run inside a sub-shell and exec will run a command and closes the current shell\*\*

Environment variables:

Every variable has name and a value. echo the name with a $ in front of it:

USER :The name of the logged-in user.

UID : The numeric user id of the logged-in user.

HOME: The user's home directory.

PWD : The current working directory.

SHEL : The name of the shell.

$ : The process id (or PIDof the running bash shell (or other) process.

PPID : The process id of the process that started this process (that is, the id of theparent proces

? : The exit code of the last command.

Paths: External commands are just files on disks. So where does bash knows where to find command? By echo $paths command.

Page 62 is really hard

Commands:

Uname -a switch: gives you data about the system.

Pwd: says where are you right now (directory that you are at)

Directories:

Global bash configs are stored at /etc/profile and each user has her own config at ~/.profile& ~/.bash\_profile & ~/.bash\_logo

-----103.2. Process text streams using filter (bad edits)

Candidates should be able to apply filters to text stream

Summery:

Streams:

modern programming environments and shells use three standard I/O streams:

stdin is the standard input stream, which provides input to commands.

stdout is the standard output stream, which displays output from commands.

stderr is the standard error stream, which displays error output from commands

Commands: (ctrl + D will stop the input)

Piping ( | ): If you want to give the output of command1 as the input of command2, you should PIPE them as command1 | command2.

Redirection ( > ): Another useful way of controlling the streams is >. This help you to redirect your output(mostly to a file).

Cat: this command simply outputs its input stream.

Wc: is word count. -l switch with count lines.

head & tail: Shows the head (top) of a file or its tail (bottom). The default lines to show is 10 but you can specify with -20.

expand & unexpand & tr: Expand will replace the tabs in a stream with spaces;

Unexpand will do the reverse: The tr command translates A to 1, B to 2 and C to 3 in a stream you have to tr 'ABC' '123'. It is a pure filter so if you need to give it file to work on, you have to use cat.

-: You should know that if you put pipe (or keyboard stdin).- instead of a filename, the data will be replaced from the

jadi@funlife:~/w/lpic/101$ wc -l mydata | cat mydata - mydata #so here it will show mydata info then numbers of line in mydata then it will show mydata again.

Pr “file”: this formats text for classic printers.

Nl: Simply numbers lines.example: nl mydata | head -3 #here first 3 lines (of head) will be numbered.

Fmt: Will reformat a text file within margins (say 80 columns width or 60 if you use -w60).

sort & uniq: sort Will sorts its input(s).(by alpha-beta. If you want to sort by sort numerically put -n or -r to reverse ) and uniq removes duplicate entries from its input. Normal behaviour is removing only the duplicated lines but you can change the behaviour for example by giving -f1 to force it to not check fist field.

\*\*As you can see, the input HAVE TO BE sorted for uniq to work Page 76\*\*

Cut: cut command will cut a column of one file. Good for separating fields.

Paste: The paste command pastes lines from two or more files side-by-side!

Join: Our final field-manipulating command is join, which joins files based on a matching field. The files should be sorted on the join field.

Sed: sed is stream editor. It uses regular expressions and is great tool for replacing text. If you need to replace A with B only once in each line in a stream you have to say sed 's/A/B/'.

Split: Will split files.

split [OPTION]... [FILE [PREFIX]].

Mandatory arguments to long options are mandatory for short options too.   
 -a, --suffix-length=N   generate suffixes of length N (default 2)   
 -b, --bytes=SIZE        put SIZE bytes per output file   
 -C, --line-bytes=SIZE   put at most SIZE bytes of records per output file   
 -d  use numeric suffixes starting at 0, not alphabetic  --numeric-suffixes[=FROM]  same as -d, but allow setting the start value   
 -x use hex suffixes starting at 0, not alphabetic   
     --hex-suffixes[=FROM]  same as -x, but allow setting the start value   
 -e, --elide-empty-files  do not generate empty output files with '-n'   
     --filter=COMMAND    write to shell COMMAND; file name is $FILE   
 -l, --lines=NUMBER      put NUMBER lines/records per output file   
 -n, --number=CHUNKS     generate CHUNKS output files; see explanation below   
 -t, --separator=SEP     use SEP instead of newline as the record separator; '\0' (zero) specifies the NUL character   
 -u, --unbuffered        immediately copy input to output with '-n r/...'   
     --verbose           print a diagnostic just before each output file is opened.

Od: shows files in formats other than text). Normal behaviour is OctalDump. -t will tell what format to print (-t a for showing only named characters or -t c For showing escaped chars) -A for choosing how to show offsets (-A```Decimal,Octal,Hex or N````one) Od is very useful to find problems in your text files - say finding out if you are using tabs or correct line endings.

-----103.4. Use streams, pipes and redirects

Candidates should be able to redirect streams and connect them in order to efficiently process textual data. Tasks include redirecting standard input, standard output and standard error, piping the output of one command to the input of another command, using the output of one command as arguments to another command and sending output to both stdout and a file.

Summery:

Redirecting standard IO

* stdout is the standard output stream, which displays output from commands (file descriptor 1)
* stderr is the standard error stream, which displays error output from commands (file descriptor 2)
* stdin is the standard input stream, which provides input to commands (file descriptor 0)

What are these file descriptions? There are used to control the output. If you need to control where your output goes, you can add n> or n>>. n> redirects file description n to a file or device. If the file already exists it is overwritten and if it does not exists, it will be created.

n>> redirects file description n to a file or device. If the file already exists the stream will be appended to the end of it and if it does not exists, it will be created.

Redirecting both stdout and stderr to one location

Use &> and &>> to say both stderr and stdout.It is also possible to use &1 and &2 and &0 to refer to current place of stdout, stderr & stdin. In this case ls > file1 2>&1 means redirect output to file1 and output stderr to same place as stdout (file1) Be careful! ls 2>&1 > file1 means print stderr to current location of stdout (screen) and then change the stdout to file1.

\*\*Be careful! ls 2>&1 > file1 means print stderr to current location of stdout (screen) and then change the stdout to file1\*\*

here-documents:

Many shells, have here-documents (also called here-docs) as a way of input. You use << and a WORD and then whatever you input is considered stdin till you give only the WORD in one line. (very important when you use scripts and automated tasks)

Xargs:

This command reads input from stdin and uses them as arguments. You can limit the number of arguments with --max-args (same as -n) switch. Page 98 -l switch HARD

$ ls | xargs echo these are files:

these are files: errors f file1 fiona habib mahmoodrm minoo mojtaba.

Commands:

Tee : What if you need to see the output on screen and also save it to a file? Like: $ ls -1 | tee allfiles myfiles. -a switch to prevent from over writing.

Directories:

/dev/null : In linux, /dev/null is like a trash-can.

-----103.7 Search text files using regular expressions

Candidates should be able to manipulate files and text data using regular expressions. This objective includes creating simple regular expressions containing several notational elements. It also includes using regular expression tools to perform searches through a filesystem or file content.

Summery:

Regex:

Regex is a pattern to describe what we want to match from a text.

There is two kind of regex in GNU grep: Basic and Extended.

Repeating:

* The \* means repeating the previous character for 0 or more
* The + means repeating the previous character for 1 or more
* the ? means zero or one repeats
* {n,m} The item is matched at least n times, but not more than m times
* The | mean either this or that

Ranges:

There are easy ways to commonly used classes. Named classes open with [: and close with:].

* [:alnum:] Alphanumeric characters
* [:blank:] Space and tab characters
* [:digit:] The digits 0 through 9 (equivalent to 0-9)
* [:upper:] and [:lower:] Upper and lower case letters, respectively.
* ^ (negation) As the first character after [ in a character class negates the sense of the remaining characters
* A common form is .\* which matches any character (zero or any length).
* ^ means beginning of the string
* $ means the end of the string (page 113 for examples)

Grep switches:

-c just show the count

-v reverse the search

-n show line numbers

-l show only file names

-i case insensitive

Extra Lecture

3.6 Basic vs Extended Regular Expressions

In basic regular expressions the meta-characters ‘?’, ‘+’, ‘{’, ‘|’, ‘(’, and ‘)’ lose their special meaning; instead use the backslashed versions ‘\?’, ‘\+’, ‘\{’, ‘\|’, ‘\(’, and ‘\)’.

Traditional egrep did not support the ‘{’ meta-character, and some egrep implementations support ‘\{’ instead, so portable scripts should avoid ‘{’ in ‘grep -E’ patterns and should use ‘[{]’ to match a literal ‘{’.

GNU grep -E attempts to support traditional usage by assuming that ‘{’ is not special if it would be the start of an invalid interval specification. For example, the command ‘grep -E '{1'’ searches for the two-character string ‘{1’ instead of reporting a syntax error in the regular expression. POSIX allows this behavior as an extension, but portable scripts should avoid it. Fixed grep: If you need to search the exact string (and not interpret it as a regex), use grep -F Or fgrep so the fgrep this$ wont go for the end of the line and will find this$that too.

Sed: sed understands regex! If is good to use -r switch to tell sed that we are using them.

-----103.8 Perform basic file editing operations using vi

Candidates should be able to edit text files using vi. This objective includes vi navigation, basic vi modes, inserting, editing, deleting, copying and finding text.

Summery:

vi moded:

vi has 2 different modes:

Command mode: is where you go around the file, search, delete text, copy paste, replace, ... and give other commands to the vi. Some commands start with a : and some are only a keypress.

Insert mode: is where what you type, goes into the file at the cursors position.

If you want to go to the Command mode, press ESC key. There are several ways to go to the Insert mode from Command mode (including the 'i' key).

Moving the cursor

* H One character to the left (only current line)
* J One line down
* K One line up
* L One character to the right (only current line)
* W Next word on the current line
* E Next end of word on the current line
* B Previous beginning of the word on the current line
* Ctrl-f Scroll forward one page
* Ctrl-b Scroll backward one page

\*\*you can type a number before a command then the command will run as many as that number.\*\*

Jumping

* G With no number, will jump to the end & 10G will jump to line 10
* H 5H will go to the 5th line from the top of the screen
* L 3L will move the cursor to the 3rd line to the last line of the screen

Editing text

* I Enter the insert mode
* A Enter the insert mode after the current position of the cursor
* R replace only one character
* O open a new line below the cursor and go to the insert mode
* O open a new line above the cursor and go to the insert mode
* C clear to a location and go to the insert mode the replace till there and then clear to a location and go to the insert mode the replacenormal insert ( cw will overwrite the current word)
* D delete. you can mix with w (dw) to delete a word. Same as cw but dw does not to to the insert mode
* Dd Delete the current line
* X Delete character at the position of the cursor
* P Paste the last deleted text after the cursor
* P Paste the last deleted text before the cursor
* Xp swaps the character at the cursor position with the one on its right

Searching

* / Search forward (/happiness will find the next happiness)
* ? Search backward
* N repeat previous search.(You can also use / and ? without any parameters)

Exiting:

* :q! Quit editing without saving = runaway after any mistake
* Write the file (whether modified or not). Attempt to overwrite existing files or
* :w! read-only or other unwritable files
* :w Write to a new name myfile.txt
* ZZ Exit and save the file if modified
* :e! Reload the file from disk
* :! Run a shell command

Commands:

Vi “file”: best text editor ever. --version switch will tell you what version you are using.

101.1. Determine and configure hardware settings

Candidates should be able to determine and configure fundamental system hardware.

Summery:

HAL:

HAL is Hardware Abstraction Layer. It abstracts your hardware details for you.

Dbus:

A line like a bus that connects all parts of the OS to each other. dbus lets different parts of the system to communicate with each other.

Udev:

if you plug in any device, it will have a file in /dev. It has the protocol’s about how to use device to script it at /dev.

Hotplug:

Hot plug is when you insert a hardware into a running computer and cold plug is when you have to turn your computer off to install a hardware.

Commands:

These commands show list of modules and hardware’s on the system.

Lsmod: Shaows kernel modules.

Lspci: Shows PCI devices that are connected to the computer.

Lsusb: Shows all the USB devices connected to the system.

Lspcmcia: Shows available PCMCIA cards on this computer.

Lshal: Shows HAL data.

Lshw: Shows hardware. Test it!

sudo blkid -f: Device UUIDs

\*\*If you need to add a module to your kernel (say a new driver for a hardware) or remove it (uninstall a driver) you can use rmmod and modprobe.(or insmod is so old but it installs module) you can use -f switch to FORCE rmmod to remove the module even if it is in use.\*\*

Directions:

Dev: There are a lot of devices in /dev/ and if you plug in any device, it will have a file in /dev. udev lets you control what will be what in /dev./dev/

Sysfs:

The /sys directory is where HAL keeps its database of everything connected to the system. (we have the devices based on their technology) block bus class dev devices firmware fs hypervisor kernel module power./sys/

Proc:

This is where kernel keeps its settings and properties. This directory is created on ram and files might have write accessible. (The numbers are the process IDs) (cpuinfo, mounts, meminfo)./proc/Modules/

Modules:

If you need to load some modules every time your system boots do the method.

/etc/modules/ ((add names))

/etc/modprobe.d/ ((add configs))

101.2. Boot the system

Candidates should be able to guide the system through the booting process.

Summery:

BIOS:

BIOS is Basic Input Output System and does the first steps of the PC bootup. For example, is does a POST (Power on Self-Test) and decides which hardware should boot the system.

Kernel:

boot parameters supply the kernel (operation system) with information about hardware parameters that it might not determine on its own. Then OS (kernel) will start.

Commands:

pstree: The Tree of all Processes.

Directories:

Bootloader

Bootloader can be GRUB (1&2) or LILO

/etc/lilo.conf/boot/grub/grub.cfg , /boot/grub/menu.lst

Init

When the kernel finishes loading, it usually starts /sbin/init. This program remains running until the system is shut down. It is always assigned process ID 1.

/sbin/init , /etc/init

Upstart

jobs are defined here and the command is initctl list:

/etc/systemd/system

systemd

config files has unit type suffix (say cups.service or rpcbind.socket) and are located here- and the command is systemctl ))Dmesgdmesg command will show the full data from kernel ring buffer up to now. Butfile in below directory will show only the data during the boot

/var/log/dmesg

/var/log

101.3. Change runlevels and shutdown or reboot system

Candidates should be able to manage the runlevel of the system. This objective includes changing to single user mode, shutdown or rebooting the system. Candidates should be able to alert users before switching run level and properly terminate processes. This objective also includes setting the default run level. It also includes basic feature knowledge of potential replacements to init.

Summery:

Run levels:

This is the format: id:runlevels:action:process

* id: 2 or 3 chars
* runlevels: which runlevel these commands refers to (empty means all)
* action: respawn, wait, once, initdefault.
* Process: off course will be the command that should be done

default run level can be seen in this file which says init what to do, sets default runlevel and…being phased out!

* 0- Halt
* 1- Single user mode (recovery)
* 2- Debian/Ubuntu default
* 3- RHEL/Fedora/SUSE text mode
* 4- free
* 5- RHEL/Fedora/SUSE graphical mode
* 6- Reboot

Shutdown:

first sends a warning message to all logged-in users and blocks any further logins. It then signals init to switch runlevels. After 5 seconds(for waiting for them to save the progress), or another delay if specified, init sends a SIGKILL signal to forcibly end each remaining process.

* default is 5 seconds delay and then going to runlevel 1
* -h will halt the system
* -r will reboot the system
* time is hh:mm or n (minutes) or now
* whatever you add, will be broadcasted to logged in users
* if the command is running, ctrl+c or the "shutdown -c" will cancel it
* an example: shutdown -r 60 Reloading updated kernels

for more advance users:

* -t60 will delay 60 seconds between SIGTERM and SIGKILL
* if you cancel a shutdown, users won’t get the news! you can use "wall" command to tell
* them that the shutdown is canceled

upstart

understands events. Events are used to trigger tasks or services (jobs). Examples are connecting a usb or starting the Apache server only after having network and filesystem.

Systemd

uses sockets and a socket will be open for each daemon process but will start the daemon only when needed. Understands dependencies. Faster and parallel.

Commands:

runlevel: show your run level that you are in right now.

telinit: will change your run level.

initctl list: ups,tart INIT process list

systemctl: systemd INIT process list

Halt, poweroff, reboot: for doing just that.

Shutdown: for doing anything about shutting down or restart

Directions:

Upstart

jobs are defined in here and subdirectories./etc/init­/

Runlevels

runlevels are controlled from these directories.

/etc/inittab is being replaced by upstart and systemd but is still part of the exam.

grep "^id:" /etc/inittab #on init systems

/etc/init.d #all scrips are here

ls /etc/rc2.d/start/stop/on

102.1. Design hard disk layout

Description: Candidates should be able to design a disk partitioning scheme for a Linux

system.

Summery:

Primary, Extended & Logical Partition:

Linux numbers the primary partitions 1, 2, 3 & 4. If you define an extended partition, logical partitions inside it will be called 5, 6, 7. The newer GUID Partition Table (or GPT) solves this problem. If you format your disk with GTP you can have 128 primary partitions (no need to extended and logical)

LVM:

LVM helps you create one partition from different disks and add or remove space to them. The main concepts are:

* Physical Volume (pv): a whole drive or a partition. It is better to define partitions and not use whole disks - unpartitioned.
* Volume Groups (vg): this is the collection of one or more pvs. OS will see the vg as one big disk. PVs in one vg, can have different sizes or even be on different physical disks.
* Logical Volumes (lv): OS will see lvs as partitions. You can format an lv with your OS and use iyt.

Design Hard disk layout:

Swap: as extended memory around 8gb.(2gb)

Boot: as starting partition after BIOS about 500mb.(100mb)

/Root: well all other stuff gets in here.

A little more detail:

1. in a network station, /home can be mounted from a network drive (NFS, SMB, SSH, ..). This lets users to sit at any station, login and have their own home mounted from a network drive. Swap can be mounted from network or local.
2. And on servers. In many cases we separate the /var because logs and many other files are there and being updated and Some people also separate the /usr and write-protect it.

Commands:

Fdisk /dev/sda: this command starts the fdisk program so you do whatever you need with any hard disk. -P switch: show lot of good information about hard disk like usage or type or partitions.

Parted: does the same thing as fdisk does. With -p Switch But Note: parted does not understands GPT.

Unix directories:

Directory Description

bin Essential command binaries

boot Static files of the boot loader

dev Device files

etc Host-specific system configuration

home Home directory of the users

lib Essential shared libraries and kernel modules

media Mount point for removable media

mnt Mount point for mounting a filesystem temporarily

opt Add-on application software packages

root Home directory of the root user

sbin Essential system binaries

srv Data for services provided by this system

tmp Temporary files

usr Secondary hierarchy

var Variable data

102.2 Install a boot manager

Candidates should be able to select, install and configure a bootmanager.

Summery:

LILO:

LILO (LInux LOader) is the older of three main Linux boot loaders

/etc/lilo.conf (configuration)

/usr/sbin/liloconfig (generate an initial config)

When this config file is created, we have to issue the Bootable; lilo command to make the disk:

Lilo -v -v

Now you can boot your system from that floppy and you will get a menu to choose one of the boot options (at config file). some useful switches to use in lilo:

* -q show information about the map file. map file is located at /boot/map and contains the boot configs
* -R boot the system on the next reboot only. Used for remote systems
* -l list information about the Kernel
* -u uninstall lilo and restore previous boot record

\*pressing TAB when choosing an item in LILO menu, will let you edit that item.

GRUB (version 1 or the Legacy grub)

/boot/grub/grub.conf (config file)

/boot/grub/menu.lst (is a symbolic link to it)

After creating the configuration, you need to install the grub on a disk do this with one of these command forms: grub-install /dev/fd0

Commands:

Default: the default system to boot; starts from 0

Timeout: how long to wait before autobooting

Splashimage: background image

Password: Security is important! will ask this password

Title: Name of the entry.

Root: The partition to boot. Counting starts from 0. root(hd0,2) is the 3rd partition on the first disk.

Kernel: which kernel image should be loaded

Initrd: the name of the initial RAM disk. Modules needed by the kernel before the file system is mounted.

Savedefault: remember the last booted item

Chainloader: another file will act as stage 1 loader. Used for booting Windows systems.

Directions:

GRUB2

Its main config file is:

/boot/grub/grub.cfg

and can be created by

grub-mkconfig or /boot/grub/grub.cfg

Grub also is highly dependent on a core.img file in /boot/grub.

When you run grub-install /dev/sda, grub2 builds a core image, build a configuration file and install GRUB 2.

\*\*There is a command called update-grub as a frontend to grub-mkconfig which looks into the /etc/default/grub and creates a grub.cfg file.\*\*

102.3. Manage shared libraries

Candidates should be able to determine the shared libraries that executable programs depend on and install them when necessary.

Summery:

Linking:

if you need to read text from standard input, you need to link a library which provides this. linking has two forms:

Static linking:

is when you add this library to your executable program.

Dynamic linking(Shared)

is when you just say in your program "We need this and that library to run this program"(so all programs share the same library). This way your program is smaller but you need to install those libraries separately. This makes programs more secure (because libraries can be updated centrally by apt for example).

symbolic links for libraries:

All programs are getting daily updates and we want them to use latest library for them to function well, so what if i coded my program to use old version of library? Off course with symbolic links. So basically, this happens:

I will check the same lib on my system. First, I'll find where the libudev.so.1 is on my system:

# locate libudev.so.1

/lib/i386-linux-gnu/libudev.so.1

Now my program uses that address to find best library for itself but in that address, there is a link to very latest library that I need.

# ls -la /lib/i386-linux-gnu/libudev.so.1

lrwxrwxrwx 1 root root 16 Nov 13 23:05 /lib/i386-linux-gnu/libudev.so.1 -> libudev.

so.1.4.0 In general catch file will be like this and point just like below:

# ldconfig -p | head

1358 libs found in cache `/etc/ld.so.cache'

libzvbi.so.0 (libc6,x86-64) => /usr/lib/x86\_64-linux-gnu/libzvbi.so.0

libzvbi-chains.so.0 (libc6,x86-64) => /usr/lib/x86\_64-linux-gnu/libzvbi-chains.so.0

LD\_LIBRARY\_PATH (load library path of my own)

For some reason if you need to use your own lib file because:

* You are running an old software which needs an old version of a library.
* You are developing a shared library and want to test is without installing it
* You are running a specific program (say from opt) which needs to access its own libraries

A Colon (:) separated list of directories will tell your program where to search for needed libraries before checking the libraries in ld.so.cache. So how to do it?

Export:

LD\_LIBRARY\_PATH=/usr/lib/myoldlibs:/home/jadi/lpic/libs/

and then run any command or program now, the system will search /usr/lib/myoldlibs and then /home/jadi/lpic/libs/ before going to the main system libraries (defined in ld.so.cache).

Commands:

Ldd: If a program is dynamically or statically linked and What libraries a program need. The Ldconfig: process all these files to make the loading of libraries faster. This command creates ld.so.cache to locate files that are to be dynamically loaded and linked. -p switch prints all the catch data on screen.

Directions:

Libraries

/lib

/lib64

Dynamic library configs

/etc/ld.so.conf or /etc/ld.so.conf.d/ (all the links that help sys to be faster to access lib config files) ((all the library config files in system))

\*\*if you change the ld.so.conf (or sub-directories) you need to run ldconfig\*\*

102.4. Use Debian package management

Candidates should be able to perform package management using the Debian package tools.

Summery:

Concept of the package management system:

Most distributions have their own Package Manager for installing pre-build programs from defined repositories. Debian based distros use .deb files with apt, dpkg, aptitude and Fedora,RedHat, RHEL, SUSE, .. use RPM and Yum package managers.

Installing packages

apt-get resolved dependencies, it know what is needed to install this package and installs them.

package information with dpkg:

If you want to see what is inside a .deb file(working with .deb package files only)

Commands:

apt-get install “program”: install .deb program or it will update the program if it is already installed.-s switch make a simulated version of installation and will be removed after first use. --download-only switch will download the files without installing them and stores them at /var/cache/apt/archive/.

apt-get update: Updating sources information

Apt-get upgrade: upgrade all programs to latest version.

apt-get remove “program”: Removing debian packages(program)

\*\*removing a package will not remove its dependencies\*\*

Apt-get autoremove “program”: if you want to remove automatically installed dependencies you will use the very command Or you can use Apt-get autoremove solo so it removes unnecessary dependency files.

aptitude search “program”: searching for packages.

apt-get dist-upgrade: upgrades the distro.

apt-config : is a app that change apt configuration.

dpkg-reconfigure “program”: will undo whatever you have done at apt-config and make the default app config again.

Dpkg: another apt tool is dpkg. dpkg can install, remove, configure and query packages. -s “program” switch is for status for knowing is program installed successfully.

--contents switch shows what’s going on in .deb file (shows the code)

-P or --purge in apt or dpkg: will remove package and all dependency's.(In Not Good Way. All other apps that use same library cant use those library's any longer)

Dpkg -l switch: shows the files and directories a package(program) installed.

Dpkg -s “file” switch: will show which package(program) installed this file that i’m choosing.

Which “command”: for finding out what file will be used on any given command.

aptitude show or install or remove or search: (in internet and says what is it for or what is right spelling) “program”. It is graphical app too. Show is great help:Package: jcal,Version: 0.4.1-2build1,State: installed,Automatically installed: no ,Depends: libc6 (>= 2.4), libjalali0,Description: UNIX-cal-like tool to display Jalali calendar ,Jcal is a UNIX-cal-like tool to display Jalali (Persian) calendar, Homepage.

Directions:

Installing packages:

/var/cache/apt/archive/ (downloaded packages)

Note: as most other tools, you can configure the default configs at /etc/apt/apt.conf and there is a program apt-config for this purpose.

Package location

where these packages come from? from a Repository of different Repositories which are defined at bellow and files located at second line bellow

/etc/apt/sources.list

/etc/apt/sources.list.d/

Dpkg

dpkg can install, remove, configure and query packages and as always

/etc/dpkg/dpkg.cfg (config file)

/var/lib/dpkg (tree of all dpkg and config will use this directory)

102.5 Use RPM and YUM package management

Candidates should be able to perform package management using RPM and YUM tools.

Summery:

RedHat Package Manager (RPM) and Yellowdog Updater Modified (YUM) are fedora /redhat / rhel / centos / .. tools to manage packages.YUM adds extra features likes automatic updates, dependency management and works with repositories.Failed because of dependencies. RPM understands the dependencies but does not installs it itself. We need YUM and its just better. rpm needs a full file name and yum needs only the package name. How can we find these info?with command: rpm or yum info “app”.

So RPM can check the MD5 or SHA1 of files. The option is --checksig (-K)(and v for verbose)

rpm2cpio

The cpio is kind of an archive, just like zip or rar or tar. the rpm2cpio can convert rpm files tocpio archives so you can open them using cpio command.

The SUSE uses YaST and many modern desktops (KDE & Gnome)use PackageKit. Package Kit installs and updates packages on graphical interfaces on mostlinux systems (Debian, Fedora, Arch, ...)

Commands:

Yum -i(switch) “program”: for installing a app with dependency's.-y switch will answer yes aromatically when we’re installing apps. -e switch is for removing apps the interesting part is that it will show apps that use same dependency and cant use it any longer so you need to uninstall them(or just install dependency for them again).

Yum upgrade “\*a part of name of any app\*” and Yum update.

\*\*if using RPM, you can upgrade a system using -U or -F instead of -i. This is the difference:

-i is for install, -U is upgrade or install, -F is upgrade if installed. Note that -F wont install / upgrade the package if it is not already installed.\*\*

Yum list & rpm -q(querying ((info))) & yum info: will show information just like aptitude show “program”.

Yum search “program name or part of it”: will serach for the program in net and will give you all information about it.

Rpm --requires or -R switch “app” or yum deplist “app”: show apps

yumdownloader --resolve “app”: Downloads the app.

Cpio : opens .rpm file just like a rar or zip file. And rpm2cpio can convert any type of rpm file to cpio format so it gets to open.

\*\*it is also possible to find all installed packages with rpm -qa (query all). In most cases we pipe this with sort or grep and `less :

$ rpm -qa | grep vim

vim-minimal-7.4.027-2.fc20.x86\_64\*\*

Its hard for me right now Page 54.

Directories:

/etc/yum.repos.d/ (YUM dependencies and repository source)

103.3 Perform basic file management

Candidates should be able to use the basic Linux commands to manage files and directories.

Summery:

Wildcards and globbing:

* \* means any string
* ? means any single character
* [ABC] matches A, B & C
* [a-k] matches a, b, c, ..., k (both lower-case and capital)
* [0-9a-z] matches all digits and numbers
* [!x] means NOT X.

Compressing files:

Zip:

we mostly use gzip and gunzip in Linux. gzip creates the new compressed file with the same name but with .gz ending gzip removes the original files after creating the compressed file.

Bzip2:

is another compressing tool. Works just the same but with another compression algorithm.

Archiving files:

Tar: TapeARchive or tar is the most common archiving tool. In automatically create an archive file from a directory and all its subdirs.

* -cf myarchive.tar: create file named myarchive.tar
* -xf myarchive.tar: extract a file called myarchive.tar
* -z: compress the archive with gzip after creating it
* -b: compress the archive with bzip2 after creating it
* -v: verbose! print a lot of data about what you are doing
* -r: appeng new files to the current type available archive

Cpio: Gets a list of files and creates archive (one file) of it which can be opened later.

* -o: makes cpio to create an output from its input
* cpio does not goes into the folders.
* -i: is for extract

Dd: The Dd command copies data from one location to another.

Commands:

Ls: Is used to list directories & files.

* -l: more info for each file
* -1: will print one file per line
* -t: sorts based on date
* -r: reverses the search (so -tr is reverse time (newer files at the bottom).

cp source destination: copy

Mv: move

Rm: remove

* -f:(--force) will cause cp to try overwrite the target.
* -i: (--interactive) will ask Y/N question (deleting / overwriting).
* -b: (--backup) will make backups of overwritten files
* -p: will preserve the attributes.

Mkdir: creates directories.

* -p: will make inside in inside job.

Rmdir: will remove a directory.

* -p: will remove all directories inside too.

\*\*-r in any of commands will do recursive for example in rm -r all things inside of directory will be removed and -f will force it. \*\*

Touch: will make a file ((check the man for more switches))

Find: for searching on files with many wild-cards.

* -type f: will search for a regular file
* -type d: will search for a directory
* -type l: will search for a symbolic link
* -size “size”
* -Iname and -name: iname will search for any uppercase and lowercase but in name its matters.
* -mtime:
* -atime -6: file was last accessed less than 6\*24 hours ago
* -ctime +6: file was changed more than 6\*24 hours ago
* -mtime -6: file content moditication less than time is 6\*24 ago
* -mmin -90: file's data was last modified less than 90 minutes ago
* -amin, -cmin

Some interesting commands:

This will remove all empty files in this directory and its sub-directories:

find. -empty -exec rm '{}' \;

this will rename all htm files to hfml

find. -name "\*.htm" -exec mv '{}' '{}l' \;

File: Identify a file.

-----

103.5. Create, monitor and kill processes

Candidates should be able to perform basic process management.

Summery:

foreground and background jobs:

By adding & after a command we are adding the command to background and if we are in the middle of a command process by ctrl + z we are stopping the command and adding it to background, lets say its our first job then bg%1 will start the command 1 in background for us, and you can bring the command back to foreground with fg%1.

finding processes:

We can Grep on that and see who is running gedit and what is its process ID:

$ ps -ef | grep gedit

but there is also a more direct way:

$ ps -C gedit -o user,pid,tty,time,command

Commands:

Nohup: lets you run your commands even after you logged out and writes its output to nohup.out.

\*\*It is common to use 2> to redirect the nohup errors to a file:

nohup script.sh > mynohup.out 2>&1 &\*\*

Free -m switch: will show you info about the system memory.

Uptime: will show how long has system been up.

Kill: You can control processes by signals. (ctrl + z & c)

* 1(SIGHUP): Informing the process that its controlling terminal (like an (ssh connection) is terminated
* 15(SIGTERM): normal termination request
* 9(SIGKILL) forcefully kills the process.

Killall: Will send the given signal (or 15) to all the processes with the given name.

Ps: shows running processes on your computer.

Ps -aux: shows ALL processes on this system.

\*\*Every process has a ProcessID (PID) and a PPID (Parent Process ID). \*\*

103.6. Modify process execution priorities

Candidates should be able to manage process execution priorities.

Summery:

Prioritizing:

There is NI column (on top cpmmand), it shows how nice the process is. The nicer the process, the less CPU it asks. Nice can be from -20 to 19 (a process with nice = -20 is ANGRY and asking for a lot of CPU while a process with nice = 19 is SUPER NICE and lets other processes use most of the CPU and the default number for most of commands is 0).in order to give a command less than 0 priority we need to be root and If you run a command with nice without any parameters, the nice value will be 10.

Setting priorities when running commands:

If you need to change the niceness level of a program you can running it with Nice command and -n switch (for nice).

Commands:

Renice: will replace self-priority with priority given. ( renice -n -10 13605 )

104.1. Create partitions and filesystems

Candidates should be able to configure disk partitions and then create filesystems on media such as hard disks. This includes the handling of swap partitions.

Summery:

Blocked devices (any device that you can save anything on it):

In long ls format, the first b indicates Block Device:

$ ls -l /dev/loop1 /dev/sd[a-z]

brw-rw---- 1 root disk 7, 1 Jan 8 10:46 /dev/loop1

brw-rw---- 1 root disk 8, 0 Jan 8 10:46 /dev/sda

Fdisk:

Device Boot Start End Sectors Size Id Type

/dev/sda1 \* 2048 43094015 43091968 20.6G 83 Linux

/dev/sda2 43094016 92078390 48984375 23.4G 83 Linux

/dev/sda3 92080126 625141759 533061634 254.2G 5 Extended

/dev/sda5 92080128 107702271 15622144 7.5G Linux 82 swap/Solaris

/dev/sda6 107704320 625141759 517437440 246.8G 83 Linux

The Boot flag shows which partition starts the boot on DOS PCs and has no importance on LILO & GRUB.ID indicated the partition format (82 is swap, 83 is linux data, ..)

Commands:

Always start fdisk like this for more info

Fdisk -l: “hard you want to edit”.

Fdisk -l -m: “m will give you all help you need”.

-p switch: will show you all partitions.

Mkfs: You can format your partition. full list of installed on your system is here (/sbin/mk) (Mkswap is for making swap.) and -t switch is for typing the type of format.

105.1. Customize and use the shell environment

Candidates should be able to customize shell environments to meet users’ needs. Candidates should be able to modify global and user profiles.

Summery:

Environment variables:

login vs non-login shell :

when you login every time you need to do login so the apps work. But with non-login shell you can use apps without doing another login after the first one.

login shell:

This happens when you give your user and pass to enter a shell:

1. /etc/profile is run (one of the sys files {you should not change this file because it will be replaced after an update)
2. A line in /etc/profile runs whatever is in /etc/profile.d/\* Now the global profile is loaded and system will go for user specific profiles: (Profile.d is a place to add all powerfull codes (as Scrips .sh), so it will load when system starts and user log’s in.
3. /home/USERNAME/.bash\_profile (in 3 4 and 5 shell will check witch one can it run (is there any file as these 3 paths?)
4. /home/USERNAME/.bash\_login
5. /home/USERNAME/.profile
6. At the end if non of those files are available system will load: /home/USERNAME/.bashrc (BTW .bashrc is a place to add settings to users)

\*\* files that start with a dot (.) can’t be seen with normal ls command \*\*

Interactive (non-login) shell:

if you run a shell in an interactive mode (non-login) shell say from a GUI, only two things will be run:

1. /etc/bash.bashrc (or in some systems /etc/bashrc) {this is a place to check is shell is being interactive then it will start the customization with .bashrc}
2. /home/USERNAME/.bashrc

adding global configs for login shell:

you can add your global new config files int /etc/profile.d/ (with .sh at the end).

adding global configs for interactive/non-login shell:

you can use /etc/bash.bashrc file (some systems /etc/bashrc).

Lists:

Bash has arrays Too!

Commands:

Aliases: It is kind of a shortcut for codes. example: alias ll='ls -ltrh'

. (and source) :This is a shortcut for the bash source command. It runs the executable in front of it as part of the current environment not as a child shell in normal executables).

Functions: Just as normal codding you can do functions in bash too. Func (xorgs) {code part}

Set: allows you to change the values of shell options and set them. Using set we can configure how bash works. These are some samples:

-b Cause the status of terminated background jobs to be reported immediately

-e return in case a pipline, command, ... return non-zero

-n Read commands but do not execute them; this may be used to check a script for syntax errors.

-t Exit after reading and executing one command.

-C Prevent output redirection using ‘>’, ‘>&’, and ‘<>’ from overwriting existing files.

Export: when we are exporting something like a command, all the shell child’s can use it too.

Unset: This command has nothing to do with set command! This can unset(remove) the defined variables or functions.

Env: can set, remove or display variables or even run a command in a modified environment (changed by set): some examples

env [OPTION]... [NAME=VALUE] ... [COMMAND [ARGS]...]

and the Options are:

-u NAME --unset=NAME

That will Remove variable NAME from the environment

-I --ignore-environment

That will Start with an empty environment, ignoring the inherited

environment.

Directions:

/etc/skel

This directory contains files which will be used as a starting template for each new user.

.bash\_logout

runs when you logout from a login shell. It will remove the commands you just did so next user don’t see them.

105.2 Customize or write simple scripts

Candidates should be able to customize existing scripts, or write simple new Bash scripts.

Summery:

**Shell Scripts:**

Are a way of automating tasks.

**Shebang:** #!

In many cases we run shells with #!/bin/bash or #!/bin/sh

**Variables:**

You can define variables like this VARNAME=VALUE.

**Command substitution:**

Sometimes you need to have a variable with the result of something to a variable. In this case you can use $() construct: FILES=$(ls -1)

**executing scripts**

If the file is executable, we can run them using the ./script\_name.sh if we are in the same directory,or include their directory in $PATH variable.

Note: you know that for making a file executable we can do chmod 755 filename or

chmod +x filename.

**Conditions**

if [condition]

then

do something

do another thing

else

do new things

even funnier things

fi

Conditions can be TRUE or FALSE. A very simple conditions is if [ "Linux" = "Linux" ]

|  |  |  |
| --- | --- | --- |
|  | **conditions** | **what is means** |
|  | |  |
| "a" = "b" | | if two strings are equal (here it will return False) |
|  | |  |
| "a" != "b" | | string a *is not equal* to string b |
|  |  |  |
| 4 | -lt 40 | if 4 is *lower than* 40 (True) |
|  |  |  |
| 5 | -gt 15 | if 5 is *greater than* 15 (False) |
|  |  |  |
| 5 | -ge 5 | if 5 is *greater or equal* 5 |
|  |  |  |
| 5 | -le 3 | if 5 is *lower or equal* to 3 |
|  |  |  |
| 9 | -ne 2 | 9 is *not equal* with 2 (True) |
|  | |  |
| -f FILENAME | | if file FILENAME exists |
|  | |  |
| -s FILENAME | | if file exists and its size is more than 0 |
|  | |  |
| -x FILENAME | | file exists and is executable |
|  |  |  |

learn the syntax! Specially the *spaces* and *=* for checking if two strings are equal.

**Read:**

Using read we can read the user input. Look at this:

#!/bin/sh

echo "what is your name?"

read NAME

echo "Hello $NAME"

**for loop**

for VAR in SOME\_LIST;

do

some stuff with $VAR

some other stuff

done

**while loop**

while [condition]

do

do something

do anohter thing

done

**mailing the root user**

For sending mail, you need to install mailutils. Then the mail command will send emails.

If you need to send emails in a script, just do:

$ echo "Body!" | mail -s "Subject" root

105.3 SQL data management

Candidates should be able to query databases and manipulate data using basic SQL

commands. This objective includes performing queries involving joining of 2 tables and/or sub selects.

Summery:

**Databases:**

This module is about SQL language and MySQL is one of the many SQL databases

**mysql command line:**

You only need to know that mysql is a command line program to interactively connect to a mysql-server . I use it like this:

$ mysql -u root -p #(-u is for user and -p is for password)

**using a database:**

the use command to select which database you are going to use commands on.

Commands:

mysql> SHOW DATABASES; shows all the DBs.

Mysql> use “this db”: selects the very db.

mysql> SHOW TABLES;

mysql> SELECT \* FROM phonebook; choose \* from jadval.

mysql> SELECT \* FROM info WHERE mood = 'happy' AND weight >= 80; mood is soton in jadval so it will find happy soton and the vazn.

Like ‘sa\*%’ : will find the similar stuff in strings.

Limit “number” : will show the result till the number that we are giving it.

ORDER BY “soton or ST or desc(to Decrise)”

This is used if you want to **sort** the data based on one field.

GROUP BY:

This will *group* the output. گروه بندیش کن برا اساس یک چیز

مثلا وقتی میگیم گروه بندی کن افراد رو بر اساس mood آن ها تنها یک مورد از مود ها را میبینیم .

Select count(\*), mood from info group by mood:

تعداد را بر اساس مد میاورد مثلا 3 تا آدم شاد.

INSERT: It adds a new row to a table.

Like this: INSERT INTO phonebook (name, phone, email) VALUES ('ghasem', '+982112345678',''); these codes does not stop until we put; at the very end.

DELETE: mysql> DELETE FROM phonebook WHERE name = 'ghasem';

UPDATE: It updates (changes) row and again WHERE is usefull. So first you will choose the DB then you do it like this (with set):

UPDATE phonebook SET email='haale@lpic.fake' WHERE name = 'haale';

JOIN: with this one you can join two DBs to check them at the same time.(only to check and not to edit) and it will be like this:

SELECT \* FROM phonebook JOIN info ON phonebook.name = info.name;

Another example:

mysql> SELECT phonebook.name, phone, mood FROM phonebook JOIN info ON phonebook.name =info.name WHERE mood = 'happy';

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

| name | phone | mood |

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

| jadi | +9890something | happy |

| sina | +687randomnum | happy |

| nasrin | +9898989898 | happy |

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

3 rows in set (0.00 sec)

Quit ; mysql> quit , as simple as that, it will exit mysql.

106.1 Install and configure X11

Candidates should be able to install and configure X11.

Summery:

X:

The X Window System is a network transparent window system which runs on a wide range of computing and graphics machines; ( this represents the very first ghrapical servers, all of them starts or ends with X)

/etc/X11/xorg.conf is the directory of it that has many sections.

Section "Files" : was about the fontPath of the system. (in some very old pcs instead of putting the path for the font there was ip address so a server do the rendering part inorder that pc don’t do that ( because it coudent)

Section "Module" : was about, what are the modules am I suppose to load. (“glx” was the 3d options)

Section "InputDevice" : this part was all about every thing that a laptop should read the very first boot like keyboard, mouse or touchpad. Heres a example:

Section "InputDevice"

Identifier "Generic Keyboard"

Driver "kbd"

Option "CoreKeyboard"

Option "XkbRules" "xorg"

Option "XkbModel" "pc105"

Option "XkbLayout" "us"

EndSection

And another example for graphical driver:

Section "Device"

Identifier "ATI Technologies, Inc. Radeon Mobility 7500 (M7 LW)"

Driver "radeon"

BusID "PCI:1:0:0"

Option "DynamicClocks" "on"

Option "CRT2HSync" "30-80"

Option "CRT2VRefresh" "59-75"

Option "MetaModes" "1024x768 800x600 640x480 1024x768+1280x1024"

EndSection

\*\*Note: The vesa points to a low resolution, always working driver. It is used for troubleshooting. (so this driver was always be able to boot the graphical part\*\*

Section "Screen"

Identifier "Screen0"

Device "Screen0 ATI Technologies, Inc. Radeon Mobility 7500 (M7 LW)"

Monitor "Generic Monitor"

DefaultDepth 24 ----------------------------------------🡪 this is about colors

SubSection "Display"

Depth 1

Modes "1024x768"

EndSubSection

At the end we have to glue all of the above in one place as ServerLayout(and do all these so we can use the system) :

Section "ServerLayout"

Identifier "DefaultLayout"

Screen "Default Screen"

InputDevice "Generic Keyboard"

InputDevice "Configured Mouse"

InputDevice "Synaptics Touchpad"

EndSection

Commands:

Xwininfo: The xwininfo command is a window information utility for X. gives you some information about that *window* like its size, position, color depth, ... .

Xdpyinfo: (display info) This give you information about the running X session. Things like screens, color depth,version, name, ... (it has more details)

Xhost “ip address”: so this is just like ssh but the deffrent is that the ghraphical programs run on other pc but will see output on ower pc.

* + switch will: access control disabled, clients can connect from any host.
* switch will: access control enabled, only authorized clients can connect.

DISPLAY: This variable tell graphical program where to show their graphical output (where to draw their inputs). In normal cases this is set on my own machine:

$ echo $DISPLAY

:0

but if another X is listening to all IPs (after xhost + ) or listening to my machine (after xhost 192.168.42.85 ) I can change the DISPLAY environment and connect my graphical output to that machine.by doing this :

$ export DISPLAY=192.168.42.85:0---🡪 this 0 is our display as in last code.

$ xeyes # the eyes will be shown on 192.168.42.85 machine

Note: This wont work if you test it on a modern machine. Most X11s do not listen on any port these days.

106.2 Setup a display manager

Candidates should be able to describe the basic features and configuration of the LightDM display manager. This objective cover awareness of the display managers XDM (X Display Manger), GDM (Gnome Display Manager) and KDM (KDE Display Manager).

Summery:

Display Manager:

A Display Manager is a graphical interface which lets you login into your system when you turn your computer on. There are many different display managers (say XDM, SDDM, KDM,GDM, ...).

Lightdm:

Many of the distros use LightDM as their display/login manager.

It shows the default user (last logged in user) and asks for password.

/etc/lightdm All of the lightdm configs are in /etc/lightdm .

Some distributions like Ubuntu are using a lightdm.conf.d directory instead of a straight forward lightdm.conf and put their configs there.

\*\*Note: you can always : yum search greeter, to find out witch one do you like more.

greeter-session=lightdm-webkit2-greeter

the line above is most important one. You got to change this in order to apply the theme. And there is

user-session=gnome

that will choose the desktop theme.\*\*\*

controlling DMs:

The lightdm works as a service. You can start, stop & restart it or even use systemctl

disable lightdm to disable it on next boots.

106.3 Accessibility

Demonstrate knowledge and awareness of accessibility technologies.

Summery:

Linux is for everyone: ( even ppl that have disabilities)

1. AccessX : (to use keyboard and mouse)
2. Visual Settings help people with vision problems.
3. AssistiveTechnologies are things like text-to-speech (tts), there are 2 good tts: ocra and emacspeak.

107.1 Manage user and group accounts and related system files

Candidates should be able to add, remove, suspend and change user accounts.

Summery:

Changing password;

Each user can change her password using the passwd command. But only root can put weak passwords, it’s really easier for root to do these stuffs.

Users and groups:

You can define groups, give privileges to them and make users members of these groups. For example, there can be a "printer" group who has access to printings and you can add user "jadi" to this group.

* Each user can be a member of many different groups
* Each file belongs to one user and one group

Managing Users

Adding users:

Adding a user is done using the useradd command.

|  |  |  |
| --- | --- | --- |
| **switch** | **meaning** |  |
|  |  |  |
| -d | home directory (-d /home/user) |  |
|  |  |  |
| -m | create home directory |  |
|  |  |  |
| -s | specify shell |  |
|  |  |  |
| -G | add to additional groups |  |
|  |  |  |
| -c | comment. most of the time, users actual name. Use quotes if comments have |  |
| spaces or special characters in them |  |
|  |  |
|  |  |  |

When a new user directory is being created, the system will copy the contents of /etc/skel to their home dir. /etc/skel is used as a template for the home of users.

Modifying users:

It supports most of the useradd switches. For example, you can change *jadi*'s login shell by issuing usermod -s /bin/csh jadi. But there are 3 more switches:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **switch** |  | **meaning** | |  |
|  |  |  |  |  |
| -L | lock this account | | |  |
|  |  |  |  |  |
| -U | Unlock the account | | |  |
|  |  |  |  |  |
| -aG | add to more groups (say |  | ) |  |
| usermod -aG wheel jadi |  |
|  |  |  |  |  |

An example to chamge user name: usermod -c ‘alireza jan’Alireza.

Note: If you do usermod -G wheel,users jadi , jadi will be ONLY the member of these two groups. That is why we use -aG newgoup to ADD a new group to what jadi is a member of. -G is like saying "jadis groups are ..." and -aG is like "add this group to whatever groups jadi is a member of".

Deleting users:

userdel jadi (if you add -r switch too, the home direcoty and mail spool will be erased too!)

Managing Groups:

It is kind of same as users, you can do groupadd , groupdel and groupmod(to change or add id to group). Each group as an id a name.

# groupadd -g 1200 newgroup

adds a group called *newgroup* with id 1200.

\*\*Note: If root deletes a group with members, people won’t be deleted! They will just won’t be the members of that group anymore. \*\*

Directories:

Important files

/etc/passwd: This is the file which contains all the user names and their shells, etc, …

the format in this file is like this:

username:password:userid:primary group id:Name and comments:home dir:shell

In old days the password or the hashed password was actually shown in this file but

nowadays that is moved to the /etc/shadow file.These days if there is a x instead of password, it means *go look at the /etc/shadow* file.

\*\*Note: some commands are defined as a user but (like lightdm) in their shell, they are using /bin/false command in order that nobody can sign in as user with them (we can do this to normal users too , by adding (/sbin/nologin) to the user’s shell. these users have the ownership of some stuff like mail, BTW the password for these type of users in shadow is!! (that means when the password hash is being generated no password hash can be !!, at the end it means you can’t add any password. \*\*

/etc/shadow: This file contains password (hashed passwords) of the users/

is only readable for root and members of the shadow group:

# ls -ltrh /etc/passwd /etc/shadow

-rw-r--r-- 1 root root 1.9K Oct 28 15:47 /etc/passwd

-rw-r----- 1 root shadow 851 Oct 29 19:06 /etc/shadow

Note: we got to do tail command to see the stuff here.and ! means there is no password.

# tail /etc/shadow

scard:!:16369::::::

sshd:!:16369::::::

statd:!:16369::::::

tftp:!:16369::::::

uucp:\*:16369::::::

lightdm:\*:16369::::::

jadi:$6$enk5I3bv$uSQrRpen7m9xDapYLgwgh3P/71OLZUgj31n8AwzgIM2lA5Hc/BmRVAMC0eswdBGkseuXS

vmaz0lsYFtduvuqUo:16737:0:99999:7:::

svn:!:16736::::::

privoxy:!:16736::::::

\*\*users that are in group of wheel only can use sudo su\*\*

there are some numbers and encrypted passwords here.



|  |  |
| --- | --- |
| **filed** | **meaning** |

16737 When was the last time this password changes

1. User won’t be able to change the password 0 days after each change

99999 After this many days, the user HAVE to change his password

...and the user will be informed 7 days before the expiration to change his

7

password

Note: there numbers are "days after 1st of January 1970" or the Epoch time in days.For example 16737 means 16373 days after 1st Jan 1970. Strange but practical! We can always change these numbers with chage jadi.and these are the switches for the very command:

|  |  |  |
| --- | --- | --- |
| **switch** | **meaning** |  |
|  |  |  |
| -l | list information |  |
|  |  |  |
| -E | Set the expiration date. Date can be a number, in YYYY-MM-DD format or -1 |  |
| which will mean *never* |  |
|  |  |
|  |  |  |

/etc/group: This file contains the groups and their IDs.( # tail /etc/group)

checking user info :

# id jadi

uid=1000(jadi) gid=100(users) groups=1000(input),100(users)

\*\*remember changing this file by normal editing is wrong you Should use vi or special software to do so\*\*

Another solution is getent (توی فلان فلان چیز رو به من بده) (for get entry). It can query important *databases* for specific entries. These databases include /etc/passwd, /etc/hosts, /etc/shadow, /etc/group, ...

funlife:~ # getent group tor

tor:x:479:

funlife:~ # getent passwd jadi

jadi:x:1000:100:jadi:/home/jadi:/bin/bash

funlife:~ # getent shadow jadi

jadi:$6$enk5I3bv$uSQrRpen7m9xDapYLgwgh3P/71OLZUgj31n8AwzgIM2lA5Hc/BmRVAMC0eswdBGkseuXS

vmaz0lsYFtduvuqUo:16737:0:99999:7:::

107.2 Automate system administration tasks by scheduling jobs

Candidates should be able to use cron or anacron to run jobs at regular intervals and to use at to run jobs at a specific time.

Summery:

Crontab format:

Some part of a job of Linux admin is to do scrips or commands repeatedly. For example, when you log in terminal echo Hi!! To you or do a backup.

Crontab files are responsible to run some commands on specific intervals. Each line has 5 fileds to specify the run time and whatever after it is considered the command to be run.

The format is like this : A B C D E command and arguments

|  |  |  |
| --- | --- | --- |
| **filed** | **Meaning** | **values** |
|  |  |  |
| A | minute | 0-59 |
|  |  |  |
| B | hour | 0-23 |
|  |  |  |
| C | day of month | 1-31 |
|  |  |  |
| D | month | 1-12 (or names, see below) |
|  |  |  |
| E | day of week | 0-7 (0 or 7 is Sunday, or use names) |
|  |  |  |

Note : Also if you have @reboot or @daily instead of time fields, the command will be run once after the reboot or daily.

Example:

5 0 \* \* \* $HOME/bin/daily.job >> $HOME/tmp/out 2>&1 🡪 (we can add script at $ part too) 🡪 When a cron runs, the output will be emailed to the owner of the cron.

user specific crons:

Cron is a linux service. To see your crons you can use crontab -l (list) and for editing It you can use crontab -e to edit the file with vi. The files will be saved at /var/spool/cron/tabs/ or `/var/spool/crontabs'.

At:

what will happen if you needed to run a command only once? That’s when you going to use at command. An example:

$ at now + 1 min

warning: commands will be executed using /bin/sh

at> touch /tmp/at

at> <EOT>

job 3 at Thu Oct 29 22:12:00 2015

Note: If you want to check what is in the queue you can use atq and then try atrm 4 to delete job number 4.

system wide cron:

There is file called /etc/crontab. This looks like a normal user file opened with crontab -e but has one extra filed:

Format is like this: A B C D E USER command and arguments

This file should be edited with an editor directly

# cat /etc/crontab

SHELL=/bin/sh

PATH=/usr/bin:/usr/sbin:/sbin:/bin:/usr/lib/news/bin

MAILTO=root

#

# check scripts in cron.hourly, cron.daily, cron.weekly, and cron.monthly

#

-\*/15 \* \* \* \* root test -x /usr/lib/cron/run-crons && /usr/lib/cron/run-crons >/dev

/null 2>&1

Note: Have a look at first two line. It configures the shell which will run the commands and the PATH variable plus who will get the output emails.

System hourly, daliy, weekly, monthly, .. crons:

( کرون های ساعتی و ماهی و هفته ای میبایست اینطوری کار کنند و توی فایل زیر مورد خود کد میگیرند)

We have some system level crontab files in /etc/cron.d/ too. In other words, whatever file which is copied there, will be treated just like /etc/crontab file (a system wide cron file). This make systems much cleaner and lets programs to copy one file there instead of editing the /etc/crontab. So here system will guaranty, files will be checked and done in these Specific times but not certain times.

Anacron:

Anacron has a higher priorities than cron, so the objective (that we give it) Will be done For Sure,for example when pc is in halt mode, after the very first boot anacron will be activate and do the jobs.

Note: If you need to take a backup once a week it is better to use anacron instead of cron. anacron checks the timestamps at BOOT TIME and do not handle hourly crons.

controlling access using files:

There are also 4 more files

to control who can and can not use cron and at. The files are:

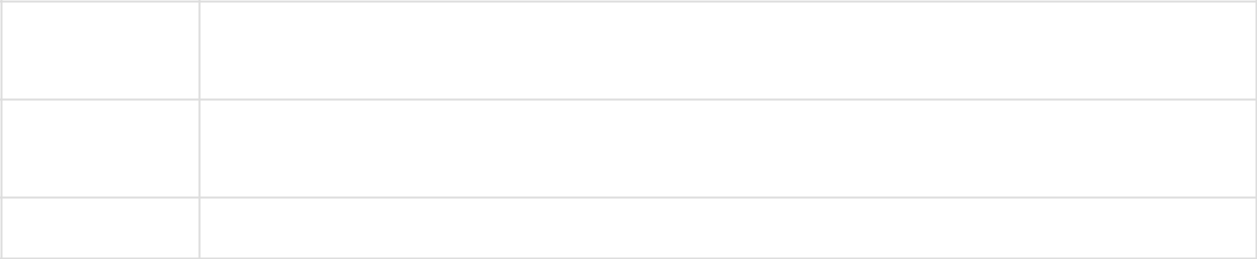
/etc/cron.allow

/etc/cron.deny

/etc/at.allow

/etc/at.deny

In most systems none of these files exist but if you create them, they will become active as follow:



**file**

**functionality**

**extension**

ONLY users mentioned in this file are allowed to run this service. All.

Allow other users will be denied

|  |  |
| --- | --- |
| .deny | Everybody can use the service except the users mentioned in this file |
|  |  |

107.3 Localisation and ternationalisation

Candidates should be able to localise a system in a different language than English. As well, an understanding of why LANG=C is useful when scripting.

Summery:

Timezone:

On linux systems you can use date and cal commands to check the date and the calendar. Example:

[jadi@funlife ~]$ date +'%Y%m%d-%H%M'

20160103-2239

Timezone determines what is your time difference comparing with a reference timezone. This way you can talk about times regardless from your location.

You can configure your timezone as always there is a command line way. The old one used to be **tzconfig** but it is not used anymore.

This process will suggest you to set a variable called TZ as follow to set *your own* time zone, but not the systems:

TZ='Asia/Tehran'; export TZ

(so we can add this line to our .profile so it stays)

Configuring timezone:

There is a directory at /usr/share/zoneinfo/ containing all the timezone info in binary. (its more like a interduce).

1. cat /etc/timezone (we can do manual change here with directory of last code)

Asia/Tehran

and there is a short link at this place:

1. # ls -ltrh /etc/localtime

-rw-r--r-- 1 root root 1.7K Jan 2 18:10 /etc/localtime

This file should be replaced by the correct file from usr/share/zoneinfo/. It is nicer to make a symbolic link rather than copying the actual file. This will prevent the conflicts during next upgrades.

usr/share/zoneinfo/) فایلی هست مربوط به زمان کل سیستم که بعد از انجام مراحل 1 و 2 و سپس انجام دستور زیر جهت ساختن یک لینک سافت و آنلینک کردن لوکال تایم استفاده میشود.)

1. Unlink localtime
2. Ln -s /usr/share/zoneinfo/asia/…/localtime

Configuring Languages:

You can check the status of current selected system language by issuing locale:

(استانداردهای استفاده شده را می آورد مثل نوع جدا کردن اعداد،نشان دادن زبان و زمان و ...)

$locale

LANG=en\_US.UTF-8 🡪زبان و کشور

LANGUAGE=

LC\_CTYPE="en\_US.UTF-8"

LC\_NUMERIC="en\_US.UTF-8"

LC\_TIME="en\_US.UTF-8"

LC\_COLLATE="en\_US.UTF-8"

LC\_MONETARY="en\_US.UTF-8"

LC\_MESSAGES="en\_US.UTF-8"

LC\_PAPER="en\_US.UTF-8"

LC\_NAME="en\_US.UTF-8"

LC\_ADDRESS="en\_US.UTF-8"

LC\_TELEPHONE="en\_US.UTF-8"

LC\_MEASUREMENT="en\_US.UTF-8"

LC\_IDENTIFICATION="en\_US.UTF-8"

LC\_ALL=

جهت تغییر همه این متغیرات طبق یک استاندارد :

Unset LC\_ALL

Export LC\_ALL = ‘en-gb UTF8’

اینجا استاندارد را قرار میدهیم.

اگر هم که میخواهید از دیفالت استفاده کنیم میبایست متغیر ال سی آل را حتما Unset کنیم.

LANG=C: (for scripting)

Another important point to know is the LANG=C settings. This shows two things:

1. All language settings will be default (en.US)

2. Binary sort order

It is also possible to do a LC\_ALL=C.

\*\*Note: in order to download missing language (or standard) this is a command for you: DPG-reconfigure “package”.

Character Encoding: (keyboard characters)

ACSII: would give us 128 characters

which was enough for numbers, punctuation and digits!

ISO-8859: It had more characters for Thai, Arabic and other languages.

UTF-8: (almost solved the problems) The Unicode Transformation Format characters not only for all written languages but also for fun characters like ¾, ♠, π and ⚤. It is backward compatible with the ASCII and uses 8 bit code units (not 8 bit coding!). In most cases it is a good idea to use UTF-8 and be sure that your system will work in practically all cases.

Iconv command: If you needed to convert coding to each other is you command. Example:

iconv -f WINDOWS-1258 -t UTF-8 /tmp/myfile.txt 🡪 -f=from,-t=to

timedatectl command: shows all important timezones.

108.1 Maintain system time

Candidates should be able to properly maintain the system time and synchronize the clock via NTP.

Summery:

How a computer keeps its time:

When the system boots, the OS reads this hardware time and it sets its system time to the hardware clock and uses it from there on. Hardware clock can be the localtime (your computers timezone) or UTC time (standard time). You can check this by /etc/adjtime. If we see UTC here it means our hardware is using global clock. The hwclock can be used to show the time based on the hwtime.

# date -s "Jan 4 22:22:22 2016" 🡪 (this is how to manually change local clock)

Mon Jan 4 22:22:22 IRST 2016

Older OSs used to set the hardware clock on localtime zone instead of timezone. This can be achived by:

# hwclock --localtime --set --date="01/05/2015 22:04:00"

If you want to fix it, just issue:

# hwclock -u -w

In this command -u tell the hardware clock that this is a UTC time and -w tells "sync with systemtime".

NTP Protocol:

uses NTP servers to find out the accurate time shown by best atomic clocks on this planet. One of the most famous severs used by ntp people is pool.ntp.org.

ntpdate: (usually get sets in cron)

The most straight forward command to set the systemclock is ntpdate “the clock site or ip”

After this, we need to set the hwclock by sudo hwclock -w.

Ntpd:

Instead of manually you can use a linux service called ntp to

keep your time using some time servers. (in cent os its ntpd) so jou got to install service then run it. Well you cant use both at same time.

root@funlife:~# ntpdate pool.ntp.org

4 Jan 22:14:25 ntpdate[18670]: the NTP socket is in use, exiting

As you can see, now the ntp is using the NTP port and ntpdate has problems starting up. Main configuration file of ntp is located at /etc/ntp.conf:

# cat /etc/ntp.conf

# /etc/ntp.conf, configuration for ntpd; see ntp.conf(5) for help

driftfile /var/lib/ntp/ntp.drift

# Enable this if you want statistics to be logged.

ntpq

The ntpq queries the ntp service. One famous switch is -p (for Print) that shows the pool we are using to sync the clock. In this output a \* means that the ntp is using this server as the main reference, + means that this is a good server and - shows an out of range server which will be neglected.

**108.2 System logging**

Candidates should be able to configure the syslog daemon. This objective also includes configuring the logging daemon to send log output to a central log server or accept log output as a central log server. Use of the systemd journal subsystem is covered. Also, awareness of rsyslog and syslog-ng as alternative logging systems is included.

**History:**

We will cover the **syslog** but most system have replaced it with **rsyslog** and **systemd journal**s. The logging in linux is orginized based on three concepts: facilities, priorities and actions.

به نوبت یعنی کی بوده و چی گفته و به کی گفته! مثال:

اینترنت در حال لوگ زدنه. اینترنت میگه من مشکل دارم در حد warning . حالا باید این مشکلم را به چه کسی بگم و چه کنم؟

* facilities: auth, user, kern, cron, daemon, mail, user, local1, local2, ...
* priorities:
* panic, emergخیلی سطح بالا
* alert (خطرناکه پیگیری شود)
* Crit(حیاتی)
* error, err (به ارور خورده ولی کار میکنه)
* Warn, warning (یک خطر پیش آمده)
* Notice (دقت کنید)
* Info (سطح پایین در حد اینکه فلانی وارد شد)
* Debug (طرف درخواست داد و من چک کردم)

priorities (میزان اهمیت) هر مرحله برای ارسال، مراحل بالاتر هم ارسال میکند.

On the **action** part we can have things like these:

|  |  |  |  |
| --- | --- | --- | --- |
| **action** | **sample** | **meaning** |  |
|  |  |  |  |
| filename | /usr/log/logins.log | will write the log to this file |  |
|  |  |  |  |
| username | jadi | will notify that person on the screen |  |
|  |  |  |  |
| @ip | @192.168.1.100 | will send this log to this log server and that log server |  |
| will decide what to do with it based on its configs |  |
|  |  |  |
|  |  |  |  |

At the end the whole thing will be shaped like this: kern.panic @ip Or \*.\* /var/log/messages

Note: If you need to log ONLY one specific level, add an equal sign (=) before the priority like this local3.=alert /var/log/user.alert.log .

It is important to know that the binary which logs the *\*kern* category is a standalone daemon. This daemon is called klogd and uses same configuration files. Why? so even after everything is crashed, klogd can log the kernel crashes.

**syslog and rsyslog:**

Most modern system use **rsyslog** instead of **syslog.** **The main configuration file in rsyslog is /etc/syslog.conf.**

\*.\*;auth,authpriv.none -/var/log/syslog

اینجا علامت – به معنی دخیره شده در رم است که بعدا در هارد ذخیره شود.

there is a /etc/rsyslog.d/ and it is better for different softwares and admins to add their specific configs there, instead of editing the main configuration file.

**creating rsyslog listener:**

If you need to start a rsylog listener and catch other systems log messages, it is enough to add an -r switch to rsyslog options. Just edit the /etc/default/rsyslog and change options from "" to "-r".and restart the daemon: # systemctl restart rsyslog.

**Journalctl:**