Experiment: 1a

Objective: to know the use of basic linux commands

1. a) Study of Unix/Linux general purpose utility command list

man, who, cat, cd, cp, ps, ls, mv, rm, mkdir, rmdir, echo, more, date, time, kill, history, chmod,chown, finger, pwd, cal, logout, shutdown.

man

Short for "manual," man allows a user to format and display the user manual built into

Linux distributions, which documents commands and other aspects of the system.

Syntax

man [option(s)] keyword(s)

Example

man ls

who: identifies the users currently logged in

The "who" command lets you display the users that are currently logged into your UNIX computer system. The following information is displayed: login name, workstation name, date and time of login. Entering who am i or who am I displays your login name, workstation name,

date and time you logged in.

Synopsys

who [OPTION]... [FILE | ARG1 ARG2]

Example

who am i

cat: concatenate or display files

Synopsys

cat [- q] [- s] [- S] [- u] [- n[- b]] [- v [- [- t]] [- | File ...]

The cat command reads each File parameter in sequence and writes it to standard output. If youdo not specify a file name, the cat command reads from standard input. You can also specify afile name of – (minus) for standard input.

cd:

The cd command, which stands for "change directory", changes the shell's current working

directory.

Syntax

cd directory

Example

cd new

cd .

cd ..

pwd:print name of current/working directory

Syntax: pwd [OPTION]...
example: pwd -L
options:-L, --logical use PWD from environment, even if it contains symlinks
-P, --physical avoid all symlinks
--help display this help and exit
--version output version information and exit

clear -clear the terminal screensyntax:clear

*history:*it displays commands executed by userssyntax: history
example:history

date - print or set the system date and timesyntax:
date [OPTION]... [+FORMAT]
date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
options:
-d, --date=STRING display time described by STRING, not `now'
-f, --file=DATEFILE like --date once for each line of DATEFILE
-r, --reference=FILE display the last modification time of FILE
examples

$ date --date='@2147483647':Convert seconds since the epoch (1970-01-01 UTC) to a date

$ TZ='America/Los_Angeles' date :Show the time on the west coast of the US (use tzselect(1) to find TZ)

$ date --date='TZ="America/Los_Angeles" 09:00 next Fri' :Show the local time for 9AM next Friday on the west coast of the US

time - run programs and summarize system resource usagesyntax: time [-apqvV] [-f FORMAT]
options:
[-o FILE]-o FILE, --output=FILE
Write the resource use statistics to FILE instead of to the standard error stream. By default, this overwrites the file,

destroying the file's previous contents. This option is useful

for collecting information on interactive programs and programs that produce output on the standard error stream.

-a, --append

Append the resource use information to the output file instead of overwriting

it. This option is only useful with the `-o' or `--output' option.

More

file perusal filter for crt viewing.more is a filter for paging through text onescreenful at a time. This version is especially primitive. Users should realize that less(1) provides more(1) emulation plus extensive enhancements.

syntax: more [-dlfpcsu] [-num] [+/pattern] [+linenum] [file ...]

options:

-d display help instead of ring bell
-f count logical, rather than screen lines
-l suppress pause after form feed
-p suppress scroll, clean screen and disblay text
-c suppress scroll, display text and clean line ends
-u suppress underlining
-s squeeze multiple blank lines into one
-NUM specify the number of lines per screenful
+NUM display file beginning from line number NUM
+/STRING display file beginning from search string match

-V output version information and exit

echo -display a line of text

Syntax:

echo [SHORT-OPTION]... [STRING]...
echo LONG-OPTION

options:

-n do not output the trailing newline
-e enable interpretation of backslash escapes
-E disable interpretation of backslash escapes (default)
--help display this help and exit
--version output version information and exit

rmdir -remove empty directories

Syntax: rmdir [OPTION]... DIRECTORY...

options:

-p, --parents

remove DIRECTORY and its ancestors; e.g., `rmdir -p a/b/c' is

similar to `rmdir a/b/c a/b a' -v, --verbose

output a diagnostic for every directory processed --help display this help and exit

mkdir - make directories

Syntax: mkdir [OPTION]... DIRECTORY...

options:

-m, --mode=MODE

set file mode (as in chmod), not a=rwx - umask -p, --parents

no error if existing, make parent directories as needed

-v, --verbose

print a message for each created directory

rm - remove files or directories

Syntax: rm [OPTION]... FILE...

options:

-f, --force

ignore nonexistent files, never prompt

-i prompt before every removal

-I prompt once before removing more than three files, or when

removing recursively. Less intrusive than -i, while still giv- ing protection against most mistakes

mv - move (rename) files

syntax: mv [OPTION]... SOURCE... DIRECTORY options;

-b like --backup but does not accept an argument -f, --force

do not prompt before overwriting -i, --interactive

prompt before overwrite

cal :displays a calendar and the date of Easter. If arguments are not specified, thecurrent month is displayed.

syntax: cal [-hjy] [-A number] [-B number] [[month] year] examples:
cal -hj 09 2015

cal -hjy

options:

-A number

Display the number of months after the current month. -B number

Display the number of months before the current month. -C Switch to cal mode.

-y Display a calendar for the specified year. -h Turns off highlighting of today.

-J Display Julian Calendar, if combined with the -e option, display date of Easter according to the Julian Calendar.

login, logout: - write utmp and wtmp entries

The utmp file records who is currently using the system. The wtmp file records all logins and logouts. See utmp(5).

The function login() takes the supplied struct utmp, ut, and writes it to both the utmp and the wtmp file.

The function logout() clears the entry in the utmp file again.

shutdown: bring the system down

syntax: shutdown [OPTION]... TIME [MESSAGE]

example:

shutdown -h 17:25
shutdown -r +5

Options:

-r Requests that the system be rebooted after it has been brought down.

-h Requests that the system be either halted or powered off after it has been brought down, with the choice as to which left up to the system.

-c Cancels a running shutdown. TIME is not specified with this option, the first argument iWrite a C program that illustrates how to executetwo commands concurrently with a command pipe. s MESSAGE.

1.b) Study of vi editor.

The vi editor is available on almost all Unix systems. vi can be used from any type of terminal because it does not depend on arrow keys and function keys--it uses the standard alphabetic keys for commands.

vi (pronounced "vee-eye") is short for "vi"sual editor. It displays a window into the file being edited that shows 24 lines of text. vi is a text editor, not a "what you see is what you get" word processor. vi lets you add, change, and delete text, but does not provide such formatting capabilities as centering lines or indenting paragraphs.

This help note explains the basics of vi:

opening and closing a file

moving around in a file

elementary editing

===== **Starting vi** =====

You may use vi to open an already existing file by typing

vi filename

where "filename" is the name of the existing file. If the file is not in your current directory, you must use the full pathname.

Or you may create a new file by typing *vi newname*

where "newname" is the name you wish to give the new file.

To open a new file called "testvi," enter *vi testvi*

On-screen, you will see blank lines, each with a tilde (~) at the left, and a line at the bottom giving the name and status of the new file:

~
~

"testvi" [New file]

===== vi Modes =====

vi has two modes:

command mode

insert mode

In command mode, the letters of the keyboard perform editing functions (like moving the cursor,deleting text, etc.). To enter command mode, press the escape &<Esc> key.
In insert mode, the letters you type form words and sentences. Unlike many word processors, vi
starts up in command mode.

===== **Entering Text** =====

In order to begin entering text in this empty file, you must change from command mode to insert mode. To do this, type *i'*

Nothing appears to change, but you are now in insert mode and can begin typing text. In

general, vi's commands do not display on the screen and do not require the Return key to be pressed.

Type a few short lines and press &<Return> at the end of each line. If you type a long line, you will notice the vi does not word wrap, it merely breaks the line unceremoniously at the edge of the screen. If you make a mistake, pressing <Backspace> or <Delete> may remove the error, depending on your terminal type.

===== **Moving the Cursor** =====

To move the cursor to another position, you must be in command mode. If you have just

finished typing text, you are still in insert mode. Go back to command mode by pressing <Esc>.

If you are not sure which mode you are in, press <Esc> once or twice until you hear a beep.

When you hear the beep, you are in command mode.

The cursor is controlled with four keys: h, j, k, l.

Key Cursor Movement

h left one space

j down one line

k up one line

l right one space

When you have gone as far as possible in one direction, the cursor stops moving and you hear a beep. For example, you cannot use l to move right and wrap around to the next line, you must use j to move down a line. See the section entitled "Moving Around in a File" for ways to move more quickly through a file.

Basic Editing

Editing commands require that you be command mode. Many of the editing commands have a different function depending on whether they are typed as upper- or lowercase. Often, editing commands can be preceded by a number to indicate a repetition of the command.

Deleting Characters

To delete a character from a file, move the cursor until it is on the incorrect letter, then type *_x'*
The character under the cursor disappears. To remove four characters (the one under the cursor and the next three) type 4x

To delete the character before the cursor, type X (uppercase)

Deleting Words

To delete a word, move the cursor to the first letter of the word, and type *dw*

This command deletes the word and the space following it. To delete three words type *3dw*

Deleting Lines

To delete a whole line, type *dd*
The cursor does not have to be at the beginning of the line. Typing dd deletes the entire line containing the cursor and places the cursor at the start of the next line. To delete two lines, type *2dd*. To delete from the cursor position to the end of the line, type *D* (uppercase)

Replacing Characters

To replace one character with another:

2: Move the cursor to the character to be replaced.

3: Type r

4: Type the replacement character.

The new character will appear, and you will still be in command mode.

Replacing Words

To replace one word with another, move to the start of the incorrect word and type *cw*

The last letter of the word to be replaced will turn into a $. You are now in insert mode and may type the replacement. The new text does not need to be the same length as the original.

Press <Esc> to get back to command mode. To replace three words, type *3cw*

Replacing Lines
To change text from the cursor position to the end of the line:

8: Type C (uppercase).

9: Type the replacement text.

10: Press <Esc>.

Inserting Text

To insert text in a line:

- Position the cursor where the new text should go.
- Type i
- Enter the new text. The text is inserted BEFORE the cursor.
- Press <Esc> to get back to command mode.

Appending Text

To add text to the end of a line:

- Position the cursor on the last letter of the line.
- Type a
- Enter the new text. This adds text AFTER the cursor.
- Press <Esc> to get back to command mode.

Opening a Blank Line

To insert a blank line below the current line, type *o* (lowercase) To insert a blank line above the current line, type *O* (uppercase)

Joining Lines

To join two lines together:

- Put the cursor on the first line to be joined.
- Type J

To join three lines together:

- Put the cursor on the first line to be joined.
- Type 3J

===== **Undoing** =====

To undo your most recent edit, type *u*

To undo all the edits on a single line, type *U* (uppercase)

Undoing all edits on a single line only works as long as the cursor stays on that line. Once you move the cursor off a line, you cannot use U to restore the line.

===== **Moving Around in a File** =====

There are shortcuts to move more quickly though a file. All these work in command mode.

Key Movement

w forward word by word b

backward word by word $ to end

of line

0 (zero) to beginning of line H to

top line of screen

M to middle line of screen L to

last line of screen

G to last line of file 1G to

first line of file

<Control>f scroll forward one screen

<Control>b scroll backward one screen

<Control>d scroll down one-half screen

<Control>u scroll up one-half screen

===== **Moving by Searching** =====

To move quickly by searching for text, while in command mode:

- Type / (slash).
- Enter the text to search for.
- Press <Return>.

The cursor moves to the first occurrence of that text.

To repeat the search in a forward direction, type *n*

To repeat the search in a backward direction, type *N*

===== **Closing and Saving a File** =====

With vi, you edit a copy of the file, rather than the original file. Changes are made to the original

only when you save your edits.

To save the file and quit vi, type **ZZ**

The vi editor is built on an earlier Unix text editor called ex. ex commands can be used within vi. ex commands begin with a : (colon) and end with a <Return>. The command is displayed on the status line as you type. Some ex commands are useful when saving and closing files.

To save the edits you have made, but leave vi running and your file open:

- Press <Esc>.
- Type :w
- Press <Return>.

To quit vi, and discard any changes your have made since last saving:

- Press <Esc>.
- Type :q!
- Press <Return>.

1. c) Study of Bash shell, Bourne shell and C shell in Unix/Linux operating system.

Types of Shells in Linux
In addition to graphical user interfaces like Gnome, KDE and MATE, the Linux operating system also offers several shells. These command-line interfaces provide powerful environments for software development and system maintenance. Though shells have many commands in common, each type has unique features. Over time, individual programmers come to prefer one type of shell over another; some develop new, enhanced shells based on previous ones. UNIX also has an ecosystem of different shells; Linux carries this practice into the open-source software arena.

The Bourne shell
The Bourne shell, called "sh," is one of the original shells, developed for Unix computers by Stephen Bourne at AT&T's Bell Labs in 1977. Its long history of use means many software developers are familiar with it. It offers features such as input and output redirection, shell scripting with string and integer variables, and condition testing and looping.

The Bash shell
The popularity of sh motivated programmers to develop a shell that was compatible with it, but with several enhancements. Linux systems still offer the sh shell, but "bash" -- the "Bourne-again Shell," based on sh -- has become the new default standard. One attractive feature of bash is its ability to run sh shell scripts unchanged. Shell scripts are complex sets of commands that automate programming and maintenance chores; being able to reuse

these scripts saves programmers time. Conveniences not present with the original Bourne shell include command completion and a command history.

C Shell

Developers have written large parts of the Linux operating system in the C and C++ languages.
Using C syntax as a model, Bill Joy at Berkeley University developed the "C-shell," csh, in 1978. Ken Greer, working at Carnegie-Mellon University, took csh concepts a step forward with a new shell, tcsh, which Linux systems now offer. Tcsh fixed problems in csh and added command completion, in which the shell makes educated "guesses" as you type, based on your system's directory structure and files. Tcsh does not run bash scripts, as the two have substantial differences.

The Korn shell
David Korn developed the Korn shell, or ksh, about the time tcsh was introduced. Ksh is compatible with sh and bash. Ksh improves on the Bourne shell by adding floating-point arithmetic, job control, and command aliasing and command completion. AT&T held proprietary rights to ksh until 2000, when it became open source.

1.d) Study of Unix/Linux file system (tree structure).
A file system is a logical collection of files on a partition or diskUNIX uses a hierarchical file system structure, much like an upside-down tree, with root (/) at the base of the file system and all other directories spreading from there.

A UNIX filesystem is a collection of files and directories that has the following properties −
It has a root directory (/) that contains other files and directories.
Each file or directory is uniquely identified by its name, the directory in which it resides, and a unique identifier, typically called an inode.

By convention, the root directory has an inode number of 2 and the lost+found directory has an inode number of 3. Inode numbers 0 and 1 are not used. File inode numbers can be seen by specifying the -i option to ls command.

It is self contained. There are no dependencies between one filesystem and any other.

The directories have specific purposes and generally hold the same types of information for easily locating files. Following are the directories that exist on the major versions of Unix −

Directory Description

/ This is the root directory which should contain only the directories needed at the top

level of the file structure.

/bin This is where the executable files are located. They are available to all user.

/dev These are device drivers.

/etc Supervisor directory commands, configuration files, disk configuration files, valid

user lists, groups, ethernet, hosts, where to send critical messages.

/lib Contains shared library files and sometimes other kernel-related files.

/boot Contains files for booting the system.

/home Contains the home directory for users and other accounts.

/mnt Used to mount other temporary file systems, such as cdrom and floppy for the CDROM drive and floppy diskette drive, respectively

/proc Contains all processes marked as a file by process number or other information that is dynamic to the system.

/tmp Holds temporary files used between system boots

/usr Used for miscellaneous purposes, or can be used by many users. Includes administrative commands, shared files, library files, and others

/var Typically contains variable-length files such as log and print files and any other type of file that may contain a variable amount of data

/sbin Contains binary (executable) files, usually for system administration. For example fdisk and ifconfig utlities.

/kernel Contains kernel files

1.e) Study of .bashrc, /etc/bashrc and Environment variables.

Following is the partial list of important environment variables.

Variable Description

DISPLAY Contains the identifier for the display that X11 programs should

use by default.

HOME Indicates the home directory of the current user: the default

argument for the cd built-in command.

IFS Indicates the Internal Field Separator that is used by the parser for

word splitting after expansion.

LANG LANG expands to the default system locale; LC_ALL can be used to

override this. For example, if its value is pt_BR, then the languageis set to (Brazilian) Portuguese and the locale to Brazil.

LD_LIBRARY_PATH On many Unix systems with a dynamic linker, contains acolonseparated list of directories that the dynamic linker should search for shared objects when building a process image after exec, before searching in any other directories.

PATH Indicates search path for commands. It is a colon-separated list of directories inwhich the shell looks for commands.

PWD Indicates the current working directory as set by the cd command.

RANDOM Generates a random integer between 0 and 32,767 each time it is

referenced.

SHLVL Increments by one each time an instance of bash is started. This

variable is useful for determining whether the built-in exit command ends the current session.

TERM Refers to the display type

TZ Refers to Time zone. It can take values like GMT, AST, etc.

UID Expands to the numeric user ID of the current user, initialized at

shell startup.