TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A LAB REPORT ON

Basic Linux Commands

SUBMITTED BY

Sushovan Shakya

THA075BEI046

SUBMITTED TO

Department of Electronics and Computer Engineering

26th August, 2021

BASIC LINUX COMMANDS

OBJECTIVE

- To understand basic Linux commands.
- To learn about shell script programming.
- To learn to compile and run C code in Linux terminal.

THEORY

There are many common Linux commands that will be helpful to you, if you ever even use the command line interface in Linux. Most average users just use the graphical user interface instead which usually has many tools and front-ends to Linux common commands. This Linux tutorial on command commands will help even the average user in case X server crashes, fails, is not properly configured, etc. So continue reading for some of the more common Linux bash commands.

Some of the more common Linux shell commands are listed below for more information on each command you can always run man [command] and this will bring up the manager for that command, you can also click on the commands listed for somecommon examples and syntax.

man [command] you will want to actually replace [command] with the shell command you want to read the man page for: man ls will give you the man page for the Linux shell command ls.

- linux ls command – is used to list files on the filesystem.
- file – command that will check the file type, this will output to you what the file type is no matter what the extension is.
- mkdir command – used to make directories on the filesystem.
- cd – is used for changing into a different directory in the Linux shell
- cp – is the Linux copy command, this shell command is used to copy files/directories from one location on the filesystem to another.
- mv – the Linux terminal command to move files/directories. Like the cp command, but deletes the original source
- rm – shell command in Linux to remove files/directories.

- Linux cat command- this command is used to print/view the contents of a file to the screen/terminal.
- grep – command used to search/find contents of a file and print/view on your terminal|screen.
- Linux more and less – commands that will allow you to read output of files, unlike cat that will output the entire file at once, even if it is too large for your terminal more and less will output only as many lines as the shell you are in can output, and allow you to scroll through the file contents.
- chown – Linux command to change ownership of a file/directory.
- Linux chmod – command that allows you to change mode of user access/permissions, basically set read, write, and execute permissions.
- Linux ps – lists the current running processes on your Linux system
- Linux kill and killall commands – used to kill/terminate running processes

Shell Commands and Shell script Programming

The term "shell" sometimes confuses beginners. When referred to by Linux users, the term "shell" means using the command line interface. It is important to know that using a shell is similar to using the DOS prompt. Linux also has a GUI (Graphical User Interface - pronounced gew-eee) that is similar to Windows. Using the shell is often frustrating for beginners because you must know the commands, what they do, and the proper way to enter them. This means using the correct syntax. The Linux OS is controlled by the kernel, which is the heart of the entire system. However, the kernel can only understand machine code. This is why a shell must be used. The shell interprets commands given by the user and translates them into machine code that thekernel can understand.

Types of Shells

There are several different types of shells available. Each shell has its own pro'sand con's, but each shell can perform the same basic tasks. The main difference between them is the prompt, and how they interpret commands. Below shows the most common shells and their attributes.

Bourne Shell: The original Bourne shell is named after its developer at Bell Labs, Steve Bourne. It was the first shell used for the UNIX operating system, and it has been largely surpassed in functionality by many of the more recent shells. However, all UNIXand many Linux versions allow users to switch to the original Bourne Shell, known simply as "sh".

C Shell: The C shell, as its name might imply, was designed to allow users to write shell script programs using syntax very similar to that of the C programming language. It is known as "csh."

TC Shell: TC shell is an expansion upon the C shell. It has all the same features, but adds the ability to use keystrokes from the Emacs word processor program to edit text on the command line. For example, users can press Esc-D to delete the rest of the highlighted word. It is also known as "tcsh."

Korn Shell: Korn Shell was also written by a developer at Bell Labs, David Korn. It attempts to merge the features of the C shell, TC shell and Bourne shell under one package. It also includes the ability for developers to create new shell commands as the need arises. It is known as "ksh."

Bourne-Again Shell: The Bourne-Again shell is an updated version of the original Bourne shell that was created by the Free Software Foundation for its open source GNU project. For this reason, it is a widely used shell in the open source community. Itssyntax is similar to that used by the Bourne shell, however it incorporates some of the more advanced features found in the C, TC and Korn shells. Among the added featuresthat Bourne lacked are the ability to complete file names by pressing the TAB key, the ability to remember a history of recent commands and the ability to run multiple programs in the background at once. It is known as "bash."

Script programming:

A shell script is a plain-text file that contains shell commands. It can be executed by typing its name into a shell, or by placing its name in another shell script. To be executable, a shell script file must meet some conditions: The file must have a special first line that names an appropriate command processor. The file must be made executable by changing its permission bits. A shell script file may optionally have an identifying suffix, like ".sh". This only helps the user remember which files are which. The command processor responsible for executing the file uses the executable bit, plusthe file's first line, to decide how to handle a shell script file.

Basic Linux commands

```
kachila@pop-os:~/Desktop/os_lab$ ls
'Lab 1'  'Lab 2'  'Lab 3'  'Lab 4'  'Lab 5'
kachila@pop-os:~/Desktop/os_lab$ cd 'Lab 1'
kachila@pop-os:~/Desktop/os_lab/Lab 1$ touch test.txt
kachila@pop-os:~/Desktop/os_lab/Lab 1$ nano test.txt
kachila@pop-os:~/Desktop/os_lab/Lab 1$ cat test.txt
Hello World!
kachila@pop-os:~/Desktop/os_lab/Lab 1$ mkdir test
kachila@pop-os:~/Desktop/os_lab/Lab 1$ ls
test  test.txt
```

```
kachila@pop-os:~/Desktop/os_lab/Lab 1$ file test.txt
test.txt: ASCII text
kachila@pop-os:~/Desktop/os_lab/Lab 1$ ps
    PID TTY          TIME CMD
   5620 pts/1    00:00:00 bash
  11966 pts/1    00:00:00 ps
kachila@pop-os:~/Desktop/os_lab/Lab 1$ kill
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or ki
ll -l [sigspec]
```

Compiling a C program through terminal:

```
kachila@pop-os:~/Desktop/os_lab/Lab 1$ gcc -o lab lab.c
kachila@pop-os:~/Desktop/os_lab/Lab 1$ ./lab
Hello World, once again!
```

Linux commands for creating shell script file and executing it:

```
#!/usr/bin/bash

# echo command
echo "Hello World!"

# string variables
name="kachila"
empty=""
echo "My name is $name $empty"
~
~
~
~
```

```
kachila@pop-os:~/Desktop/os_lab/Lab 1$ touch script1.sh
kachila@pop-os:~/Desktop/os_lab/Lab 1$ vim script1.sh
kachila@pop-os:~/Desktop/os_lab/Lab 1$ chmod a+x script1.sh
kachila@pop-os:~/Desktop/os_lab/Lab 1$ ./script1.sh
Hello World!
My name is kachila
```

```
#!/usr/bin/bash

read -p "What is your name? " some_name
echo "Hello $some_name, nice to meet you!"
~
~
~
```

```
kachila@pop-os:~/Desktop/os_lab/Lab 1$ vim script2.sh
kachila@pop-os:~/Desktop/os_lab/Lab 1$ chmod a+x script2.sh
kachila@pop-os:~/Desktop/os_lab/Lab 1$ ./script2.sh
What is your name? Sushovan
Hello Sushovan, nice to meet you!
kachila@pop-os:~/Desktop/os_lab/Lab 1$
```

```
echo "Performing arithmetic operations: "
echo "2+3 ="
expr 2 + 3
echo "20%3 = "
expr 20%3
echo expr '6 + 3'
~
~
~
```

```
kachila@pop-os:~/Desktop/os_lab/Lab 1$ vim script3.sh
kachila@pop-os:~/Desktop/os_lab/Lab 1$ chmod a+x script3.sh
kachila@pop-os:~/Desktop/os_lab/Lab 1$ ./script3.sh
Performing arithmetic operations:
2+3 =
5
20%3 =
20%3
expr 6 + 3
```

DISCUSSION AND CONCLUSION

In this section of the lab, we learned about some basic Linux commands. We learned how to navigate through a Linux system and to access different files in a Linux distro through the terminal. We also learned how to compile a C file using the Linux terminal and practiced some shell script programming commands. We could now edit different types of files through Linux terminal.

Thus the use of some basic Linux commands were understood.