

LAB 01

Summary

| Items | Description |
|--------------------------------|-----------------------------------|
| Course Title | Programming Fundamentals |
| Lab Title | SHELL COMMANDS |
| Duration | 3 Hours |
| Operating System/Tool/Language | Ubuntu/ g++/ C++ |
| Objective | To get familiar with the compiler |

COMMAND LINE UTILITIES

When Linus Torvalds introduced Linux and for a long time thereafter, Linux did not have a graphical user interface: It ran on character-based terminals only. Command line utilities are often faster, more powerful, or more complete than their GUI counter-parts. Sometimes there is no GUI counterpart to a text based utility. As it is a text-based interface so it consumes low RAM to run while GUI requires a high specification for running.

SHELL

A shell is simply a program, which is used to start other programs. It takes the commands from the keyboard and gives them to the operating system to perform the particular task. There are many different shells, but all derive several of their features from the Bourne shell, a standard shell developed at Bell Labs for early versions of Unix. Linux uses an enhanced version of the Bourne shell called bash or the "Bourne-again" shell. The bash shell is the default shell on most Linux distributions, and **/bin/sh** is normally a link to bash on a Linux system.

THE SHELL WINDOW

After logging in, open a shell window (often referred to as a **terminal**). The easiest way to do so from a GUI like Ubuntu's Unity is to open a terminal application, which starts a shell inside a new window. The window displays a prompt at the top that usually ends with a dollar sign \$. If # is the last character, it means you are running the command as root user.

BASIC COMMANDS

In this section we will have an insight of some basic commands. Different commands take multiple arguments and options (where option starts with a dash - sign).

1. Sudo

sudo (SuperUser DO) Linux command allows you to run programs or other commands with administrative privileges, just like “Run as administrator” in Windows.

2. Apt-Get

apt-get is the one of the most important Ubuntu commands every beginner must know. It is used to install, update, upgrade and remove any package. apt-get basically works on a database of available packages.

```
techlog360@techlog360: ~  
techlog360@techlog360:~$ sudo apt-get update  
[sudo] password for techlog360:  
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]  
Hit:2 http://archive.ubuntu.com/ubuntu bionic InRelease  
Get:3 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]  
Get:4 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [448 kB]  
Get:5 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]  
Get:6 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [678 kB]  
Get:7 http://archive.ubuntu.com/ubuntu bionic-updates/main Translation-en [249 kB]  
Get:8 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [969 kB]  
52% [8 Packages 9073 B/969 kB 1%] [4 Packages 288 kB/448 kB 64%]
```

3. -d

To operate the system on a specific date, you can change the date by using “-d”.

4. echo

The echo command prints its arguments to the standard output.

```
$ echo Hello World
```

Output Hello World on your terminal screen.

5. ls

The ls command lists the contents of a directory. The default is the current directory. Use ls -l for a detailed (long) listing where -l is an option. Output includes the owner of the file (column 3), the group (column 4), the file size (column 5), and the modification date/time (between column 5 and the filename).

6. touch

The touch command creates a file. If the file already exists, touch does not change it, but it does update the file’s modification time stamp.

```
$ touch file
```

7. cp

cp copies files. For example, to copy file1 to file2, enter this:

```
$ cp file1 file2
```

where file1 and file2 should be in current working directory.

```
$ cp hSourcei hdestinationi
```

where source and destination are full path from the root directory. To copy a number of files to a directory (folder) named dir, try this instead:

```
$ cp file1 ... fileN dir
```

8. cat

It simply outputs the contents of one or more files. The general syntax of the cat command is as follows:

```
$ cat file1 file2 ...
```

where file1 and file2 should be in current working directory or otherwise write down full path starting from root. When you run this command, cat prints the contents of file1, file2, and any other files that you specify (denoted by ...), and then exits. The command is called cat because it performs concatenation when it prints the contents of more than one file.

9. mv

It renames a file. For example, to rename file1 to file2, enter this:

```
$ mv file1 file2
```

You can also use mv to move a number of files to a different directory:

```
$ mv file1 ... fileN dir
```

10. rm

To delete (remove) a file, use rm. After a file is removed, it's gone from the system and generally cannot be undeleted.

```
$ rm file
```



11. pwd

The pwd (print working directory) program simply outputs the name of the current working directory.

12. date

Displays current time and date

13. clear

Clears the terminal screen.

14. exit

Exit the Shell

15. mkdir

The mkdir command creates a new directory dir:

```
$ mkdir dir
```

16. rmdir

The rmdir command removes the directory dir:

```
$ rmdir dir
```

If dir isn't empty, this command fails. `rm -rf dir` is used to delete a directory and its contents where `-r` option specifies recursive delete to repeatedly delete everything inside dir, and `-f` forces the delete operation.

17. cd

The current working directory is the directory that a process (such as shell) is currently in. The `cd` command changes the shell's current working directory:

```
$ cd dir
```

If you omit dir, the shell returns to your home directory, the directory you started in when you first logged in.

18. man

Linux systems come with a wealth of documentation. For basic commands, the manual pages (or man pages) will tell you what you need to know. For example, to see the manual page for the `ls` command, run `man` as follows:

\$ man ls

OUTPUT REDIRECTION

To send the output of command to a file instead of the terminal, use the > redirection character:

\$ command > file

The shell creates file if it does not already exist. If file exists, the shell erases the original file first. You can append the output to the file instead of overwriting it with the » redirection syntax:

\$ command » file

NAVIGATING DIRECTORIES

Unix has a directory hierarchy that starts at /, sometimes called the root directory. The //;;;directory separator is the slash (/).

When you refer to a file or directory, you specify a path or pathname. There are two different ways of writing files path:

1- Absolute Path

When a path starts with root, it is a full or absolute path.

/home/cslab/Desktop/Lab01

2- Relative Path

A path starts at current directory is called a relative path.

./Lab01

DIRECTORY REFERENCES

1. Home Directory (~)

2. Root Directory(/)

3. Current Working Directory(.)

One dot (.) refers to the current directory; for example, if you're in /usr/lib, the path dot(.) is still /usr/lib, and ./X11 is /usr/lib/X11.

4. Parent Directory(..)

A path component identified by two dots (..) specifies the parent of a directory. For example, if you're working in /usr/lib, the path .. would refer to /usr. Similarly, ../bin would refer to /usr/bin.

FILE MODES AND PERMISSIONS

Every Linux file has a set of permissions that determine whether you can read, write, or run the file. Running `ls -l` displays the permissions. The file's mode represents the file's permissions and some extra information. There are four parts to the mode, as illustrated in Figure1.

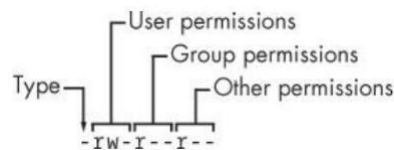


Figure 1: File Mode

The first character of the mode is the file type. A dash (-) in this position, as in the example, denotes a regular file, meaning that there's nothing special about the file. This is by far the most common kind of file. Directories are also common and are indicated by a d in the file type slot. The rest of a file's mode contains the permissions, which break down into three sets: user, group, and other, in that order. For example, the rw-characters in the example are the user permissions, the r-characters that follow are the group permissions, and the final r- characters are the other permissions.

| File Mode | Explanation |
|-----------|--|
| r | Means that the file is readable. |
| w | Means that the file is writable. |
| x | Means that the file is executable (you can run it as a program). |
| - | Means nothing |

The user permissions (the first set) pertain to the user who owns the file. The second set, group permissions, are for the file's group (somegroup in the example). Any user in that group can take advantage of these permissions. Everyone else on the system has access according to the third set, the other permissions.

MODIFYING PERMISSIONS

To execute a script first we will make it executable. To change permissions, use the chmod command. Only the owner of a file can change the permissions. There are two ways to write the chmod command:

1. `chmod {a,u,g,o}{+,-}{r,w,x}`

{all, user, group, or other}, {read, write, and execute} '+' means add permission and '-' means remove permission.

For example, to add group (g) and others (o) read (r) permissions to file, you could run these two commands:

```
$ chmod g+r file
```

```
$ chmod o+r file
```

Or you could do it all in one shot:

```
$ chmod go+r file
```

To remove these permissions, use go-r instead of go+r.

2. `chmod h3DigitNumberi`

Every digit sets permission for the owner(user), group and others as shown in Table 2. To set the permission decipher the number first and then execute the command as

```
$ chmod 700 f
```

| User | | | Group | | | Other | | | 3 Digit No. | Description |
|------|---|---|-------|---|---|-------|---|---|-------------|---|
| r | W | x | r | w | x | r | w | x | | |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 400 | user: read |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 040 | group: read |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 711 | user: read/write/execute; group, other: execute |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 644 | user: read/write; group, other: read |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 600 | user: read/write; group, other: none |

Examples of chmod with 3digit



To further ease up your skill, these Ubuntu Terminal keyboard shortcuts would help.

| Ubuntu Terminal Shortcuts | Function |
|---|---|
| Ctrl + Shift + T | Open new tab on current terminal |
| Ctrl + Shift + W | Close the current tab |
| Ctrl + A | Move cursor to beginning of line |
| Ctrl + E | Move cursor to end of line |
| Ctrl + U | Clears the entire current line |
| Ctrl + W | Delete the word before the cursor |
| Ctrl + R | Allows you to search your history for commands matching what you have typed |
| Ctrl + C | Kill the current process |
| Ctrl + L | Clears the terminal output |
| Alt + F | Move forward one word |
| Alt + B | Move backward one word |
| Ctrl + Shift + C | Copy the highlighted command to the clipboard |
| Ctrl + Shift + V or Shift + Insert | Paste the contents of the clipboard |
| Up/Down Arrow keys | To scroll through your command history, allowing you to quickly execute the same command multiple times |

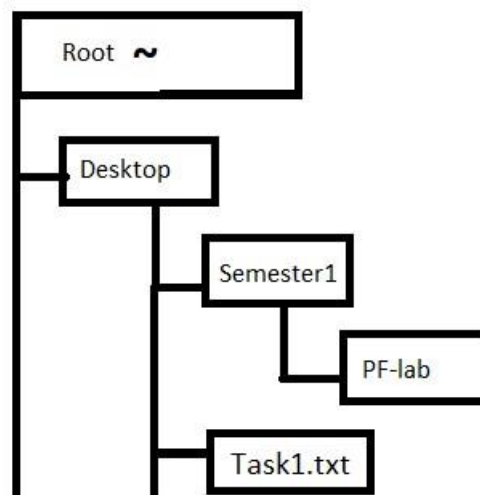
LAB TASKS

Task 1

Run all the commands given above and attach the screenshots.

Task 2

Create the following hierarchy as shown below:



After creating the above hierarchy do the following:

1. Now copy the Task1.txt from Desktop to PF-lab Directory without using cd command.
2. Create a copy of this task1.txt with name task1-1.txt in PF-lab Directory
3. Now move task1-1.txt to Home
4. Change path to home directory