LAB 1 SHELL COMMANDS

1 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.

2 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.

3 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. **echo**
   The echo command prints its arguments to the standard output.

```
$ echo Hello World
```

   Output Hello World on your terminal screen.

2. **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).

2. **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 ful path from the root directory. To copy a num-ber of files to a directory (folder) named dir, try this instead:

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

4. 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.

5. 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
```

6. 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
```

7. **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
```

8. **pwd**
   The pwd (print working directory) program simply outputs the name of the cur-rent working directory.

9. **date**
   Displays current time and date

10. **clear**
    Clears the terminal screen.

11. **exit**
    Exit the Shell

12. **mkdir**
    The mkdir command creates a new directory dir:

```
$ mkdir dir
```

13. **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.

14. **cd**
    The current working directory is the directory that a process (such as shell) is cur-rently 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.

15. 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
```

4 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
```

5 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

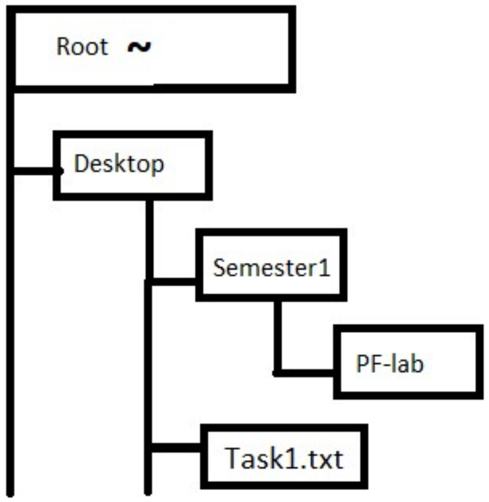
1. Home Directory (~)
2. Root Directory(/)
3. Current Working Directory(.)

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.

LAB TASK



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