Assignment 1-A

Name: Soham Sant **Roll No.:** 33271

Title: Basic Linux Commands

Aim: Study of Basic Linux Commands: echo, ls, read, cat, touch, test, loops, arithmetic comparison, conditional loops, grep, sed etc.

Linux Basic Commands :

1. **pwd command** :

   The 'pwd' command stands for 'print working directory'. It doesn't accept any option or argument and displays the detail of current working directory.

```
(base) mllab13@mllab13:~/Desktop$ pwd
/home/mllab13/Desktop
(base) mllab13@mllab13:~/Desktop$
```

2. **cd command** :

   The cd command is one of the important Linux commands you must know and it will help you to navigate through directories. Just type **cd** followed by directory as shown below.: `cd <directory path>`

```
(base) mllab13@mllab13:~$ pwd
/home/mllab13
(base) mllab13@mllab13:~$ cd Desktop
(base) mllab13@mllab13:~/Desktop$ pwd
/home/mllab13/Desktop
(base) mllab13@mllab13:~/Desktop$ cd
(base) mllab13@mllab13:~$ pwd
/home/mllab13
(base) mllab13@mllab13:~$
```

3. **ls command** :

   The ls command is used to list files and directories in the current working directory. This is going to be one of the most frequently used Linux commands you must know of.

```
(base) mllab13@mllab13:~$ ls


aa.cpp~   a.txt      lab1.c~       my1.c~   sample1.cpp~
aadity.c~ bangle.cpp~  lab.c~      mybre.c~   sample.cpp~
adwait.cpp~ ddalgo.c~    main.c~       my.c~    shapes.cpp~
anaconda3 Desktop      mock.cpp~       Pictures   snap
ass5.cpp~ Documents    mouse.c~      prjwal   tasm
Assignmentos1 Downloads    MPLABXProjects  PSDL-Setup-Ububtu  TC
Assignmentos1~  lab~       music       Public   Templates

(base) mllab13@mllab13:~$
```

4. **cat,echo,and less command** :

When you want to output the contents of a file, or print anything to the terminal output, we make use of the cat or echo commands. Let's see their basic usage. I've added some text to our New-File that we created earlier.

`cat <file name>`

`echo <Text to print on terminal>`

```
(base) mllab13@mllab13:~/Desktop$ cat NewFile
hi this is a new file

(base) mllab13@mllab13:~/Desktop$ echo Newfile
Newfile
(base) mllab13@mllab13:~/Desktop$ echo hello
hello
(base) mllab13@mllab13:~/Desktop$
```

5. **cp command** :

   The cp and mv commands are equivalent to the copy-paste and cut-paste in Windows. But since Linux doesn't really have a command for renaming files, we also make use of the mv command to rename files and folders.

   `cp <source> <destination>`

```
(base) mllab13@mllab13:~/Desktop$ ls
 a         main.cpp~   NewFile~      test.c~
'idk what this is'   NewFile   sample.cpp~   xyz.cpp~
(base) mllab13@mllab13:~/Desktop$ cp a copy
(base) mllab13@mllab13:~/Desktop$ ls
 a     'idk what this is'   NewFile    sample.cpp~   xyz.cpp~
 copy   main.cpp~     NewFile~   test.c~
(base) mllab13@mllab13:~/Desktop$
```

6. **mv command** :

   Since we were moving the file within the same directory, it acted as rename. The file name is now changed.

   `mv <source> <destination`

```
(base) mllab13@mllab13:~/Desktop$ ls
 a     'idk what this is'   NewFile    sample.cpp~   xyz.cpp~
 copy   main.cpp~     NewFile~   test.c~
(base) mllab13@mllab13:~/Desktop$ mv a b
(base) mllab13@mllab13:~/Desktop$ ls
 b     'idk what this is'   NewFile    sample.cpp~   xyz.cpp~
 copy   main.cpp~     NewFile~   test.c~
(base) mllab13@mllab13:~/Desktop$
```

7. **mkdir command** :

   The mkdir command allows you to create directories from within the terminal. The default syntax is **mkdir** followed by the directory name.

   `mkdir <folder name>`

```
(base) mllab13@mllab13:~/Desktop$ ls
 b     'idk what this is'   NewFile    sample.cpp~   xyz.cpp~
 copy   main.cpp~     NewFile~   test.c~
(base) mllab13@mllab13:~/Desktop$ mkdir newFolder
(base) mllab13@mllab13:~/Desktop$ ls
 b     'idk what this is'   NewFile    newFolder     test.c~
 copy   main.cpp~     NewFile~   sample.cpp~   xyz.cpp~
(base) mllab13@mllab13:~/Desktop$ cd newFolder
(base) mllab13@mllab13:~/Desktop/newFolder$ pwd
/home/mllab13/Desktop/newFolder
(base) mllab13@mllab13:~/Desktop/newFolder$
```

8. **rmdir command** :

   The **rmdir command** is useful when you want to remove the empty directories from the filesystem in Linux. This command lets you specify the terminal to **remove a particular directory** right from the terminal.

   `rmdir <options> <directory>`

   In the **<options>**, you can use various types of flags as per your requirements while removing **<directory>**

   The above command displays the various options such as:

   - **-p**: This option removes the directory, including all its ancestors
   - **-v, –verbose**: Displays verbose information for every directory.
   - **–ignore-fail-on-non-empty**: This option does not report a failure that occurs because a directory is non-empty.
   - **–version**: This option displays the version information and exit.

```
(base) mllab13@mllab13:~/Desktop$ ls
 b     'idk what this is'   NewFile    newFolder     test.c~
 copy   main.cpp~     NewFile~   sample.cpp~   xyz.cpp~
(base) mllab13@mllab13:~/Desktop$ rmdir newFolder
(base) mllab13@mllab13:~/Desktop$ ls
 b     'idk what this is'   NewFile    sample.cpp~   xyz.cpp~
 copy   main.cpp~     NewFile~   test.c~
(base) mllab13@mllab13:~/Desktop$(base) mllab13@mllab13:~/Desktop$ ls
 b     'idk what this is'   NewFile    newFolder     test.c~
 copy   main.cpp~     NewFile~   sample.cpp~   xyz.cpp~
(base) mllab13@mllab13:~/Desktop$ rmdir newFolder
(base) mllab13@mllab13:~/Desktop$ ls
 b     'idk what this is'   NewFile    sample.cpp~   xyz.cpp~
 copy   main.cpp~     NewFile~   test.c~
(base) mllab13@mllab13:~/Desktop$
```

9. **rm command** :

   rm is a general command in Unix and other Unix-like systems. It is used to delete objects like symbolic links, directories, and computer files from the file systems.

```
(base) mllab13@mllab13:~/Desktop$ ls
 b     'idk what this is'   NewFile    sample.cpp~   xyz.cpp~
 copy   main.cpp~     NewFile~   test.c~
(base) mllab13@mllab13:~/Desktop$ rm b copy
(base) mllab13@mllab13:~/Desktop$ ls
'idk what this is'   NewFile    sample.cpp~   xyz.cpp~
 main.cpp~       NewFile~   test.c~
(base) mllab13@mllab13:~/Desktop$
```

10. **touch command** :

    To create a new file, the touch command will be used. The **touch** keyword followed by the file name will create a file in the current directory.

```
(base) mllab13@mllab13:~/Desktop$ ls
'idk what this is'   main.cpp~   sample.cpp~   test.c~   xyz.cpp~
(base) mllab13@mllab13:~/Desktop$ touch NewFile
(base) mllab13@mllab13:~/Desktop$ gedit NewFile
(base) mllab13@mllab13:~/Desktop$ ls
'idk what this is'   NewFile    sample.cpp~   xyz.cpp~
 main.cpp~       NewFile~   test.c~
(base) mllab13@mllab13:~/Desktop$
```

11. **locate command** :

You can use this command to locate a file, just like the search command in Windows. What's more, using the -i argument along with this command will make it case-insensitive, so you can search for a file even if you don't remember its exact name.

```
(base) mllab13@mllab13:~/Desktop$ locate -i newfile
/home/mllab13/.eclipse/360744294_linux_gtk_x86_64/configuration/org.eclipse.osgi/387/0/.cp/icons/full/etool16/newfile_wiz.png
/home/mllab13/.eclipse/360744294_linux_gtk_x86_64/configuration/org.eclipse.osgi/506/0/.cp/icons/etool16/newfile_wiz.gif
/usr/share/perl5/Archive/Zip/NewFileMember.pm
/usr/share/texlive/texmf-dist/tex/latex/newfile
/usr/share/texlive/texmf-dist/tex/latex/newfile/newfile.sty
(base) mllab13@mllab13:~/Desktop$
```

12. **find command** :

    The **find** command in UNIX is a command line utility for walking a file hierarchy. It can be used to find files and directories and perform subsequent operations on them. It supports searching by file, folder, name, creation date, modification date, owner and permissions. By using the '-exec' other UNIX commands can be executed on files or folders found.

    SYNTAX :

```
$ find [where to start searching from]
[expression determines what to find] [-options] [what to find]
```

```
(base) mllab13@mllab13:~$ find ./Desktop -name a
./Desktop/file/a
(base) mllab13@mllab13:~$
```

13. **grep command** :

    If you wish to search for a specific string within an output, the grep command comes into the picture. We can pipe (**|**) the output to the grep command and extract the required string.

    `<Any command with output> | grep "<string to find>"`

```
(base) mllab13@mllab13:~/Desktop$ cat NewFile
hi this is a new file
this is 2nd Line
adding more lines will help with understanding of the working pf grep command

(base) mllab13@mllab13:~/Desktop$ cat NewFile | grep "Line"
this is 2nd Line
(base) mllab13@mllab13:~/Desktop$
```

14. **sudo command** :

    This command is equivalent to having logged in as root (based on what permissions you have as a sudoer).

    Just add the word **sudo** before any command that you need to run with escalated privileges and that's it. It's very simple to use, but can also be an added security risk if a malicious user gains access to a sudoer.

```
non-root-user@ubuntu:~# sudo <command you want to run>
Password:
```

```
(base) mllab13@mllab13:~$ sudo -l
Matching Defaults entries for mllab13 on mllab13:
    env_reset, mail_badpass,
```

```
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User mllab13 may run the following commands on mllab13:
    (ALL : ALL) ALL
(base) mllab13@mllab13:~$
```

15. **df command** :

    Use df command to get a report on the system's disk space usage, shown in percentage and KBs. If you want to see the report in megabytes, type df -m.

```
(base) mllab13@mllab13:~$ df -m
Filesystem     1M-blocks  Used Available Use% Mounted on
udev                3907     0      3907   0% /dev
tmpfs                787     3       785   1% /run
/dev/sda5         195777 56715    129047  31% /
tmpfs               3931   365      3567  10% /dev/shm
tmpfs                  5     1         5   1% /run/lock
tmpfs               3931     0      3931   0% /sys/fs/cgroup
/dev/loop0           165   165         0 100% /snap/gnome-3-28-1804/161
/dev/loop16          315   315         0 100% /snap/eclipse/66
/dev/loop6           141   141         0 100% /snap/gnome-3-26-1604/104
/dev/loop7             3     3         0 100% /snap/gnome-calculator/945
/dev/loop11            3     3         0 100% /snap/gnome-calculator/934
/dev/loop9             2     2         0 100% /snap/gnome-system-monitor/184
/dev/loop12          155   155         0 100% /snap/brave/236
/dev/loop17            1     1         0 100% /snap/gnome-logs/115
/dev/loop30            1     1         0 100% /snap/gnome-logs/119
/dev/loop23            2     2         0 100% /snap/gnome-system-monitor/181
/dev/loop20           68    68         0 100% /snap/jupyter/6
/dev/loop19          117   117         0 100% /snap/core/14946
/dev/loop21            1     1         0 100% /snap/bare/5
/dev/loop29            1     1         0 100% /snap/gnome-characters/785
/dev/loop32           56    56         0 100% /snap/core18/2751
/dev/loop31           74    74         0 100% /snap/core22/634
/dev/loop35           56    56         0 100% /snap/core18/2785
/dev/loop22           74    74         0 100% /snap/core22/817
/dev/loop8           156   156         0 100% /snap/brave/248
/dev/loop24          119   119         0 100% /snap/core/15511
/dev/loop28          480   480         0 100% /snap/netbeans/80
/dev/loop34            1     1         0 100% /snap/gnome-characters/789
/dev/loop13           92    92         0 100% /snap/gtk-common-themes/1535
/dev/loop10          486   486         0 100% /snap/gnome-42-2204/120
/dev/loop4           350   350         0 100% /snap/gnome-3-38-2004/143
/dev/loop25          315   315         0 100% /snap/eclipse/67
/dev/loop26           64    64         0 100% /snap/core20/1974
/dev/loop3           350   350         0 100% /snap/gnome-3-38-2004/140
/dev/loop15           64    64         0 100% /snap/core20/1891
/dev/loop1            82    82         0 100% /snap/gtk-common-themes/1534
/dev/loop33          219   219         0 100% /snap/gnome-3-34-1804/93
/dev/loop18          461   461         0 100% /snap/gnome-42-2204/102
/dev/loop2           165   165         0 100% /snap/gnome-3-28-1804/198
/dev/loop5           219   219         0 100% /snap/gnome-3-34-1804/90
/dev/loop14          467   467         0 100% /snap/netbeans/76
/dev/loop27          141   141         0 100% /snap/gnome-3-26-1604/111
tmpfs                787     1       787   1% /run/user/121
tmpfs                787     1       787   1% /run/user/1000
(base) mllab13@mllab13:~$
```

16. **du command** :

    If you want to check how much space a file or a directory takes, the du (Disk Usage) command is the answer. However, the disk usage summary will show disk block numbers instead of the usual size format. If you want to see it in bytes, kilobytes, and megabytes, add the -h argument
    to the command line.

```
(base) mllab13@mllab13:~/Desktop$ du
4 ./file
80  ./idk what this is
36  ./.ipynb_checkpoints
160 .
(base) mllab13@mllab13:~/Desktop$
```

17. **head command** :

    The head command is used to view the first lines of any text file. By default, it will show the first ten lines, but you can change this number to your liking. For example, if you only want to show the first five lines, type head -n 5 filename.ext.

```
(base) mllab13@mllab13:~/Desktop$ head NewFile
hi this is a new file
this is 2nd Line
adding more lines will help with understanding of the working pf grep command

(base) mllab13@mllab13:~/Desktop$ head -n 1 NewFile
hi this is a new file
(base) mllab13@mllab13:~/Desktop$
```

18. **tail command** :

    This one has a similar function to the head command, but instead of showing the first lines, the tail command will display the last ten lines of a text file. For example, tail -n filename.ext.

```
(base) mllab13@mllab13:~/Desktop$ tail NewFile
hi this is a new file
this is 2nd Line
adding more lines will help with understanding of the working pf grep command

(base) mllab13@mllab13:~/Desktop$ tail -n 1 NewFile

(base) mllab13@mllab13:~/Desktop$ tail -n 2 NewFile
adding more lines will help with understanding of the working pf grep command

(base) mllab13@mllab13:~/Desktop$
```

19. **diff command** :

    Short for difference, the diff command compares the contents of two files line by line. After analysing the files, it will output the lines that do not match. Programmers often use this command when they need to make program alterations instead of rewriting the entire source code.
    The simplest form of this command is `diff <file 1> <file 2>`

```
(base) mllab13@mllab13:~/Desktop$ diff NewFile nwFil
3c3
< adding more lines will help with understanding of the working pf grep command
---
> adding more lines will help with understanding of the working pf grep command --this line is edited
(base) mllab13@mllab13:~/Desktop$
```

20. **tar command** :

    The tar command in Linux is used to create and extract archived files in Linux. We can extract multiple different archive files using the tar command.

    To create an archive, we use the -c parameter and to extract an archive, we use the -x parameter. Let's see it working.

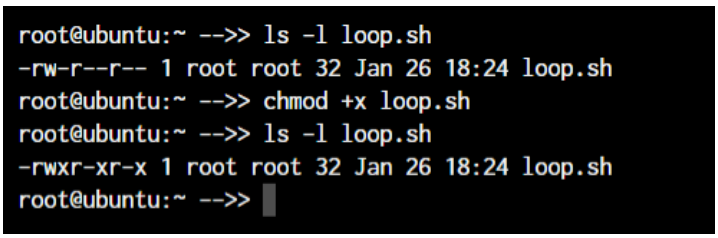
```
 soham@LAPTOP-VCSHPE1S:/mnt/c/Users/soham/OneDrive/Desktop$ tar -cvf compress
.tar mykitty file
mykitty/
file/
soham@LAPTOP-VCSHPE1S:/mnt/c/Users/soham/OneDrive/Desktop$ ls
PAPA  compress.tar  f1.txt  hello.sh  nwFile         unkillablecockroach
cat   desktop.ini   file    mykitty   sohamdirtyboi
soham@LAPTOP-VCSHPE1S:/mnt/c/Users/soham/OneDrive/Desktop$ tar -xvf compress.tar
mykitty/
file/
soham@LAPTOP-VCSHPE1S:/mnt/c/Users/soham/OneDrive/Desktop$ ls
PAPA  compress.tar  f1.txt  hello.sh  nwFile         unkillablecockroach
cat   desktop.ini   file    mykitty   sohamdirtyboi
```

21. **chmod command** :

    The chmod and chown commands give us the functionality to change the file permissions and file ownership are the most important Linux commands you should know.

    The main difference between the functions of the two commands is that the chmod command allows changing file permissions, while chown allows us to change the file owners.

    `chmod +x loop.sh`

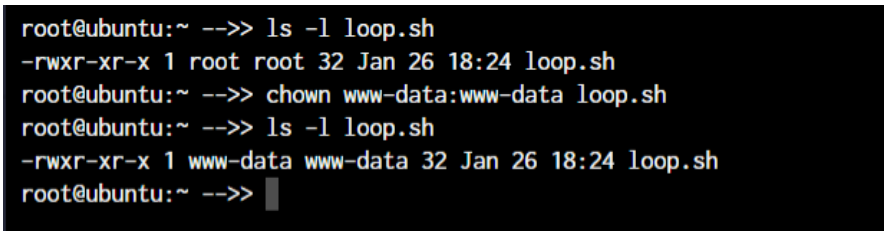


22. **chown command** :

    The chmod and chown commands give us the functionality to change the file permissions and file ownership are the most important Linux commands you should know.

    The main difference between the functions of the two commands is that the chmod command allows changing file permissions, while chown allows us to change the file owners.

    `chmod root:root loop.sh`



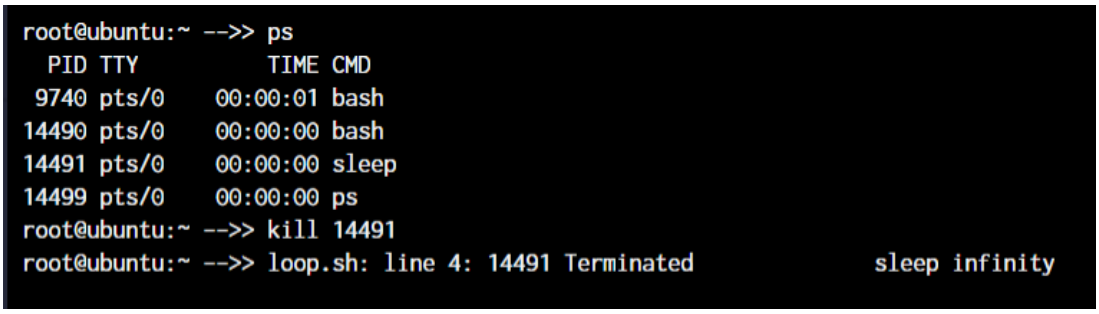
23. **kill command** :

    Now, to kill a process with the **kill** command, you can type **kill** followed b the PID of the process.

    `ps`

    `kill <process ID>`

    `killall <process name>`



24. **man command** :

    The man command is a very useful Linux command you must know. When working with Linux, the packages that we download can have a lot of functionality. Knowing it all is impossible.

    The man pages offer a really efficient way to know the functionality of pretty much all the packages that you can download using the package managers in your Linux distro.

    `man <command>`

25. **type command** :

    Linux 'type' command tell us whether a command given to the shell is a built-in or external command.

```
soham@LAPTOP-VCSHPE1S:/mnt/c/Users/soham$ type sudo
sudo is /usr/bin/sudo
soham@LAPTOP-VCSHPE1S:/mnt/c/Users/soham$
```

Conclusion:- We Understood basic commands of UNIX/LINUX and how to write shell script

Reference:-

https://www.hostinger.in/

https://www.geeksforgeeks.org/

https://www.javatpoint.com/