<u>Experiment : 2</u>

<u>BASIC LINUX COMMANDS</u>

<u>**man**</u> command in Linux is used to display the user manual of any command that we can run on the terminal.

Syntax :

```
$man [OPTION]... [COMMAND NAME]...
```

- No Option: It displays the whole manual of the command.
- $ man [COMMAND NAME]
- Just press “q” key to quit out of a man page.

To Display the manual page for the item (program) ls :

```
user@user:~$  man ls
```

output: It will display man page of ls command

===

<u>**pwd**</u> command in Linux is used to find Present Working Directory, starting from the root. pwd is shell built-in command

Syntax:

```
user@user:~$ pwd
```

Example:

```
user@user:~$ pwd
/home/user
```

===

<u>**mkdir**</u>

mkdir command in Linux allows the user to create directories (also referred to as folders in some operating systems) in the terminal

Syntax:

mkdir dirName - Creates a directory with name dirName.

Example:

```
user@user:~$ mkdir temp
```

Creates the directory temp.

==

rmdir command in Linux allows the user to remove directories

Syntax:

rmdir dirName - Removes a directory dirName.

Example:

```
user@user:~$ rmdir temp
```

- Removes the directory temp.

==

cd (Change Directory)

Change your current working directory.

Syntax:

```
cd    dirPath
```

Example:

```
user@user:~$ cd Desktop/
```

output:

```
user@user:~/Desktop$
```

cd ~ - Put you to your home directory

Example:

```
user@user:~$ cd ~
```

output:

```
user@user:~$
```

cd .. -Move to your parent directory.

Example:

```
user@user:~/Desktop/new$ cd ..
```

output:

```
user@user:~/Desktop$
```

==

cp - **cp** stands for **copy**. This command is used to copy files or group of files or directory

Syntax:

cp [OPTION] Source Destination
cp [OPTION] Source Directory
cp [OPTION] Source-1 Source-2 Source-3 Source-n Directory

- This command is used to copy files or group of files or directory. cp command works on three principal modes of operation and these operations depend upon number and type of arguments passed in cp command :

```
user@user:~$ cp  -i   source_file  destination_file
```

With the "-i" option, if the file "destination_file" exists, you will be prompted before it is overwritten.

==

ls - Lists directory contents of files and folders from which it runs.

Syntax:

```
ls [options] [paths]
```

Example:

```
user@user:~$ ls
Desktop  Documents   Downloads  Music  Pictures  Public  Templates Videos
```

Syntax:

ls -a - list all files including hidden file starting with '.' and hidden files ls -al - List all files in the current working directory in long list(rwx..)

Example:

```
user@user:~$ls -al
total 116
drwxr-xr-x 21 user user 4096 Jan 30 06:57 .
drwxr-xr-x 3 root root 4096 Jan 30 06:29 ..
-rw------- 1 user user   36 Jan 30 06:59 .bash_history
-rw-r--r-- 1 user user  220 Jan 30 06:29 .bash_logout
-rw-r--r-- 1 user user 3392 Jan 30 06:29 .bashrc
……
……
```

ls -r (reverse) - Lists the contents of the directory in reverse sort order.

Example:

```
user@user:~$ ls -r
Videos  Templates  Public  Pictures  Music  Downloads  Documents  Desktop
```

ls -a - Display the hidden files.

ls -R (recursive) - Lists the contents of all directories recursively

Example:

```
user@user:~$ ls -R
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
./Desktop:
./Templates:
```

./Videos:

ls -l - Display current permission information.

Example:

```
user@user:~$ls -l
total 908
-rw-r--r-- 1 ccf ccf   557 Oct  9 12:10  4sum.cpp
-rw-r--r-- 1 ccf ccf 34 Nov 14 15:06  akshay.txt
-rwxr-xr-x 1 ccf ccf 17128 Nov 23 15:31  a.out
-rw-r--r-- 1 ccf ccf 34 Jan 13 14:38  a.txt
.
.
```

==

mv -Move a file from one location to other or to rename a file

Syntax:

mv [Option] source destination

Example:

```
user@user:~$ mv  -i  myfile yourfile
```

-I option is **Interactive** move. Move the file from "myfile" to "yourfile". This effectively changes the name of "myfile" to "yourfile" and ask the user whether we want to overwrite or not.

The -i option makes the command ask the user for confirmation before moving a file that would overwrite an existing file, you have to press **y** for confirm moving, any other key leaves the file as it is.

==

rm - stands for remove. rm command is used to remove files, directories.

Syntax:

rm [OPTION]... FILE...
Exmple:
user@user:~$ **rm -i d.txt**

-i (Interactive Deletion): The -i option makes the command ask the user for confirmation before removing each file, you have to press **y** for confirm deletion, any other key leaves the file un-deleted.

===

find command is a command line utility. 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.

Syntax :

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

Example:

user@user:~$ $ find ./GFG -name sample.txt

It will search for sample.txt in GFG directory.

- **-empty :** Search for empty files and directories.
- **-size +N/-N :** Search for files of 'N' blocks; 'N' followed by 'c'can be used to measure size in characters; '+N' means size > 'N' blocks and '-N' means size < 'N' blocks.
- **-user name :** Search for files owned by user name or ID 'name'.

===

cat - Sends file contents to standard output. This is a way to list the contents of short files to the screen. It works well with piping.

Example:

user@user:~$ cat > file

Werr
Ewr

user@user:~$ cat file

Werr
ewr

==

more command is used to view the text files in the command prompt, displaying one screen at a time in case the file is large (For example log files).

Syntax:

more [-options] [-num] [+/pattern] [+linenum] [file_name]

While viewing the text file use these controls:

Enter key: to scroll down line by line.
Space bar: To go to the next page.
b key: To go to back one page.

- **[-options]**: any option that you want to use in order to change the way the file is displayed. Choose any one from the followings: (-d, -l, -f, -p, -c, -s, -u)
- **[-num]**: type the number of lines that you want to display per screen.
- **[+/pattern]**: replace the pattern with any string that you want to find in the text file.
- **[+linenum]**: use the line number from where you want to start displaying the text content.
- **[file_name]**: name of the file containing the text that you want to display on the screen.

==

**less - ** View a file or list of files. The position within files can be changed, and files can be manipulated in various ways. It is similar to more, but has the extended capability of allowing both forward and backward navigation through the file.

syntax : less filename

Example1:

```
user@user:~$ less +4 sample.txt
```

Output:

```
Be cool and calm
Happy birthday
I am  Bond 00 7
Eurekaa !!! I Got it !
Yes
No
NOoooooooo
Yesssssssssssssss yes
~
~
sample.txt (END)
```

To Starts up the file from the given number(less +number file_path):

Example 2:

```
user@user:~$ ls -la | less
```

Output:

```
drwxr-xr-x  2 root     root       4096 Apr  1  2012 .wireshark
drwxrwxr-x  4 himanshu himanshu   4096 Jun 30 01:33 workspace
-rw-------  1 himanshu himanshu    137 Jul  1 18:20 .Xauthority
-rw-------  1 himanshu himanshu   3286 Jul  1 18:34 .xsession-errors
-rw-------  1 himanshu himanshu   5556 Jul  1 16:46 .xsession-errors.old
```

==

head : It prints the first 10 lines of the specified files.

options:

-c num: Prints the first ‘num’ bytes from the file specified
-n num: Prints the first ‘num’ lines instead of first 10 lines.

Example:

```
user@user:~$ cat state.txt

Andhra Pradesh
Arunachal Pradesh
Assam
```

```
Bihar
Chhattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
Jharkhand
Karnataka
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Mizoram
Nagaland
Odisha
Punjab
Rajasthan
Sikkim
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttarakhand
West Bengal

user@user:~$ head state.txt

Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir

user@user:~$ head -n 5 state.txt

Andhra Pradesh
```

```
Arunachal Pradesh
Assam
Bihar
Chhattisgarh

user@user:~$ head -c 6 state.txt

Andhra
```

===

tail

It prints the last 10 lines of the specified files. Useful for things like getting the most recent entries from a log file.

options:

-c num: Prints the last ‘num’ bytes from the file specified
-n num: Prints the last ‘num’ lines instead of first 10 lines.

Example:

```
user@user:~$ tail state.txt

Odisha
Punjab
Rajasthan
Sikkim
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttarakhand
West Bengal
user@user:~$ tail -n 3 state.txt

Uttar Pradesh
Uttarakhand
West Bengal

user@user:~$ tail -c 6 state.txt
Bengal
```

==

cut

The cut command in UNIX is a command line utility for cutting sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by byte position, character and delimiter.

Example:

1)

```
user@user:~$ echo 'baz' | cut -b 2
a
user@user:~$ echo 'tfoobar' | cut -c 1,6
ta
```

2)How to cut based on a delimiter

To cut using a delimiter use the -d option. This is normally used in conjunction with the -f option to specify the field that should be cut.

```
user@user:~$ cat  city.txt

John,Smith,34,London
Arthur,Evans,21,Newport
George,Jones,32,Truro

user@user:~$ cut -d ',' -f 1 city.txt
John
Arthur
George

user@user:~$ cut -d ',' -f 1,4 city.txt
John,London
Arthur,Newport
George,Truro
```

==

wc - stands for word count. As the name implies, it is mainly used for counting purpose.

Syntax:

```
wc [OPTION]... [FILE]...
```

Example:

```
user@user:~$ cat state.txt

Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
user@user:~$ wc state.txt
```

Output:

```
5  7 63 state.txt
```

- It is used to find out number of lines, word count, byte and characters count in the files specified in the file arguments.
- By default it displays four-columnar output.
- First column shows number of lines present in a file specified, second column shows number of words present in the file, third column shows number of characters present in file and fourth column itself is the file name which are given as argument.

===

Paste - command is one of the useful commands in Linux operating system. It is used to join files horizontally.

Syntax:

```
paste [OPTION]... [FILES]...
```

Example:

```
user@user:~$ cat state
Arunachal Pradesh
Assam
Andhra Pradesh
Bihar
Chhattisgrah

user@user:~$ cat capital
Itanagar
Dispur
```

```
Hyderabad
Patna
Raipur
```

Without any option paste merges the files in parallel. The paste command writes corresponding lines from the files with tab as a deliminator on the terminal.

```
user@user:~$ paste number state capital
1       Arunachal Pradesh       Itanagar
2       Assam   Dispur
3       Andhra Pradesh  Hyderabad
4       Bihar   Patna
5       Chhattisgrah    Raipur
```

===

Echo $

echo is a command that outputs the strings it is being passed as arguments. It is a command available in various operating system shells and typically used in shell scripts and batch files

Syntax :

echo [option] [string]

Example:

```
user@user:~$  echo “The value of variable x = $x”
The value of variable x = 10
```

===

read - command in Linux system is used to read from a file descriptor. Basically, this command read up the total number of bytes from the specified file descriptor into the buffer.

Example :

```
user@user:~$ echo "what is your name..?";read name;echo "hello $name"
```

we are acquiring the user’s name and then showing the user’s name with a greeting.

===

Linux commands for redirection, pipes

I/O Redirections

```
user@user:~$ ls > file_list.txt
```

ls command is executed and the results are written in a file named file_list.txt. Since the output of ls was redirected to the file, no results appear on the display.

```
user@user:~$ ls >> file_list.txt
```

The new results are added to the end of the file, thus making the file longer each time the command is repeated. If the file does not exist when you attempt to append the redirected output, the file will be created.

==

Pipes

With pipes, the standard output of one command is fed into the standard input of another.
Example:

```
user@user:~$ ls -l | less
```

The output of the ls command (List in long format) is fed into less. By using this "| less" trick, you can make any command have scrolling output.

```
user@user:~$ ls -lt | head
```

List in long format and sort on time is the output of first command. It is fed to head command. Displays the 10 newest files in the current directory

==

File Permissions

Chmod - The chmod command is used to change the access mode of a file.

Syntax:

```
chmod [reference][operator][mode] file...
```

u	owner	file's owner
g	group	users who are members of the file's group
o	others	
a	all	

Description

r	Permission to read the file.(4)
w	Permission to write (or delete) the file.(2)
x	Permission to execute the file, or, in the case of a directory, search it(1)

Operator Description

+	Adds the specified modes to the specified classes
-	Removes the specified modes from the specified classes
=	The modes specified are to be made the exact modes for the specified classes.

Example:

```
user@user:~$ ls -l new_ file.txt

          ccf@FISATPC0360:~$ ls -l new_file.txt
          -rw-r--r-- 1 ccf ccf 20 Jan 22 11:47 new_file.txt
user@user:~$ chmod a+x  new_file.txt

          ccf@FISATPC0360:~$ ls -l new_file.txt
          -rwxr-xr-x 1 ccf ccf 20 Jan 22 11:47 new_file.txt
```

Deny execute permission to everyone

```
user@user:~$  chmod a-x sample.txt
```

Allow read permission to everyone.

```
 user@user:~$ chmod a+r sample.txt
```

Make a shell script executable by the user/owner.

```
user@user:~$  chmod u+x samplescript.sh
```

==

Chown - command is used to change the owner or the group of a particular file or directory.

Example:

```
user@user:~$ ls -l new_file.txt
-rwxr-xr-x 1 ccf ccf 20 Jan 22 11:47 new_file.txt

user@user:~$ chown master new_file.txt
-rwxr-xr-x 1 master ccf 20 Jan 22 11:47 new_file.txt
```

where the *master* is another user in the system

==

Df - **Linux df** command is used to display the disk space used in the file system. The '**df**' stands for "disk filesystem." It defines the number of blocks used, the number of blocks available, and the directory where the file system is mounted.

Syntax :

```
df [OPTION]...[FILE]...
```

Example:

// using df for a specific file //

```
$df kt.txt
Filesystem   1K-blocks   Used       Available     Use%   Mounted on
/dev/the2    1957124    1512        1955612        1%    /snap/core
```

==

top command is used to show the Linux processes. It provides a dynamic real-time view of the running system. Usually, this command shows the summary information of the system and the list of processes or threads which are currently managed by the Linux Kernel.

Here,

- **PID:** Shows task's unique process id.
- **PR:** Stands for priority of the task.
- **SHR:** Represents the amount of shared memory used by a task.
- **VIRT:** Total virtual memory used by the task.
- **USER:** User name of owner of task.
- **%CPU:** Represents the CPU usage.
- **TIME+:** CPU Time, the same as 'TIME', but reflecting more granularity through hundredths of a second.
- **SHR:** Represents the Shared Memory size (kb) used by a task.
- **NI:** Represents a Nice Value of task. A Negative nice value implies higher priority, and positive Nice value means lower priority.
- **%MEM:** Shows the Memory usage of task.

==

Ps **-** Linux provides us a utility called **ps** for viewing information related with the processes on a system which stands as abbreviation for **"Process Status".** ps command is used to list the currently running processes and their PIDs along with some other information depends on different options.

==

ssh

It is a protocol used to securely connect to a remote server/system.

Example:

To connect to a remote machine, load the terminal or any SSH client and type **ssh** followed by the IP address or name

```
user@user:~$  ssh 192.168.56.101       or      ssh test.server.com
```

==

scp **-** (secure copy) command in Linux system is used to copy file(s) between servers in a secure way. By default, SCP is using the "AES-128" to encrypt files.

Example:

```
user@user:~$scp -P 2249 Label.pdf mrarianto@202.x.x.x:.
```

To specify a specific port to use with SCP: Usually, SCP is using port 22 as a default port. But for security reason, you can change the port into another port. For example, we are going to use port 2249.

==

Ssh-keygen - Use the ssh-keygen command to generate a public/private authentication key pair. Authentication keys allow a user to connect to a remote system without supplying a password. Keys must be generated for each user separately

```
# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:z6zTVQ/PJYt2o96DrVYClmfcqBG8Pdb8nzqY2m2HjeY
root@geeklab
The key's randomart image is:
+---[RSA 2048]----+
|        .      |
|         o     |
|          * =  |
|         * O B .|
|       S. B + O.|
|        +. = = =|
|        .+ooB+.o|
|       ..oo=Bo+.|
|       .o.+*E=. |
+----[SHA256]-----+
```

The ssh-key command in the example generated two keys in the ~/.ssh directory:

```
$ ls ~/.ssh
id_rsa
id_rsa.pub
```

To log on to, or copy files to, a remote system without supplying a password, copy the public key (~/.ssh/id_rsa.pub in this example) to

~/.ssh/authorized_keys on the remote system. Set the remote **~/.ssh** directory permissions to **700**. You can then use the ssh or scp tools to access the remote system without supplying a password.

===

ssh-copy-id -Copy the public key to remote-host using ssh-copy-id
user@user:~$ **ssh-copy-id -i ~/.ssh/id_rsa.pub remote-host**
user@user's password:

Now try logging into the machine, with "ssh 'remote-host'", and check in:
.ssh/authorized_keys

Note: ssh-copy-id **appends** the keys to the remote-host's .ssh/authorized_key.

Login to remote-host without entering the password
user@user:~$ **ssh remote-host**
Last login: Sun Nov 16 17:22:33 2008 from 192.168.1.2
[Note: SSH did not ask for password.]

jsmith@remote-host$ [Note: You are on remote-host here]

===

`useradd` Command creates a new user account according to the options specified on the command line and the default values set in the `/etc/default/useradd` file.

The general syntax for the `useradd` command is as follows:

```
useradd [OPTIONS] USERNAME
```

Only root or users with sudo privileges can use the `useradd` command to create new user accounts.

Example:

```
$ sudo useradd username
```

===

The command '**usermod**' is used to modify or change any attributes of a already created user account via command line.

The ‘**usermod**‘ command is simple to use with lots of options to make changes to an existing user. Use usermod command by modifying some existing users in Linux box with the help of following options.

- **-c** = We can add comment field for the useraccount.
- **-d** = To modify the directory for any existing user account.
- **-e** = Using this option we can make the account expiry in specific period.
- **-g** = Change the primary group for a User.
- **-G** = To add a supplementary groups.
- **-a** = To add anyone of the group to a secondary group.
- **-l** = To change the login name from tecmint to tecmint_admin.
- **-L** = To lock the user account. This will lock the password so we can’t use the account.

---

userdel command in Linux system is used to delete a user account and related files. This command basically modifies the system account files, deleting all the entries which refer to the username LOGIN. It is a low-level utility for removing the users.

Syntax:

```
userdel [options] LOGIN
```

userdel -f: This option forces the removal of the specified user account. It doesn’t matter that the user is still logged in. It also forces the *userdel* to remove the user’s home directory and mail spool, even if another user is using the same home directory or even if the mail spool is not owned by the specified user.

Example:

```
$ sudo userdel -f neuser
```

===

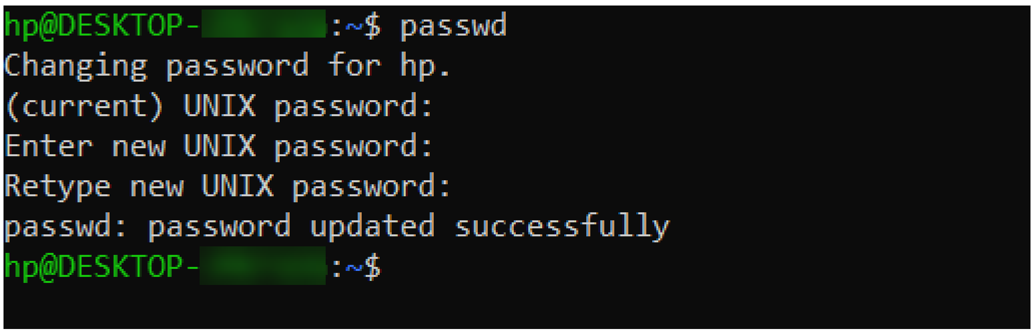
passwd command in Linux is used to change the user account passwords. The root user reserves the privilege to change the password for any user on the system, while a normal user can only change the account password for his or her own account.

Syntax:

```
 passwd [options] [username]
```

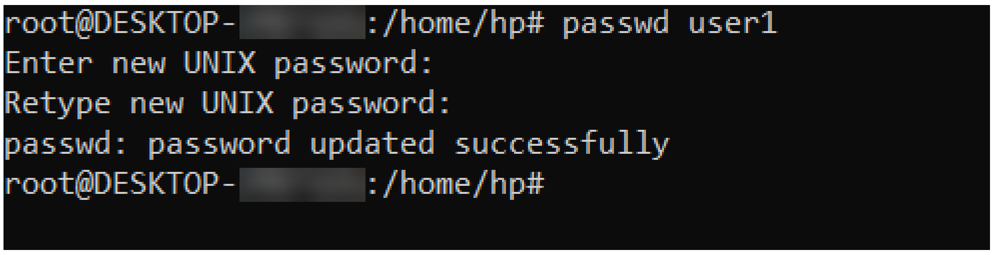
Example:

```
Command: passwd
```



```
hp@DESKTOP-[redacted]:~$ passwd
Changing password for hp.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
hp@DESKTOP-[redacted]:~$
```

```
Command [root]: passwd user1
```



```
root@DESKTOP-[redacted]:/home/hp# passwd user1
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@DESKTOP-[redacted]:/home/hp#
```

sudo can be used to invoke root privileges by normal users, and can change the password for root itself.

```
$sudo passwd root
```

```
=====================================================
```