

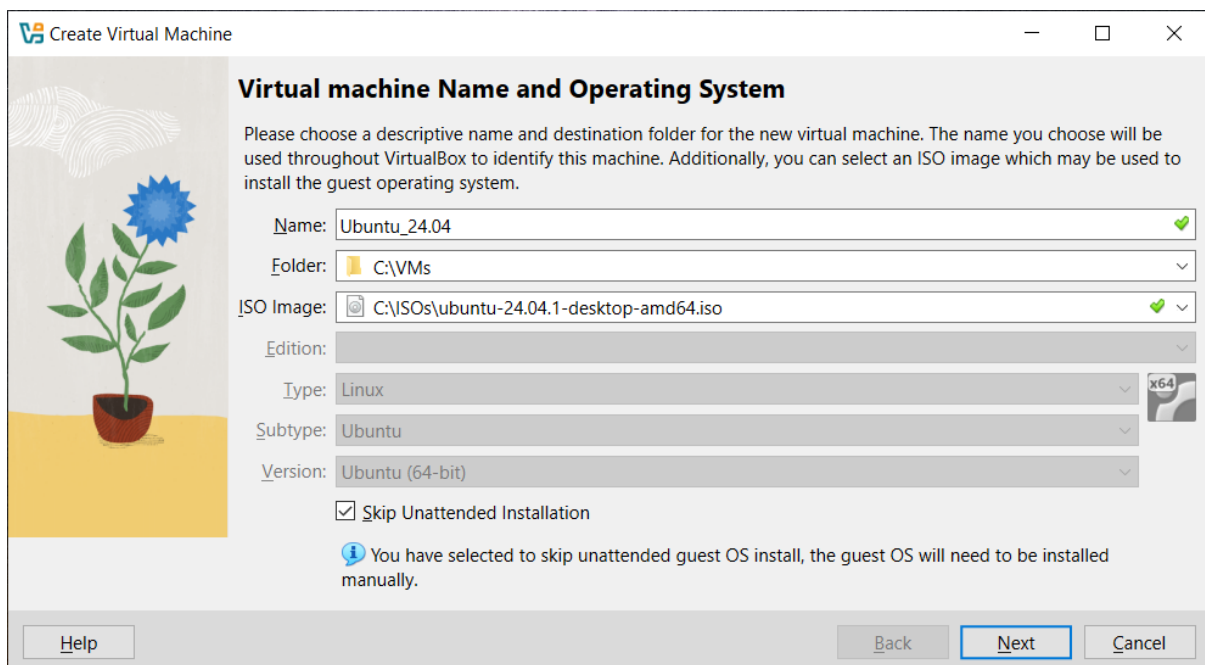
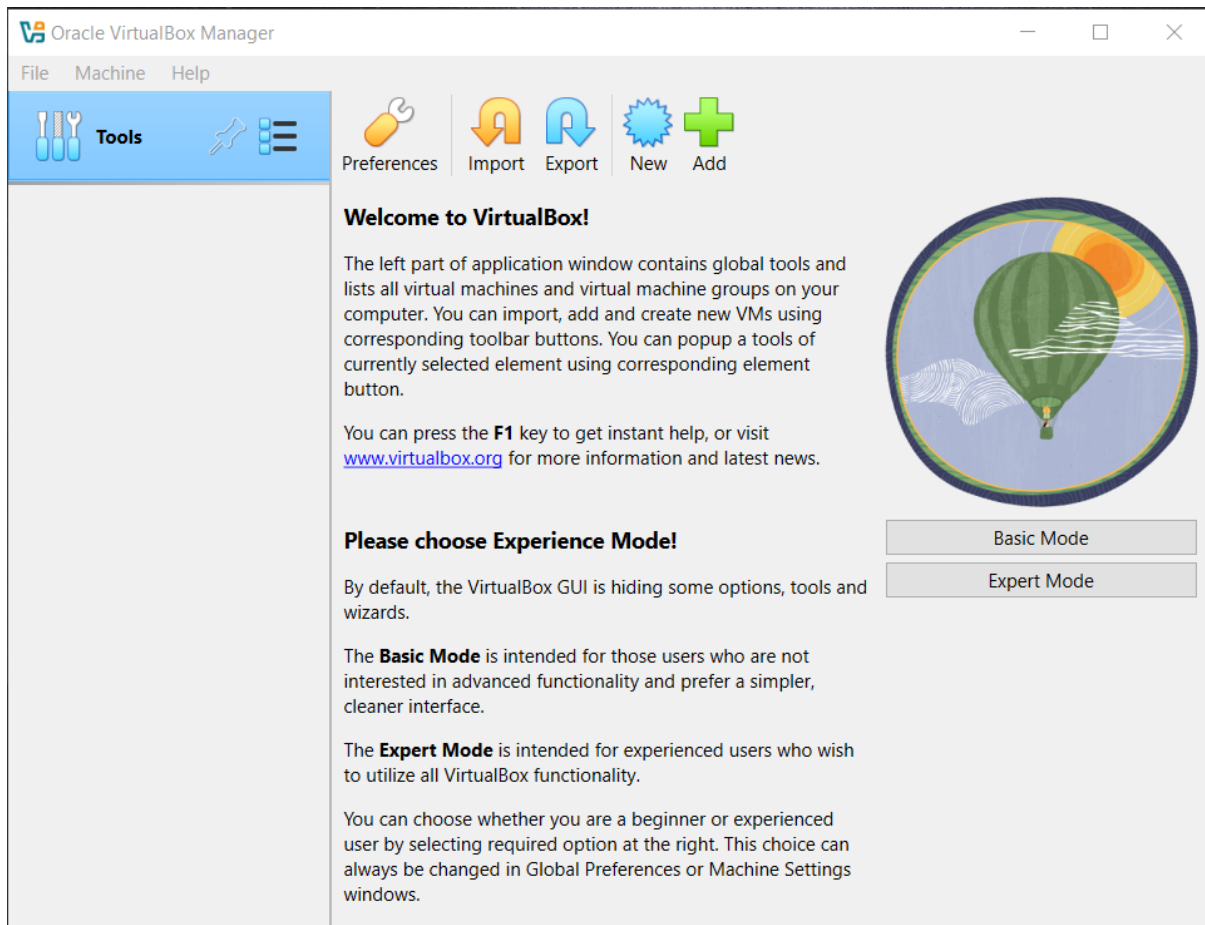
Practical 1

Aim: Installation of Linux/Unix Operating System.

Theory:

- Linux is an open source operating system (OS). Linux was designed to be similar to UNIX, but has evolved to run on a wide variety of hardware from phones to supercomputers.
- Every Linux-based OS involves the Linux kernel—which manages hardware resources—and a set of software packages that make up the rest of the operating system. Linux makes very efficient use of the system's resources.
- Linux runs on a range of hardware, right from supercomputers to watches. You can give new life to your old and slow Windows system by installing a lightweight Linux system, or even run a NAS or media streamer using a particular distribution of Linux.
- Linux is perfect for everyday tasks like browsing, emailing, photo management, financial management, and much more.

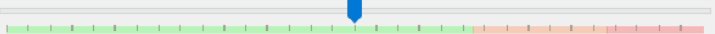
Procedure:

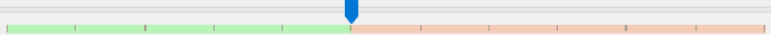


Create Virtual Machine

Hardware

You can modify virtual machine's hardware by changing amount of RAM and virtual CPU count. Enabling EFI is also possible.

Base Memory:  8192 MB

Processors:  6

☐ Enable EFI (special OSes only)

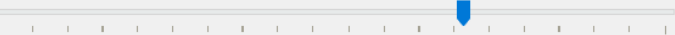
[Help](#) [Back](#) [Next](#) [Cancel](#)

Create Virtual Machine

Virtual Hard disk


If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select an existing one. Alternatively you can create a virtual machine without a virtual hard disk.

☒ Create a Virtual Hard Disk Now

Disk Size:  40.00 GB

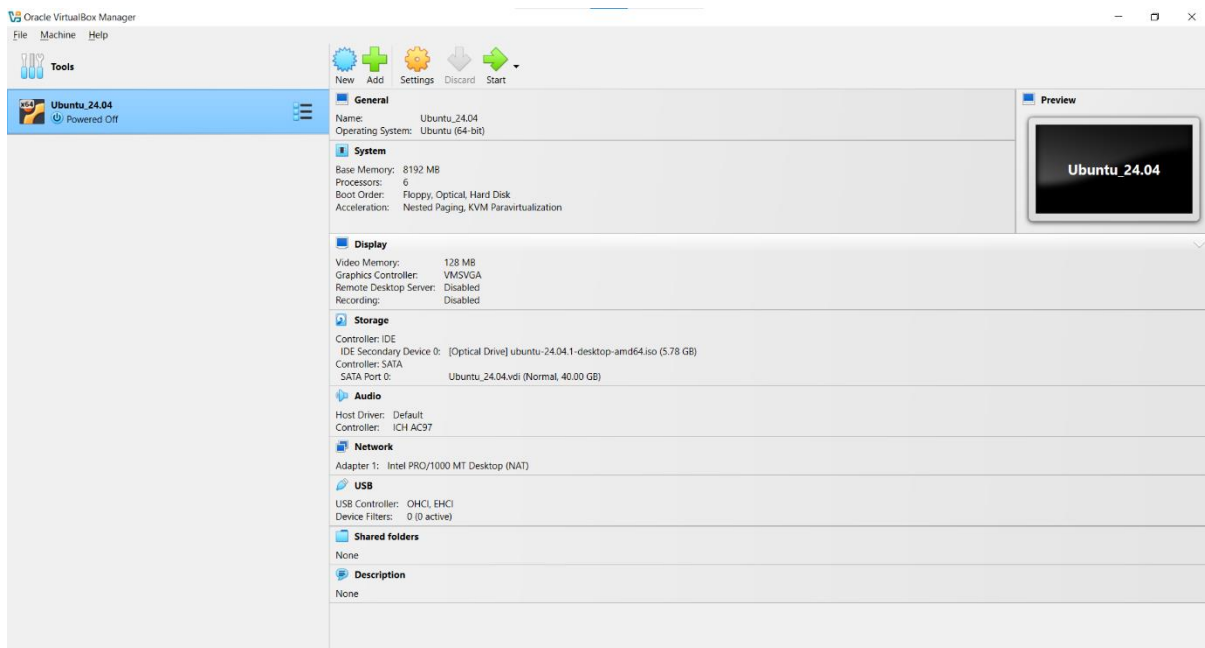
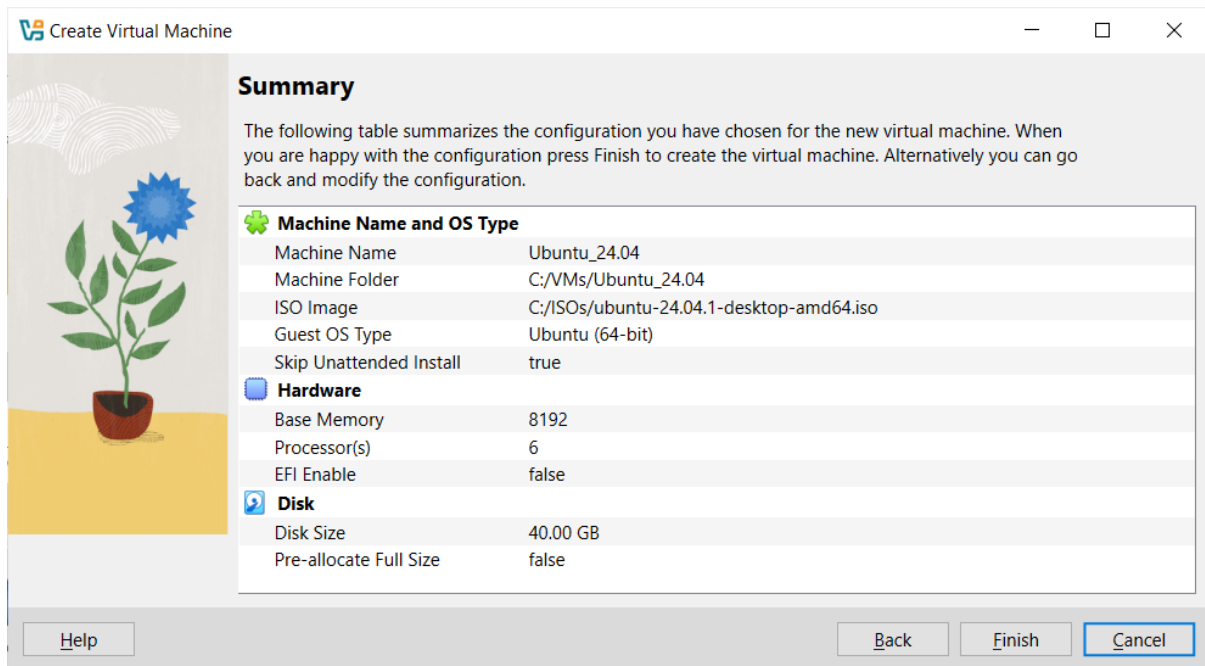
☐ Pre-allocate Full Size

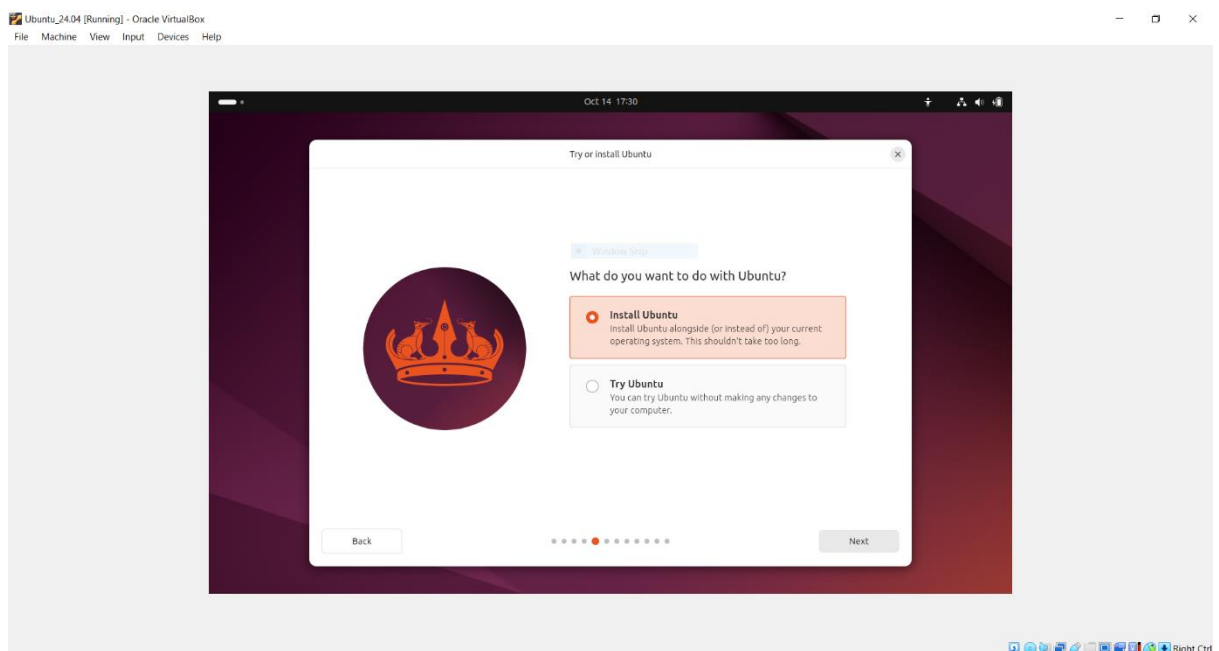
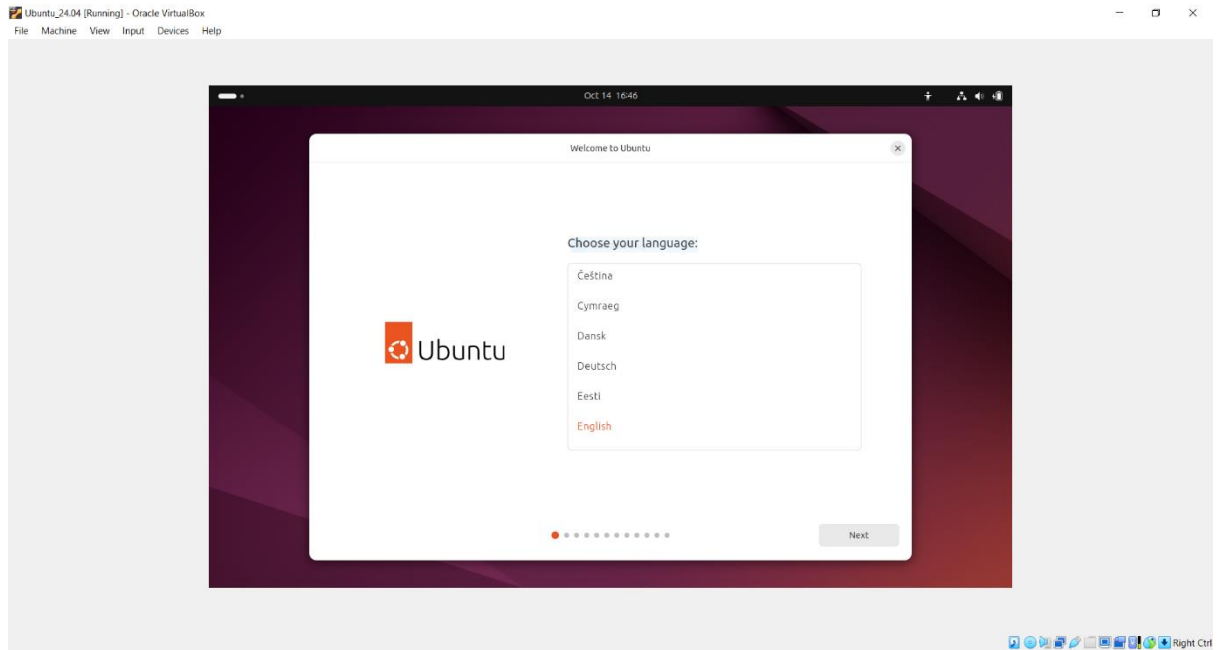
☐ Use an Existing Virtual Hard Disk File

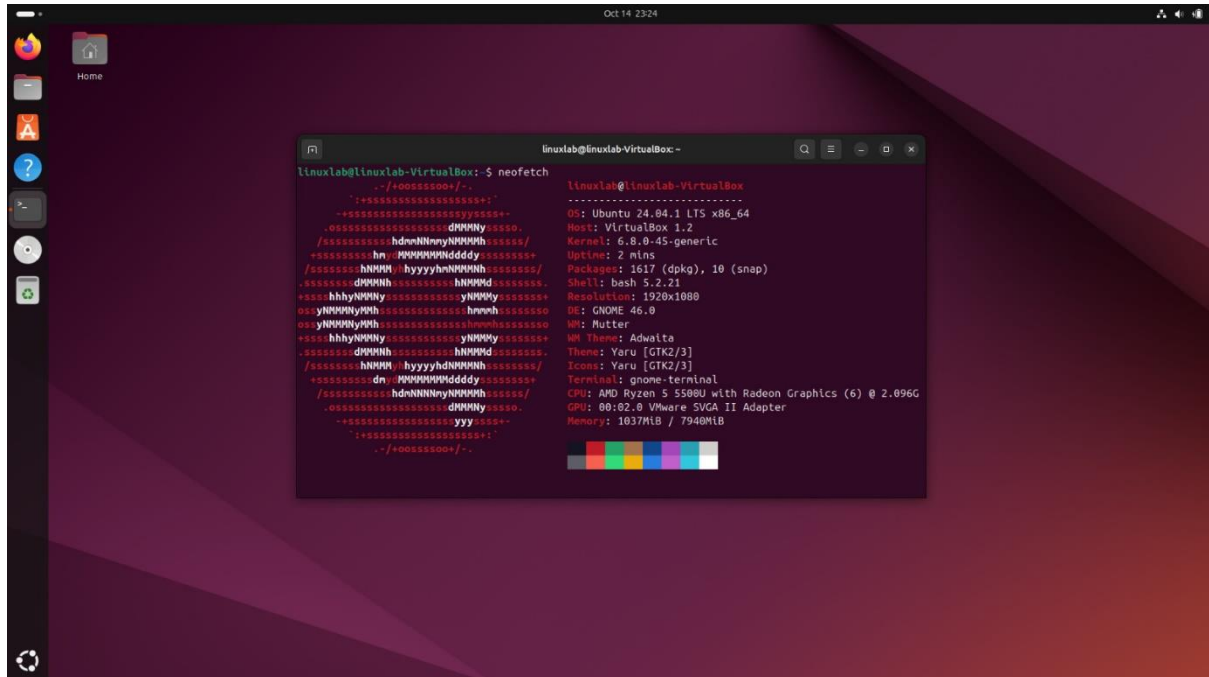
Empty 

☐ Do Not Add a Virtual Hard Disk

[Help](#) [Back](#) [Next](#) [Cancel](#)







Conclusion: Ubuntu has been successfully installed.

Practical 2

Aim: Study of logging/logout details.

Theory:

The process of logging in and logging out are fairly straightforward and similar to Windows.

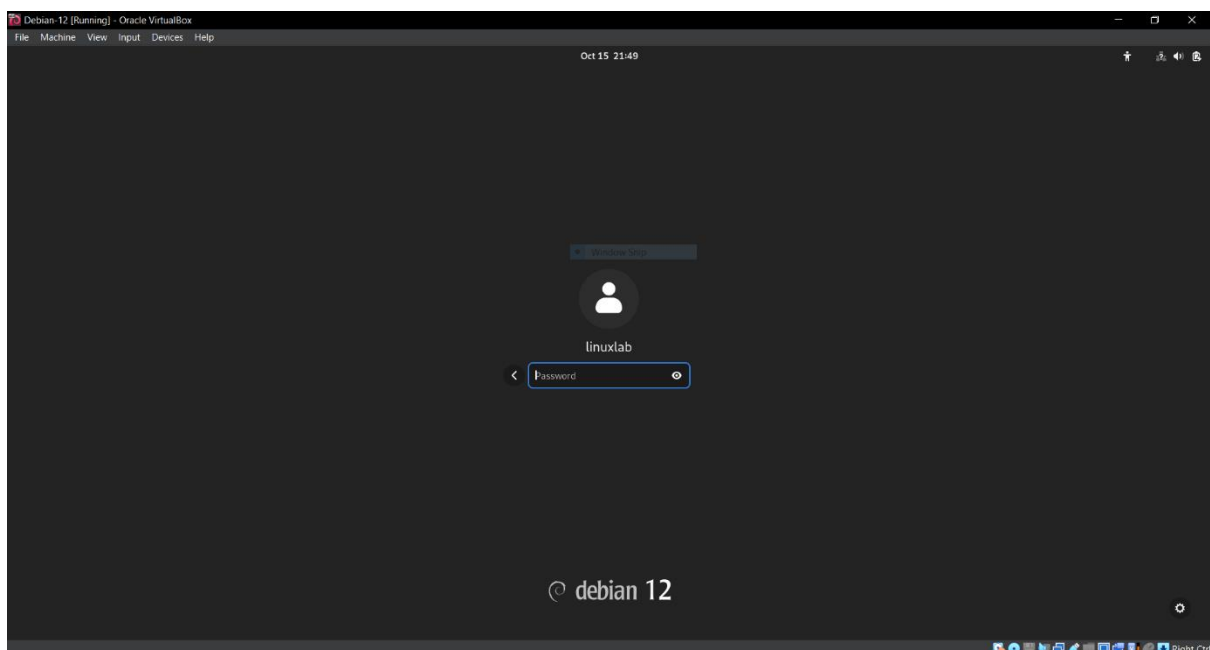
Logging and logout details are vital components of system administration and security management in multi-user environments. These logs provide a historical record of user activity, including login times, logout times, and session durations, which can be crucial for tracking user behavior and identifying security breaches.

The primary tools for monitoring user sessions in Linux include commands like `last`, `who`, and `w`, which retrieve information from log files such as `/var/log/wtmp` and `/var/log/btmp`. The `wtmp` file records all login and logout events, while the `btmp` file tracks failed login attempts, helping administrators detect unauthorized access attempts.

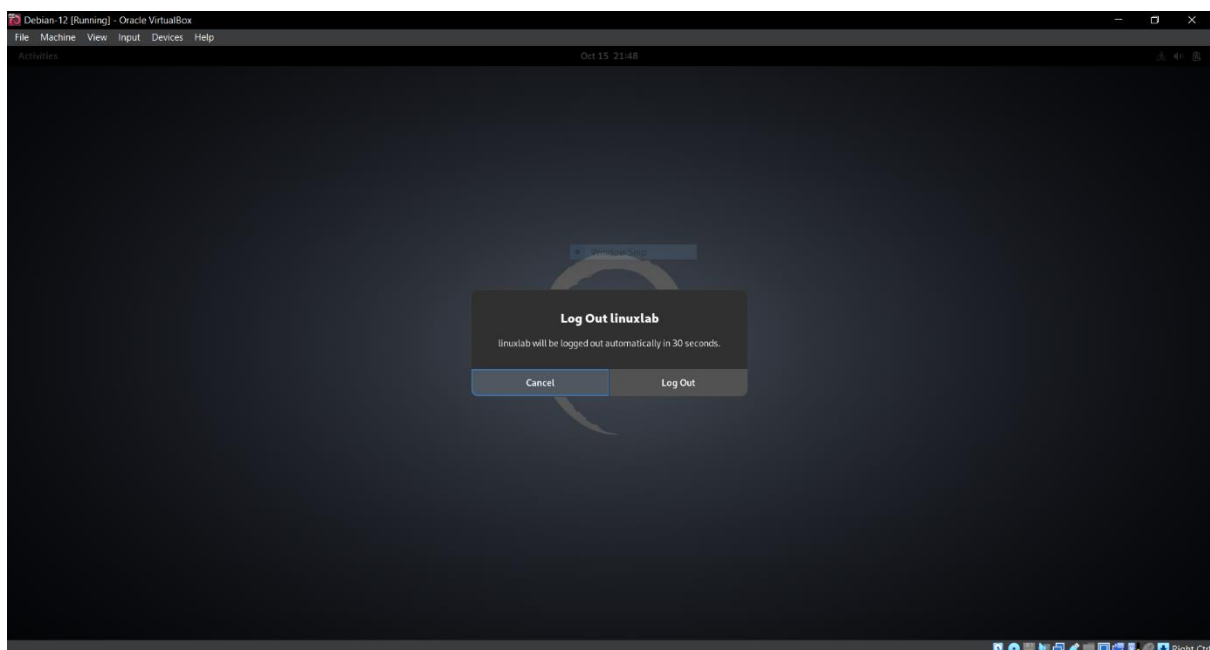
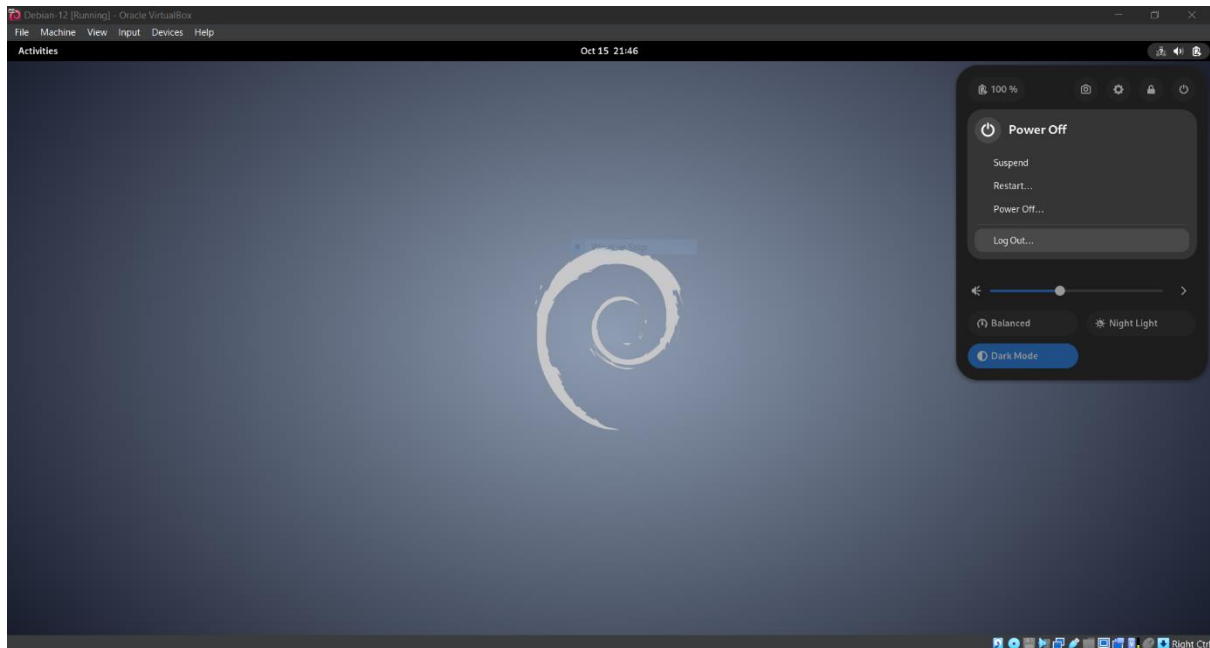
Monitoring login/logout details aids in resource management, ensuring users have appropriate access while minimizing the risk of unauthorized activities. Analyzing these logs can reveal patterns of usage, identify potential security threats, and facilitate compliance with security policies. Overall, maintaining a comprehensive record of user logins and logouts is essential for ensuring system integrity and security in any computing environment.

Procedure:

Logging In:



Logging Out:



Commands that can be used to view login/logout details:

1. last

- displays a list of all users who have logged in and out of the system.
- this command reads from the `/var/log/wtmp` file, showing user login/logout times and durations.


```
linuxlab@debian:~$ last
linuxlab tty2          tty2          Tue Oct 15 17:56      still logged in
reboot  system boot      6.1.0-26-amd64 Tue Oct 15 17:55      still running
linuxlab tty2          tty2          Tue Oct 15 17:36 -    down      (00:00)
reboot  system boot      6.1.0-26-amd64 Tue Oct 15 17:35 -    17:36    (00:01)
linuxlab tty2          tty2          Tue Oct 15 17:33 -    down      (00:07)
reboot  system boot      6.1.0-26-amd64 Tue Oct 15 17:32 -    17:40    (00:07)
linuxlab tty2          tty2          Tue Oct 15 17:30 -    down      (00:03)
reboot  system boot      6.1.0-26-amd64 Tue Oct 15 17:30 -    17:34    (00:03)
linuxlab tty2          tty2          Tue Oct 15 17:14 -    crash     (00:15)
reboot  system boot      6.1.0-26-amd64 Tue Oct 15 17:14 -    17:34    (00:20)
linuxlab tty2          tty2          Tue Oct 15 16:58 -    down      (00:12)
reboot  system boot      6.1.0-26-amd64 Tue Oct 15 16:58 -    17:11    (00:12)
linuxlab tty2          tty2          Tue Oct 15 16:56 -    down      (00:01)
reboot  system boot      6.1.0-26-amd64 Tue Oct 15 16:53 -    16:58    (00:05)
linuxlab tty2          tty2          Tue Oct 15 16:47 -    16:56    (00:09)
reboot  system boot      6.1.0-26-amd64 Tue Oct 15 16:47 -    16:56    (00:09)
linuxlab tty1          Tue Oct 15 16:35 -    down      (00:11)
reboot  system boot      6.1.0-26-amd64 Tue Oct 15 16:35 -    16:47    (00:11)

wtmp begins Tue Oct 15 16:35:36 2024
```

2. lastb

- displays a list of failed login attempts.
- this command reads from the `/var/log/btmp` file.

```
linuxlab@debian:~$ sudo lastb
[sudo] password for linuxlab:

btmp begins Tue Oct 15 16:27:26 2024
```

3. who: shows currently logged-in users and their login times.

```
linuxlab@debian:~$ who
linuxlab tty2          2024-10-15 17:56 (tty2)
```

4. w: provides information about logged-in users and what they are currently doing, including their login time and idle time.

```
linuxlab@debian:~$ w
 21:25:21 up  3:29,  1 user,  load average: 0.91, 1.25, 1.17
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
linuxlab  tty2          tty2             17:56    3:29m  0.04s  0.04s /usr/libexec/gnome-session-binary
```

5. utmpdump: dumps the contents of the utmp, wtmp, or btmp files in a readable format.

```
linuxlab@debian:~$ utmpdump /var/log/wtmp
Utmp dump of /var/log/wtmp
[2] [00000] [~~ ] [reboot ] [~      ] [6.1.0-26-amd64] [0.0.0.0] [2024-10-15T11:05:36,230054+00:00]
[1] [00053] [~~ ] [runlevel] [~      ] [6.1.0-26-amd64] [0.0.0.0] [2024-10-15T11:05:38,274828+00:00]
[5] [00559] [tty1] [      ] [tty1   ] [      ] [0.0.0.0] [2024-10-15T11:05:38,303057+00:00]
[6] [00559] [tty1] [LOGIN ] [tty1   ] [      ] [0.0.0.0] [2024-10-15T11:05:38,303057+00:00]
[7] [00559] [  ] [linuxlab] [tty1   ] [      ] [0.0.0.0] [2024-10-15T11:05:45,846494+00:00]
[1] [00000] [~~ ] [shutdown] [~      ] [6.1.0-26-amd64] [0.0.0.0] [2024-10-15T11:17:00,887417+00:00]
[2] [00000] [~~ ] [reboot ] [~      ] [6.1.0-26-amd64] [0.0.0.0] [2024-10-15T11:17:05,410395+00:00]
[1] [00053] [~~ ] [runlevel] [~      ] [6.1.0-26-amd64] [0.0.0.0] [2024-10-15T11:17:11,499739+00:00]
[7] [01212] [  ] [linuxlab] [tty2   ] [      ] [0.0.0.0] [2024-10-15T11:17:16,733085+00:00]
[8] [01212] [  ] [      ] [tty2   ] [      ] [0.0.0.0] [2024-10-15T11:26:26,326648+00:00]
[1] [00000] [~~ ] [shutdown] [~      ] [6.1.0-26-amd64] [0.0.0.0] [2024-10-15T11:26:26,802336+00:00]
[2] [00000] [~~ ] [reboot ] [~      ] [6.1.0-26-amd64] [0.0.0.0] [2024-10-15T11:23:18,337118+00:00]
[1] [00053] [~~ ] [runlevel] [~      ] [6.1.0-26-amd64] [0.0.0.0] [2024-10-15T11:26:45,157494+00:00]
[7] [01213] [  ] [linuxlab] [tty2   ] [      ] [0.0.0.0] [2024-10-15T11:26:49,116259+00:00]
[1] [00000] [~~ ] [shutdown] [~      ] [6.1.0-26-amd64] [0.0.0.0] [2024-10-15T11:28:27,229125+00:00]
[2] [00000] [~~ ] [reboot ] [~      ] [6.1.0-26-amd64] [0.0.0.0] [2024-10-15T11:28:21,941310+00:00]
[1] [00053] [~~ ] [runlevel] [~      ] [6.1.0-26-amd64] [0.0.0.0] [2024-10-15T11:28:40,982413+00:00]
[7] [01934] [  ] [linuxlab] [tty2   ] [      ] [0.0.0.0] [2024-10-15T11:28:44,196632+00:00]
[1] [00000] [~~ ] [shutdown] [~      ] [6.1.0-26-amd64] [0.0.0.0] [2024-10-15T11:41:20,830795+00:00]
[2] [00000] [~~ ] [reboot ] [~      ] [6.1.0-26-amd64] [0.0.0.0] [2024-10-15T11:44:19,823383+00:00]
[1] [00053] [~~ ] [runlevel] [~      ] [6.1.0-26-amd64] [0.0.0.0] [2024-10-15T11:44:36,177142+00:00]
[7] [02001] [  ] [linuxlab] [tty2   ] [      ] [0.0.0.0] [2024-10-15T11:44:39,264880+00:00]
```

6. finger: provides detailed information about users, including their login times and idle status.

```
linuxlab@debian:~$ sudo apt install finger
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  finger
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 19.8 kB of archives.
After this operation, 50.2 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bookworm/main amd64 finger amd64 0.17-17 [19.8 kB]
Fetched 19.8 kB in 0s (66.0 kB/s)
Selecting previously unselected package finger.
(Reading database ... 115119 files and directories currently installed.)
Preparing to unpack .../finger_0.17-17_amd64.deb ...
Unpacking finger (0.17-17) ...
Setting up finger (0.17-17) ...
Processing triggers for man-db (2.11.2-2) ...
linuxlab@debian:~$ finger
Login      Name      Tty      Idle   Login Time   Office      Office Phone
linuxlab   tty2      3:30    Oct 15 17:56 (tty2)
linuxlab@debian:~$
```

Conclusion: Used commands that can be used to get login/logout details.

Practical-3

Aim: Study of Unix/Linux general purpose utility command list obtained from (man, who, cat, cd, cp, ps, ls, mv, rm, mkdir, rmdir, echo, more, date, time, kill, history, chmod, chown, finger, pwd, cal, logout, shutdown) commands.

Procedure:

1. man command

Used to display the user manual of any command that we can run on the terminal.

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ man
What manual page do you want?
For example, try 'man man'.
linuxlab@linuxlab-VirtualBox:~/Desktop$ man echo
linuxlab@linuxlab-VirtualBox:~/Desktop$ echo hello
hello
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

2. pwd command

Used to find current path

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ pwd
/home/linuxlab/Desktop
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

3. ls command

```
linuxlab@linuxlab-VirtualBox:~$ ls
dead.letter  Documents  Music      Public      Templates  Vifile.txt
Desktop      Downloads  Pictures   shellfile.sh Videos
linuxlab@linuxlab-VirtualBox:~$
```

It is used to list files in current director.

4. ls -ltr command

It is used to list files by time in reverse order with long listing.

```
linuxlab@linuxlab-VirtualBox:~$ ls -ltr
total 44
drwxr-xr-x 2 linuxlab linuxlab 4096 Nov  9 17:39 Templates
drwxr-xr-x 2 linuxlab linuxlab 4096 Nov  9 17:39 Public
drwxr-xr-x 2 linuxlab linuxlab 4096 Nov 10 11:24 Documents
drwxr-xr-x 2 linuxlab linuxlab 4096 Nov 10 11:25 Downloads
drwxr-xr-x 2 linuxlab linuxlab 4096 Nov 10 11:25 Music
drwxr-xr-x 2 linuxlab linuxlab 4096 Nov 10 11:25 Pictures
drwxr-xr-x 2 linuxlab linuxlab 4096 Nov 10 11:25 Videos
-rw-rw-r-- 1 linuxlab linuxlab  21 Nov 10 11:40 Vifile.txt
-rw-rw-r-- 1 linuxlab linuxlab  18 Nov 10 11:46 shellfile.sh
-rw-rw-r-- 1 linuxlab mail    372 Nov 10 13:07 dead.letter
drwxr-xr-x 2 linuxlab linuxlab 4096 Nov 10 17:47 Desktop
linuxlab@linuxlab-VirtualBox:~$
```

5. history command

This command displays all the commands that were previously being executed by the user.

```
linuxlab@linuxlab-VirtualBox:~$ history
 1  docker
 2  sudo apt install docker.io
 3  docker --version
 4  sudo systemctl status docker
 5  sudo apt-get update
 6  sudo docker run hello-world
 7  docker images
 8  sudo docker images
 9  docker ps
10  sudo docker ps
11  sudo docker ps -a
12  sudo chmod compare.sh
13  cat > compare.sh
14  bash compare.sh
15  cat > string.sh
16  bash string.shj
17  bash string.sh
18  cat >logical.sh
```

6. ping command

The ping command(packet internet groper) checks connectivity status between host to server.Ping uses ICMP(Internet Control Message protocol) and sends an ICMP echo to the server.It takes an input of an IP address or URL.

```
linuxlab@linuxlab-VirtualBox:~$ ping google.com
PING google.com (142.250.192.110) 56(84) bytes of data.
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=1 ttl=58 time=26.4 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=2 ttl=58 time=28.2 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=3 ttl=58 time=26.6 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=4 ttl=58 time=27.5 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=5 ttl=58 time=27.4 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=6 ttl=58 time=27.4 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=7 ttl=58 time=27.6 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=8 ttl=58 time=27.0 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=9 ttl=58 time=27.9 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=10 ttl=58 t
```

7. df command-The df command is used to display the disk space used in the file system. It displays the output as in the number of used blocks, available blocks, and the mounted directory.

```
linuxlab@linuxlab-VirtualBox:~$ df
Filesystem      1K-blocks    Used Available Use% Mounted on
udev            723808         0      723808   0% /dev
tmpfs           151200      1352      149848   1% /run
/dev/sda5       9736500 8242036      980160  90% /
tmpfs           755984         0      755984   0% /dev/shm
tmpfs           5120         4         5116   1% /run/lock
tmpfs           755984         0      755984   0% /sys/fs/cgroup
/dev/loop5        128        128           0 100% /snap/bare/5
/dev/loop0       224256    224256           0 100% /snap/gnome-3-34-1804/66
/dev/loop1        66432     66432           0 100% /snap/gtk-common-themes/1514
/dev/loop3        31872     31872           0 100% /snap/snapd/11036
/dev/loop4        52352     52352           0 100% /snap/snap-store/518
/dev/loop7       224256    224256           0 100% /snap/gnome-3-34-1804/72
/dev/loop6       66816     66816           0 100% /snap/gtk-common-themes/1519
/dev/loop2       56832     56832           0 100% /snap/core18/1988
/dev/loop8       52224     52224           0 100% /snap/snap-store/547
/dev/loop10      43264     43264           0 100% /snap/snapd/13831
/dev/loop9       56832     56832           0 100% /snap/core18/2246
/dev/sda1       523248         4      523244   1% /boot/efi
tmpfs           151196        20      151176   1% /run/user/1000
linuxlab@linuxlab-VirtualBox:~$
```

8. cd command

The “cd” command is used to change the current directory.

```
linuxlab@linuxlab-VirtualBox:~$ cd Desktop
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

9. cat command

The “cat” command is a multi-purpose utility in the Linux system. It can be used to create a file, display content of the file, copy the content of one file to another file, and more.

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ cat > sample.txt
abc
def
aa
ads
^C
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

10. ps command

To view the processes that you’re running, use ps command.

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ ps
  PID TTY          TIME CMD
 2160 pts/0    00:00:00 bash
 2318 pts/0    00:00:00 ps
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

11. top command

Command used to view all the processes that are running.


```
linuxlab@linuxlab-VirtualBox:~/Desktop$ top
```

top - 18:15:12 up 21 min, 1 user, load average: 0.01, 0.07, 0.09
Tasks: 181 total, 1 running, 180 sleeping, 0 stopped, 0 zombie
%Cpu(s): 9.9 us, 1.4 sy, 0.0 ni, 88.4 id, 0.0 wa, 0.0 hi, 0.3 si, 0.0 st
MiB Mem : 1476.5 total, 86.9 free, 723.3 used, 666.4 buff/cache
MiB Swap: 448.5 total, 447.5 free, 1.0 used. 613.6 avail Mem

| PID | USER | PR | NI | VIRT | RES | SHR | S | %CPU | %MEM | TIME+ | COMMAND |
|------|----------|-----|-----|---------|--------|--------|---|------|------|---------|---------|
| 1290 | linuxlab | 20 | 0 | 3439632 | 326104 | 118224 | S | 6.0 | 21.6 | 0:33.75 | gnome-+ |
| 732 | linuxlab | 20 | 0 | 529888 | 58988 | 38268 | S | 3.0 | 3.9 | 0:07.77 | Xorg |
| 2152 | linuxlab | 20 | 0 | 823280 | 50748 | 38280 | S | 2.0 | 3.4 | 0:02.33 | gnome-+ |
| 12 | root | 20 | 0 | 0 | 0 | 0 | S | 0.3 | 0.0 | 0:00.17 | ksofti+ |
| 1621 | linuxlab | 20 | 0 | 912320 | 32408 | 21688 | S | 0.3 | 2.1 | 0:00.38 | gsd-me+ |
| 2321 | linuxlab | 20 | 0 | 20488 | 3688 | 3176 | R | 0.3 | 0.2 | 0:00.03 | top |
| 1 | root | 20 | 0 | 167776 | 11584 | 8436 | S | 0.0 | 0.8 | 0:02.17 | systemd |
| 2 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | kthrea+ |
| 3 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 | rcu_gp |
| 4 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 | rcu_pa |
| 6 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 | kworke |
| 9 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 | mm_per |
| 10 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | rcu_ta |
| 11 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | rcu_ta |
| 13 | root | 20 | 0 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.85 | rcu_sc |
| 14 | root | rt | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.02 | migrat |
| 15 | root | -51 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | idle_i |
| 16 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | cpuhp/0 |
| 17 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | kdevtm |

12. mkdir command

To create a new directory, use “mkdir”.

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ mkdir linux
linuxlab@linuxlab-VirtualBox:~/Desktop$ ls
linux sample.txt
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

13. rmdir command

To remove an empty directory use “rmdir”.

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ rmdir linux
linuxlab@linuxlab-VirtualBox:~/Desktop$ ls
sample.txt
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

14. head command

“head” command displays the top part of a file. It displays the first 10 lines in a file.

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ head sample.txt
abc
def
aa
ads
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

15. head command for n lines

Command used to display the n number of lines in a file.

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ head -n 2 sample.txt
abc
def
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

16. tail command

“tail” command displays the last part of a file. It displays the last 10 lines in a file.

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ tail sample.txt
abc
def
aa
ads
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

17. tail command for n lines

Command used to display the n number of lines in a file.

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ tail -n 2 sample.txt
aa
ads
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

18. cp command

The “cp” command is used to copy a file or directory.

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ cat abcd.txt
abc
def
aa
ads
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

19. id command

The “id” command is used to display the user ID (UID) and group ID (GID).

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ id
uid=1000(linuxlab) gid=1000(linuxlab) groups=1000(linuxlab),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

20. wc command

The “wc” command is used to count the lines, words, and characters in a file.

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ wc sample.txt
 4  4 15 sample.txt
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

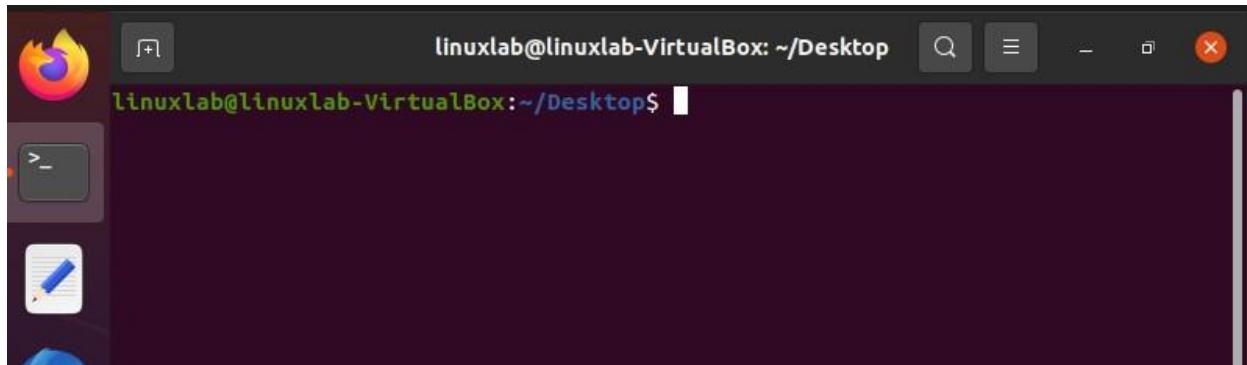
21. host command

The “host” command is used to display the IP address for a given domain name and vice versa. It performs the DNS lookups for the DNS Query.

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ host google.com
google.com has address 142.250.192.142
google.com has IPv6 address 2404:6800:4009:825::200e
google.com mail is handled by 10 aspmx.l.google.com.
google.com mail is handled by 30 alt2.aspmx.l.google.com.
google.com mail is handled by 50 alt4.aspmx.l.google.com.
google.com mail is handled by 40 alt3.aspmx.l.google.com.
google.com mail is handled by 20 alt1.aspmx.l.google.com.
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

22. clear command

Linux **clear** command is used to clear the terminal screen.



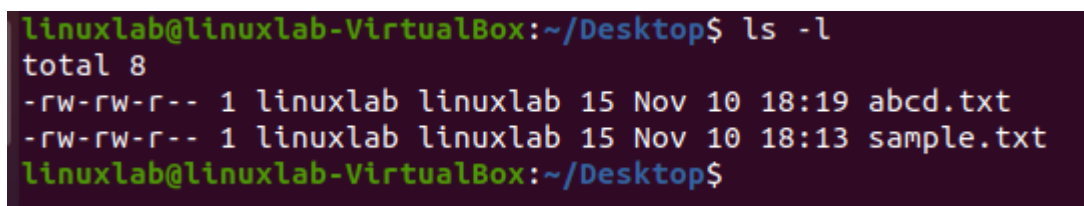
23. less command

“less” displays a file, allowing forward/backward movement within it.



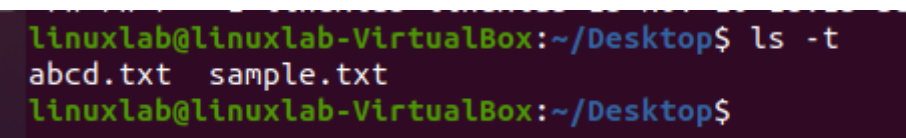
24. ls -l command

Command used to display the files in long list format.



25. ls -t command

Command used to display the files in sorting format of time modification.



26. ls -h command

Command used to display the file sizes in human readable format.

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ ls -h
abcd.txt  sample.txt
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

27. ls -r command

Command used to display the files in reverse order format.

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ ls -r
sample.txt  abcd.txt
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

28. ip command-

: Linux “ip” command is an updated version of the ipconfig command. It is used to assign an IP address, initialize an interface, disable an interface.

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ ip
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
       ip [ -force ] -batch filename
where  OBJECT := { link | address | addrlabel | route | rule | neigh | ntable |
                  tunnel | tuntap | maddress | mroute | mrule | monitor | xfrm
                  |
                  netns | l2tp | fou | macsec | tcp_metrics | token | netconf
                  | ila |
                  vrf | sr | nexthop }
       OPTIONS := { -V[ersion] | -s[tatistics] | -d[etails] | -r[esolve] |
                   -h[uman-readable] | -iec | -j[son] | -p[retty] |
                   -f[amily] { inet | inet6 | mpls | bridge | link } |
                   -4 | -6 | -I | -D | -M | -B | -O |
                   -l[oops] { maximum-addr-flush-attempts } | -br[ief] |
                   -o[neline] | -t[imestamp] | -ts[hort] | -b[atch] [filename]
                   |
                   -rc[vbuf] [size] | -n[etns] name | -N[umeric] | -a[ll] |
                   -c[olor]}
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

29. exit command

Linux **exit** command is used to exit from the current shell.

Conclusion: Studied the general-purpose utility commands for Linux

Practical 4

Aim: Study of Vi Editor

Theory:

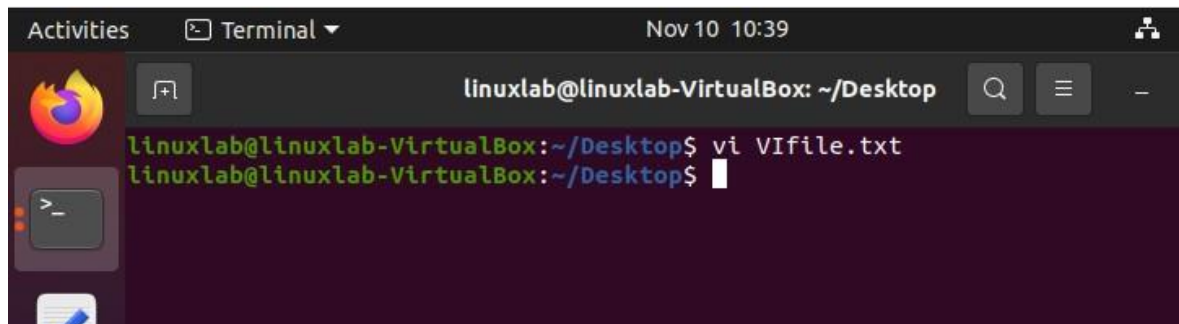
Vi Editor is used to edit files in Unix. It is done using the screen-oriented text editor, vi is one of the best ways. This editor enables you to edit lines in context with other lines in the file. An improved version of the vi editor which is called the VIM has also been made available now. Here, VIM stands for vi improved.

- vi is generally considered the de facto standard in Unix editors because –
- It's usually available on all the flavors of Unix system.
- Its implementations are very similar across the board.
- It requires very few resources.
- It is more user-friendly than other editors such as the ed or the ex.

You can use the vi editor to edit an existing file or to create a new file from scratch. You can also use this editor to just read a text file.

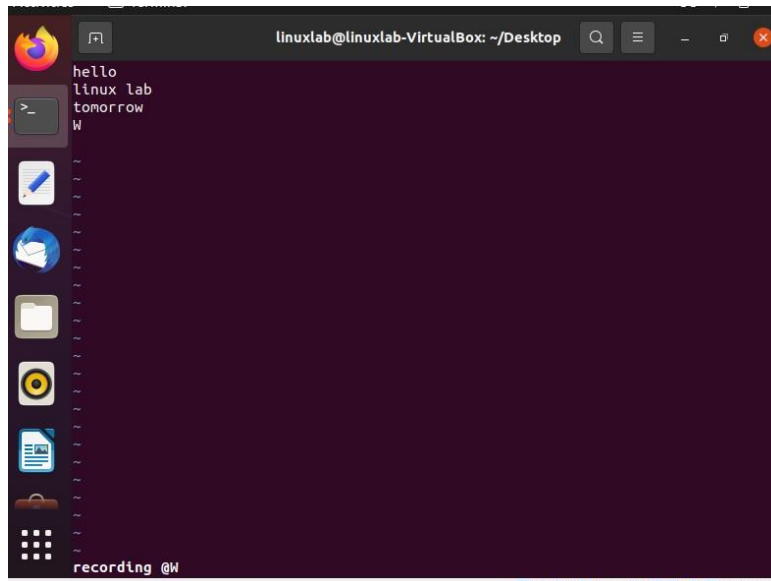
Procedure:

1. **vi filename command:** Creates a new file if it already does not exist, otherwise opens an existing file.

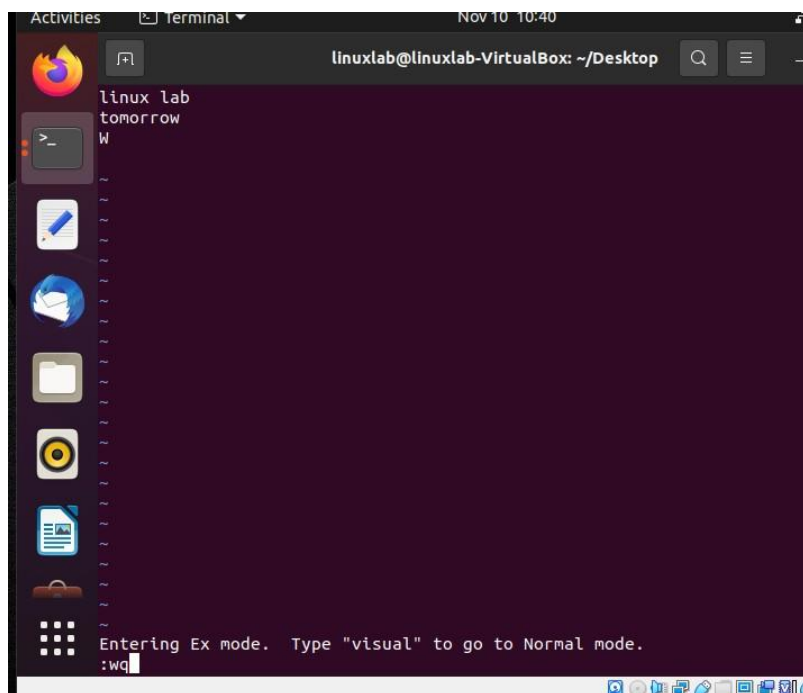


```
Activities  Terminal  Nov 10 10:39
linuxlab@linuxlab-VirtualBox: ~/Desktop
linuxlab@linuxlab-VirtualBox:~/Desktop$ vi VIfile.txt
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

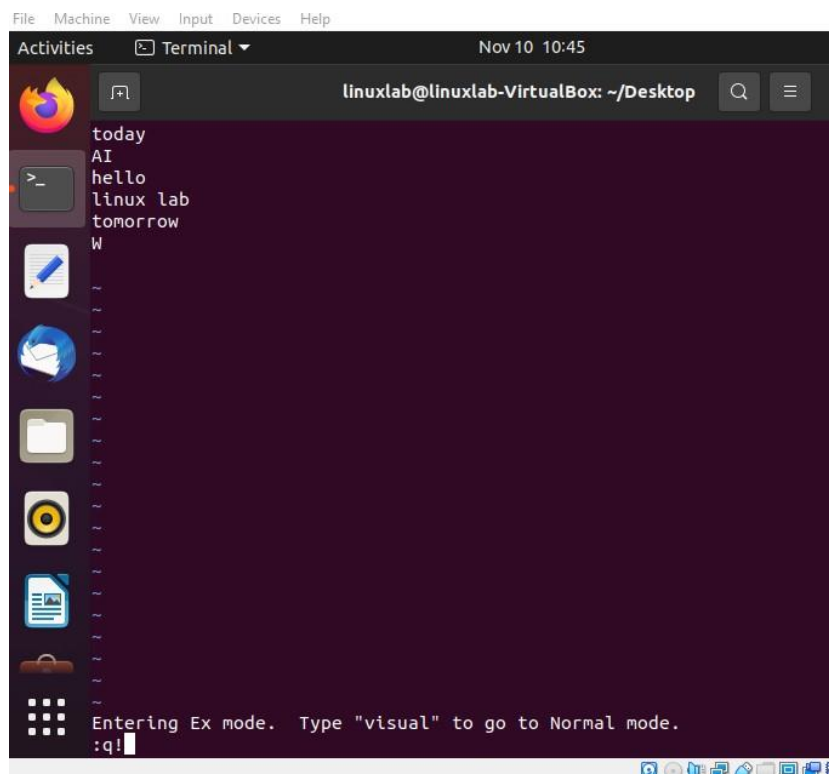
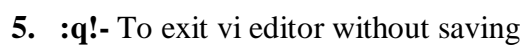
2. Inserting in vi: use 'I' to enter insertion mode



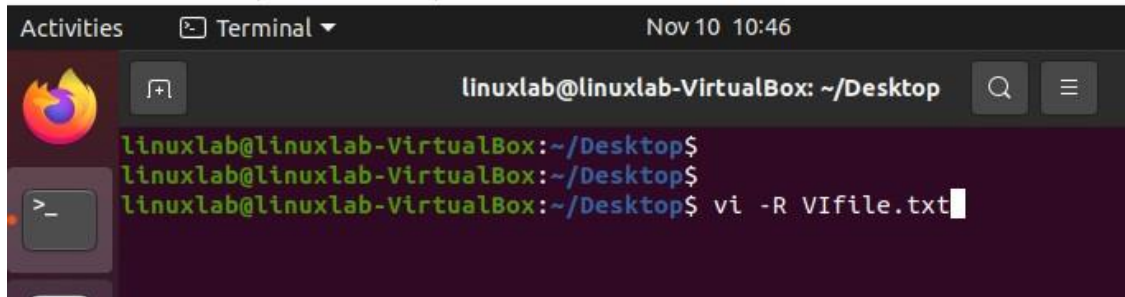
3. :wq- To exit vi editor



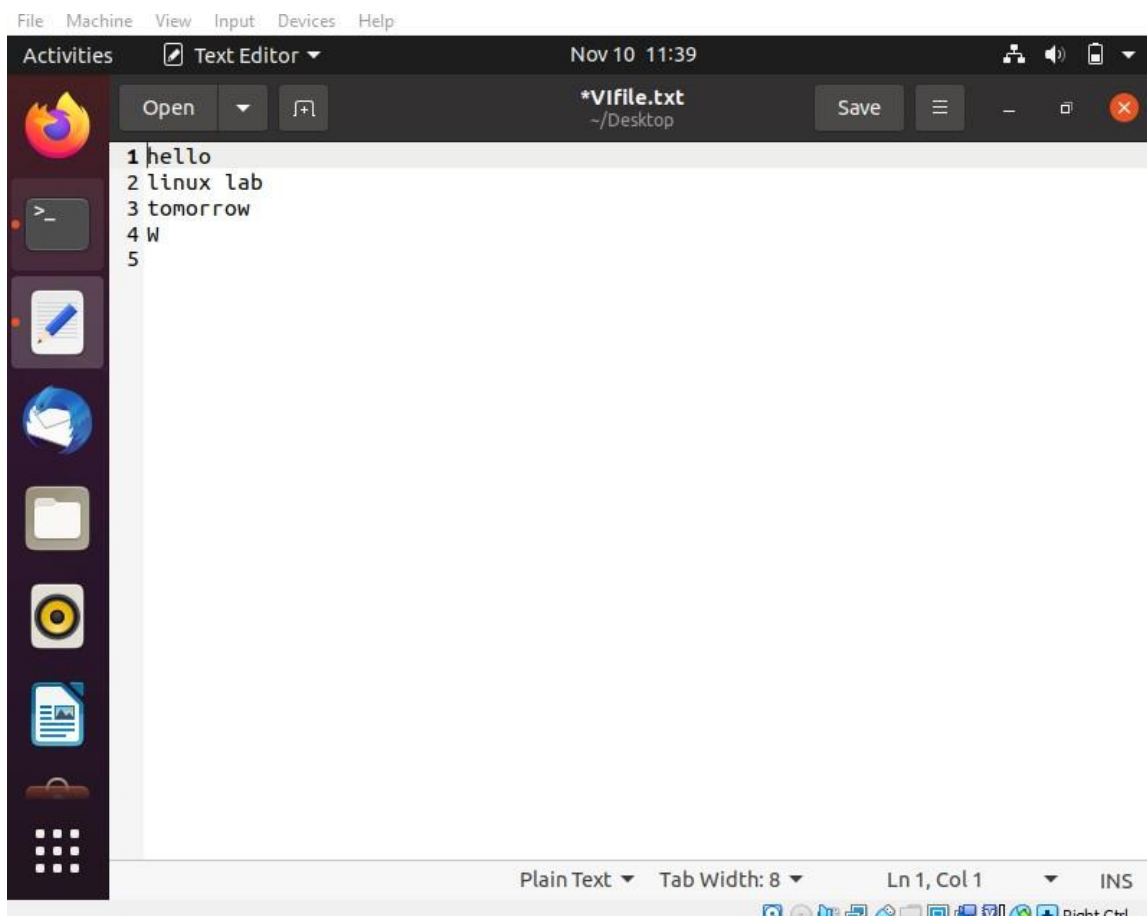
4. :w- To save edited content in vi editor



6. **vi -R filename:** Opens an existing file in the read-only mode.



```
linuxlab@linuxlab-VirtualBox: ~/Desktop
linuxlab@linuxlab-VirtualBox:~/Desktop$
linuxlab@linuxlab-VirtualBox:~/Desktop$ vi -R Vifile.txt
```



```
File Machine View Input Devices Help
Activities Text Editor Nov 10 11:39
*Vifile.txt
~/Desktop Save
1 hello
2 linux lab
3 tomorrow
4 W
5
Plain Text Tab Width: 8 Ln 1, Col 1 INS
Right Ctrl
```

Conclusion: Studied the VI Editor

Practical 5

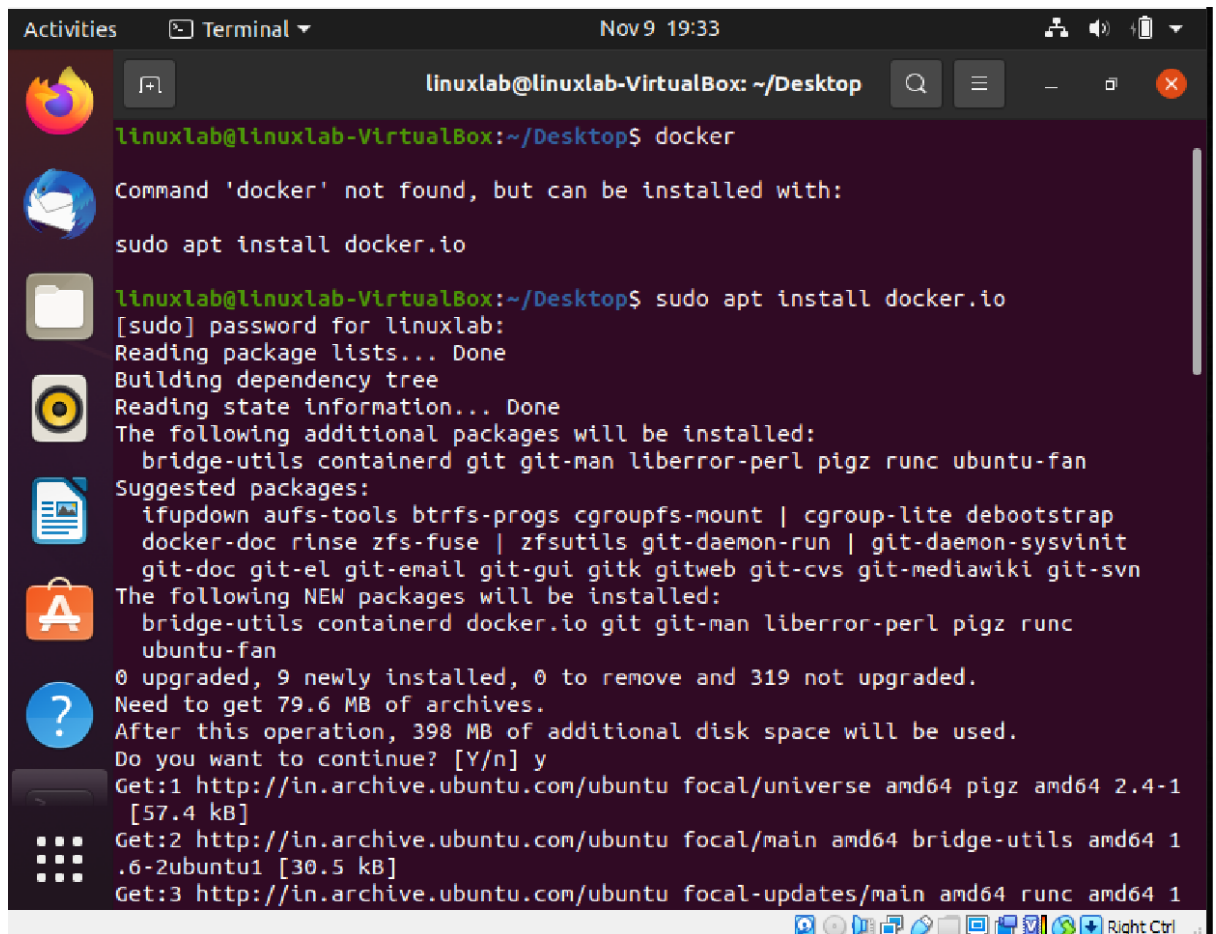
Aim: Install Docker and run :Start, stop, push, pull, log, docker ps, docker ps -a,create account, dockercompose, docker build, docker run

Theory:

- Docker is an open source containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment. Containers simplify delivery of distributed applications, and have become increasingly popular as organizations shift to cloud-native development and hybrid multicloud environments.
- Developers can create containers without Docker, but the platform makes it easier, simpler, and safer to build, deploy and manage containers. Docker is essentially a toolkit that enables developers to build, deploy, run, update, and stop containers using simple commands and work-saving automation through a single API.
- Docker is so popular today that “Docker” and “containers” are used interchangeably. But the first container-related technologies were available for years — even decades (link resides outside IBM) — before Docker was released to the public in 2013.

Procedure:

1. Installing Docker

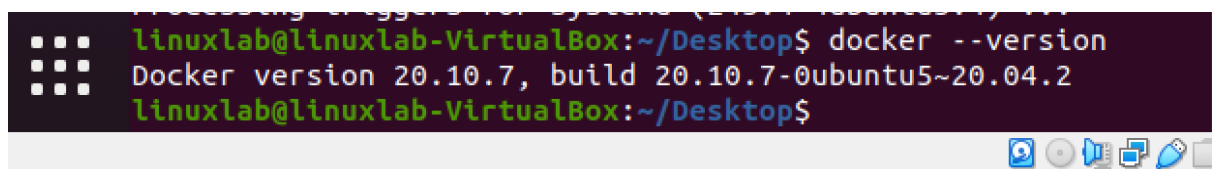


The image shows a terminal window titled 'linuxlab@linuxlab-VirtualBox: ~/Desktop' with a search bar and window controls. The user enters the command 'docker', which results in a message: 'Command 'docker' not found, but can be installed with:'. The user then enters 'sudo apt install docker.io'. The terminal shows the progress of the installation, including reading package lists, building a dependency tree, and listing additional packages to be installed (bridge-utils, containerd, git, git-man, liberror-perl, pigz, runc, ubuntu-fan). It also shows suggested packages and the disk space requirements. The user confirms the installation with 'y'. The terminal shows the progress of downloading the packages from the Ubuntu repositories.

```
linuxlab@linuxlab-VirtualBox: ~/Desktop$ docker
Command 'docker' not found, but can be installed with:
sudo apt install docker.io

linuxlab@linuxlab-VirtualBox: ~/Desktop$ sudo apt install docker.io
[sudo] password for linuxlab:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd git git-man liberror-perl pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap
  docker-doc rinse zfs-fuse | zfsutils git-daemon-run | git-daemon-sysvinit
  git-doc git-el git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  bridge-utils containerd docker.io git git-man liberror-perl pigz runc
  ubuntu-fan
0 upgraded, 9 newly installed, 0 to remove and 319 not upgraded.
Need to get 79.6 MB of archives.
After this operation, 398 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 pigz amd64 2.4-1
    [57.4 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu focal/main amd64 bridge-utils amd64 1
    .6-2ubuntu1 [30.5 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 runc amd64 1
```

2. Checking version of docker



The image shows a terminal window titled 'linuxlab@linuxlab-VirtualBox: ~/Desktop' with a search bar and window controls. The user enters the command 'docker --version'. The terminal shows the output: 'Docker version 20.10.7, build 20.10.7-0ubuntu5~20.04.2'.

```
linuxlab@linuxlab-VirtualBox: ~/Desktop$ docker --version
Docker version 20.10.7, build 20.10.7-0ubuntu5~20.04.2
linuxlab@linuxlab-VirtualBox: ~/Desktop$
```

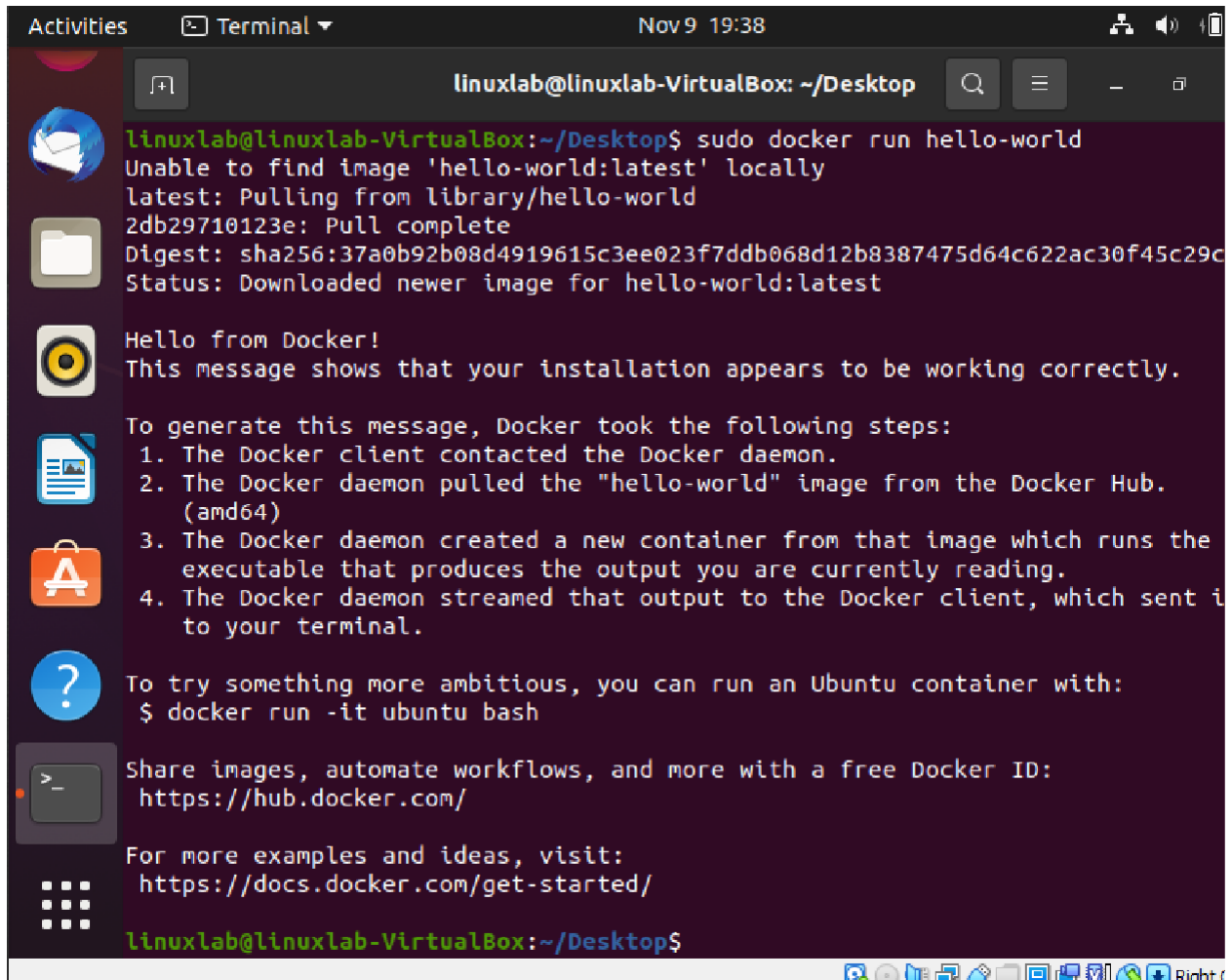
3. Checking enable status of docker(active/inactive)

```
linuxlab@linuxlab-VirtualBox: ~/Desktop
D Thunderbird Mail 20.10.7, build 20.10.7-0ubuntu5-20.04.2
linuxlab@linuxlab-VirtualBox:~/Desktop$
linuxlab@linuxlab-VirtualBox:~/Desktop$
linuxlab@linuxlab-VirtualBox:~/Desktop$
linuxlab@linuxlab-VirtualBox:~/Desktop$ sudo systemctl status docker
[sudo] password for linuxlab:
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor prese
   Active: active (running) since Tue 2021-11-09 19:33:45 IST; 1min 45s ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 2507 (dockerd)
      Tasks: 8
     Memory: 50.4M
    CGroup: /system.slice/docker.service
            └─2507 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/con

Nov 09 19:33:40 linuxlab-VirtualBox dockerd[2507]: time="2021-11-09T19:33:40.5
Nov 09 19:33:40 linuxlab-VirtualBox dockerd[2507]: time="2021-11-09T19:33:40.5
Nov 09 19:33:40 linuxlab-VirtualBox dockerd[2507]: time="2021-11-09T19:33:40.5
Nov 09 19:33:40 linuxlab-VirtualBox dockerd[2507]: time="2021-11-09T19:33:40.5
Nov 09 19:33:42 linuxlab-VirtualBox dockerd[2507]: time="2021-11-09T19:33:42.3
Nov 09 19:33:42 linuxlab-VirtualBox dockerd[2507]: time="2021-11-09T19:33:42.7
Nov 09 19:33:45 linuxlab-VirtualBox dockerd[2507]: time="2021-11-09T19:33:45.3
Nov 09 19:33:45 linuxlab-VirtualBox dockerd[2507]: time="2021-11-09T19:33:45.3
Nov 09 19:33:45 linuxlab-VirtualBox dockerd[2507]: time="2021-11-09T19:33:45.3
Nov 09 19:33:45 linuxlab-VirtualBox systemd[1]: Started Docker Application Con
Nov 09 19:33:45 linuxlab-VirtualBox dockerd[2507]: time="2021-11-09T19:33:45.5
lines 1-21/21 (END)
```

```
Activities Terminal Nov 9 19:36
linuxlab@linuxlab-VirtualBox: ~/Desktop
linuxlab@linuxlab-VirtualBox:~/Desktop$ sudo apt-get update
Hit:1 http://in.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://in.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu focal-updates/main i386 Packages [559
kB]
Get:6 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [1,
338 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Metad
ata [279 kB]
Get:8 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metada
ta [14.4 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu focal-updates/universe amd64 DEP-11 M
etadeta [362 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packa
ges [24.6 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 DEP-1
1 Metadata [944 B]
Get:12 http://in.archive.ubuntu.com/ubuntu focal-backports/universe amd64 DEP-1
1 metadata [10.4 kB]
Get:13 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [98
2 kB]
Get:14 http://security.ubuntu.com/ubuntu focal-security/main i386 Packages [303
kB]
Get:15 http://security.ubuntu.com/ubuntu focal-security/main Translation-en [18
2 kB]
Get:16 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metad
ata [29.0 kB]
```

4. Run hello world

A terminal window titled 'linuxlab@linuxlab-VirtualBox: ~/Desktop' showing the execution of the 'hello-world' Docker container. The terminal output includes the command 'sudo docker run hello-world', the message 'Unable to find image 'hello-world:latest' locally', the pull progress 'latest: Pulling from library/hello-world', the digest '2db29710123e: Pull complete', and the status 'Digest: sha256:37a0b92b08d4919615c3ee023f7ddb068d12b8387475d64c622ac30f45c29c'. It then displays 'Hello from Docker!' and a message stating that the installation appears to be working correctly. A list of four steps explains the process: 1. The Docker client contacted the Docker daemon. 2. The Docker daemon pulled the "hello-world" image from the Docker Hub. (amd64) 3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading. 4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal. It also provides a tip to try something more ambitious by running an Ubuntu container with the command '\$ docker run -it ubuntu bash' and links to Docker Hub and documentation.

```
linuxlab@linuxlab-VirtualBox: ~/Desktop$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:37a0b92b08d4919615c3ee023f7ddb068d12b8387475d64c622ac30f45c29c
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

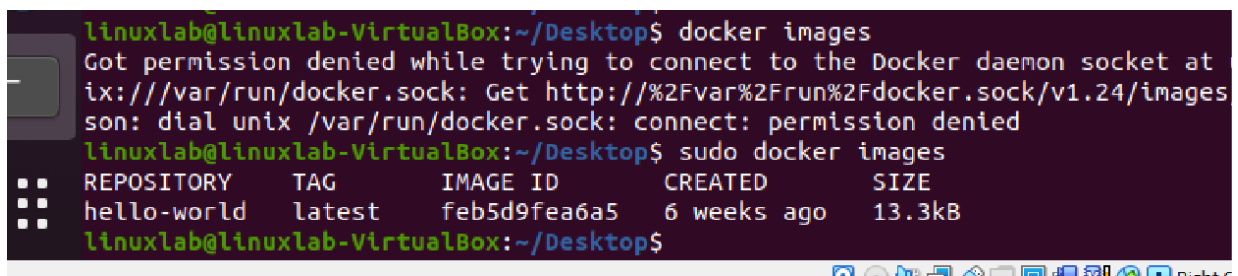
To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

linuxlab@linuxlab-VirtualBox:~/Desktop$
```

5. Run image pulled from repository

A terminal window titled 'linuxlab@linuxlab-VirtualBox: ~/Desktop' showing the execution of the 'docker images' command. The first attempt fails with the error 'Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get http://%2Fvar%2Frun%2Fdocker.sock/v1.24/images son: dial unix /var/run/docker.sock: connect: permission denied'. The second attempt, using 'sudo', succeeds and displays a table of Docker images.

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ docker images
Got permission denied while trying to connect to the Docker daemon socket at
ix:///var/run/docker.sock: Get http://%2Fvar%2Frun%2Fdocker.sock/v1.24/images
son: dial unix /var/run/docker.sock: connect: permission denied
linuxlab@linuxlab-VirtualBox:~/Desktop$ sudo docker images
REPOSITORY      TAG         IMAGE ID      CREATED       SIZE
hello-world     latest     feb5d9fea6a5  6 weeks ago  13.3kB
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

6. Checking if containers are still running in machine

docker ps: Checks if container is running

```
linuxlab@linuxlab-VirtualBox:~/Desktop$ docker ps
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/json: dial unix /var/run/docker.sock: connect: permission denied
linuxlab@linuxlab-VirtualBox:~/Desktop$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
linuxlab@linuxlab-VirtualBox:~/Desktop$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
4fc3eb5d21b3   hello-world  "/hello"   About a minute ago   Exited (0) About a minute ago   clever_galois
linuxlab@linuxlab-VirtualBox:~/Desktop$
```

No container is running

Conclusion: Installed Docker and did various commands

Practical 6

Aim: Study of Bash shell, Bourne shell and C shell in Unix/Linux operating system.

Theory:

In Unix, there are two major types of shells –

1. **Bourne shell** – If you are using a Bourne-type shell, the \$ character is the default prompt.
2. **C shell** – If you are using a C-type shell, the % character is the default prompt.

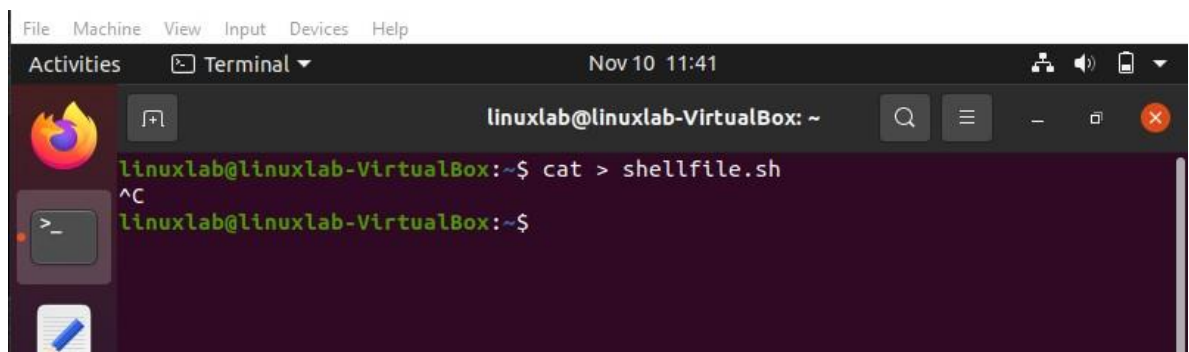
The Bourne Shell has the following subcategories: Bourne shell (sh), Korn shell (ksh), Bourne Again shell (bash), POSIX shell (sh).

- The different C-type shells follow: C shell (csh), TENEX/TOPS C shell (tcsh).
- The original Unix shell was written in the mid-1970s by Stephen R. Bourne while he was at the AT&T Bell Labs in New Jersey.
- Bourne shell was the first shell to appear on Unix systems, thus it is referred to as "the shell". Bourne shell is usually installed as **/bin/sh** on most versions of Unix. For this reason, it is the shell of choice for writing scripts that can be used on different versions of Unix.

Bash is a command language interpreter. It is widely available on various operating systems and is a default command interpreter on most GNU/Linux systems. The name is an acronym for the '**Bourne-Again Shell**'. Bash is a shell program. Bash is a command processor that typically runs in a text window where the user types commands that cause actions. Bash can also read and execute commands from a file, called a shell script. A shell program is typically an executable binary that takes commands that you type and (once you hit return), translates those commands into (ultimately) system calls to the Operating System API. Bash is not the only kind of shell. Other shells include: sh, ash, dash, ksh, tcsh, zsh, tcsh.

Procedure:

1. **Creating <filename>.sh file:**



The screenshot shows a terminal window titled 'linuxlab@linuxlab-VirtualBox: ~'. The terminal displays the command `cat > shellfile.sh` being entered at the prompt. The prompt is `linuxlab@linuxlab-VirtualBox:~$`. The terminal window has a menu bar with 'File', 'Machine', 'View', 'Input', 'Devices', and 'Help'. The title bar shows 'Nov 10 11:41' and standard window controls. The terminal output shows the command being entered and the prompt returning.

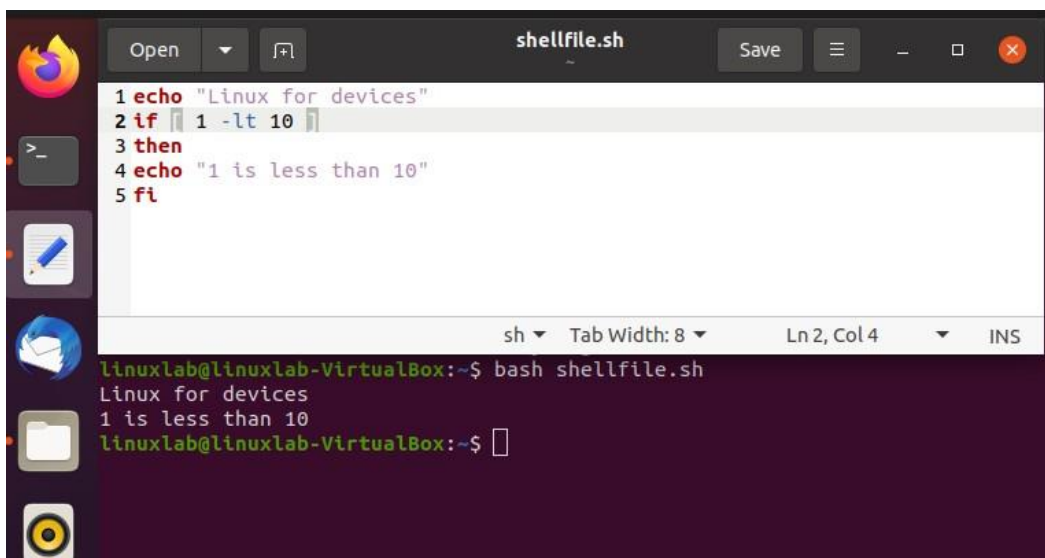
2. Inserting linux command in .sh file:



```
1 echo "Linux for devices"
2 if 1 -lt 10
3 then
4 echo "1 is less than 10"
5 fi
```

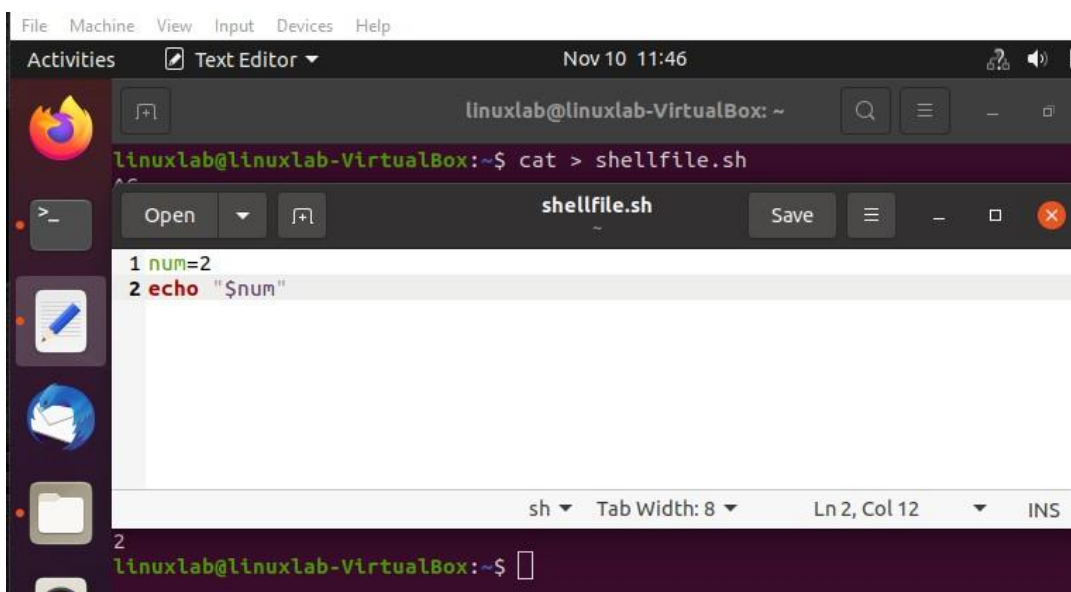
3. Saving and exited from .sh file:

4. running the .sh file:



```
linuxlab@linuxlab-VirtualBox:~$ bash shellfile.sh
Linux for devices
1 is less than 10
linuxlab@linuxlab-VirtualBox:~$
```

5. Creating Variable in .sh file and executing it:



```
linuxlab@linuxlab-VirtualBox:~$ cat > shellfile.sh
1 num=2
2 echo "$num"
2
linuxlab@linuxlab-VirtualBox:~$
```

Practical 7

Aim: Study of Unix/Linux file system (tree structure).

Theory:

The Unix/Linux file system is a hierarchical directory structure that organizes files and directories into a tree-like layout. It starts at the root directory (/) and branches into various subdirectories, each containing files and other directories. This design makes it easier to manage, locate, and access files.

The key principles behind the Unix/Linux file system are:

- **Hierarchical Structure:** Files and directories are organized in a hierarchical tree.
- **Everything is a File:** Devices, directories, and even system information are treated as files.
- **Modular Components:** Different directories serve different purposes, such as storing system binaries, user data, or configuration files.

Unix/Linux File System Structure

The Unix/Linux file system can be visualized as a tree with the root directory (/) as the top node. From there, it branches into various directories, each serving a specific function.

Some key directories in the Unix/Linux file system:

1. **/(Root):** The base of the file system hierarchy. All other directories and files stem from here.
2. **/bin:** Contains essential system binaries (commands like ls, cat, etc.).
3. **/boot:** Holds files required for booting the system, such as the kernel.
4. **/dev:** Contains device files, representing hardware devices (e.g., disks, terminals).
5. **/etc:** Stores system-wide configuration files.
6. **/home:** The home directories for regular users.
7. **/lib:** Shared library files used by binaries in /bin and other programs.
8. **/usr:** User binaries, documentation, libraries, and source code.
9. **/var:** Files that change frequently during system operation, like logs, cache, and spool files.
10. **/tmp:** Temporary files created by users and processes.
11. **/proc:** A virtual filesystem containing runtime system information.
12. **/mnt or /media:** Temporary mount points for removable devices.

Procedure:

1. **Accessing the File System:** Open a terminal on your Unix/Linux system.
2. **Navigating Directories:** Use the `cd` (change directory) command to navigate between directories.
3. **Listing Directory Contents:** Use the `ls` command to list the contents of a directory.
4. **Displaying the Tree Structure:** Use the `tree` command to view the directory structure in a tree format. Install it if it's not available by default.

```
ubuntu@ip-172-31-2-57:/home$ tree
.
├── ubuntu
│   └── snap
│       └── tree
│           ├── 54
│           ├── common
│           └── current -> 54
```

5. Checking File Types

- Use `ls -l` to check file types and permissions. Files are prefixed with:
 - - for regular files
 - d for directories
 - l for symbolic links
 - c for character devices
 - b for block devices

```
ubuntu@ip-172-31-2-57:/home$ ls -l
total 4
drwxr-x--- 5 ubuntu ubuntu 4096 Oct 15 16:15 ubuntu
```

6. **Viewing Disk Usage:** Use `du` (disk usage) to analyze the space taken up by files and directories.


```

ubuntu@ip-172-31-2-57:/home$ du
4      ./ubuntu/snap/tree/common/.local/lib/locale
8      ./ubuntu/snap/tree/common/.local/lib
12     ./ubuntu/snap/tree/common/.local
16     ./ubuntu/snap/tree/common
4      ./ubuntu/snap/tree/54
24     ./ubuntu/snap/tree
28     ./ubuntu/snap
4      ./ubuntu/.ssh
4      ./ubuntu/.cache
52     ./ubuntu
56     .

```

7. **Viewing File System Usage:** The `df` command shows the available and used disk space on all mounted file systems.

```

ubuntu@ip-172-31-2-57:/home$ df

```

| Filesystem | 1K-blocks | Used | Available | Use% | Mounted on |
|-------------|-----------|---------|-----------|------|----------------|
| /dev/root | 7034376 | 1707400 | 5310592 | 25% | / |
| tmpfs | 490200 | 0 | 490200 | 0% | /dev/shm |
| tmpfs | 196084 | 872 | 195212 | 1% | /run |
| tmpfs | 5120 | 0 | 5120 | 0% | /run/lock |
| /dev/xvda16 | 901520 | 77020 | 761372 | 10% | /boot |
| /dev/xvda15 | 106832 | 6246 | 100586 | 6% | /boot/efi |
| tmpfs | 98040 | 12 | 98028 | 1% | /run/user/1000 |

Conclusion: The Unix/Linux file system is designed for flexibility, security, and efficiency. The tree structure allows for easy navigation, organized storage, and quick access to files and directories.

Practical 8

Aim: Study of .bashrc, /etc/bashrc and Environment variables.

Theory:

The .bashrc file is a script that Bash runs whenever a new terminal session is started in interactive mode (i.e., when you open a new terminal or log in to a shell session). Located in the user's home directory (~/.bashrc), it allows users to customize their shell environment by defining aliases, functions, and environment variables, as well as configuring shell behavior. It's a powerful tool for automation and personalization of terminal sessions, allowing users to tweak settings for optimal productivity.

Typical things you can configure in .bashrc include:

- **Aliases:** Shortcuts for common commands (e.g., `alias ll='ls -lah'`).
- **Prompt customization:** Adjusting how the shell prompt looks (e.g., displaying the current directory).
- **Shell options:** Setting terminal options like history size or enabling colored output.
- **Program execution:** Automating the execution of certain scripts or commands when a terminal session starts.

and much more.

It is a lot like a startup script anytime the user logs in and starts a terminal session.

Environmental variables are dynamic values that affect the behavior of processes running on a Linux system. These variables contain system-wide and user-specific data such as the location of executable files, the user's home directory, and configuration settings for various applications.

Key examples of environmental variables:

- **PATH:** Specifies directories where the system looks for executable files.
- **HOME:** Points to the current user's home directory.
- **USER:** Contains the username of the current user.
- **SHELL:** Defines the default shell (e.g., /bin/bash).

bashrc is the personal startup/setup script for individual users, /etc/bashrc has some global settings and functions which are defined for all users on the workstation.

Procedure:

For this experiment, we will tweak the .bashrc file to add a startup prompt and assign values to some environmental variables

```
# echo Greetings User, the current date and time is $(date)
Greetings User, the current date and time is Tue 15 Oct 21:31:08 IST 2024
#
```

This small commandlet is the perfect for a fresh bash session. To see this everytime we log in, lets modify the ~/.bashrc file and add this commandlet at the end of it.

```
# vim ~/.bashrc  
#
```

```
# .bashrc  
  
# Source global definitions  
if [ -f /etc/bashrc ]; then  
    . /etc/bashrc  
fi  
  
# User specific environment  
if ! [[ "$PATH" =~ "$HOME/.local/bin:$HOME/bin:" ]]  
then  
    PATH="$HOME/.local/bin:$HOME/bin:$PATH"  
fi  
  
PATH=/go/bin:$PATH  
export PATH  
  
# Uncomment the following line if you don't like systemctl's auto-paging feature:  
# export SYSTEMD_PAGER=  
  
alias ss='sudo su'  
alias xx='exit'  
alias op='xdg-open'  
alias cls="clear"  
alias clip="xclip -selection clipboard"  
export PS1="# "  
# User specific aliases and functions  
if [ -d ~/.bashrc.d ]; then  
    for rc in ~/.bashrc.d/*; do  
        if [ -f "$rc" ]; then  
            . "$rc"  
        fi  
    done  
fi  
  
echo Greetings User, the date and time is $(date)  
  
export SESSION_START=$(date)
```

Then exit using **:q!**

We have also added a variable called `SESSION_START` to keep track of how much time the terminal has been open for.

Now if we open a new terminal, we are greeted with the following prompt.

```
Greetings User, the date and time is Tue 15 Oct 21:38:07 IST 2024
# source ~/.bashrc
Greetings User, the date and time is Tue 15 Oct 21:38:20 IST 2024
#
```

Note: Loading a new shell session can also be triggered with the source command as shown above.

Now in this session, a variable SESSION_START exists which holds the value(of date and time) when the terminal session began

```
Greetings User, the date and time is Tue 15 Oct 21:38:20 IST 2024
# echo $SESSION_START
Tue 15 Oct 21:38:20 IST 2024
#
```

We can add another prompt to the /etc/bashrc to see both of them execute.

```
# sudo vim /etc/bashrc
#
```

Since its a file all the users(including root), it cannot be changed without sudo privileges

```
# /etc/bashrc

# System wide functions and aliases
# Environment stuff goes in /etc/profile

# It's NOT a good idea to change this file unless you know what you
# are doing. It's much better to create a custom.sh shell script in
# /etc/profile.d/ to make custom changes to your environment, as this
# will prevent the need for merging in future updates.

echo "This is from the /etc/bashrc"

# Prevent doublesourcing
if [ -z "$BASHRC_SOURCED" ]; then
    BASHRC_SOURCED="Y"

# are we an interactive shell?
if [ "$PS1" ]; then
    if [ -z "$PROMPT_COMMAND" ]; then
        case $TERM in
            xterm*)
                if [ -e /etc/sysconfig/bash-prompt-xterm ]; then
                    PROMPT_COMMAND=/etc/sysconfig/bash-prompt-xterm
                else
                    PROMPT_COMMAND='printf "\033]0;%s@%s:%s\007" "${USER}" "${HOSTNAME%%.*}" "${PWD/#$HOME/~}"'
                fi
            ;;
            screen*)
                if [ -e /etc/sysconfig/bash-prompt-screen ]; then
                    PROMPT_COMMAND=/etc/sysconfig/bash-prompt-screen
                else
                    PROMPT_COMMAND='printf "\033k%s@%s:%s\033\\" "${USER}" "${HOSTNAME%%.*}" "${PWD/#$HOME/~}"'
                fi
            ;;
            *)
                [ -e /etc/sysconfig/bash-prompt-default ] && PROMPT_COMMAND=/etc/sysconfig/bash-prompt-default
            ;;
        esac
    fi
# Turn on parallel history
shopt -s histappend
# Turn on checkwinsize
shopt -s checkwinsize
[ "$PS1" = "\s-\v\\\$ " ] && PS1="\u@\h \W]\\\$ "
# You might want to have e.g. tty in prompt (e.g. more virtual machines)
# and console windows
```

Now sourcing the bashrc of any user will show both prompts

```
# source ~/.bashrc
This is from the /etc/bashrc
Greetings User, the date and time is Tue 15 Oct 21:43:46 IST 2024
#
```

Conclusion: The usage of environmental variables and utilisation of the .bashrc and /etc/bashrc files were closely studied and their contribution in making a linux system more optimal for individual performance has been established.

Practical 9

Aim: Write a shell script program to display list of user currently logged in.

Theory:

- `#!/bin/bash`: This is the shebang line that tells the system to use the Bash shell to execute the script.
- `echo "List of users currently logged in:"`: This prints a message to the terminal.
- `who`: This command lists all users currently logged into the system, along with their login details (e.g., terminal, login time).

Procedure:

1. Open a text editor and paste the code.

```
#!/bin/bash
# Script to display the list of users currently logged in

echo "List of users currently logged in:"
who
```

2. Save the file with a `.sh` extension, for example `who.sh`.

```
ubuntu@ip-172-31-2-57:/home$ sudo nano who.sh
```

3. Open a terminal and navigate to the directory where the script is saved.
4. Make the script executable by running

```
ubuntu@ip-172-31-2-57:/home$ sudo chmod +x who.sh
```

5. Run the script.

```
List of users currently logged in:
ubuntu pts/0      2024-10-15 16:06 (13.233.177.5)
```

Conclusion: Script was made and executed successfully.

PRACTICAL 10

Aim: Write a shell script program to display “HELLO WORLD”. write a shell script program to develop a scientific calculator.

Theory:

A Shell Script is a text file containing a sequence of commands for the Unix-based shell to execute. Shell scripting allows users to automate repetitive tasks by combining multiple commands in a single script file. The scripting process reduces manual effort and ensures consistent execution of commands.

One of the most common shells is the Bash (Bourne Again Shell), and Bash scripting involves writing commands that the Bash interpreter can understand and execute. Every shell script starts with a Shebang (`#!/`) to specify the interpreter. For Bash scripts, the Shebang is `#!/bin/bash`, ensuring that the system runs the script using the Bash interpreter.

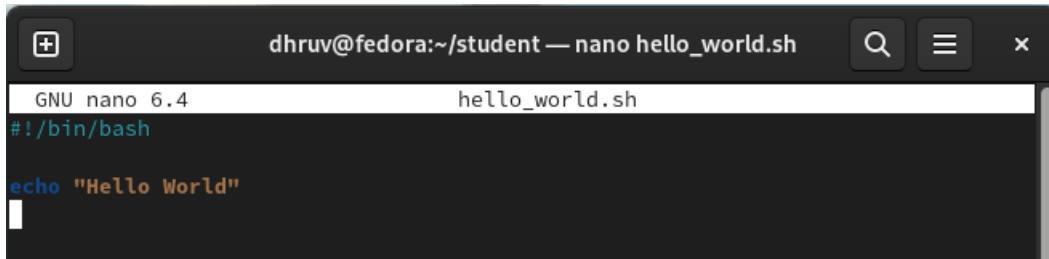
The echo command is a built-in command in Bash scripting that outputs text to the terminal. It is commonly used to display messages or the result of commands in scripts. In this practical, we will use the echo command to display "HELLO WORLD."

Procedure:

1. **Open Terminal:** Open the terminal window on your system.
2. **Create a Script File:**
 - a. Navigate to the directory where you want to save the script using the `cd` command.
 - b. Use the `touch` command to create an empty shell script file. Name it `hello_world.sh`:
Syntax: `touch hello_world.sh`

```
[dhruv@fedora student]$ touch hello_world.sh
```

3. **Open the Script for Editing:** Open the script file in a text editor like `nano`:
`nano hello_world.sh`
4. **Add the Shebang Line:** In the text editor, start by adding the Shebang line to ensure the script is executed using the Bash shell:
`#!/bin/bash`
5. **Write the Script:** Use the `echo` command to write the program that displays "HELLO WORLD":
`echo "HELLO WORLD"`

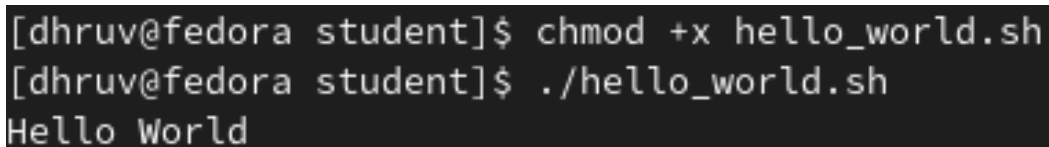


```
dhruv@fedora:~/student — nano hello_world.sh
GNU nano 6.4 hello_world.sh
#!/bin/bash
echo "Hello World"
```

6. **Save the Script:** Save the file and close the text editor.
7. **Make the Script Executable:** In the terminal, change the permissions of the script to make it executable:

`chmod +x hello_world.sh`
8. **Run the Script:** Execute the script using the following command:

`./hello_world.sh`
9. **View Output:** The script will display "HELLO WORLD" on the terminal.



```
[dhruv@fedora student]$ chmod +x hello_world.sh
[dhruv@fedora student]$ ./hello_world.sh
Hello World
```

Conclusion: Successfully created a shell script to display “Hello World”