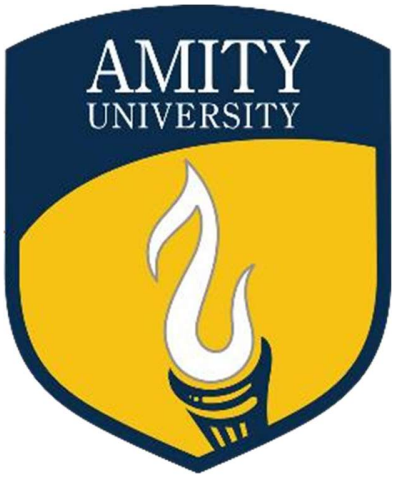
AMITY UNIVERSITY UTTAR PRADESH NOIDA

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



LINUX FOR DEVICES

[CSE 438]

PRACTICAL FILE

Submitted By:
Saumya Singh
A2305221578
B. TECH(CSE)
BATCH 2021-2025
7CSE-6Y

Submitted To:
Mr. Roshan Lal

INDEX-

S No.	Name of Program	Date of Conduct	Maximum Marks	Marks Obtained	Signature of faculty
1.	Installation of Linux Operating System.	30/07/2024			
2.	Study of Unix/Linux general purpose utility command list obtained from (man, cat, cd, cp, ps, ls, mv, rm, mkdir, rmdir, date, time, chmod, pwd, cal) commands.	13/08/2024			
3.	Study of vi editor, Study of bash shell,bourne shell and C shell in Unix/Linux operating system.	20/08/2024			
4.	Write a shell script program to display “HELLO WORLD”,	27/08/2024			
5.	Write a shell script program to develop a scientific calculator.	03/09/2024			
6.	Install Docker and run : Start, stop, push, pull, log, docker ps, docker ps –a,create account, docker compose, docker build, docker run	10/09/2024			
7.	Write a shell script program to illustrate the implementation of following: a) Integer Comparison b) String comparison c) Logical operators d) File tests e) Conditional control structure f) Loop control structures	17/09/2024			

8.	Write a shell script to a) print a number in reverse order. b)to reverse the string and reverse each string further in the list c)find the sum of all numbers in a file in Linux d) validate password strength. Here are a few assumptions for the password string.Length – minimum of 8 characters,Contain both alphabet and number,Include both the small and capital case letters.	24/09/2024			
9.	Design and develop a “Birthday Reminder” that can automatically send birthday wishes with a personalized message via email.	24/09/2024			
10.	Check whether strings in csv are palindrome.	01/10/2024			
11.	Implement NIC Bonding and Teaming in Linux	01/10/2024			

- **ls -ltr:** 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:~$
```

- **history:** 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
```

- **ping:** 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 ti
me=26.4 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=2 ttl=58 ti
me=28.2 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=3 ttl=58 ti
me=26.6 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=4 ttl=58 ti
me=27.5 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=5 ttl=58 ti
me=27.4 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=6 ttl=58 ti
me=27.4 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=7 ttl=58 ti
me=27.6 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=8 ttl=58 ti
me=27.0 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=9 ttl=58 ti
me=27.9 ms
64 bytes from bom12s17-in-f14.1e100.net (142.250.192.110): icmp_seq=10 ttl=58 t
```

- **df:** The df command is used to display the disk space used in the file system. Itdisplays 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:~$
```

- **rmdir:** To remove and empty directory use “rmdir”.

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

- **head**: “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$
```

- **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$
```

- **tail:** “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$
```

- **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$
```

- **cp:** 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$
```

- **id:** 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$
```

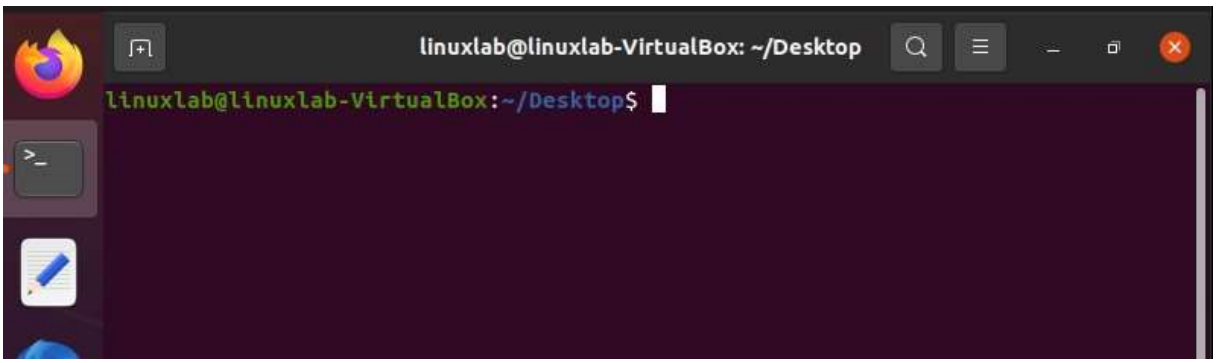
- **wc:** The “wc” command is used to count the lines, words, and characters in a file.

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

- **host:** 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.

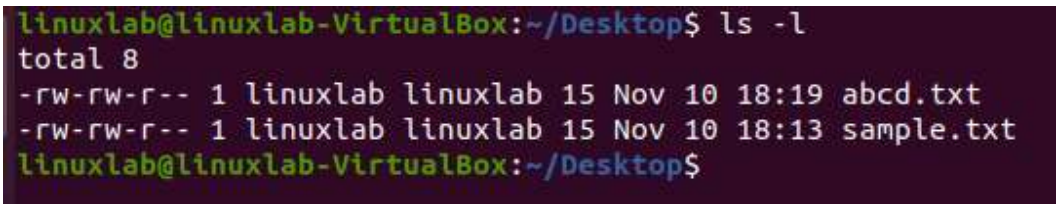
```
linuxlab@linuxlab-VirtualBox:~/Desktop$ host google.com
google.com has address 142.250.192.142
google.com has IPv6 address 2404:6800:4009:825::200e
google.com mail is handled by 10 aspmx.l.google.com.
google.com mail is handled by 30 alt2.aspmx.l.google.com.
google.com mail is handled by 50 alt4.aspmx.l.google.com.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

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

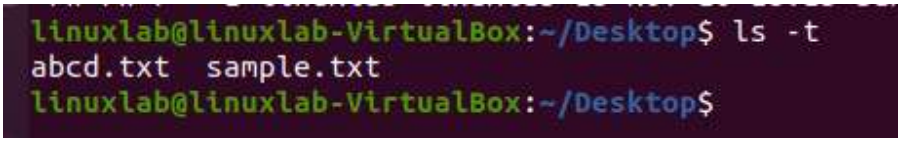


- **less:** “less” displays a file, allowing forward/backward movement within it.

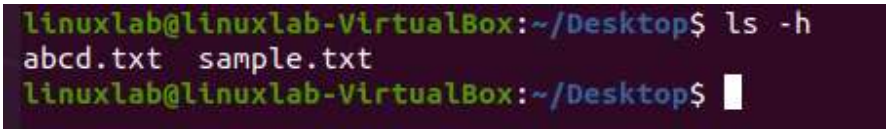
- **ls -l:** Command used to display the files in long list format.



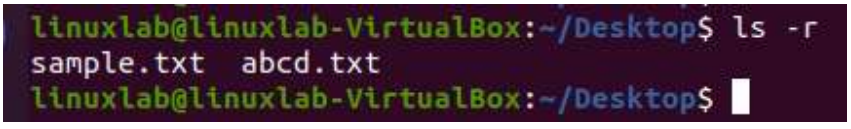
- **ls -t:** Command used to display the files in sorting format of time modification.



- **ls -h:** Command used to display the file sizes in human readable format.



- **ls -r:** Command used to display the files in reverse order format.



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

- **exit command**

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

Practical -3

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

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.

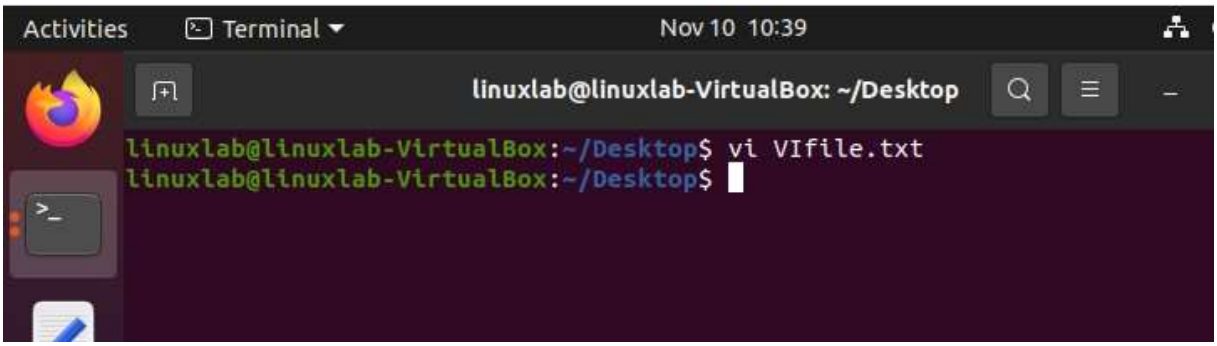
In Unix, there are two major types of shells −

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.

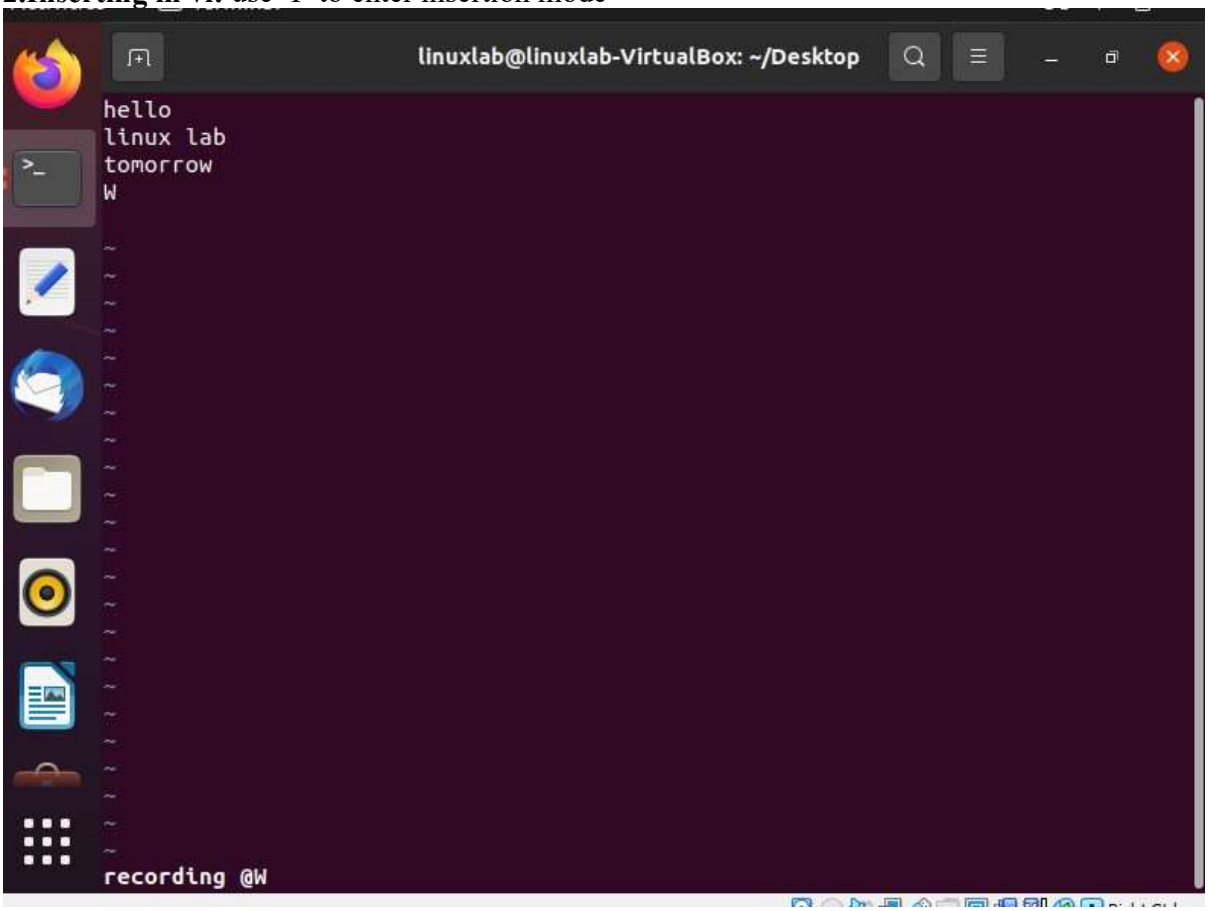
The Bourne Shell has the following subcategories: Bourne shell (sh), Korn shell (ksh), Bourne Again shell (bash), POSIX shell (sh).
The different C-type shells follow: C shell (csh), TENEX/TOPS C shell (tcsh).
The original Unix shell was written in the mid-1970s by Stephen R. Bourne while he was at the 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.

PROCEDURE:

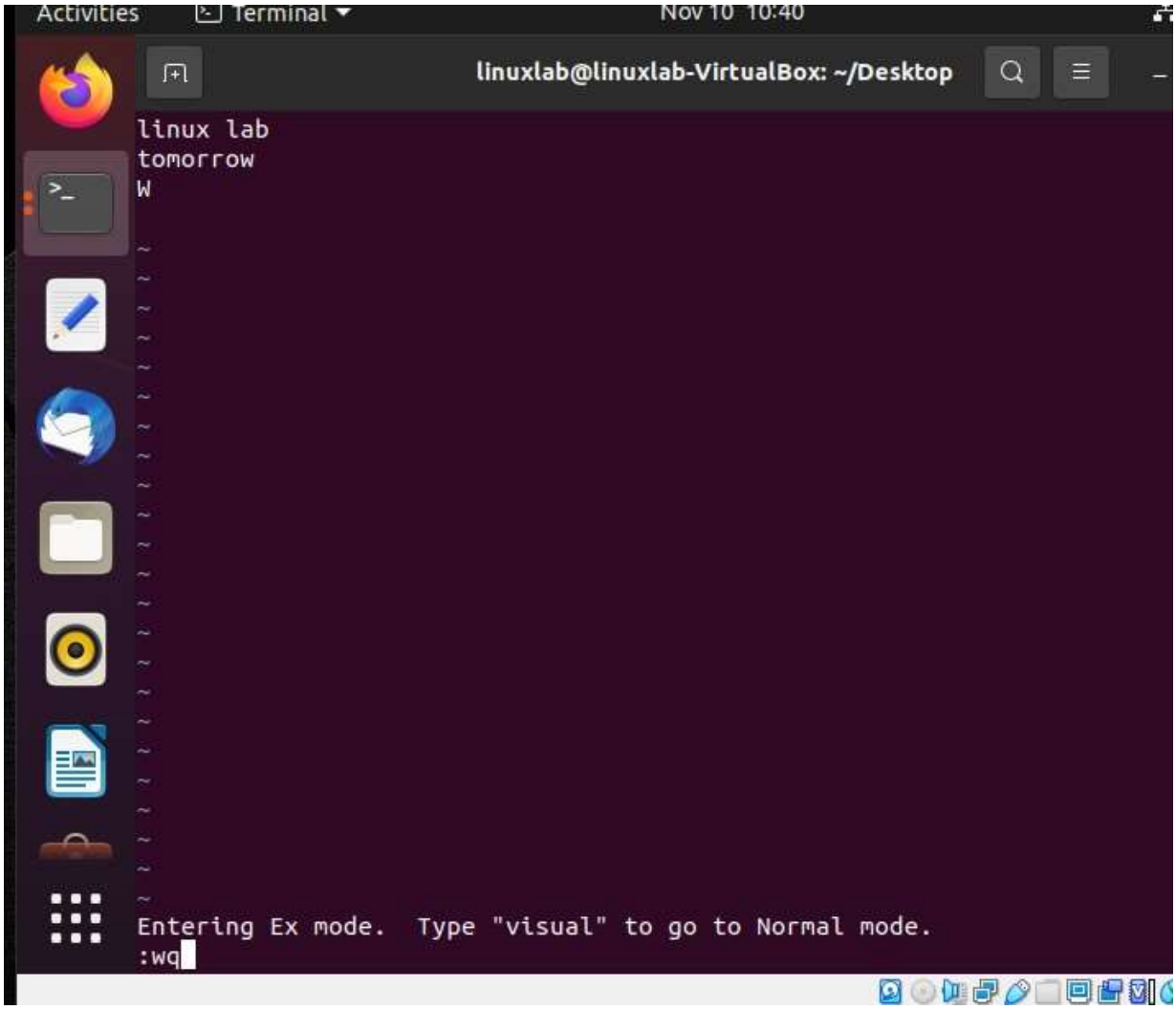
1. **vi filename command:** Creates a new file if it already does not exist, otherwise opens an existing file.



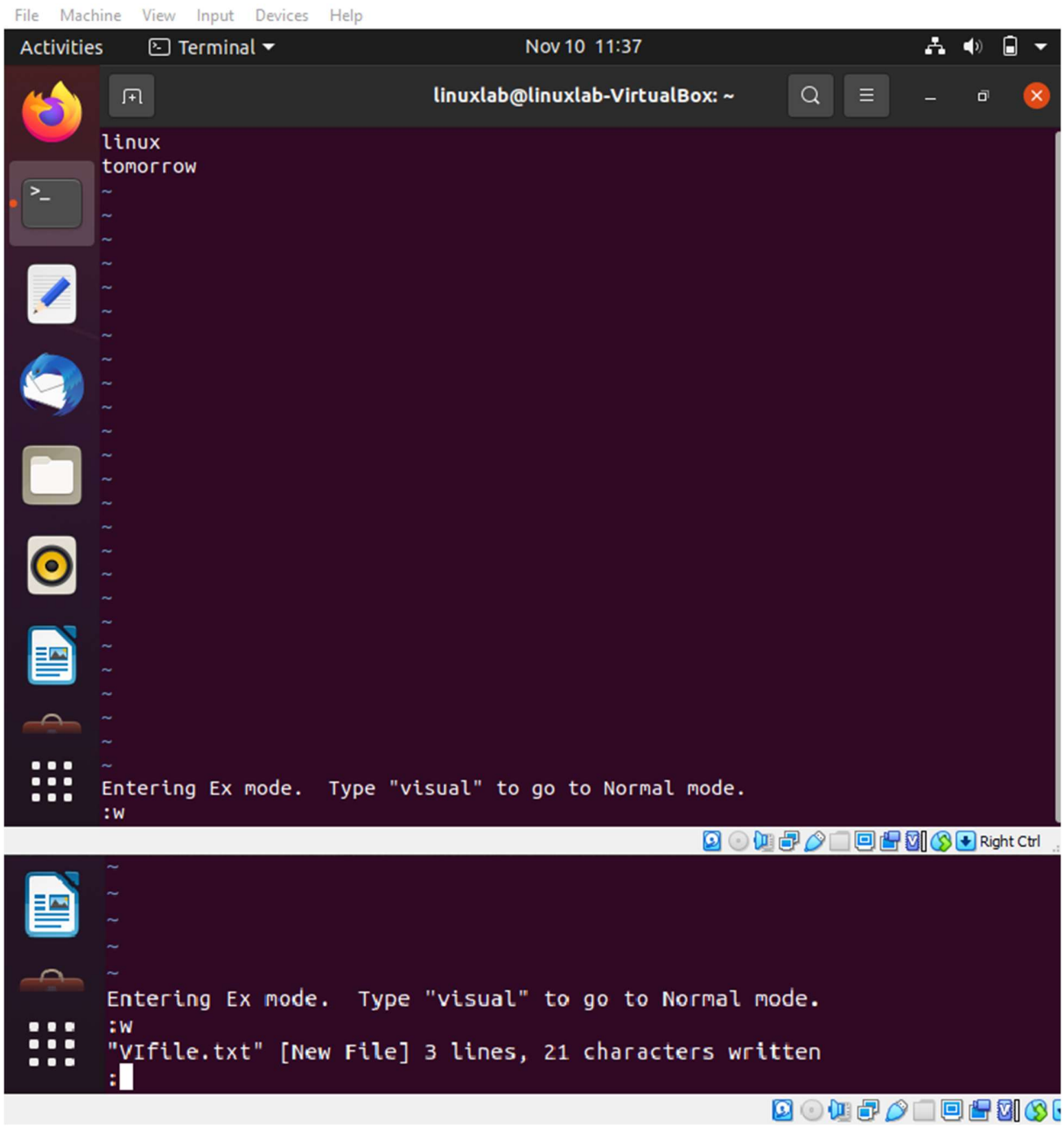
2.Inserting in vi: use ‘I’ to enter insertion mode



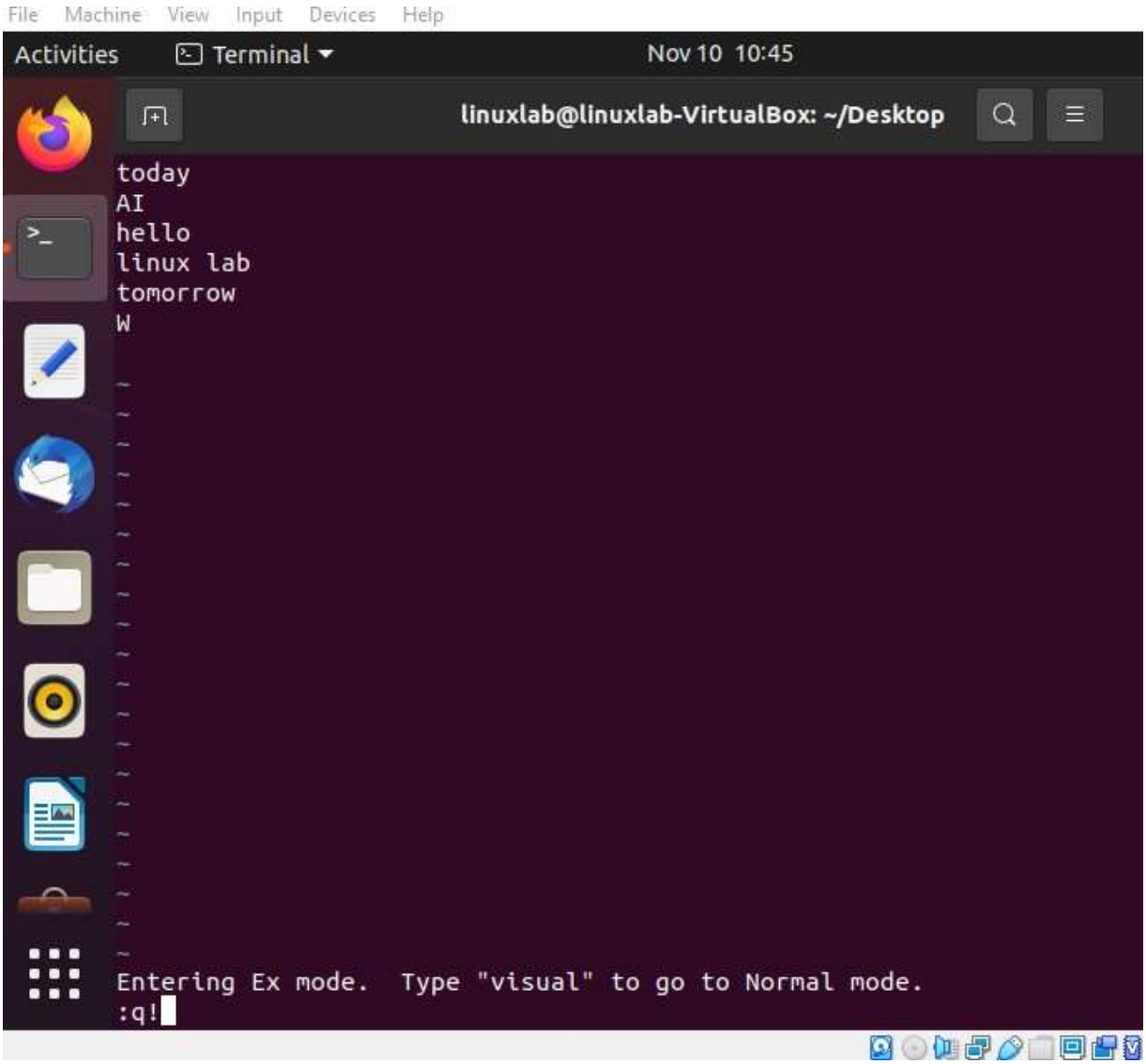
3. **:wq-** To exit vi editor



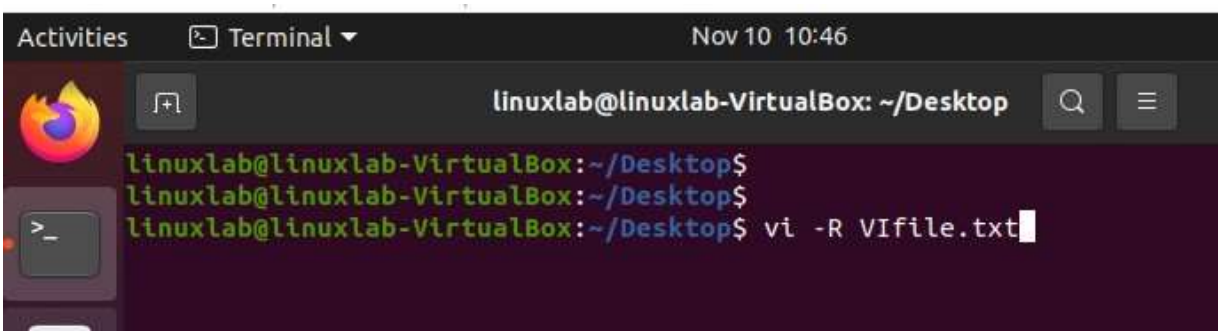
4:w- To save edited content in vi editor

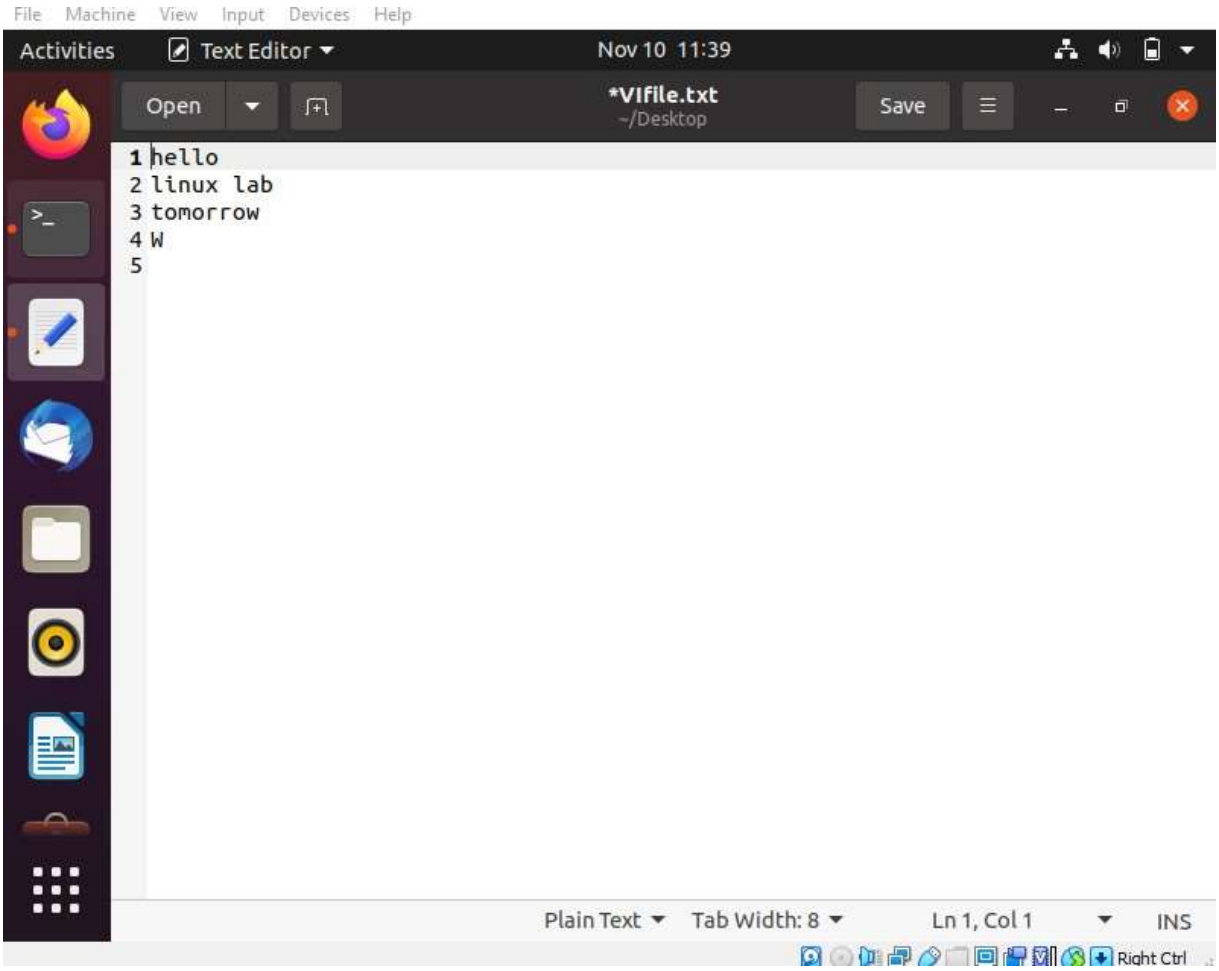

File Machine View Input Devices Help
Activities
Terminal
Nov 10 11:37
linuxlab@linuxlab-VirtualBox: ~
linux
tomorrow
Entering Ex mode. Type "visual" to go to Normal mode.
:w
Right Ctrl
Entering Ex mode. Type "visual" to go to Normal mode.
:w
"VIfile.txt" [New File] 3 lines, 21 characters written
:

5 :q!- To exit vi editor without saving



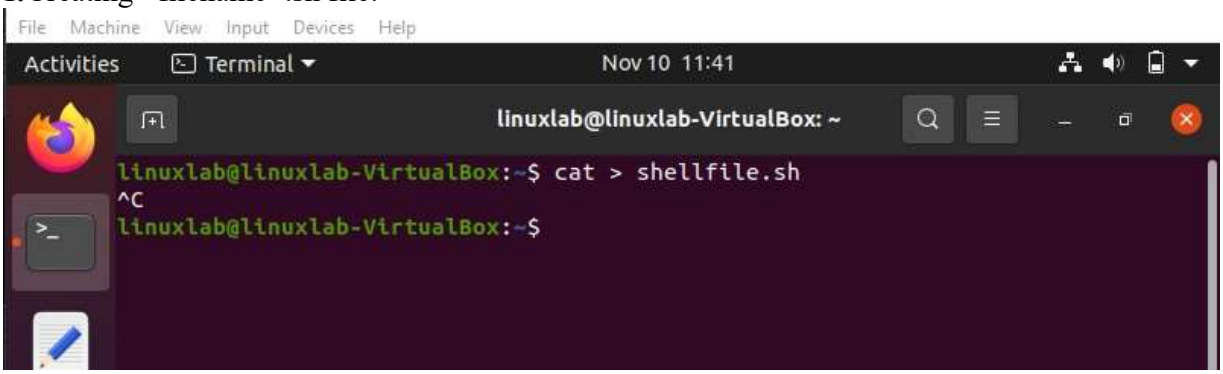
6. **vi –R filename:** Opens an existing file in the read-only mode.




File Machine View Input Devices Help
Activities
Text Editor
Nov 10 11:39
Open
*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

PROCEDURE:

1.Creating <filename>.sh file:

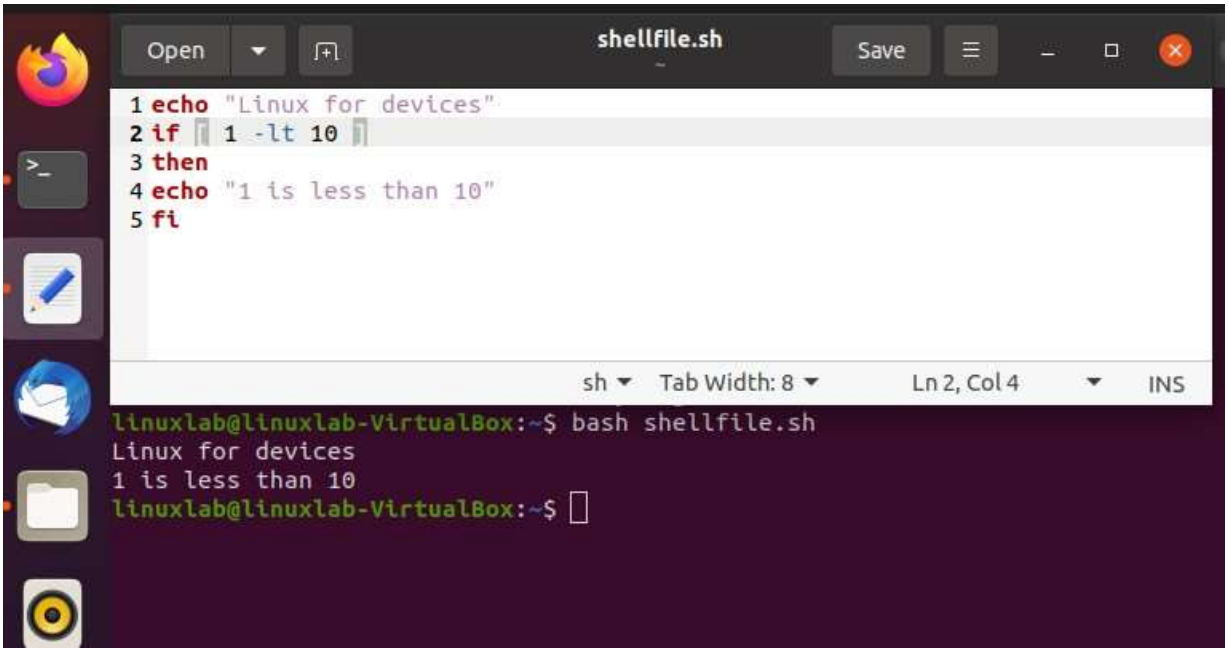


2.Inserting linux command in .sh file:

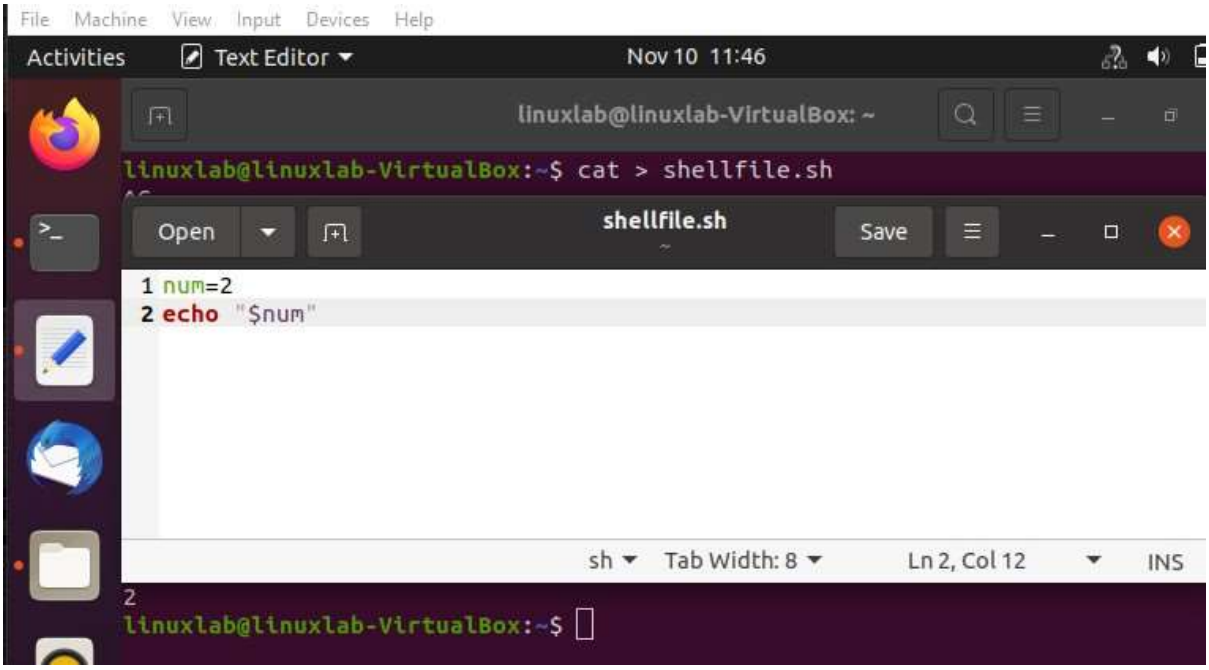
3.Saving and exiting .sh file:

- Press Ctrl + O to save the file, then press Enter.
- Press Ctrl + X to exit the editor

4. running the .sh file:



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



PRACTICAL 4 & 5

AIM- Write a shell script program to display "HELLO WORLD". write a shell script program to develop a scientific calculator.

THEORY-

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 '**B**ourne-**A**gain **S**hell'. 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, tclsh.

Shell

Shell is a macro processor which allows for an interactive or non-interactive command execution.

Scripting
Scripting allows for an automatic commands execution that would otherwise be executed interactively one-by-one.

all our scripts will include shell interpreter definition #!/bin/bash.

Script

In Computer programming, a script is a set of commands for an appropriate run time environment which is used to automate the execution of tasks.

Bash Script:

A Bash Shell Script is a plain text file containing a set of various commands that we usually type in the command line. It is used to automate repetitive tasks on Linux filesystem. It might include a set of commands, or a single command, or it might contain the hallmarks of imperative programming like loops, functions, conditional constructs, etc. Effectively, a Bash script is a computer program written in the Bash programming language.

How to create and run a Bash Script?

- To create an empty bash script, first, change the directory in which you want to save your script using cd command. Try to use text editor like gedit in which you want to type the shell commands.
- Use touch command to create the zero bytes sized script.

1. touch file_name

- To open the script in the text editor (eg., gedit), type

1. gedit file_name.sh

Here, .sh is suffixed as an extension that you have to provide for execution.

- Type the shell commands for your bash script in the newly opened text window or the text editor. Before typing bash shell commands, first, look at the base of any bash script.

Each Bash based Linux script starts by the line-

1. #! /bin/bash

Where #! is referred to as the shebang and rest of the line is the path to the interpreter specifying the location of bash shell in our operating system.

Bash use # to comment any line.

Bash use echo command to print the output.

At the end, execute the bash script prefixing with ./.

Have a look at the basic terms of a Bash Script, i.e., SheBang and echo command.

SheBang (#!)

The She Bang (#!) is a character sequence consisting of the characters number sign (#) and exclamation mark (!) at the beginning of a script.

Under the Unix-like operating systems, when a script with a shebang runs as a program, the program loader parses the rest of the lines with the first line as an interpreter directive. So, SheBang denotes an interpreter to execute the script lines, and it is known as the path directive for the execution of different kinds of Scripts like Bash, Python, etc.

Here is the correct SheBang format for the discussed Bash Script.

1. #!/bin/bash

The formatting for shebang is most important. Its incorrect format can cause improper working of commands. So, always remember these two points of SheBang formatting while creating a Script as follows:

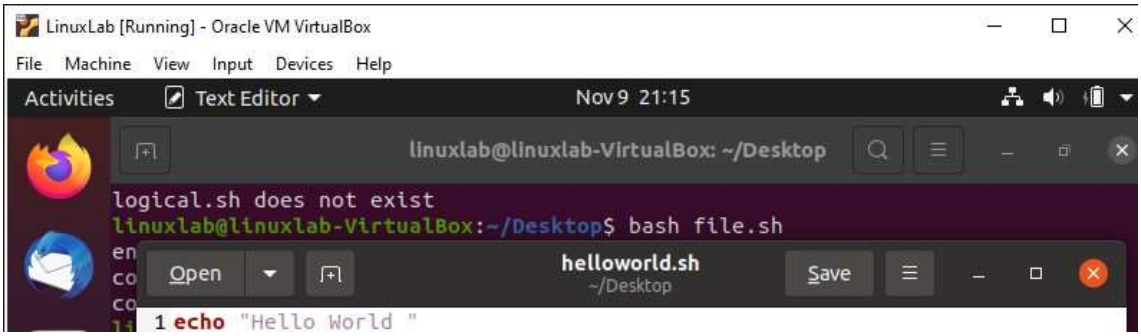
1. It should always be on the very first line of the Script.
2. There should not be any space before the hash (#), between the hash exclamation marks (#!), and the path to the interpreter.

echo

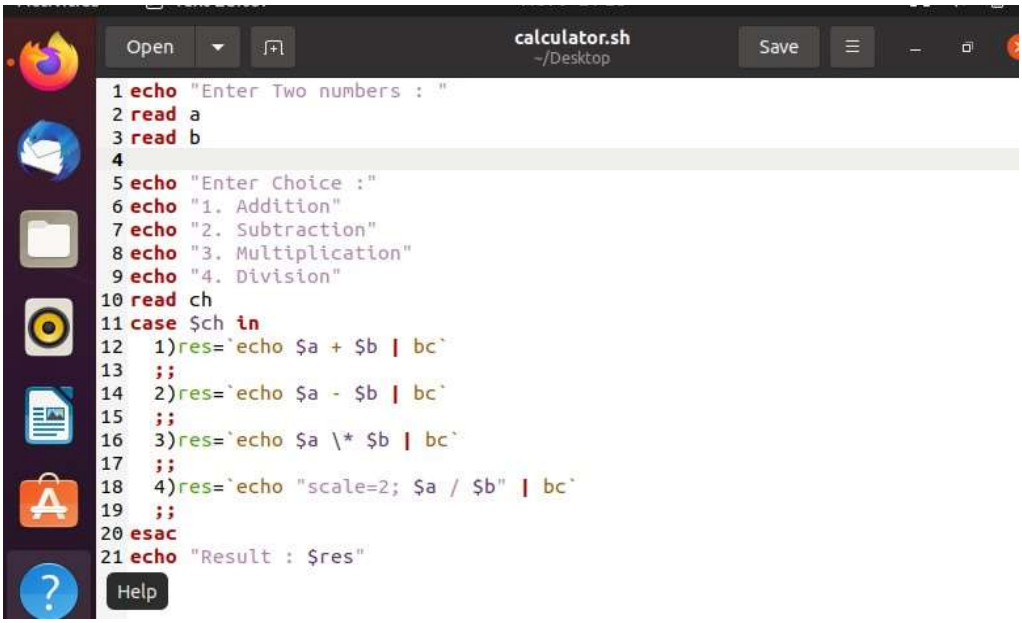
echo is a built-in command in Bash, which is used to display the standard output by passing the arguments. It is the most widely used command for printing the lines of text/String to the screen. Its performance is the same on both the platforms: Bash Shell and Command Line Terminal.

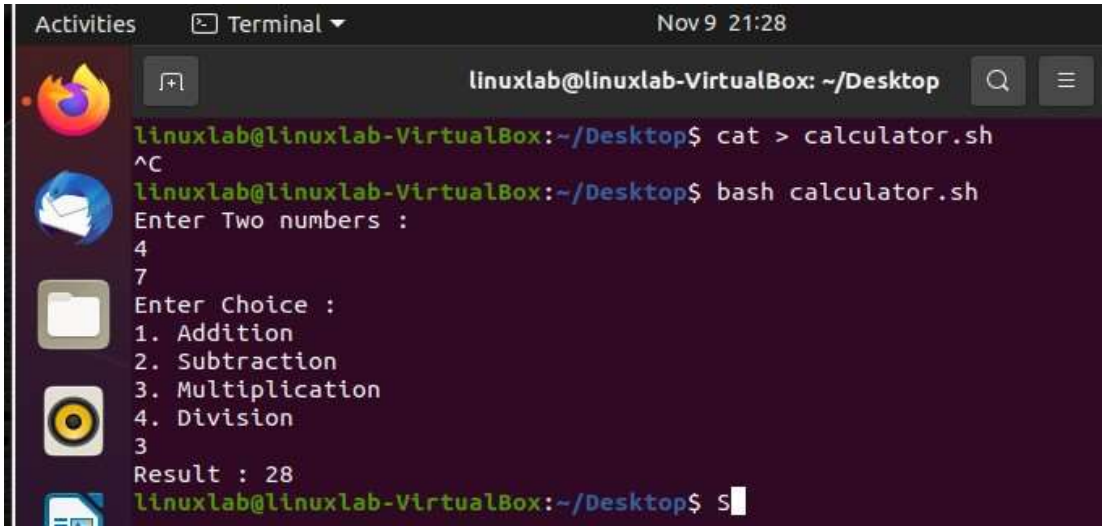
OUTPUT:

HELLO WORLD:



Calculator:


Open
calculator.sh
~/Desktop
Save
1 echo "Enter Two numbers : "
2 read a
3 read b
4
5 echo "Enter Choice :"
6 echo "1. Addition"
7 echo "2. Subtraction"
8 echo "3. Multiplication"
9 echo "4. Division"
10 read ch
11 case $ch in
12 1)res=`echo $a + $b | bc`
13 ;;
14 2)res=`echo $a - $b | bc`
15 ;;
16 3)res=`echo $a \* $b | bc`
17 ;;
18 4)res=`echo "scale=2; $a / $b" | bc`
19 ;;
20 esac
21 echo "Result : $res"
Help


Activities
Terminal
Nov 9 21:28
linuxlab@linuxlab-VirtualBox: ~/Desktop
linuxlab@linuxlab-VirtualBox:~/Desktop$ cat > calculator.sh
^C
linuxlab@linuxlab-VirtualBox:~/Desktop$ bash calculator.sh
Enter Two numbers :
4
7
Enter Choice :
1. Addition
2. Subtraction
3. Multiplication
4. Division
3
Result : 28
linuxlab@linuxlab-VirtualBox:~/Desktop$ S

PRACTICAL-6

AIM- Install Docker and run :Start, stop, push, pull, log, docker ps, docker ps –a,create account, docker compose, docker build, docker run

THEORY-

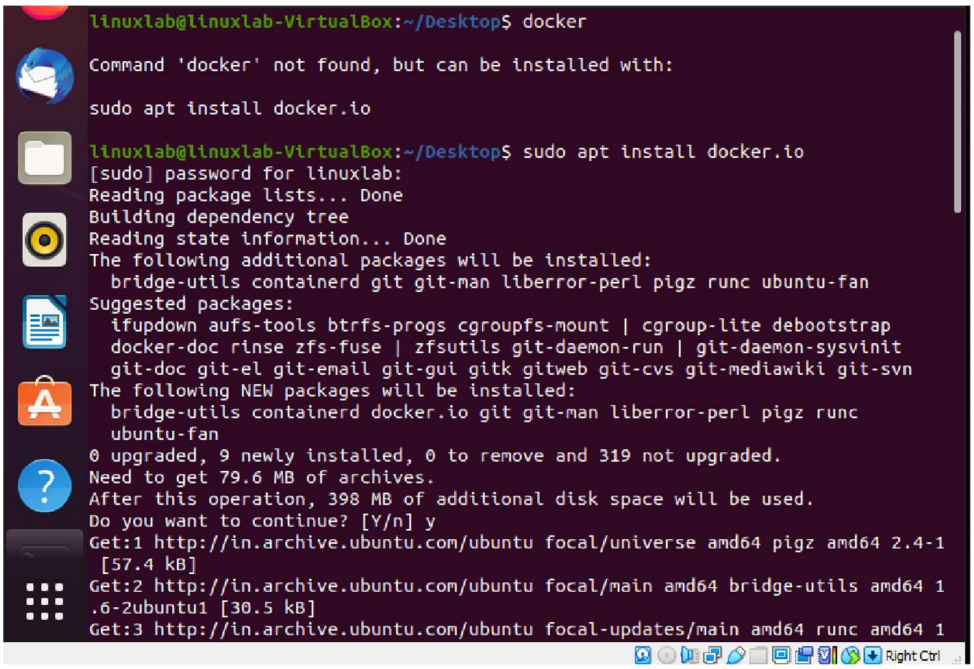
- Docker is an open source containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment. Containers simplify 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.

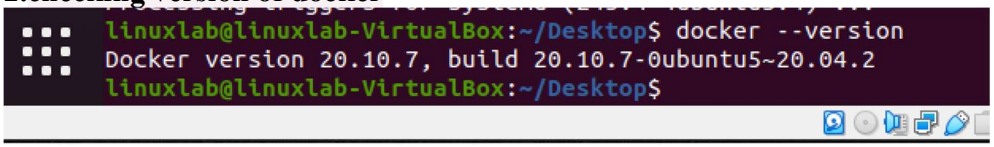
- Docker is so popular today that "Docker" and "containers" are used interchangeably. But the first container-related technologies were available for years — even decades (link resides outside IBM) — before Docker was released to the public in 2013.

PROCEDURE-

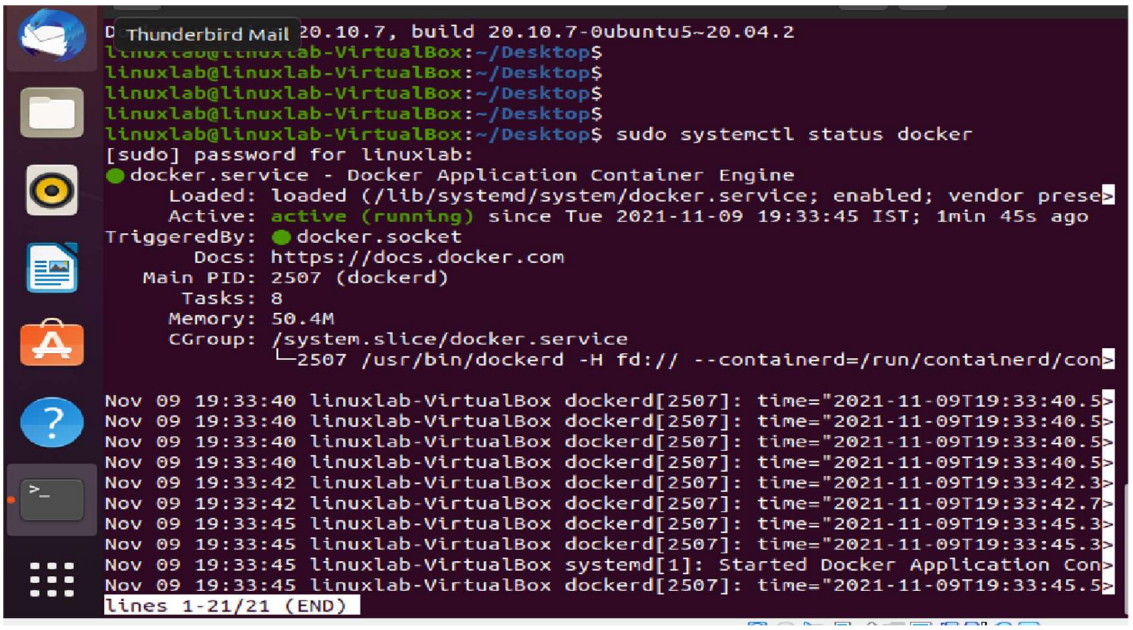
1.installing docker

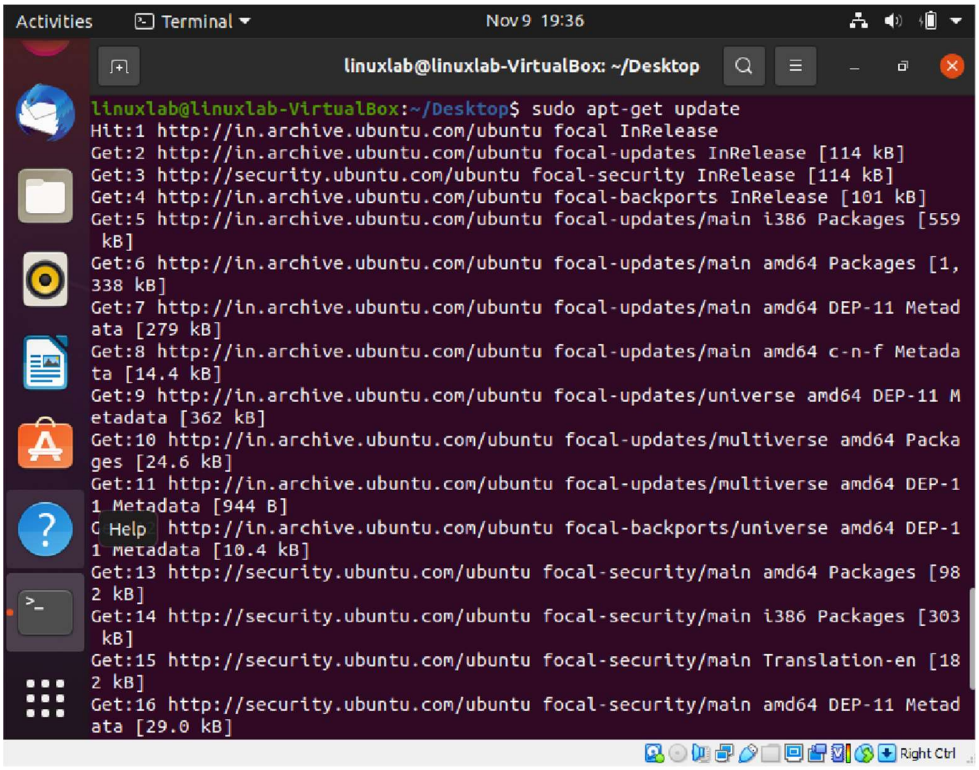


2.checking version of docker

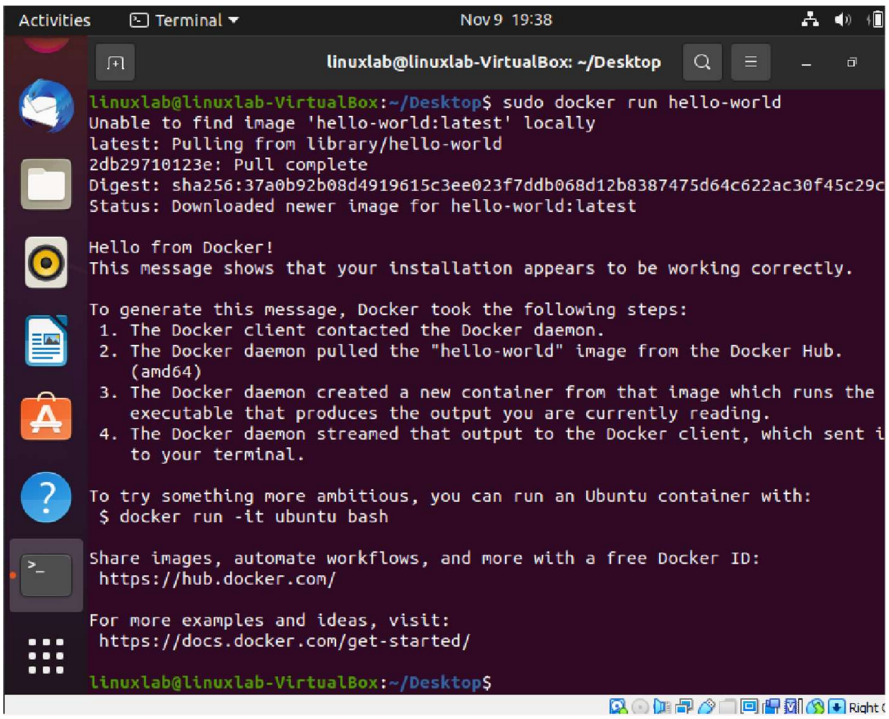


3.Checking enable status of docker(active/inactive)

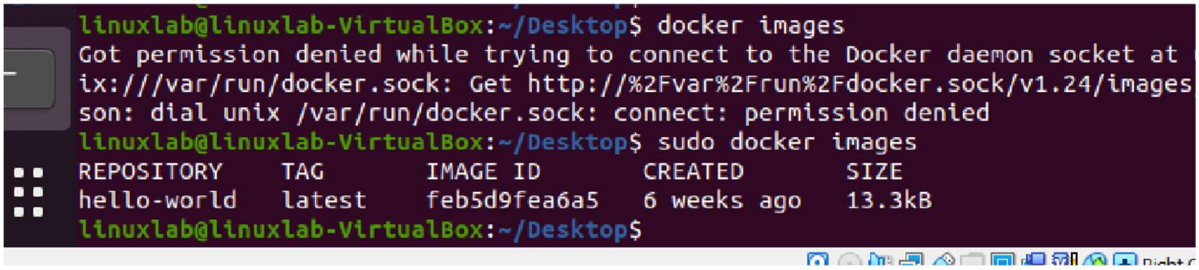




4. Run hello world



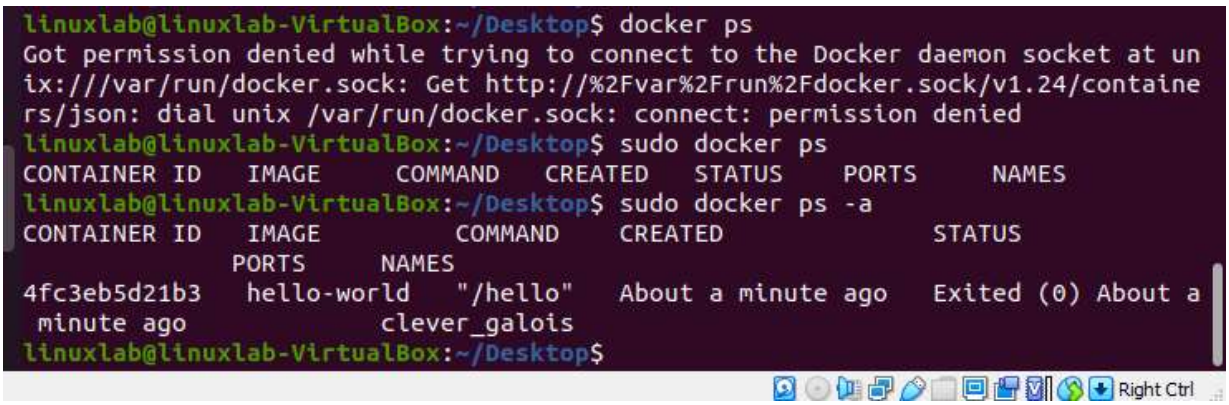
1. Run image pulled from repository



6.

Checking if containers are still running in machine

docker ps: Checks if container is running



No container is running

PRACTICAL-7

AIM: Write a shell script program to illustrate the implementation of following:

a)Integer Comparison

b)String comparison

c) Logical operators

d) File tests

e)Conditional control structure

f) Loop control structures

THEORY-

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 '**B**ourne-**A**gain **S**hell'. 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.

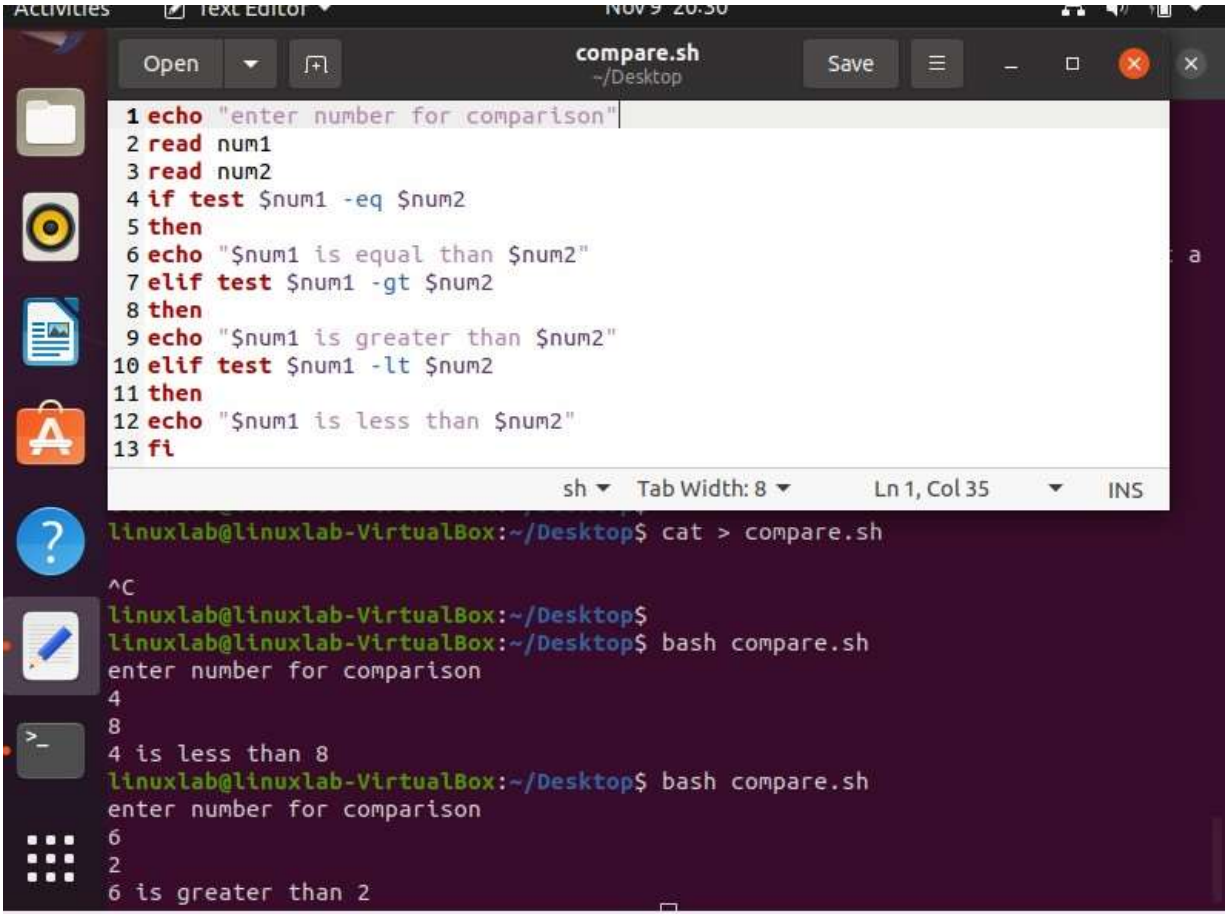
PROCEDURE:

1.Integer comparison

Different kinds of integer operator in bash are-
The general form of integer comparisons is int1 -operator int2.

Operator	Purpose
-eq	Integer equality
-ne	Integer inequality
-lt	Integer less than

-le	Integer less than or equal to
-gt	Integer greater than
-ge	Integer greater than or equal to



2.String comparison

String Comparison Operators

Comparison operators are operators that compare values and return true or false. When comparing strings in Bash you can use the following operators:

string1 = string2 and string1 == string2 - The equality operator returns true if the operands are equal.

- *Use the = operator with the test [command.*
- *Use the == operator with the [[command for pattern matching.*

string1 != string2 - The inequality operator returns true if the operands are not equal.

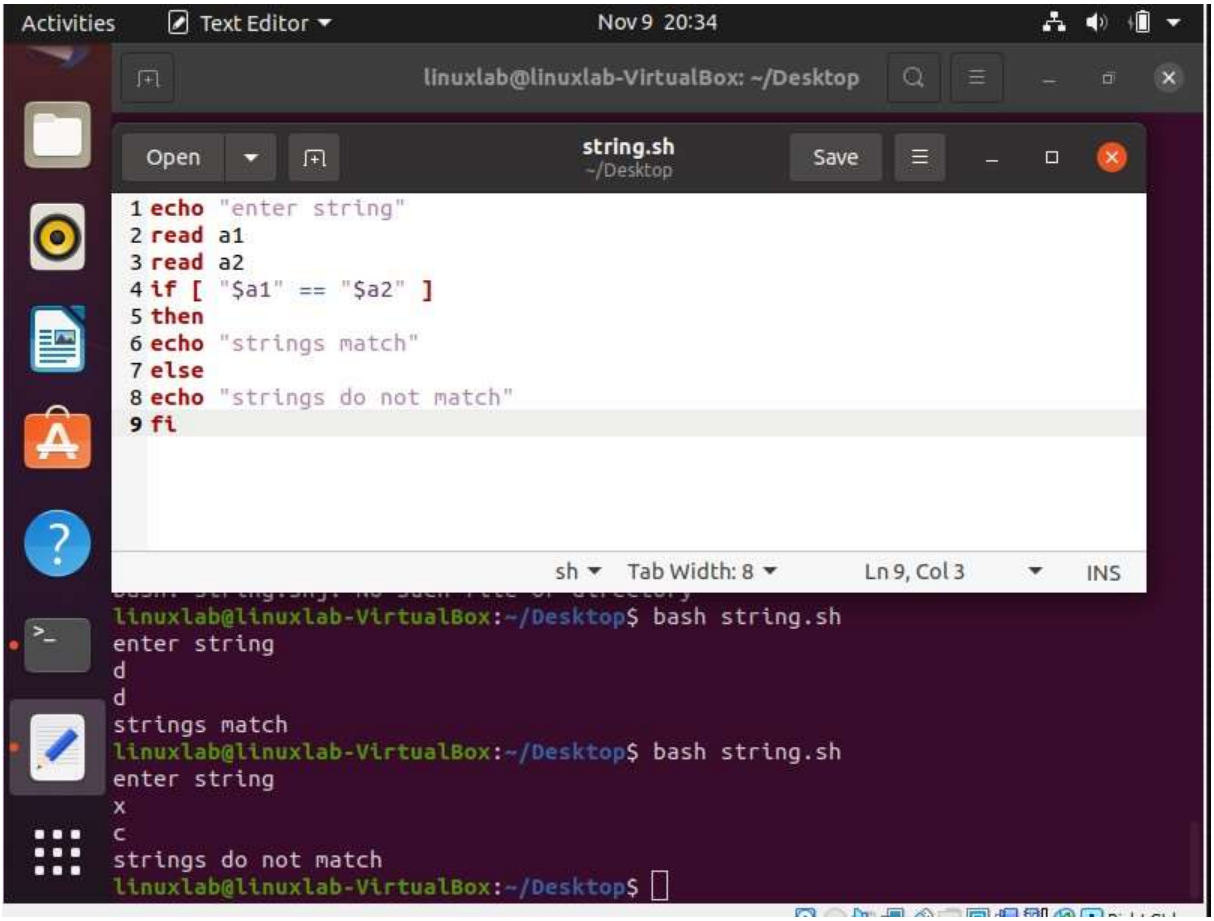
string1 =~ regex- The regex operator returns true if the left operand matches the extended regular expression on the right.

string1 > string2 - The greater than operator returns true if the left operand is greater than the right sorted by lexicographical (alphabetical) order.

string1 < string2 - The less than operator returns true if the right operand is greater than the right sorted by lexicographical (alphabetical) order.

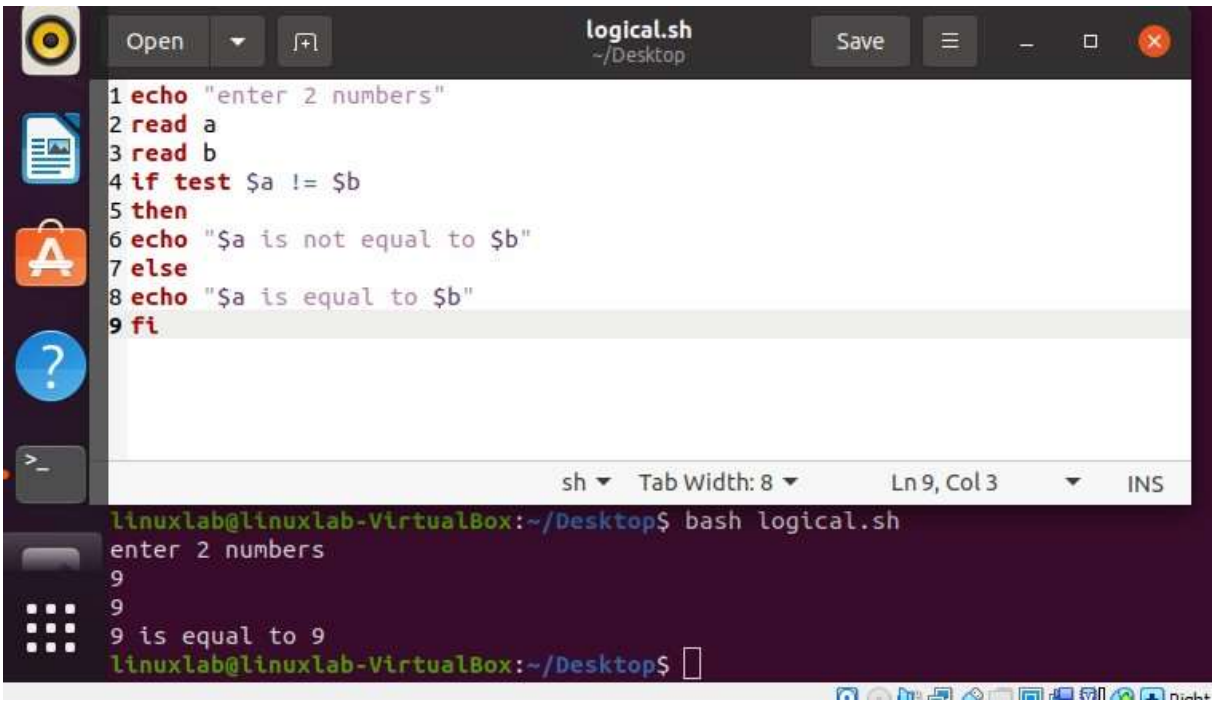
-z string - True if the string length is zero.

-n string - True if the string length is non-zero.



3.Logical operators

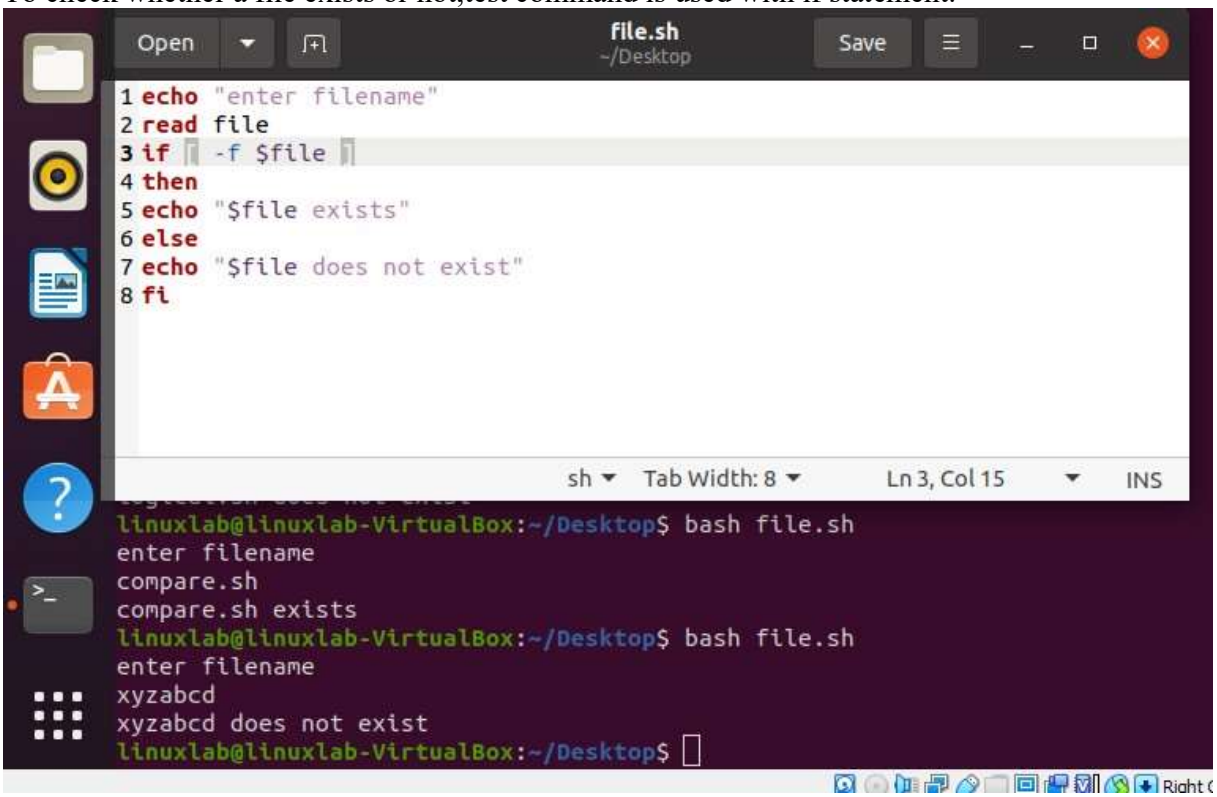
Different Logical Operator in Linux are:
!: This is logical negation. This inverts a true condition into false and vice versa.
-o: This is logical OR. If one of the operands is true, then the condition becomestrue.
-a: This is logical AND. If both the operands are true, then the condition becomestrue otherwise false.



4.File tests

To check whether a file exists,most commonly used file operators are -e and -f.
-e checks whether a file exists regardless of its type,
-f return true only if file is a regular file,not a directory or device.

To check whether a file exists or not,test command is used with if statement.



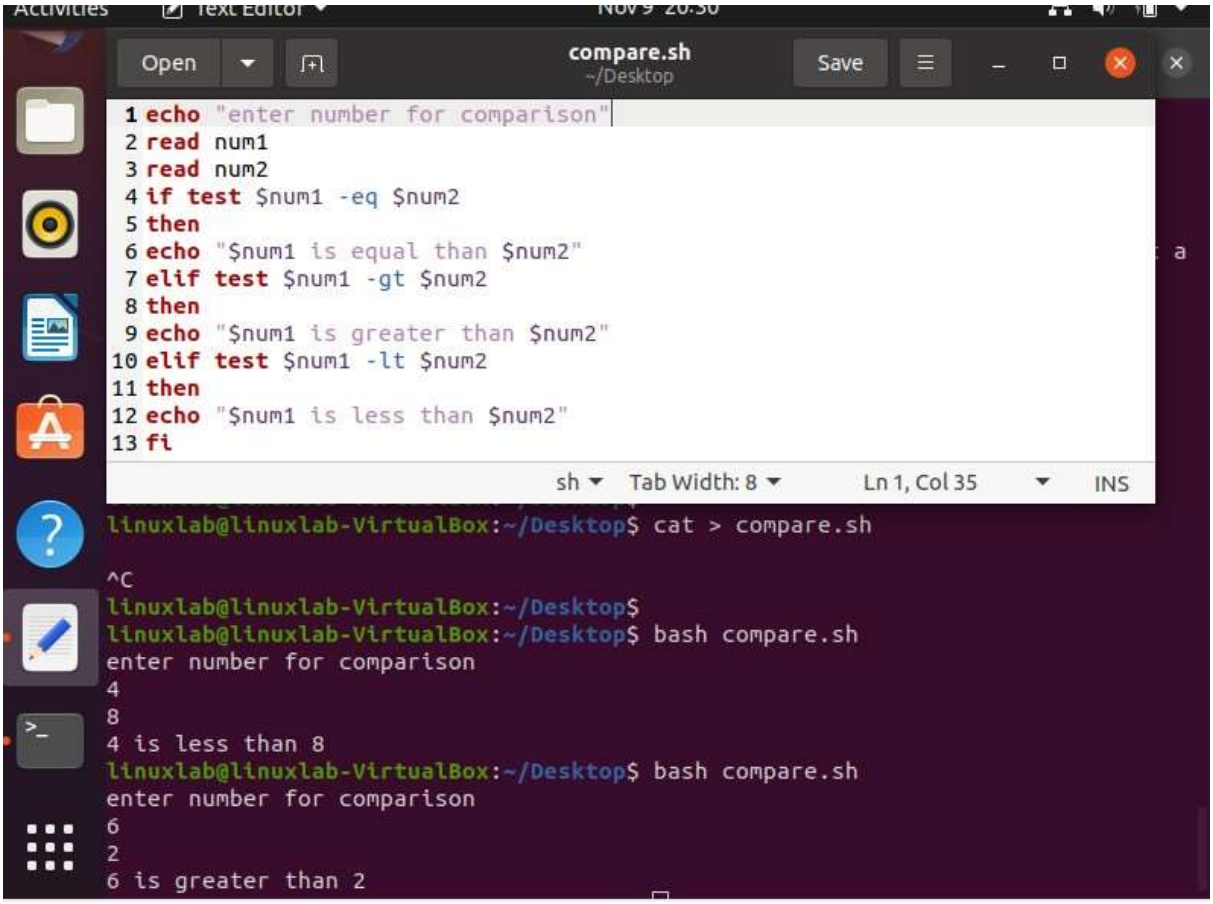
5. Conditional control structure

Conditional Statements: There are a total of five conditional statements which can be used in bash programming.

1. if statement
2. if-else statement
3. if..elif..else..fi statement (Else If ladder)
4. if..then..else..if..then..fi..fi..(Nested if)

5. switch statement

simple program for integer comparison using if statement



6.Loop control structures

This bash shows that loop terminates as soon as **a** becomes 5 −

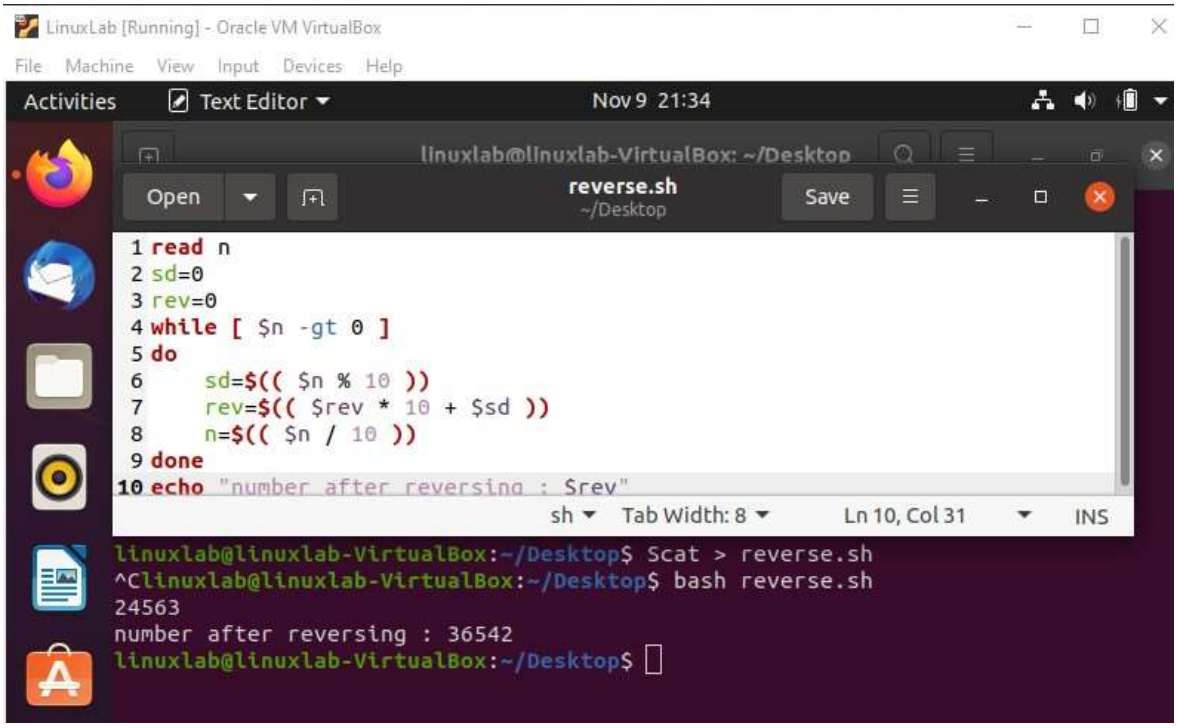
PRACTICAL-8

AIM-

Write a shell script to
a) print a number in reverse order.
b) to reverse the string and reverse each string further in the list
c) find the sum of all numbers in a file in Linux
d) validate password strength. Here are a few assumptions for the password string.Length – minimum of 8 characters,Contain both alphabet and number,Include both the small and capital case letters.

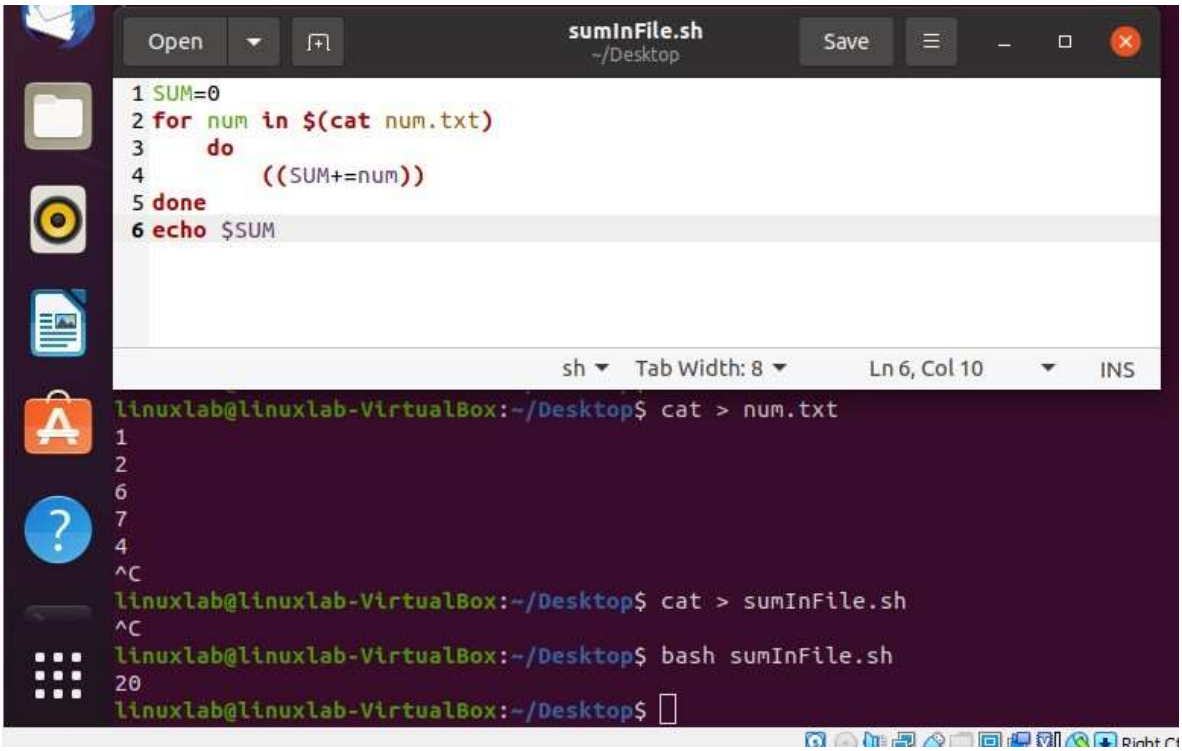
PROCEDURE-

1. Reverse a number

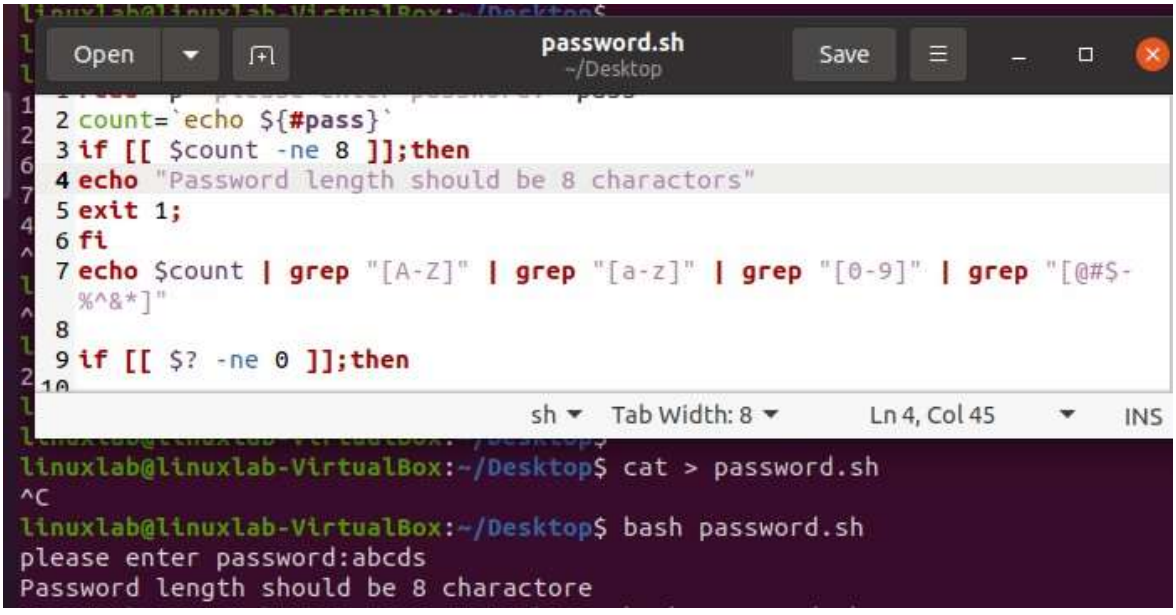


2.reverse a string and reverse each string further in the list

3.Print the sum of all numbers in file



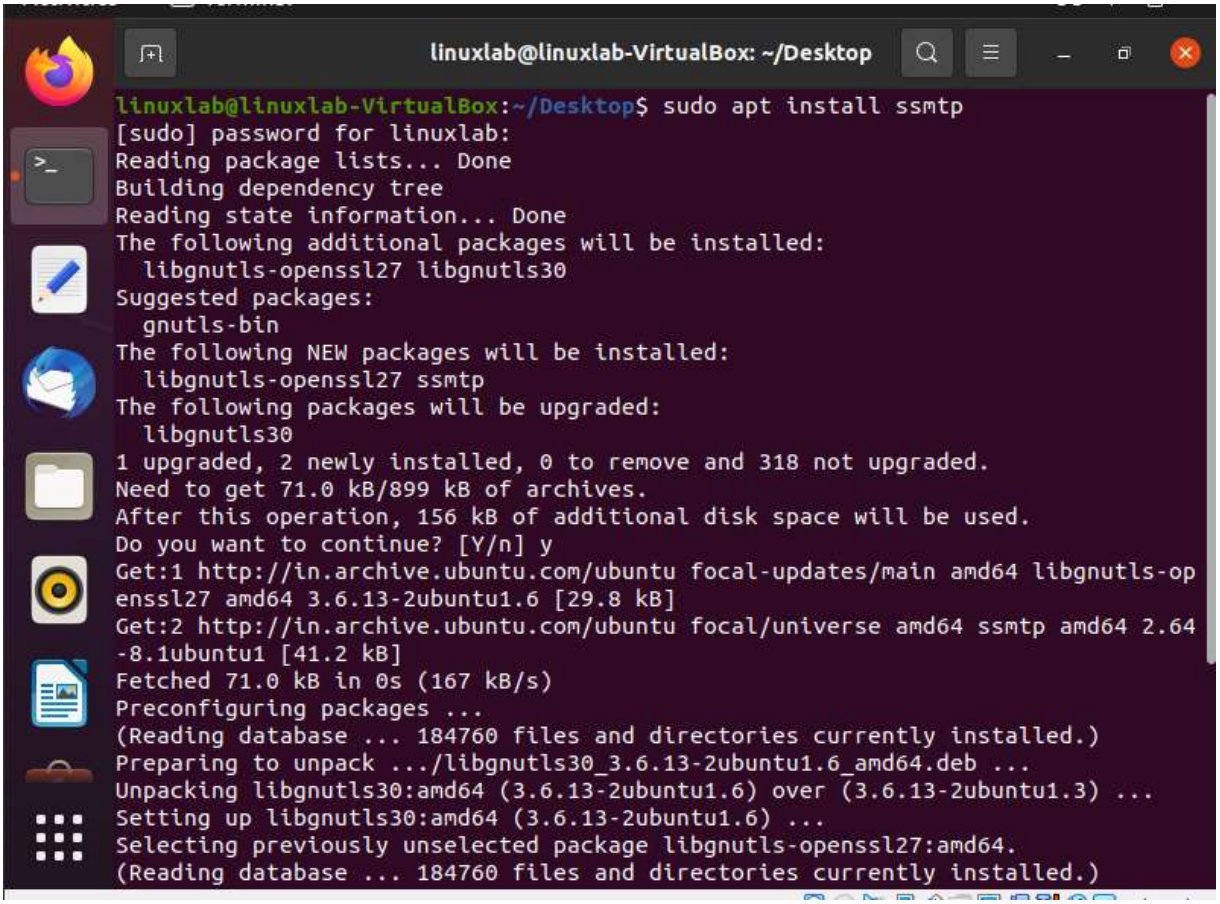
4. validate password strength. Here are a few assumptions for the password string.Length – minimum of 8 characters,Contain both alphabet and number,Include both the small and capital case letters.

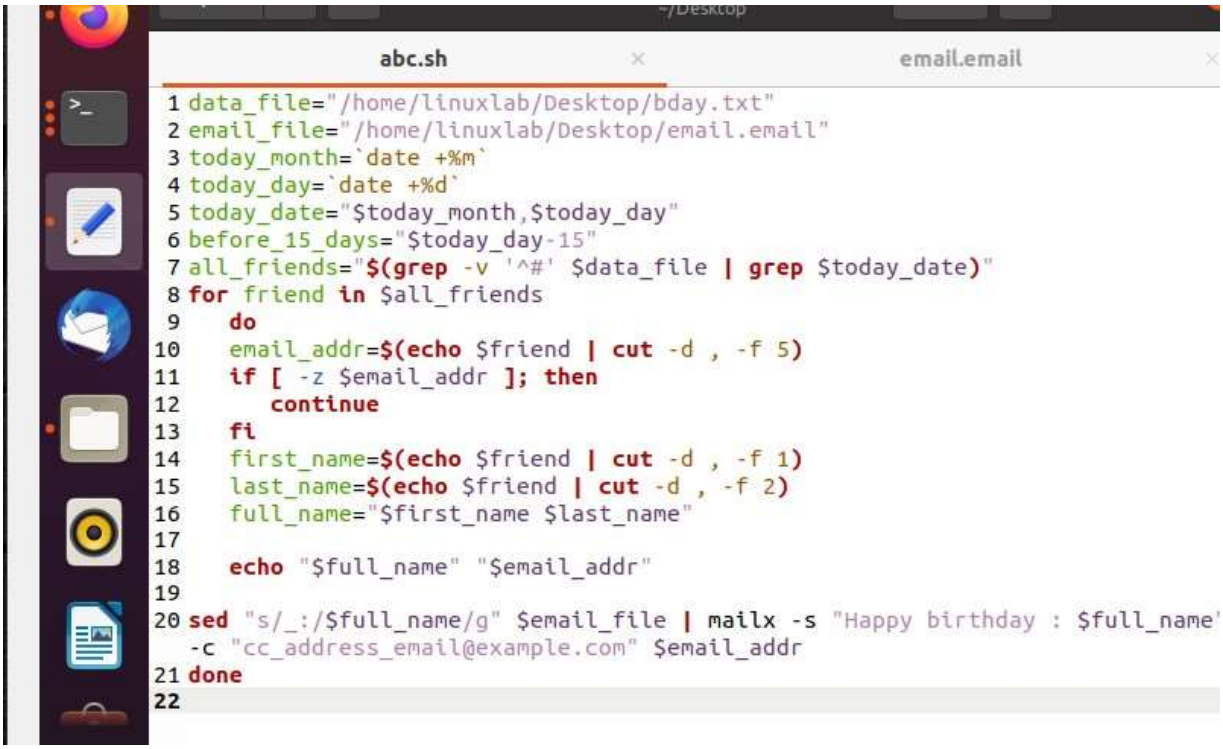


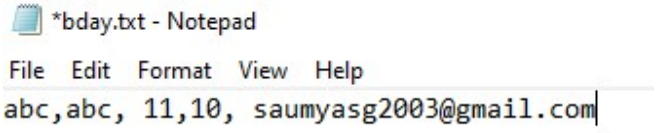
PRACTICAL-9

AIM: Design and develop a “Birthday Reminder” that can automatically send birthday wishes with a personalized message via email.

PROCEDURE-







OUTPUT-

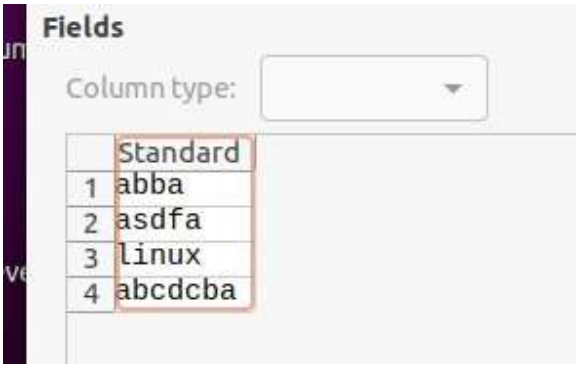
Birthday mail sent successfully.

Practical-10

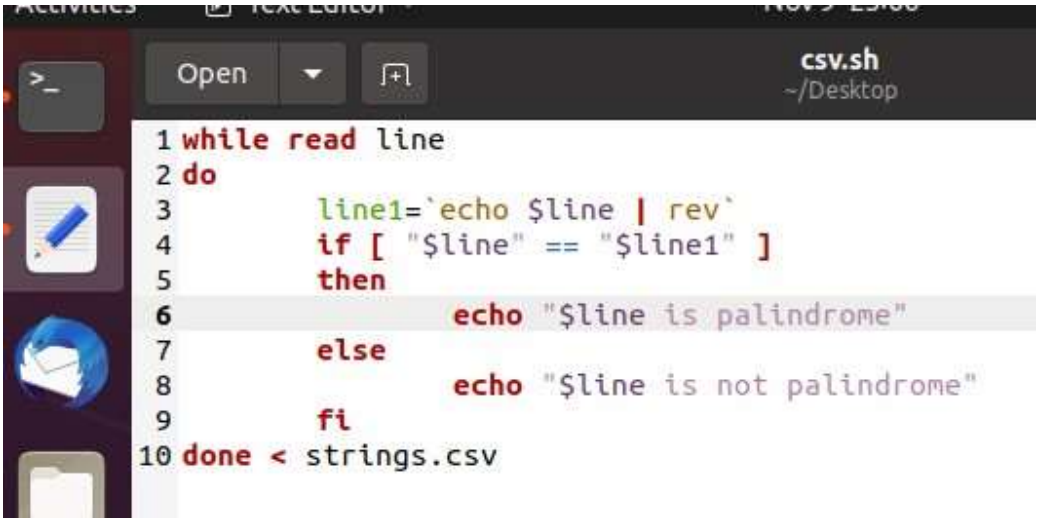
AIM: Check whether strings in csv files are palindrome or not.

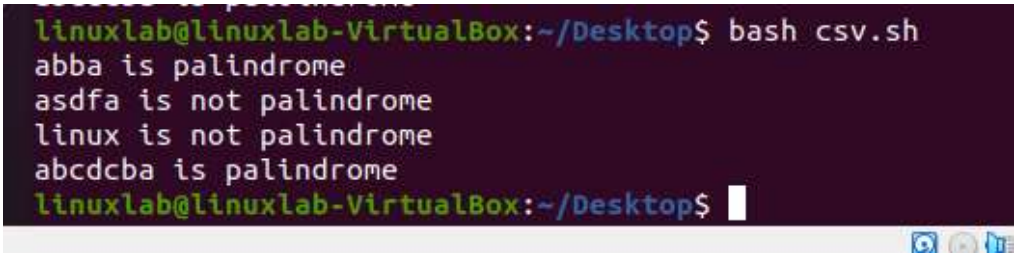
Procedure:

Strings present in the csv file:



.sh file:





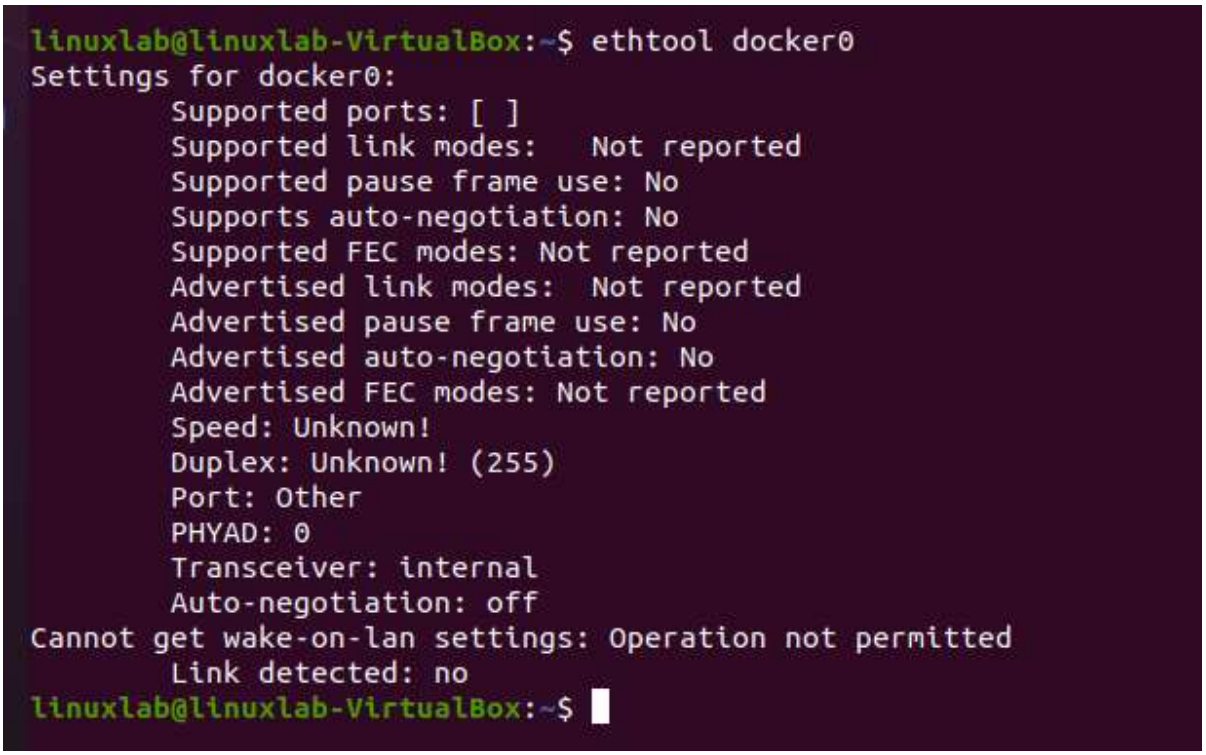
PRACTICAL-11

AIM: Implement NIC Bonding and Teaming.

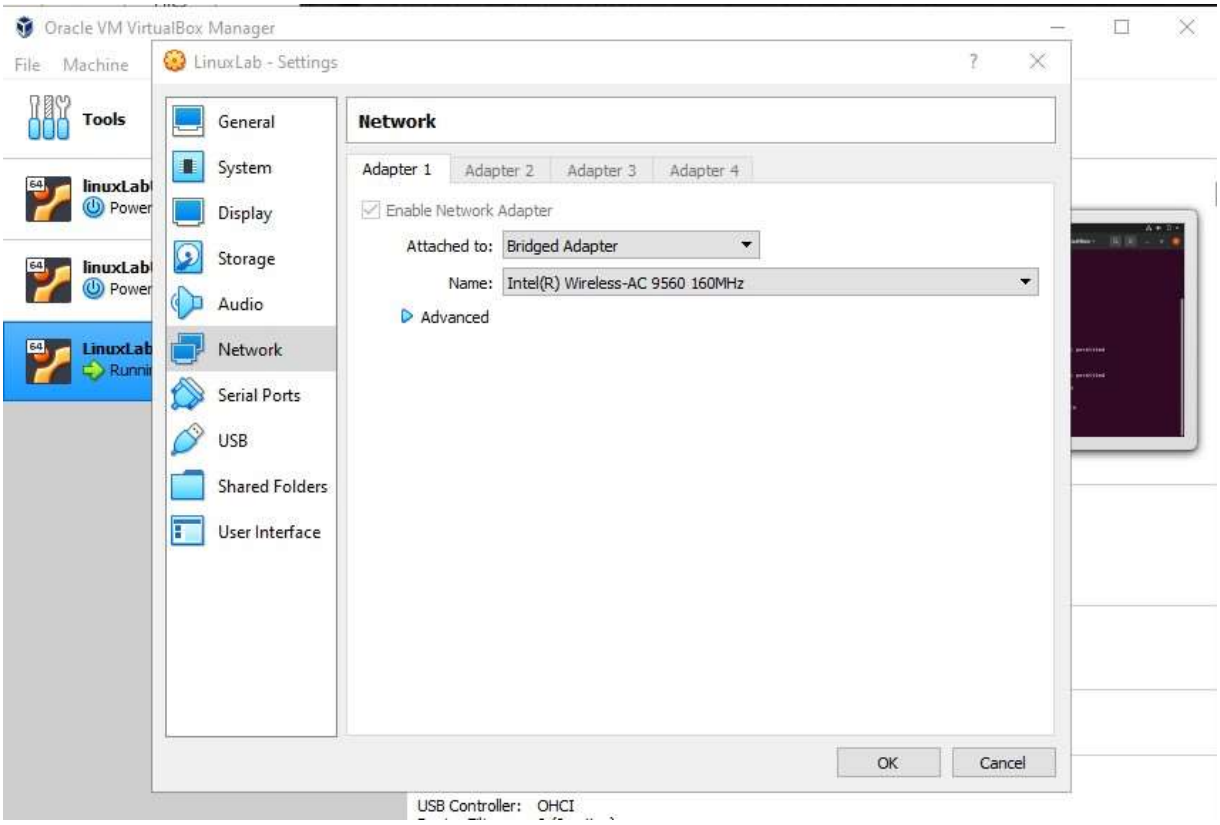
THEORY:

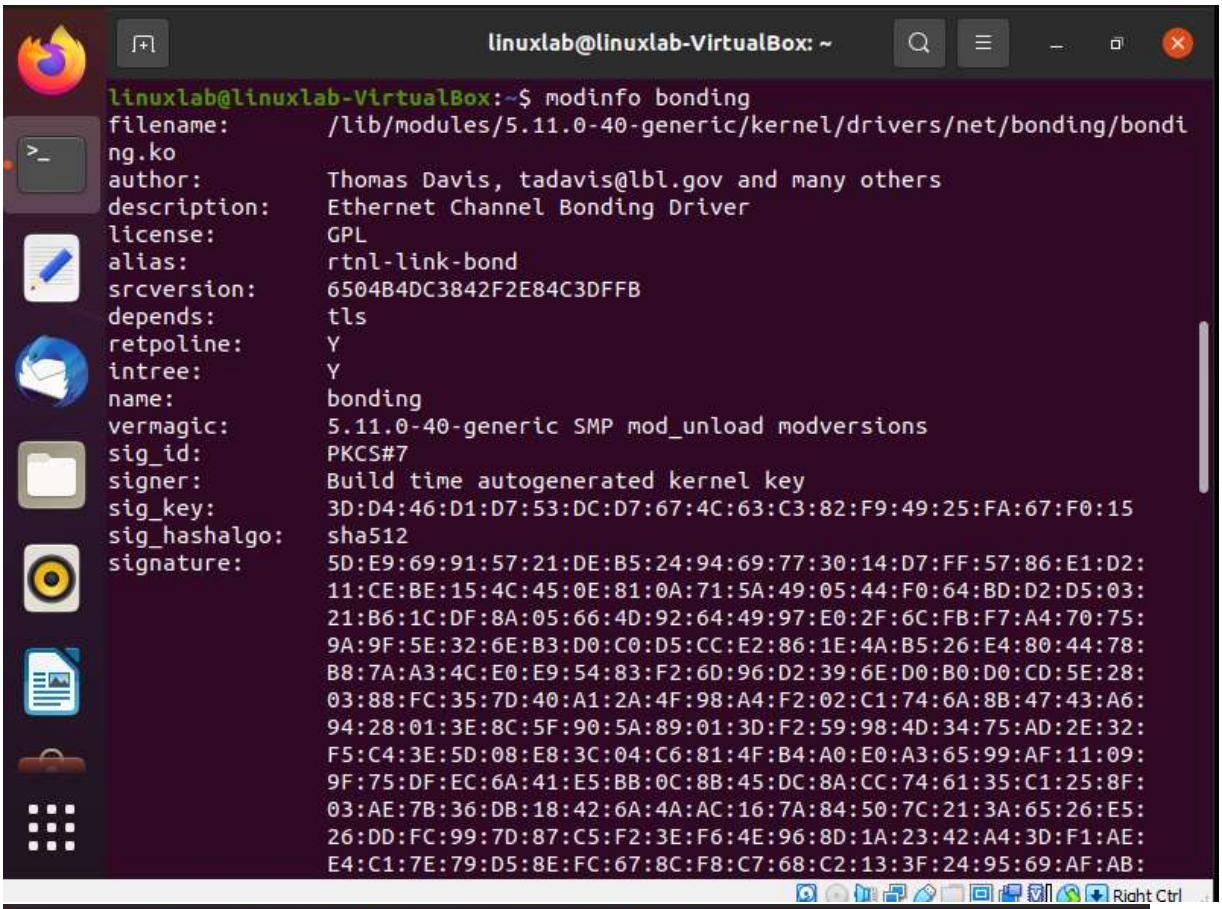
Network Interface Bonding is a mechanism used in Linux servers which consists of binding more physical network interfaces in order to provide more bandwidth than a single interface can provide or provide link redundancy in case of a cable failure. This type of link redundancy has multiple names in Linux, such as **Bonding**, **Teaming** or **Link Aggregation Groups** (**LAG**).

Bonding is nothing but Linux kernel feature that allows to aggregate multiple link interfaces (such as eth0, eth1) into a single virtual link such as bond0. The idea is pretty simple get higher data rates and as well as link failover. Linux allows administrators to bind multiple network interfaces together into a single channel using the bonding kernel module and a special network interface called a channel bonding interface. Channel bonding enables two or more network interfaces to act as one, simultaneously increasing the bandwidth and providing redundancy.

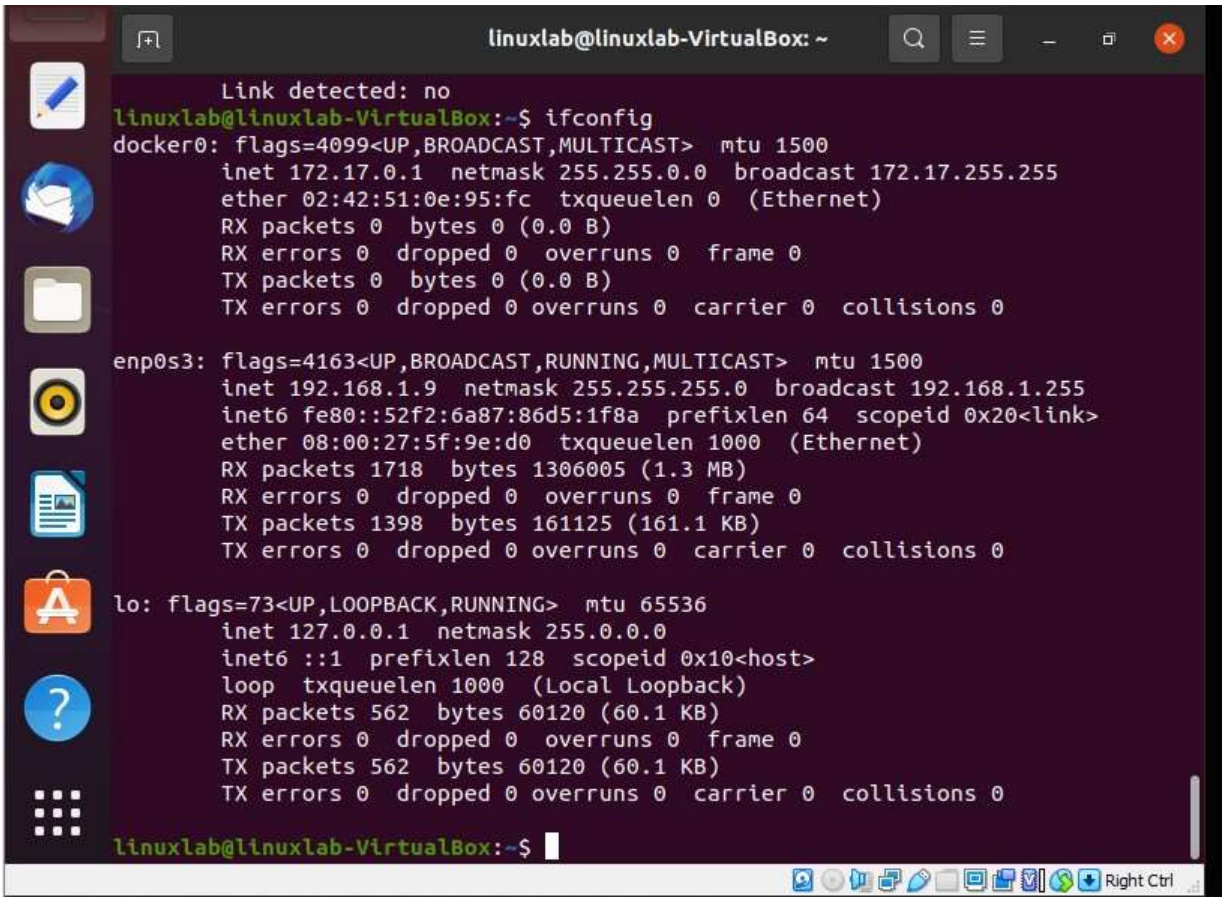


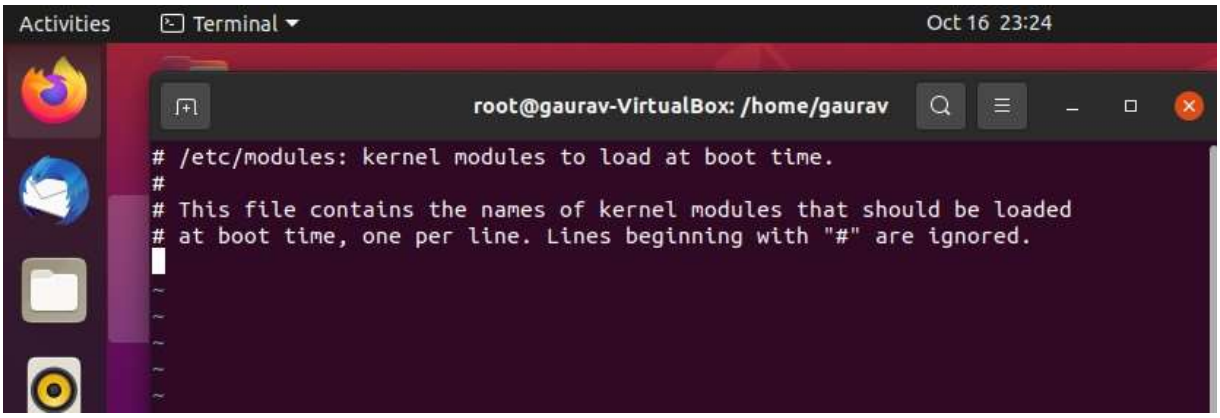
```
linuxlab@linuxlab-VirtualBox:~$ ethtool lo
Settings for lo:
Cannot get wake-on-lan settings: Operation not permitted
        Link detected: yes
linuxlab@linuxlab-VirtualBox:~$ ethtool enpis0
Settings for enpis0:
Cannot get device settings: No such device
Cannot get wake-on-lan settings: No such device
Cannot get message level: No such device
Cannot get link status: No such device
No data available
linuxlab@linuxlab-VirtualBox:~$
```

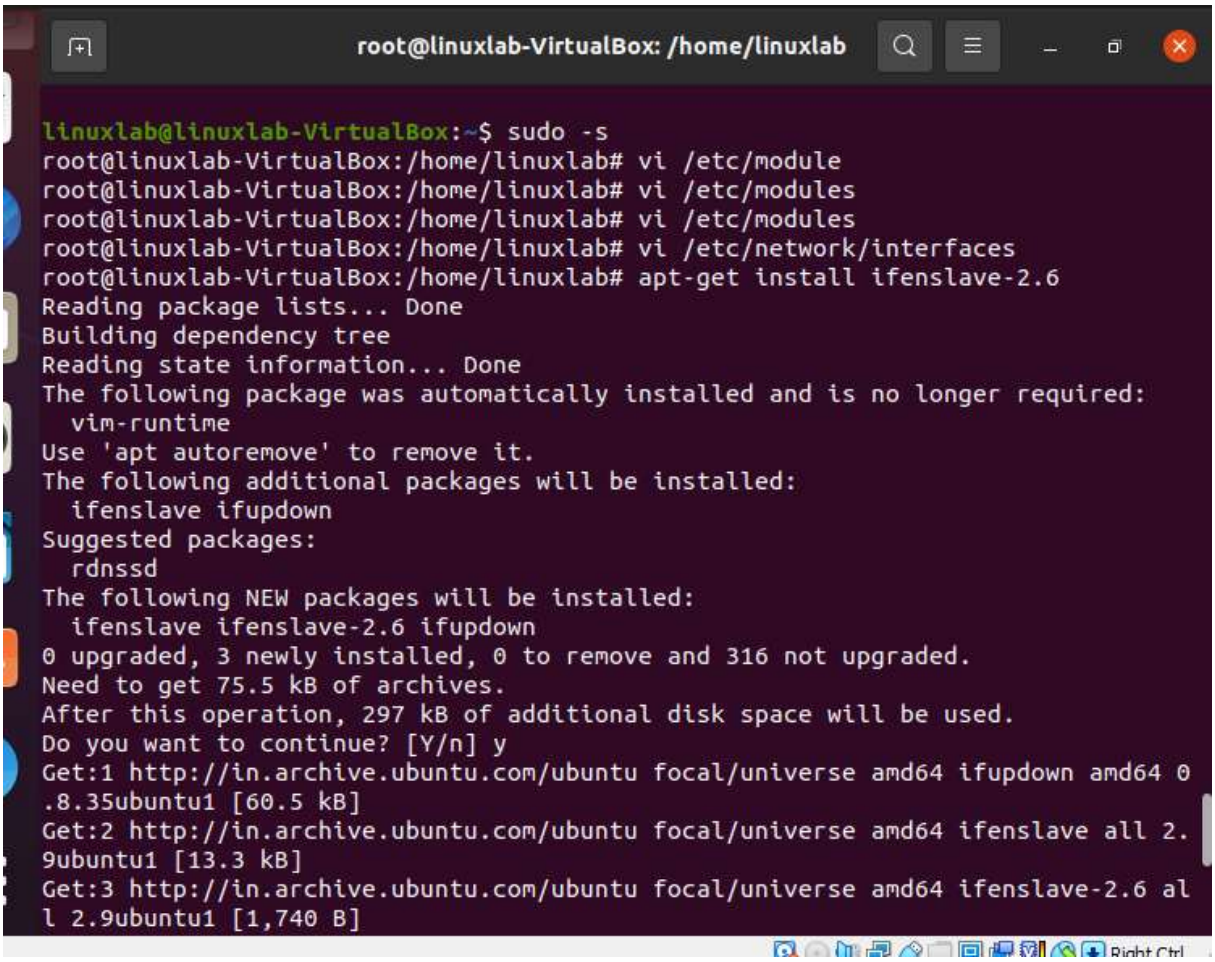





```
parm:           tx_queues:Max number of transmit queues (default = 16) (int)
parm:           num_grat_arp:Number of peer notifications to send on failover event (alias of num_unsol_na) (int)
parm:           num_unsol_na:Number of peer notifications to send on failover event (alias of num_grat_arp) (int)
parm:           miimon:Link check interval in milliseconds (int)
parm:           updelay:Delay before considering link up, in milliseconds (int)
parm:           downdelay:Delay before considering link down, in milliseconds (int)
parm:           use_carrier:Use netif_carrier_ok (vs MII ioctls) in miimon; 0 for off, 1 for on (default) (int)
parm:           mode:Mode of operation; 0 for balance-rr, 1 for active-backup, 2 for balance-xor, 3 for broadcast, 4 for 802.1
for balance-tlb, 6 for balance-alb (charp)
parm:           primary:Primary network device to use (charp)
parm:           primary_reselect:Reselect primary slave once it comes up; 0 for always (default), 1 for only if speed of prima
better, 2 for only on active slave failure (charp)
parm:           lacp_rate:LACPDU tx rate to request from 802.3ad partner; 0 for slow, 1 for fast (charp)
parm:           ad_select:802.3ad aggregation selection logic; 0 for stable (default), 1 for bandwidth, 2 for count (charp)
parm:           min_links:Minimum number of available links before turning on carrier (int)
parm:           xmit_hash_policy:balance-alb, balance-tlb, balance-xor, 802.3ad hashing method; 0 for layer 2 (default), 1 for
 3+4, 2 for layer 2+3, 3 for encap layer 2+3, 4 for encap layer 3+4 (charp)
parm:           arp_interval:arp interval in milliseconds (int)
parm:           arp_ip_target:arp targets in n.n.n.n form (array of charp)
parm:           arp_validate:validate src/dst of ARP probes; 0 for none (default), 1 for active, 2 for backup, 3 for all (char
parm:           arp_all_targets:fail on any/all arp targets timeout; 0 for any (default), 1 for all (charp)
parm:           fail_over_mac:For active-backup, do not set all slaves to the same MAC; 0 for none (default), 1 for active, 2
llow (charp)
parm:           all_slaves_active:Keep all frames received on an interface by setting active flag for all slaves; 0 for never
lt), 1 for always. (int)
parm:           resend_igmp:Number of IGMP membership reports to send on link failure (int)
parm:           packets_per_slave:Packets to send per slave in balance-rr mode; 0 for a random slave, 1 packet per s
>1 packets per slave. (int)
parm:           lp_interval:The number of seconds between instances where the bonding driver sends learning packets
eer switch. The default is 1. (uint)
gaurav@gaurav-VirtualBox:~$
```


linuxlab@linuxlab-VirtualBox: ~
Link detected: no
linuxlab@linuxlab-VirtualBox:~$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
ether 02:42:51:0e:95:fc txqueuelen 0 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.1.9 netmask 255.255.255.0 broadcast 192.168.1.255
inet6 fe80::52f2:6a87:86d5:1f8a prefixlen 64 scopeid 0x20<link>
ether 08:00:27:5f:9e:d0 txqueuelen 1000 (Ethernet)
RX packets 1718 bytes 1306005 (1.3 MB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 1398 bytes 161125 (161.1 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 562 bytes 60120 (60.1 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 562 bytes 60120 (60.1 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
linuxlab@linuxlab-VirtualBox:~$
Right Ctrl


Activities
Terminal
Oct 16 23:24
root@gaurav-VirtualBox: /home/gaurav
/etc/modules: kernel modules to load at boot time.
#
This file contains the names of kernel modules that should be loaded
at boot time, one per line. Lines beginning with "#" are ignored.


root@linuxlab-VirtualBox: /home/linuxlab
linuxlab@linuxlab-VirtualBox:~$ sudo -s
root@linuxlab-VirtualBox:/home/linuxlab# vi /etc/module
root@linuxlab-VirtualBox:/home/linuxlab# vi /etc/modules
root@linuxlab-VirtualBox:/home/linuxlab# vi /etc/modules
root@linuxlab-VirtualBox:/home/linuxlab# vi /etc/network/interfaces
root@linuxlab-VirtualBox:/home/linuxlab# apt-get install ifenslave-2.6
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
vim-runtime
Use 'apt autoremove' to remove it.
The following additional packages will be installed:
ifenslave ifupdown
Suggested packages:
rdnssd
The following NEW packages will be installed:
ifenslave ifenslave-2.6 ifupdown
0 upgraded, 3 newly installed, 0 to remove and 316 not upgraded.
Need to get 75.5 kB of archives.
After this operation, 297 kB 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 ifupdown amd64 0
.8.35ubuntu1 [60.5 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 ifenslave all 2.
9ubuntu1 [13.3 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 ifenslave-2.6 al
l 2.9ubuntu1 [1,740 B]
Right Ctrl

```
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

#enp0s3 configuration
auto enp0s3
iface enp0s3 inet manual
bond-master bond0
bond-primary enp0s3

#enp0s8  configuration
auto enp0s8
iface enp0s8  inet manual
bond-master bond0

# Bonding enp0s3 & enp0s8  create bond0 NIC
auto bond0
iface bond0 inet static
address 192.168.100.200
gateway 192.168.100.255
netmask 255.255.255.0
bond-mode active-backup
bond-miimon 100
bond-slaves none
```

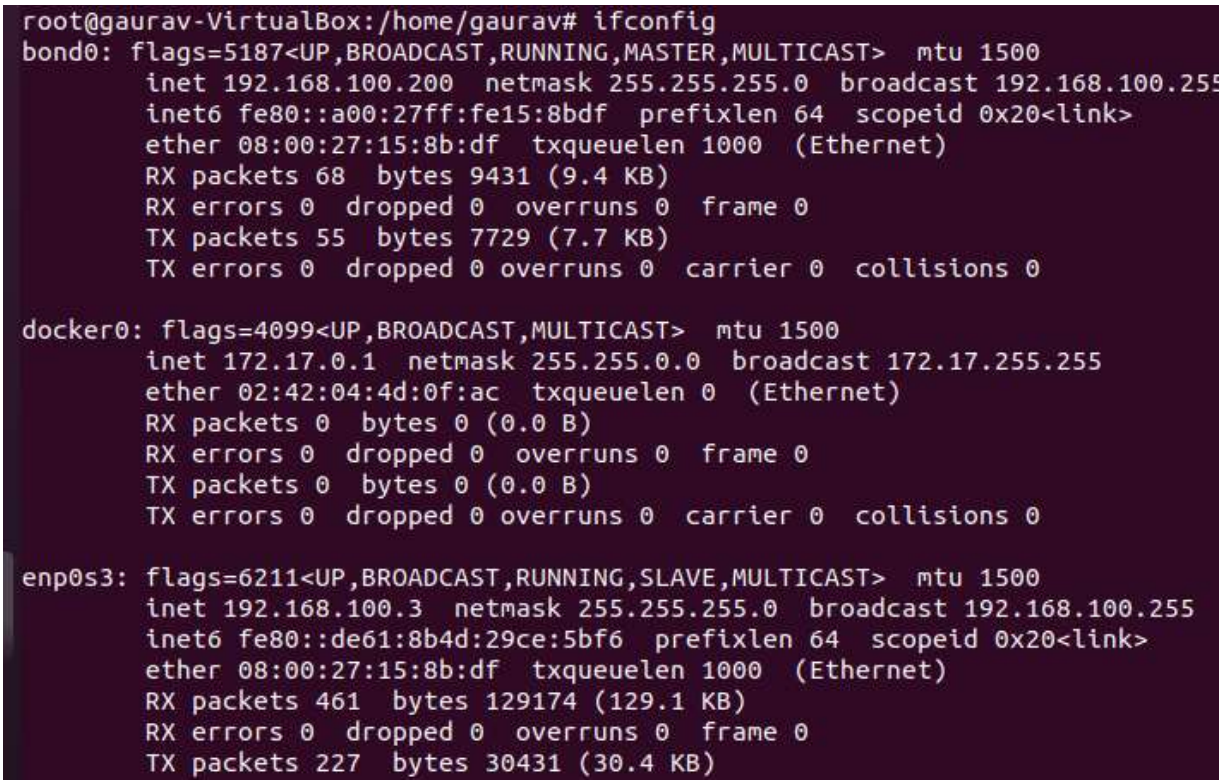
Show Applications

-- INSERT --

```
Bonding Mode: fault-tolerance (active-backup)
Primary Slave: enp0s3 (primary_reselect always)
Currently Active Slave: enp0s3
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0
Peer Notification Delay (ms): 0

Slave Interface: enp0s3
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 08:00:27:15:8b:df
Slave queue ID: 0

Slave Interface: enp0s8
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 08:00:27:e2:f6:1f
Slave queue ID: 0
```


root@gaurav-VirtualBox:/home/gaurav# ifconfig
bond0: flags=5187<UP,BROADCAST,RUNNING,MASTER,MULTICAST> mtu 1500
inet 192.168.100.200 netmask 255.255.255.0 broadcast 192.168.100.255
inet6 fe80::a00:27ff:fe15:8bdf prefixlen 64 scopeid 0x20<link>
ether 08:00:27:15:8b:df txqueuelen 1000 (Ethernet)
RX packets 68 bytes 9431 (9.4 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 55 bytes 7729 (7.7 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
ether 02:42:04:4d:0f:ac txqueuelen 0 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
enp0s3: flags=6211<UP,BROADCAST,RUNNING,SLAVE,MULTICAST> mtu 1500
inet 192.168.100.3 netmask 255.255.255.0 broadcast 192.168.100.255
inet6 fe80::de61:8b4d:29ce:5bf6 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:15:8b:df txqueuelen 1000 (Ethernet)
RX packets 461 bytes 129174 (129.1 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 227 bytes 30431 (30.4 KB)