Operating System Laboratory

P. Charan

210303126176

FACULTY OF ENGINEERING AND TECHNOLOGY

BACHELOR OF TECHNOLOGY

Operating System Laboratory (203105214)

4TH SEMESTER

COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that

Mr./Ms. ***POTTHURI.CHARAN PADMA SRIKHAR*** *with enrolment no.* ***210303126176*** *has successfully completed his/her laboratory experiments in the* ***OPERATING SYSTEM (2103105214)*** *from the department of* ***COMPUTER SCIENCE AND ENGNEERING*** *during the academic year* ***2022-2023.***

Date of Submission: .... **Staff In charge: ....**

Head Of Department: ....

INDEX

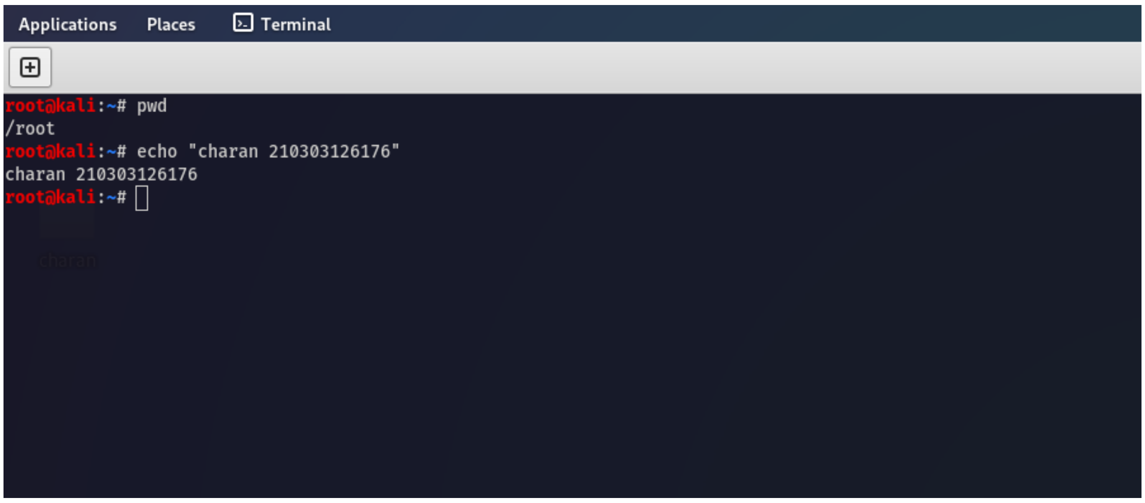
S.no	Title	Page no.	Performance date	Assessment date	Marks	Sign
1	Study of Basic commands of Linux	04-13	17-11-2022	24-11-2022		
2	Study the basics of shell programming	14-18	24-11-2022	01-12-2022		
3	Write a Shell script to print given numbers sum of all digits	18-19	01-12-2022	08-12-2022		
4	Write a shell script to validate the entered date. (e.g. Date format is: dd-mm-yyyy)	19-20	08-12-2022	15-12-2022		
5	Write a shell script to check entered string is palindrome or not.	20-21	15-12-2022	29-12-2022		
6	Write a Shell script to say Good morning/Afternoon/Evening as you log in to system	21-22	29-12-2022	05-01-2023		
7	Write a C program to create a child process	22-23	05-01-2023	12-01-2023		
8	Finding out biggest number from given three numbers supplied as command line arguments.	23-24	12-01-2023	12-01-2023		
9	Printing the patterns using for loop	24-25	12-01-2023	02—02-2023		
10	Shell script to determine whether given file exist or not.	25-26	02—02-2023	02-02-2023		
11	Write a program for process creation using C. (Use of gcc compiler)		02-02-2023			
12	implementation of FCFS & Round Robin Algorithm		16-02-2023			
13	Implementation of Banker's Algorithm		16-02-2023			

PRACTICAL – 1

Aim: To study the basic commands of kali Linux / Unix?

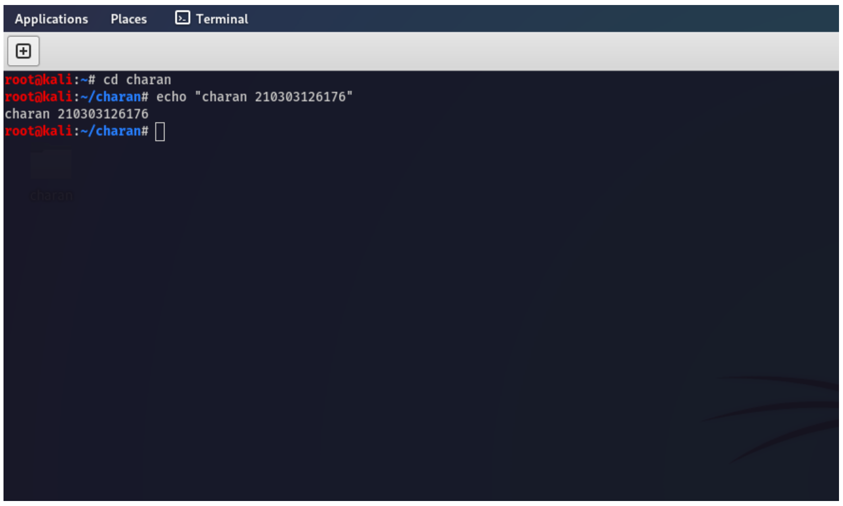
Procedure:

1. PWD
   - Description: The pwd Linux command prints the current working directory path, starting from the root **(/)**. Use the pwd command to find your way in the Linux file system structure maze or to pass the working directory in a Bash script. In this tutorial, you will learn to use the pwd command.
   - Syntax: pwd



```
root@kali:~# pwd
/root
root@kali:~# echo "charan 210303126176"
charan 210303126176
root@kali:~# 
```

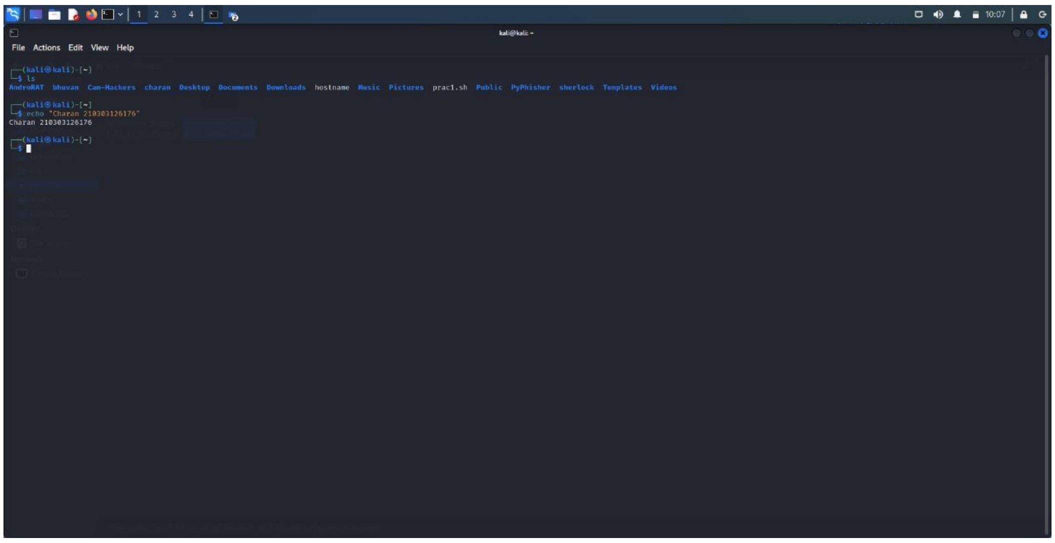
2. CD
   - Description: The cd command is used to change the current directory in both Linux and other Unix-like systems.
   - Syntax: cd [directory]



```
root@kali:~# cd charan
root@kali:~/charan# echo "charan 210303126176"
charan 210303126176
root@kali:~/charan# 
```

3. LS

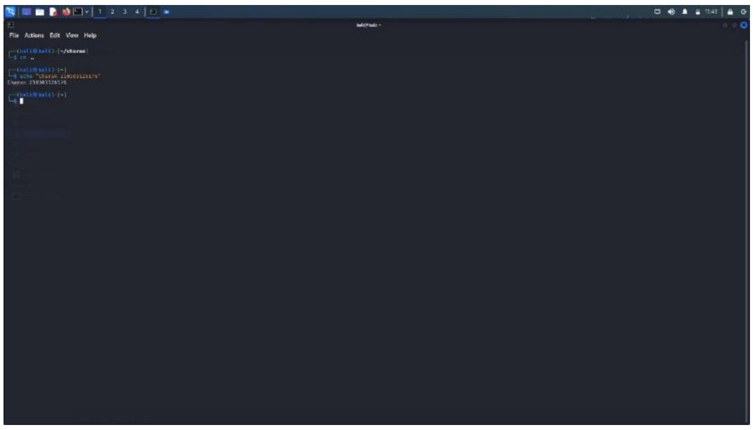
- Description: we use ls command to list files and directories. This command will print all the file and directories in the current directory.
- Syntax: ls [directory]



```
┌──(kali㉿kali)-[~]
└─$ ls
AndroRAT  bhuvan  Cam-Hackers  charan  Desktop  Documents  Downloads  hostname  Music  Pictures  prac1.sh  Public  PyPhisher  sherlock  Templates  Videos

┌──(kali㉿kali)-[~]
└─$ echo "Charan 210303126176"
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$
```

4. CD ..

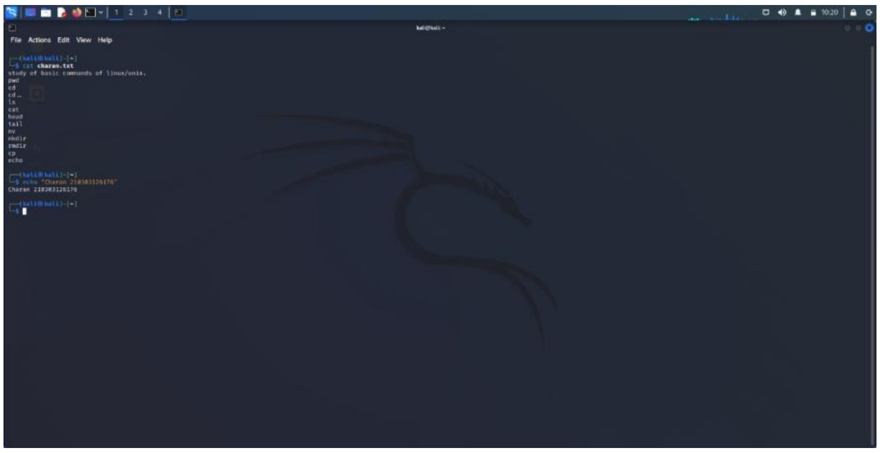
- Description: This command is used to move to the parent directory of current directory, or the directory one level up from the current directory. ".." represents parent directory.
- Syntax: cd ..



```
┌──(kali㉿kali)-[~/charan]
└─$ cd ..

┌──(kali㉿kali)-[~]
└─$ echo "Charan 210303126176"
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$
```

5. CAT

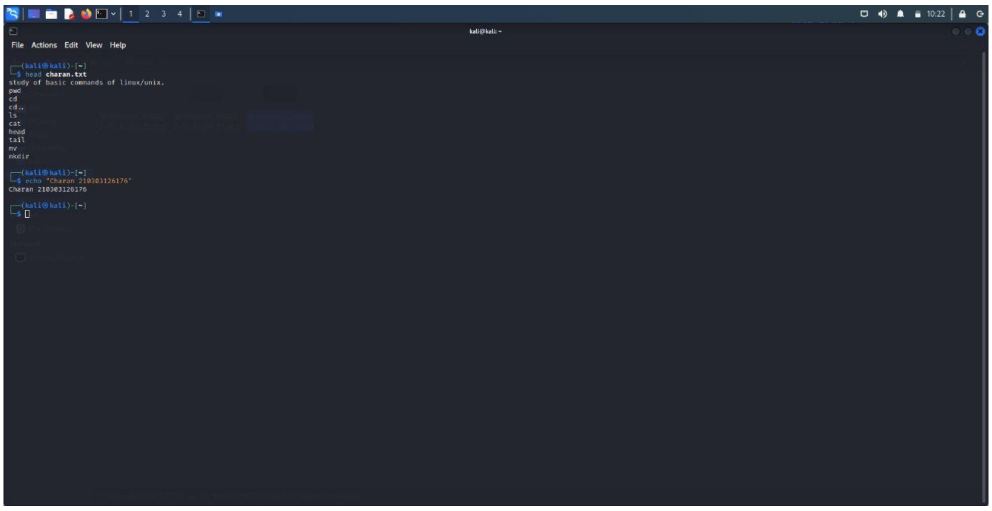
- Description: The cat command is a utility command in Linux. One of its most common usages is to print the content of a file onto the standard output stream. Other than that, the cat command also allows us to write some texts into a file.
- Syntax: cat [file-name]



```
┌──(kali㉿kali)-[~]
└─$ cat charan.txt
study of basic commands of linux/unix.
pwd
cd
cd..
ls
cat
head
tail
mv
mkdir
rmdir
cp
echo

┌──(kali㉿kali)-[~]
└─$ echo "Charan 210303126176"
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$ 
```

6. HEAD

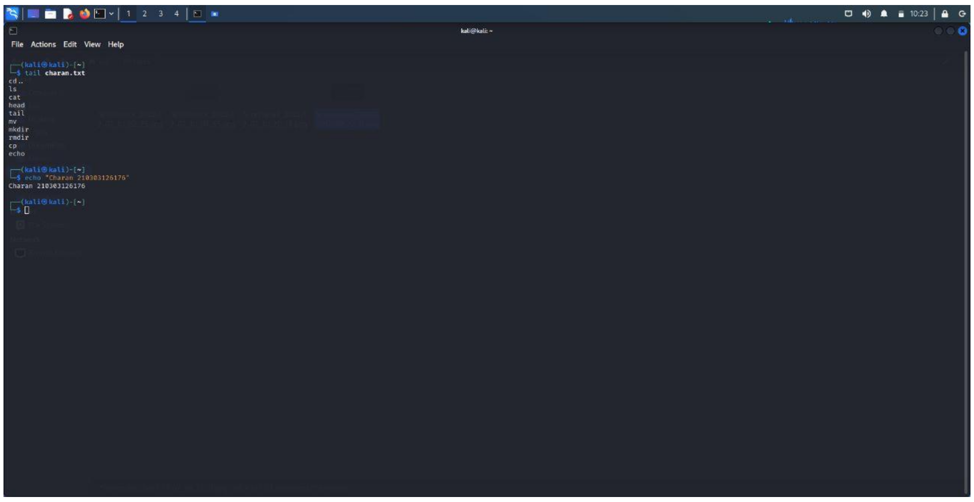
- Description: The head command, as the name implies, print the top N number of data of the given input. By default, it prints the first 10 lines of the specified files. If more than one file name is provided then data from each file is preceded by its file name.
- Syntax: head [option] [file]



```
┌──(kali㉿kali)-[~]
└─$ head charan.txt
study of basic commands of linux/unix.
pwd
cd
cd..
ls
cat
head
tail
mv
mkdir

┌──(kali㉿kali)-[~]
└─$ echo "Charan 210303126176"
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$ 
```

7. TAIL

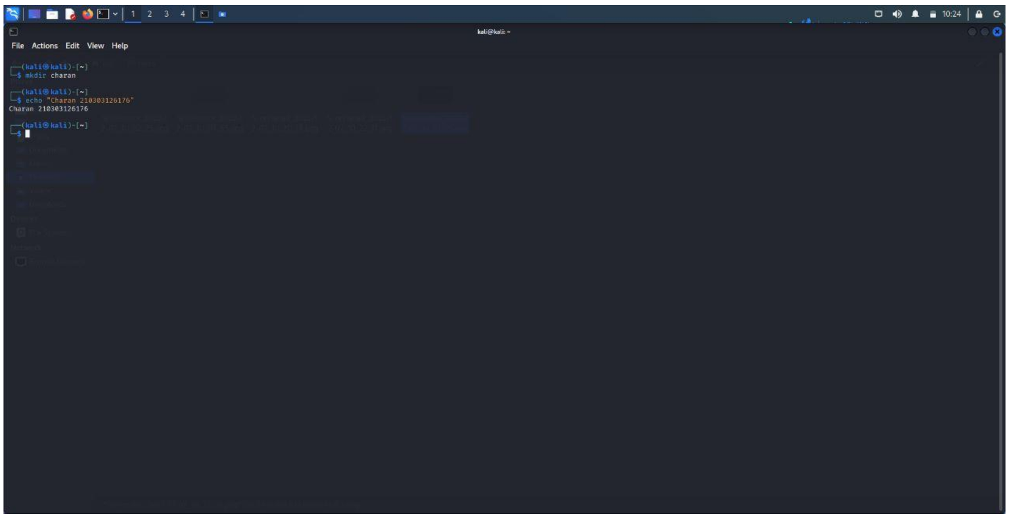
- Description: Tail is a command which prints the last few numbers of lines (10 lines by default) of a certain file, then terminates. By default, “tail” prints the last 10 lines of a file, then exits. as you can see, this prints the last 10 lines of /var/log/messages.
- Syntax: tail [option] [file]



```
┌──(kali㉿kali)-[~]
└─$ tail charan.txt
cd..
ls
cat
head
tail
mv
mkdir
rmdir
cp
echo

┌──(kali㉿kali)-[~]
└─$ echo "Charan 210303126176"
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$ 
```

8. MKDIR

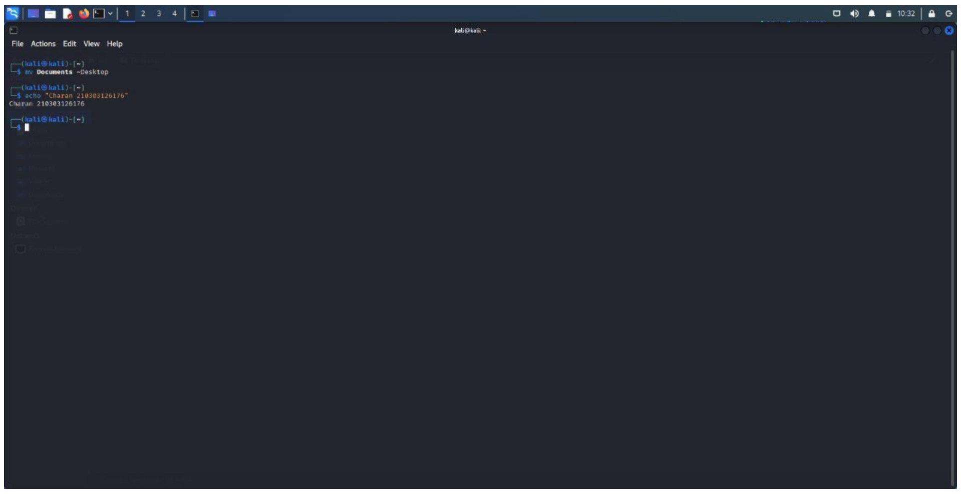
- Description: The mkdir command in Linux/Unix allows users to create or make new directories. mkdir stands for “make directory.” With mkdir , you can also set permissions, create multiple directories (folders) at once, and much more.
- Syntax: mkdir [directory name]



```
┌──(kali㉿kali)-[~]
└─$ mkdir charan

┌──(kali㉿kali)-[~]
└─$ echo "Charan 210303126176"
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$ 
```

9. MV

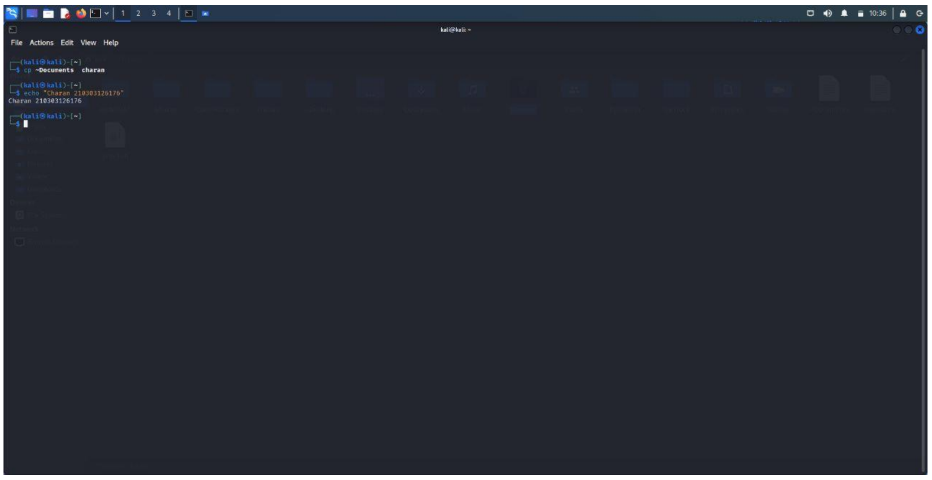
- Description: The mv command termed as “Move”, which is a command-line utility to move files or directories from source to target. It supports the moving of a single file, multiple files, and directories.
- Syntax: mv [option] source destination



```
┌──(kali㉿kali)-[~]
└─$ mv Documents ~Desktop

┌──(kali㉿kali)-[~]
└─$ echo "Charan 210303126176"
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$ 
```

10.CP

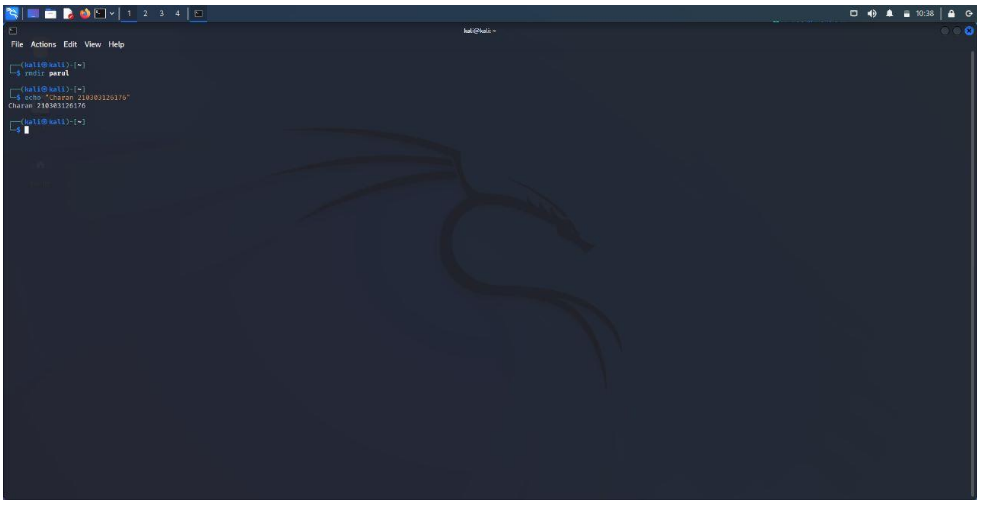
- Description: cp command copies files (or, optionally, directories). The copy is completely independent of the original. You can either copy one file to another, or copy arbitrarily many files to a destination directory. In the first format, when two file names are given, cp command copies SOURCE file to DEST file.
- Syntax: cp [option] source destination



```
┌──(kali㉿kali)-[~]
└─$ cp ~Documents  charan

┌──(kali㉿kali)-[~]
└─$ echo "Charan 210303126176"
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$ 
```

11.RMDIR

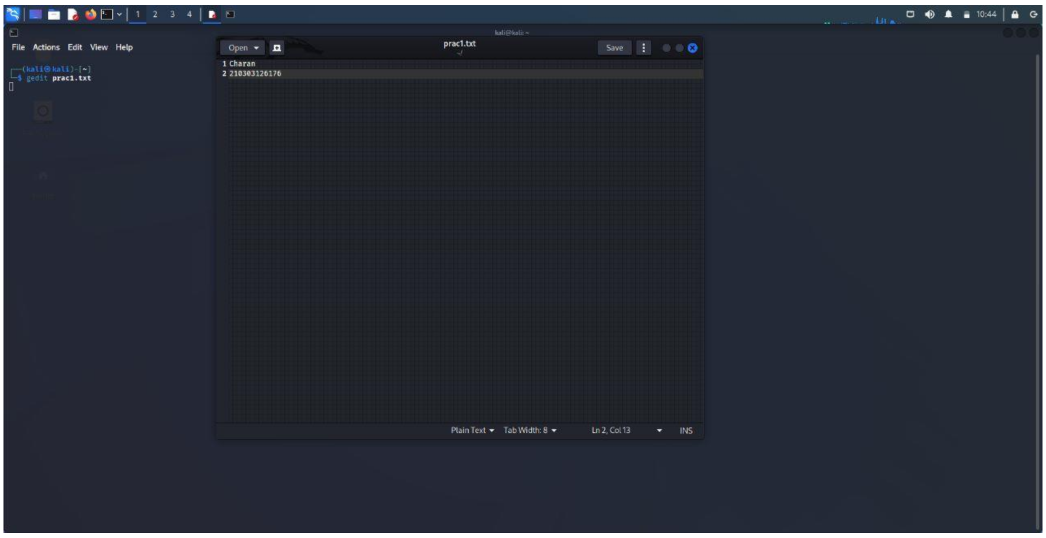
- Description: mdir command is used remove empty directories from the filesystem in Linux. The rmdir command removes each and every directory specified in the command line only if these directories are empty. So if the specified directory has some directories or files in it then this cannot be removed by rmdir command.
- Syntax: rmdir [directory name]



```
┌──(kali㉿kali)-[~]
└─$ rmdir parul

┌──(kali㉿kali)-[~]
└─$ echo "Charan 210303126176"
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$ 
```

12.GEDIT

- Description: The gedit command is used to create and open a file
- Syntax: gedit filename.txt



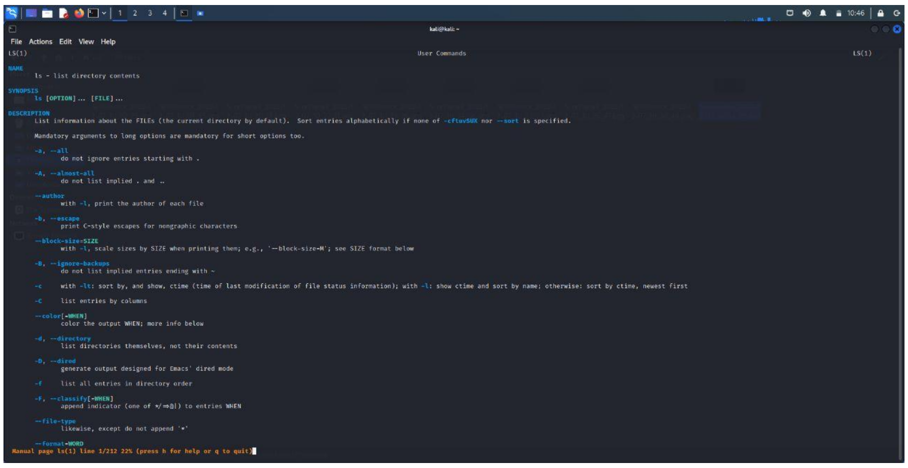
```
┌──(kali㉿kali)-[~]
└─$ gedit prac1.txt
```

prac1.txt

```
Charan
210303126176
```

13.MAN

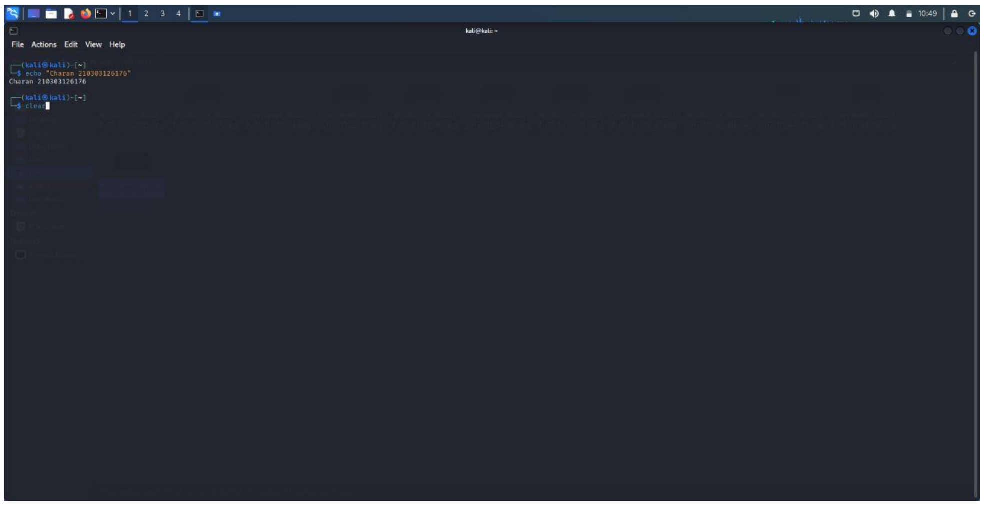
- Description: man command in Linux is used to display the user manual of any command that we can run on the terminal. It provides a detailed view of the command which includes NAME, SYNOPSIS, DESCRIPTION, OPTIONS, EXIT STATUS, RETURN VALUES, ERRORS, FILES, VERSIONS, EXAMPLES, AUTHORS
- Syntax: man command



```
LS(1)                              User Commands                              LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List information about the FILEs (the current directory by default).  Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

       Mandatory arguments to long options are mandatory for short options too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic characters

       --block-size=SIZE
              with -l, scale sizes by SIZE when printing them; e.g., '--block-size=M'; see SIZE format below

       -B, --ignore-backups
              do not list implied entries ending with ~

       -c     with -lt: sort by, and show, ctime (time of last modification of file status information); with -l: show ctime and sort by name; otherwise: sort by ctime, newest first

       -C     list entries by columns

       --color[=WHEN]
              color the output WHEN; more info below

       -d, --directory
              list directories themselves, not their contents

       -D, --dired
              generate output designed for Emacs' dired mode

       -f     list all entries in directory order

       -F, --classify[=WHEN]
              append indicator (one of */=>@|) to entries WHEN

       --file-type
              likewise, except do not append '*'

       --format=WORD
 Manual page ls(1) line 1/212 22% (press h for help or q to quit)
```

14.ECHO

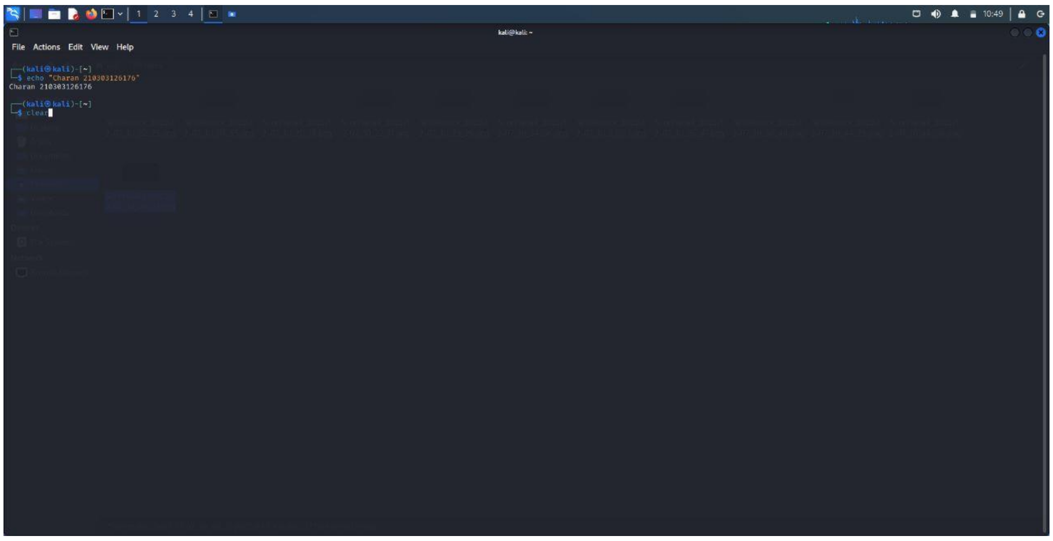
- Description: Display text on the screen
- Syntax: Display text on the screen



```
┌──(kali㉿kali)-[~]
└─$ echo "Charan 210303126176"
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$ clear
```

15.CLEAR

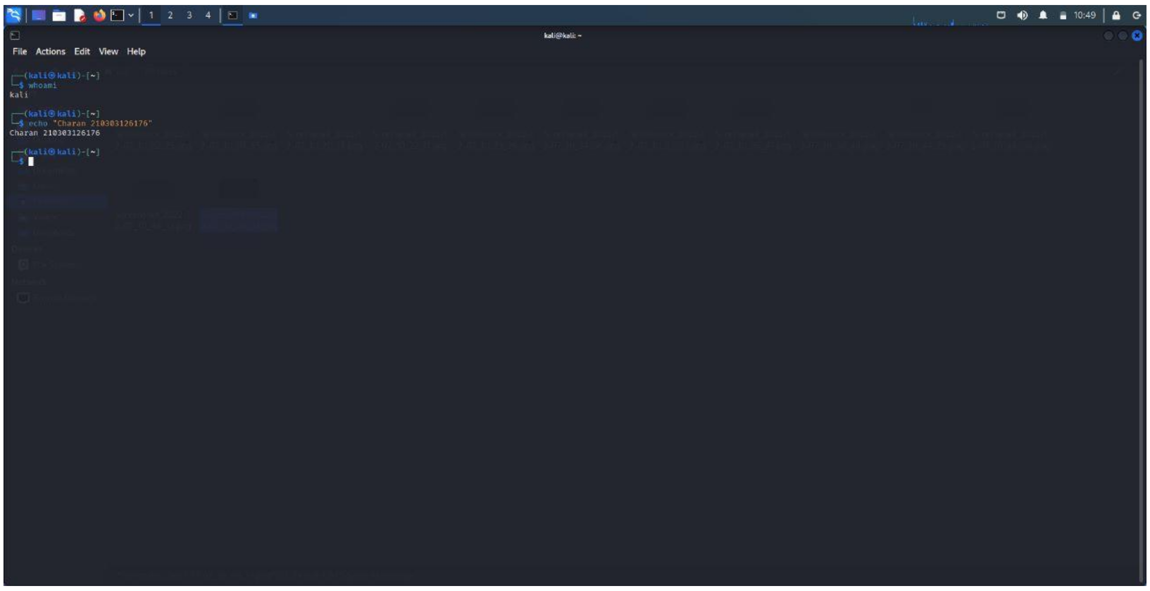
- Description: Used to clear the screen
- Syntax: clear



```
┌──(kali㉿kali)-[~]
└─$ echo "Charan 210303126176"
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$ clear
```

16.WHOAMI

- Description: whoami prints the effective user ID. This command prints the username associated with the current effective user ID
- Syntax: whoami [option]



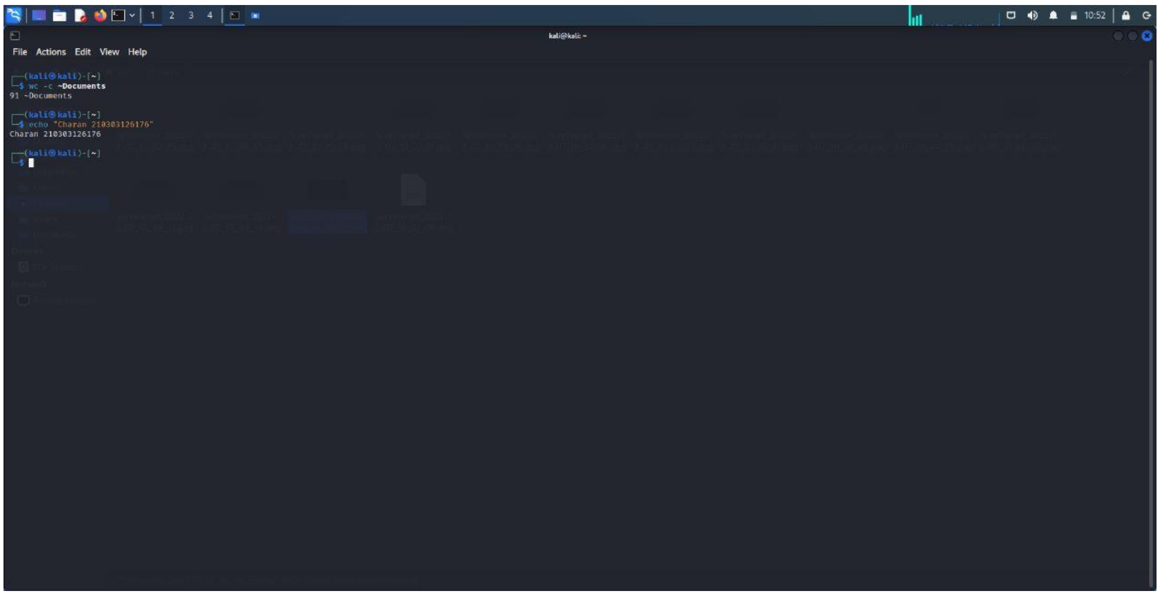
```
┌──(kali㉿kali)-[~]
└─$ whoami
kali

┌──(kali㉿kali)-[~]
└─$ echo "Charan 210303126176"
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$ 
```

17.WC

- Description: wc (word count) command, can return the number of lines, words, and characters in a file.
- Syntax: wc [option]… [file]…

Example:

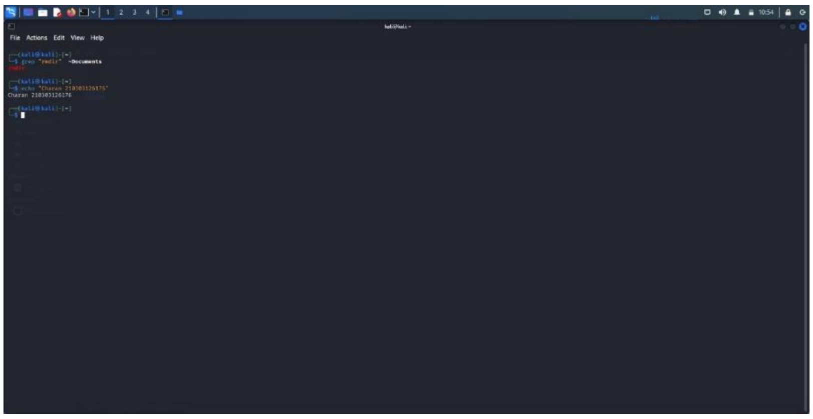
✓ Print the byte counts of file myfile.txt
wc -c myfile.txt
✓ Print the line counts of file myfile.tx
wc -l myfile.txt
✓ Print the word counts of file myfile.txt
wc -w myfile.txt



```
┌──(kali㉿kali)-[~]
└─$ wc -c ~Documents
91 ~Documents

┌──(kali㉿kali)-[~]
└─$ echo "Charan 210303126176"
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$ 
```

18.GREP

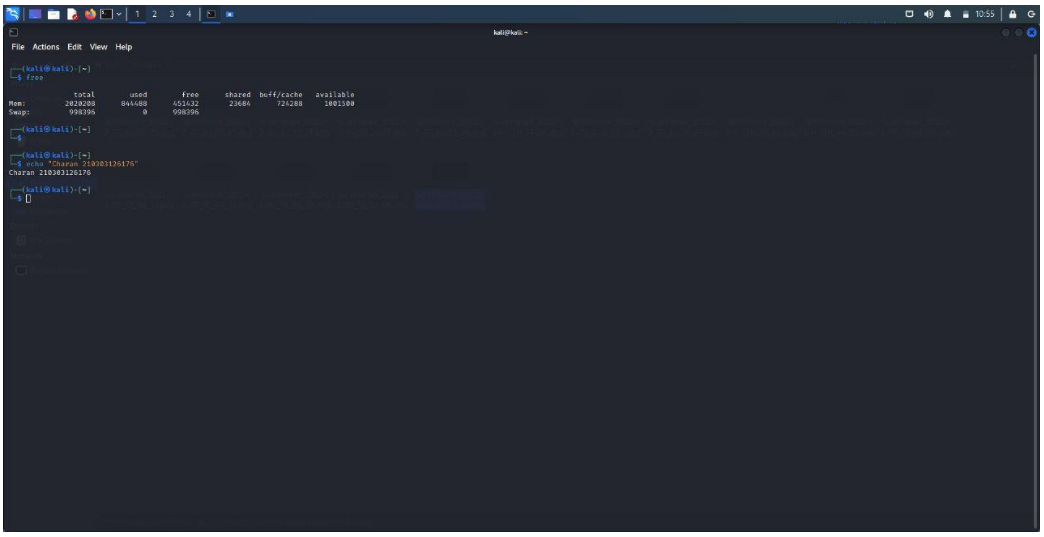
- Description: grep command uses a search term to look through a file
- Syntax: grep [option]… Pattern [file]



```
┌──(kali㉿kali)-[~]
└─$ grep "rmdir" ~Documents
rmdir

┌──(kali㉿kali)-[~]
└─$ echo "Charan 210303126176"
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$ 
```

19.FREE

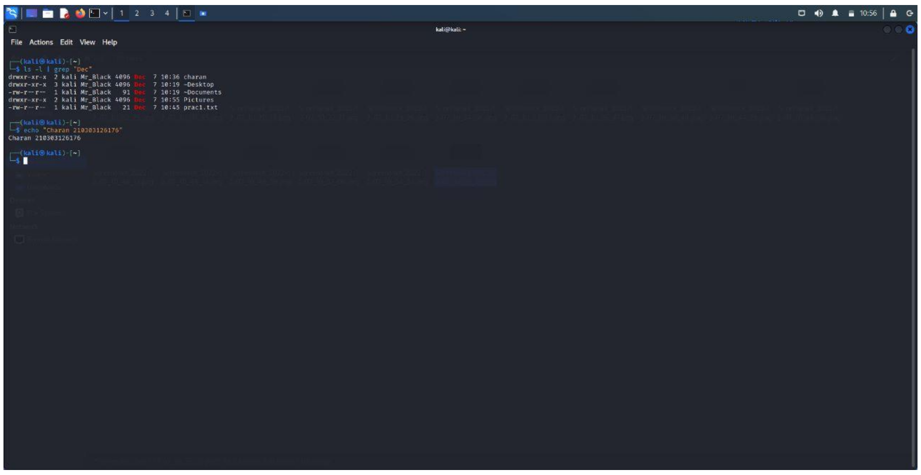
- Description: To display the RAM details in Linux machine need to write following command.
- Syntax: free



```
┌──(kali㉿kali)-[~]
└─$ free
               total        used        free      shared  buff/cache   available
Mem:         2020208      844488      451432       23684      724288     1001500
Swap:         998396           0      998396

┌──(kali㉿kali)-[~]
└─$

┌──(kali㉿kali)-[~]
└─$ echo "Charan 210303126176"
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$ 
```

20.PIPE (|)

- Description: Pipe command is used to send output of one program as a input to another. Pipes "|" help combine 2 or more commands
- Syntax: Command 1 | command 2



```
┌──(kali㉿kali)-[~]
└─$ ls -l | grep "Dec"
drwxr-xr-x  2 kali Mr_Black 4096 Dec  7 10:36 charan
drwxr-xr-x  3 kali Mr_Black 4096 Dec  7 10:19 -Desktop
-rw-r--r--  1 kali Mr_Black   91 Dec  7 10:19 ~Documents
drwxr-xr-x  2 kali Mr_Black 4096 Dec  7 10:55 Pictures
-rw-r--r--  1 kali Mr_Black   21 Dec  7 10:45 prac1.txt

┌──(kali㉿kali)-[~]
└─$ echo "Charan 210303126176"
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$ 
```

PRACTICAL – 2

Aim: Study the basics of Shell of a linux

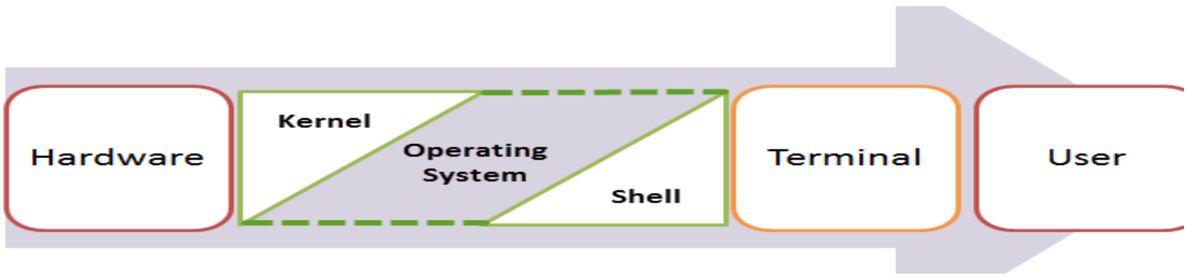
What is a Shell?

It is a list of commands in a computer program that is run by the Unix shell which is a command line interpreter. A shell script usually has comments that describe the steps.

An Operating is made of many components,

But its two prime components are –

- ✓ Kernel
- ✓ Shell



A Kernel is at the nucleus of a computer. It makes the communication between the hardware and software possible. While the Kernel is the innermost part of an operating system, a shell is the outermost one. A shell in a Linux operating system takes input from you in the form of commands, processes it, and then gives an output. It is the interface through which a user works on the programs, commands, and scripts. A shell is accessed by a terminal which runs it. When you run the terminal, the Shell issues a command prompt (usually $), where you can type your input, which is then executed when you hit the Enter key. The output or the result is thereafter displayed on the terminal. The Shell wraps around the delicate interior of an Operating system protecting it from accidental damage. Hence the name Shell.

Types Of Shells:

1. Bournee shell : This is default shell for version 7 unix. The character $ is the default prompt for the bourne shell.
2. C shell : This is a unix shell and a command processor that is run in a text window . The character % is the default prompt for the C shell. File commands can also be read easily by the C shell , which is known as a script.

How to create file in linux :

In linux there are two commands which are used to create the files in linux :

1. Gedit

2. nano

What is Shell Scripting?

Shell scripting is writing a series of command for the shell to execute. It can combine lengthy and repetitive sequences of commands into a single and simple script, which can be stored and executed anytime. This reduces the effort required by the end user. Let us understand the steps in creating a Shell Script

1. Create a file using a vi editor(or any other editor). Name script file with extension .sh

2. Start the script with #! /bin/sh

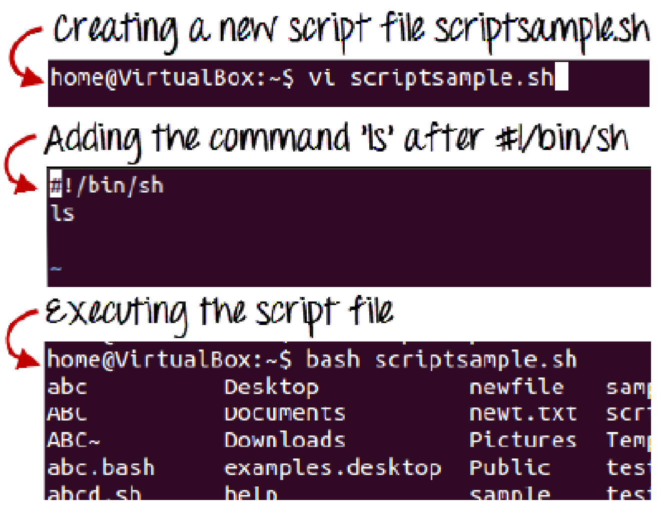
3. Write some code.

4. Save the script file as filename.sh

5. For executing the script type bash filename.sh

"#!" is an operator called shebang which directs the script to the interpreter location. So, if we use"#! /bin/sh" the script gets directed to the bourne-shell. Let's create a small script -

Let's create a small script –


Creating a new script file scriptsample.sh
Adding the command 'ls' after #!/bin/sh
Executing the script file

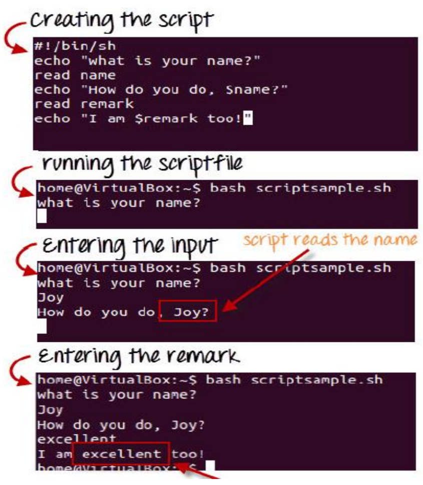
```
home@VirtualBox:~$ vi scriptsample.sh
```

```
#!/bin/sh
ls

~
```

```
home@VirtualBox:~$ bash scriptsample.sh
abc           Desktop          newfile    sam
ABC           Documents        newt.txt   scr
ABC~          Downloads        Pictures   Tem
abc.bash      examples.desktop Public     tes
abcd.sh       help             sample     tes
```

```
#!/bin/sh
ls
```

Let's see the steps to create it –



```
#!/bin/sh
echo "what is your name?"
read name
echo "How do you do, $name?"
read remark
echo "I am $remark too!"
```

```
home@VirtualBox:~$ bash scriptsample.sh
what is your name?
```

```
home@VirtualBox:~$ bash scriptsample.sh
what is your name?
Joy
How do you do, Joy?
```

```
home@VirtualBox:~$ bash scriptsample.sh
what is your name?
Joy
How do you do, Joy?
excellent
I am excellent too!
home@VirtualBox:~$
```

Command 'ls' is executed when we execute the scrip sample.sh file.

Adding shell comments

Commenting is important in any program. In Shell programming, the syntax to add a comment is

#comment
Let understand this with an example.

What are Shell Variables?

As discussed earlier, Variables store data in the form of characters and numbers. Similarly, Shell variables are used to store information and they can by the shell only.

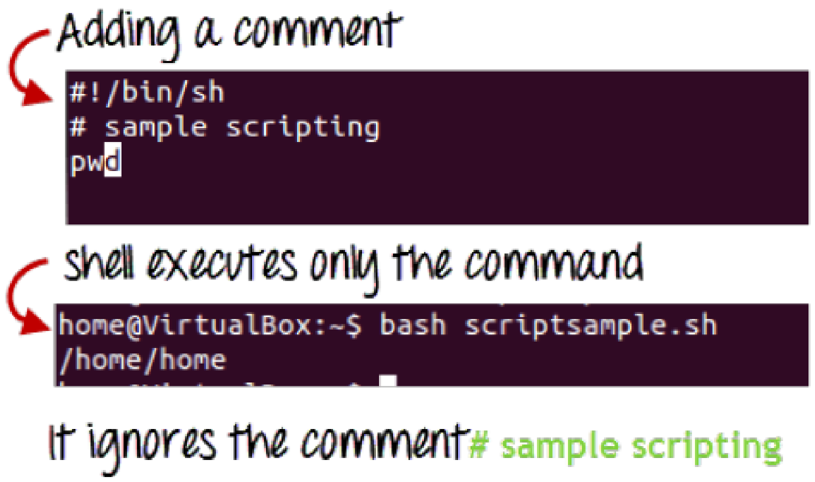
For example, the following creates a shell variable and then prints it:

variable ="Hello"
echo $variable

Below is a small script which will use a variable.

```
#!/bin/sh
echo "what is your name?"
read name
echo "How do you do, $name?"
read remark
echo "I am $remark too!"
```

Let's understand, the steps to create and execute the script


Adding a comment
Shell executes only the command
It ignores the comment# sample scripting

```
#!/bin/sh
# sample scripting
pwd
```

```
home@VirtualBox:~$ bash scriptsample.sh
/home/home
```

As you see, the program picked the value of the variable 'name' as Joy and 'remark' as excellent.

This is a simple script. You can develop advanced scripts which contain conditional statements, loops, and functions. Shell scripting will make your life easy and Linux administration a breeze.

Summary:

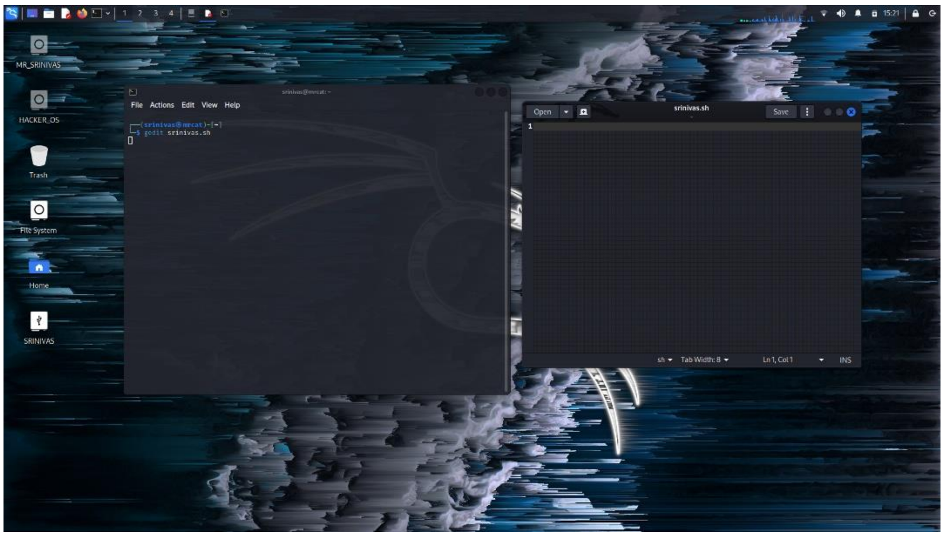
- Kernel is the nucleus of the operating systems, and it communicates between hardware and software
- Shell is a program which interprets user commands through CLI like Terminal
- The Bourne shell and the C shell are the most used shells in Linux
- Shell scripting is writing a series of command for the shell to execute
- Shell variables store the value of a string or a number for the shell to read
- Shell scripting can help you create complex programs containing conditional statements, loops, and functions .

These two commands are useful to create the files.

1. **Gedit :**

   Syntax : gedit prac1.txt

   Description: Gedit, the deafault GUI editor if you use Gnome ,also runs under KDE and other desktops . Most gNewsense and linux installations use gnome by default. To start Gedit open a terminal and type.



```
┌──(srinivas㉿mrcat)-[~]
└─$ gedit srinivas.sh
```

After Gedit command function a new window will open in that we have to give input. After giving input we have save the file and after that use command bash.

Bash :

Syntax : bash prac11.sh

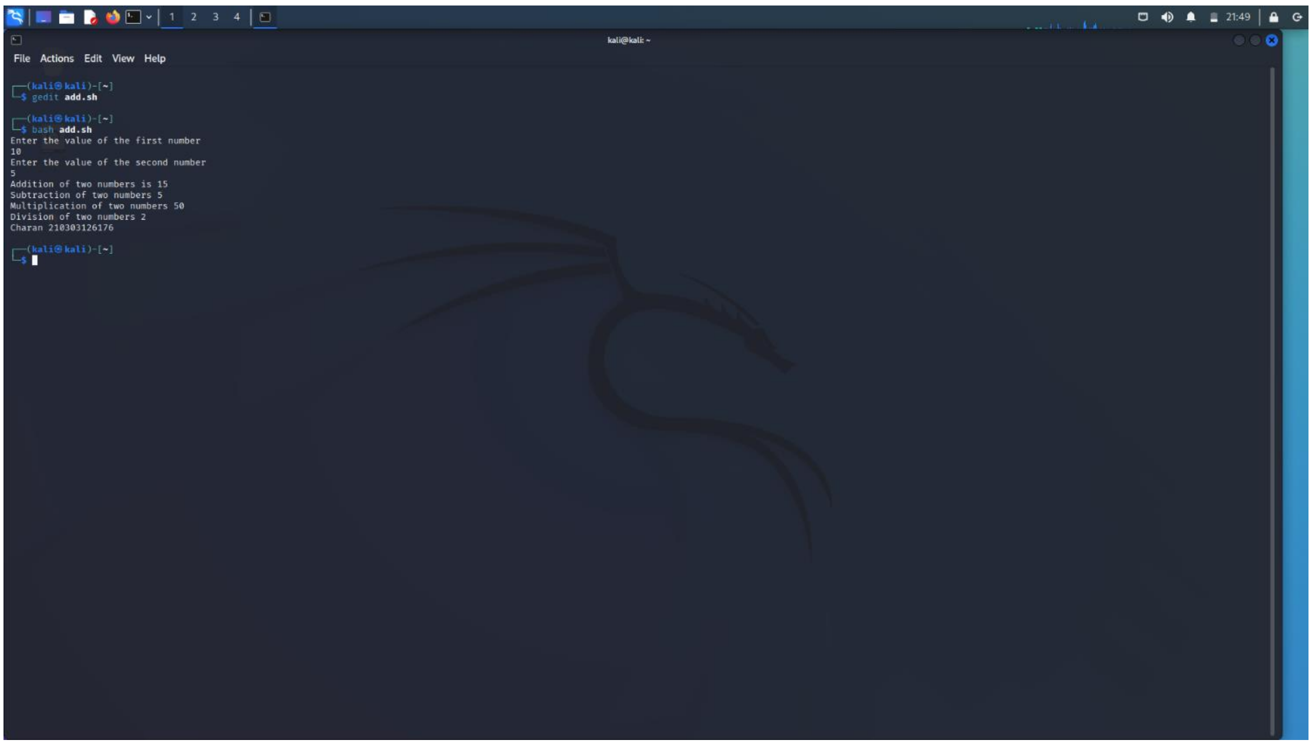
Description : it is used to read the data in existing file in the linux .

PRACTICAL – 3

Aim: Perform the operations on the Shell Script to perform sum operations:

Sample code:-

✓
```
echo "enter the value of n";
read n;
sum=0;
while [ $n -gt 0 ]
do
a=`expr $n % 10`
sum=`expr $sum + $a`
n=`expr $n / 10`
done
echo "sum is $sum";
```



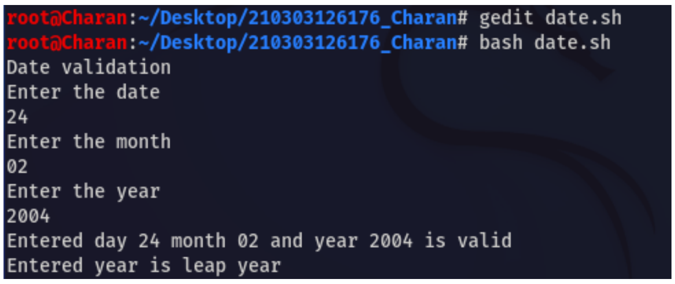
```
┌──(kali㉿kali)-[~]
└─$ gedit add.sh

┌──(kali㉿kali)-[~]
└─$ bash add.sh
Enter the value of the first number
10
Enter the value of the second number
5
Addition of two numbers is 15
Subtraction of two numbers 5
Multiplication of two numbers 50
Division of two numbers 2
Charan 210303126176

┌──(kali㉿kali)-[~]
└─$ 
```

Practical-4

Aim: -Write a shell script to validate the entered date. (eg. Date format is: dd-mm-yyyy)

Sample code: -

```
echo "Date validation";
echo "Enter the date";
read d;
echo "Enter the month";
read m;
echo "Enter the year";
read y;
a=`expr $y % 4 `;
b=`expr $y % 400`;
if(($d>=1 && $d<=31 && $m>=1 && $m<=12 && $y>=1 && $y<=2025 ))
then
echo "Entered day $d month $m and year $y is valid ";
if [ $a -eq 0 ] || [ $b -eq 0 ]
then
echo "Entered year is leap year ";
else
echo "Entered year is not a leap year ";
fi
else
echo "valid date is not found";
fi
```



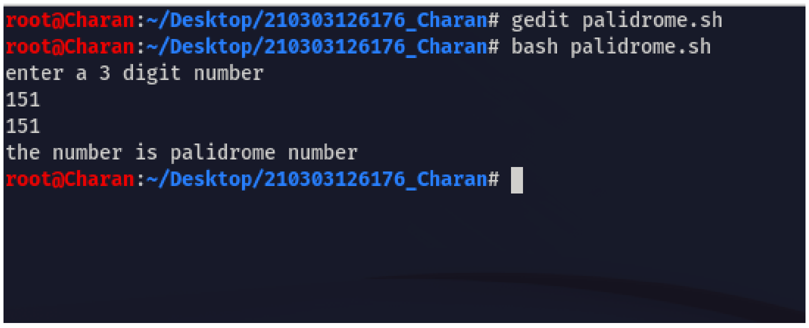
```
root@Charan:~/Desktop/210303126176_Charan# gedit date.sh
root@Charan:~/Desktop/210303126176_Charan# bash date.sh
Date validation
Enter the date
24
Enter the month
02
Enter the year
2004
Entered day 24 month 02 and year 2004 is valid
Entered year is leap year
```

Practical-5

Aim: -Write a shell script to print whether the number is palindrome or not?

Sample code: -

```
 echo "enter a 3 digit number";
read a;
c=$a;
sum=0;
while [ $a -ne 0 ]
do
        b=`expr $a % 10 `;
        sum=`expr $sum \* 10 `;
        sum=`expr $sum + $b`;
        a=`expr $a / 10 `;

done
echo "$sum";

if [ $c -eq $sum ]
then
 echo "the number is palindrome number ";
else
 echo "the number is not palindrome number";
fi
```



```
root@Charan:~/Desktop/210303126176_Charan# gedit palidrome.sh
root@Charan:~/Desktop/210303126176_Charan# bash palidrome.sh
enter a 3 digit number
151
151
the number is palidrome number
root@Charan:~/Desktop/210303126176_Charan#
```

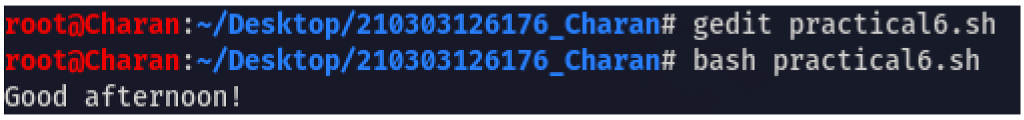
-

Practical-6

Aim: -Write a Shell script to say good morning/Afternoon/Evening as you log in to system

Sample code: -

```
current_time=$(date +%H)
if [ $current_time -lt 12 ]; then
echo "Good morning!"
elif [ $current_time -lt 17 ]; then
echo "Good afternoon!"
else
echo "Good evening";
fi
```



```
root@Charan:~/Desktop/210303126176_Charan# gedit practical6.sh
root@Charan:~/Desktop/210303126176_Charan# bash practical6.sh
Good afternoon!
```

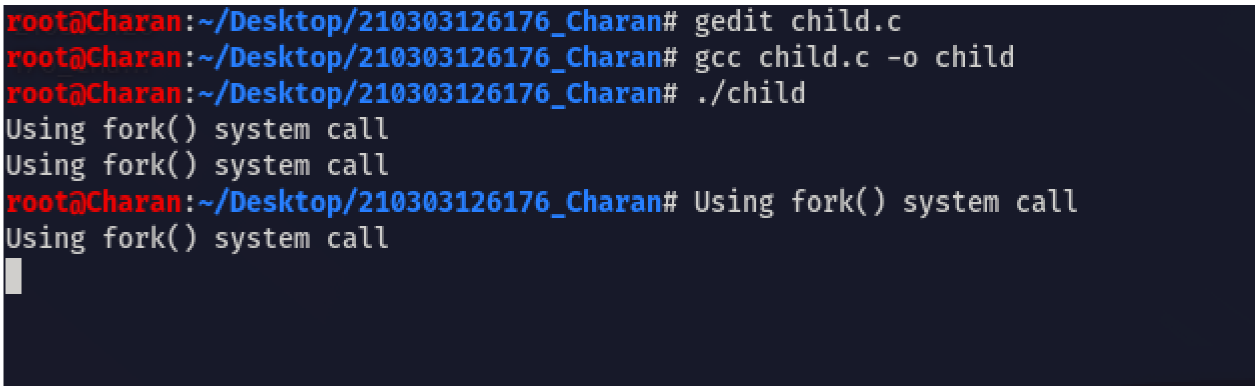
Practical-7

Aim: - Write a c program to create a child process

Sample code: -

```
#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

   fork();

   fork();

   printf("Using fork() system call \n");

   return 0;

 }
```

Output:-



```
root@Charan:~/Desktop/210303126176_Charan# gedit child.c
root@Charan:~/Desktop/210303126176_Charan# gcc child.c -o child
root@Charan:~/Desktop/210303126176_Charan# ./child
Using fork() system call
Using fork() system call
root@Charan:~/Desktop/210303126176_Charan# Using fork() system call
Using fork() system call
```

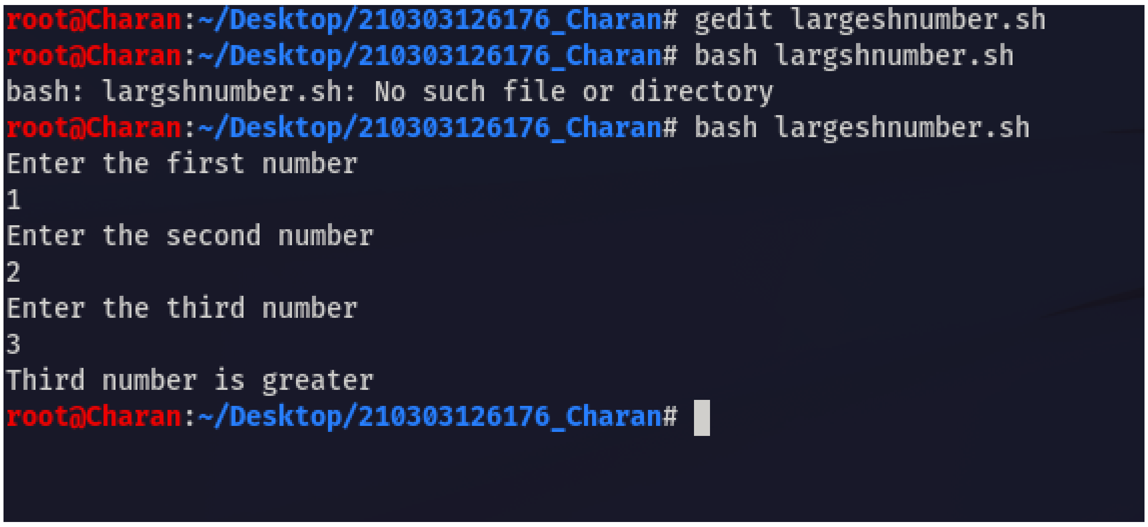
Practical-8

Aim: - Find out the largest number from three numbers

Sample Code:-

```
echo "Enter the first number "
read a
echo "Enter the second number"
read b
echo "Enter the third number"
read c
if [ $a -gt $b ] && [ $a -gt $c ]
then
   echo "First number is greater "
elif [ $b -gt $a ] && [ $b -gt $c ]
then
   echo "Second number is greater "
else
echo "Third number is greater "
fi
```

Output: -



```
root@Charan:~/Desktop/210303126176_Charan# gedit largeshnumber.sh
root@Charan:~/Desktop/210303126176_Charan# bash largshnumber.sh
bash: largshnumber.sh: No such file or directory
root@Charan:~/Desktop/210303126176_Charan# bash largeshnumber.sh
Enter the first number
1
Enter the second number
2
Enter the third number
3
Third number is greater
root@Charan:~/Desktop/210303126176_Charan#
```

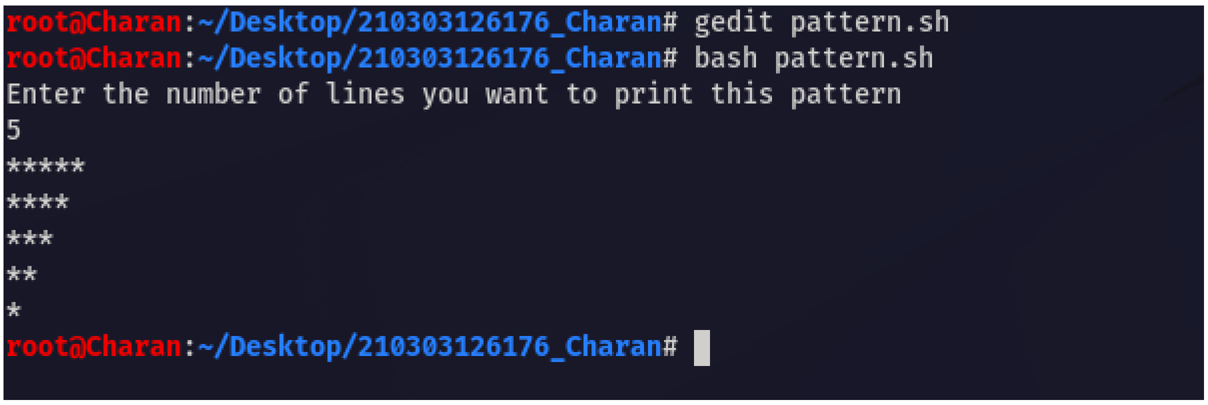
Practical-9

Aim: - Print the pattern using for loop

Sample code: -

```
echo "Enter the number of lines you want to print this pattern"
read n
for((i=1;i<=n;i++))
do
  for((j=i;j<=n;j++))
  do
    echo -ne "*"
  done
  echo
done
```

Output: -



```
root@Charan:~/Desktop/210303126176_Charan# gedit pattern.sh
root@Charan:~/Desktop/210303126176_Charan# bash pattern.sh
Enter the number of lines you want to print this pattern
5
*****
****
***
**
*
root@Charan:~/Desktop/210303126176_Charan#
```

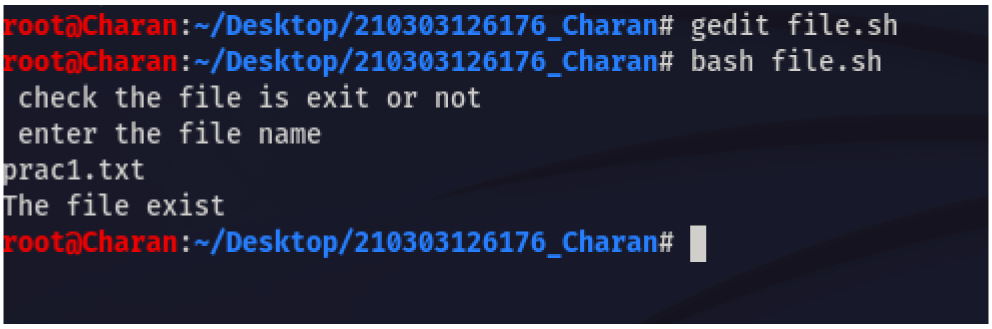
Practical-10

Aim: -Shell script to determine whether given file exist or not

Sample code: -

```
echo " check the file is exit or not ";

echo " enter the file name ";

read a;

if [ -f $a ]

then

   echo "The file exist "

else

   echo "The file dose not exist "

   echo "Do you want to create that file y/n "

   read b

   if [ $b= y ]

   then

      touch $a;

      echo $ls;

      echo "The file created successufully "

   else

     echo "Thank you "

   fi

fi
```

Output: -



```
root@Charan:~/Desktop/210303126176_Charan# gedit file.sh
root@Charan:~/Desktop/210303126176_Charan# bash file.sh
 check the file is exit or not
 enter the file name
prac1.txt
The file exist
root@Charan:~/Desktop/210303126176_Charan# 
```