FACULTY OF ENGINEERING & TECHNOLOGY

BACHELOR OF TECHNOLOGY

OPERATING SYSTEMS LAB

(303105252)

4th SEMESTER

Department of Artificial Intelligence

LABORATORY MANUAL 2024-25

CERTIFICATE

This is to certified that Ms.Kakkirala Sri Naga Venkata Lavanya with enrolment No: 2303031240478 has successfully completed his/her laboratory experiments in the **OPERATING SYSTEM (303105252)** from the department of AI & AIDS during the academic year **2024-2025.**

Date of Submission:

Staff in charge:

Head of Department:

INDEX

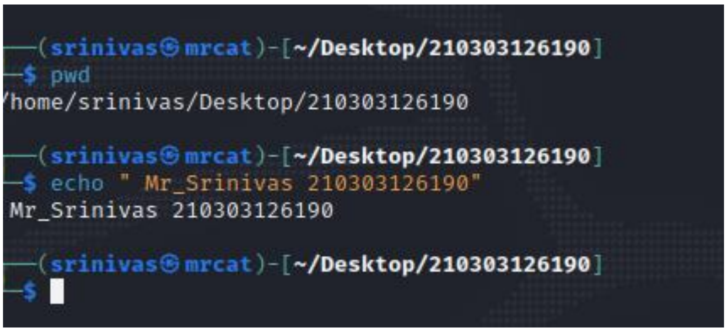
PRACTICAL – 1

Aim: Study of Basic commands of Linux.

Procedure:

1. PWD

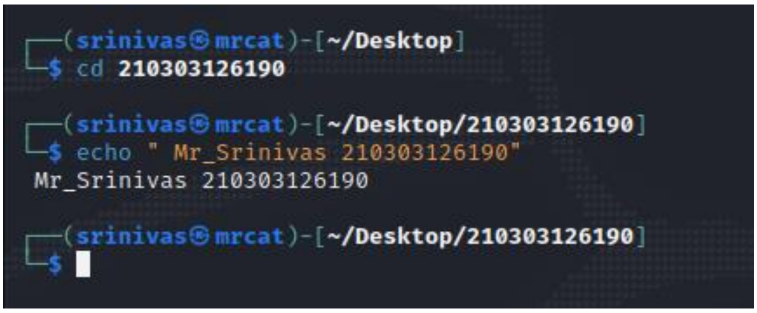
- Description: The pwd Linux command prints the current working directory path, starting from the root **(/)**. Use the pwd command to find your way in the Linux file system structure maze or to pass the working directory in a Bash script. In this tutorial, you will learn to use the pwd command.
- Syntax: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



```
──(srinivas㉿mrcat)-[~/Desktop/210303126190]
─$ pwd
/home/srinivas/Desktop/210303126190

──(srinivas㉿mrcat)-[~/Desktop/210303126190]
─$ echo " Mr_Srinivas 210303126190"
Mr_Srinivas 210303126190

──(srinivas㉿mrcat)-[~/Desktop/210303126190]
─$ █
```

2. CD

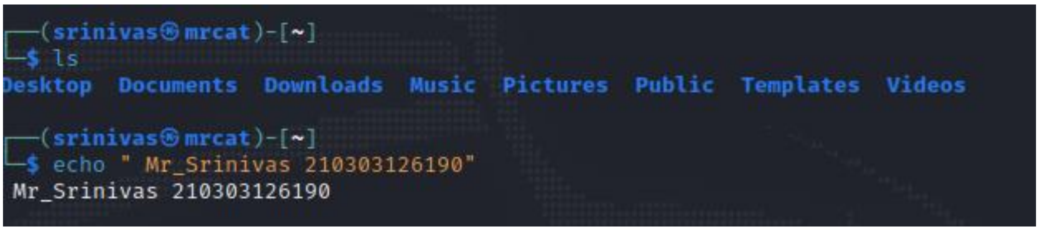
- Description: The cd command is used to change the current directory in both Linux and other Unix-like systems.
- Syntax: cd [directory]



```
┌──(srinivas㉿mrcat)-[~/Desktop]
└─$ cd 210303126190

┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ echo " Mr_Srinivas 210303126190"
Mr_Srinivas 210303126190

┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ █
```

3. LS

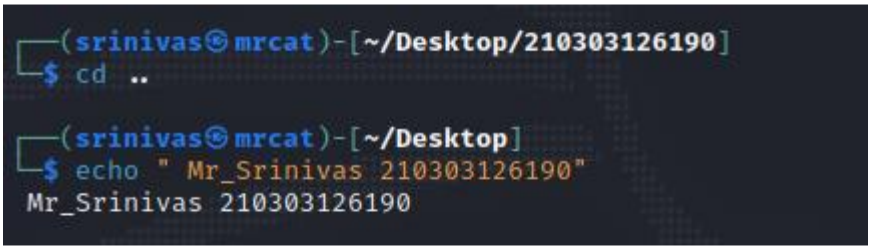
- Description: we use ls command to list files and directories. This command will print all the file and directories in the current directory.
- Syntax: ls [directory]



```
┌──(srinivas㉿mrcat)-[~]
└─$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos

┌──(srinivas㉿mrcat)-[~]
└─$ echo " Mr_Srinivas 210303126190"
 Mr_Srinivas 210303126190
```

4. CD ..

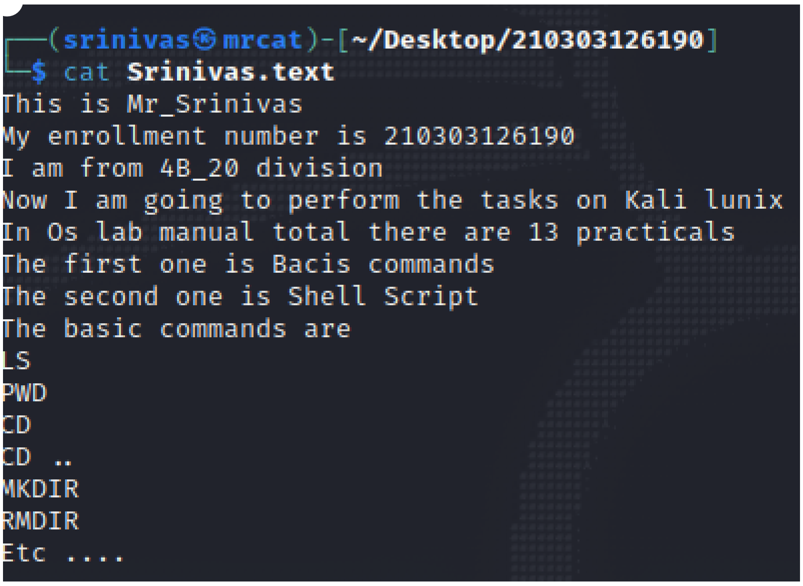
- Description: This command is used to move to the parent directory of current directory, or the directory one level up from the current directory. ".." represents parent directory.
- Syntax: cd ..



```
┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ cd ..

┌──(srinivas㉿mrcat)-[~/Desktop]
└─$ echo " Mr_Srinivas 210303126190"
 Mr_Srinivas 210303126190
```

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]

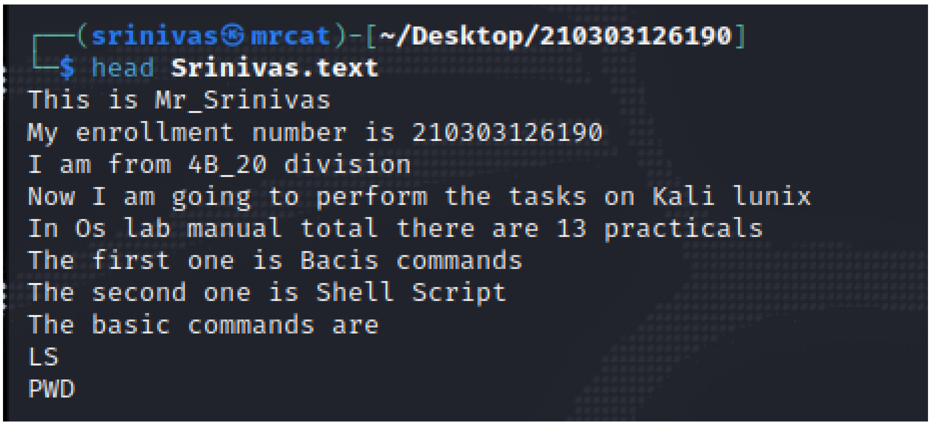


```
┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ cat Srinivas.text
This is Mr_Srinivas
My enrollment number is 210303126190
I am from 4B_20 division
Now I am going to perform the tasks on Kali lunix
In Os lab manual total there are 13 practicals
The first one is Bacis commands
The second one is Shell Script
The basic commands are
LS
PWD
CD
CD ..
MKDIR
RMDIR
Etc ....
```

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.

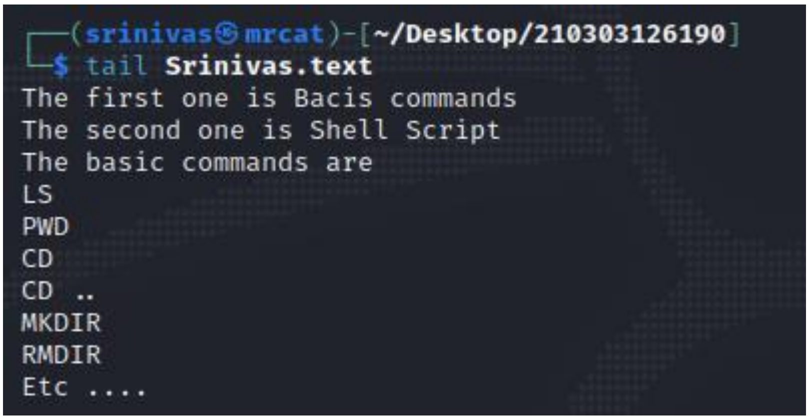
- Syntax: head [option] [file]



```
┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ head Srinivas.text
This is Mr_Srinivas
My enrollment number is 210303126190
I am from 4B_20 division
Now I am going to perform the tasks on Kali lunix
In Os lab manual total there are 13 practicals
The first one is Bacis commands
The second one is Shell Script
The basic commands are
LS
PWD
```

7. TAIL

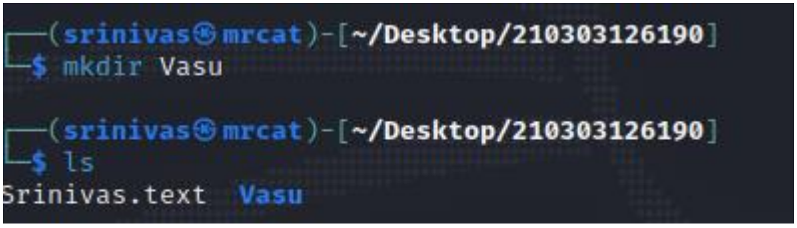
- Description: Tail is a command which prints the last few numbers of lines (10 lines by default) of a certain file, then terminates. By default, “tail” prints the last 10 lines of a file, then exits. as you can see, this prints the last 10 lines of /var/log/messages.
- Syntax: tail [option] [file]



```
┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ tail Srinivas.text
The first one is Bacis commands
The second one is Shell Script
The basic commands are
LS
PWD
CD
CD ..
MKDIR
RMDIR
Etc ....
```

8. MKDIR

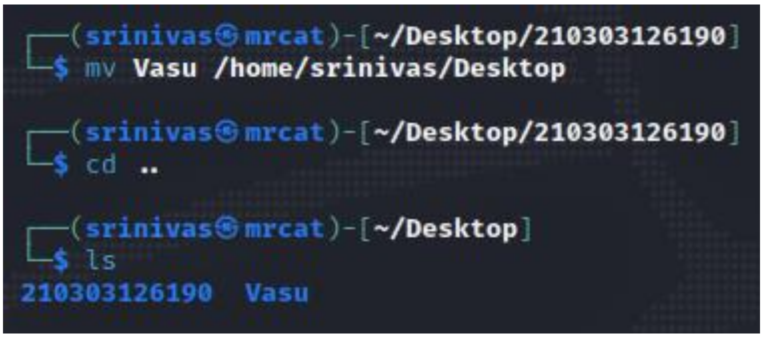
- Description: The mkdir command in Linux/Unix allows users to create or make new directories. mkdir stands for “make directory.” With mkdir , you can also set permissions, create multiple directories (folders) at once, and much more.
- Syntax: mkdir [directory name]



```
┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ mkdir Vasu

┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ ls
Srinivas.text  Vasu
```

9. MV

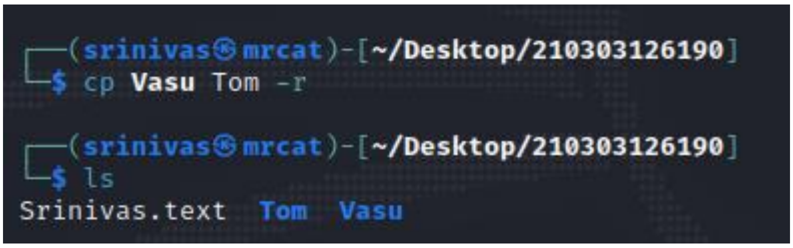
- Description: The mv command termed as “Move”, which is a command-line utility to move files or directories from source to target. It supports the moving of a single file, multiple files, and directories.
- Syntax: mv [option] source destination



```
┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ mv Vasu /home/srinivas/Desktop

┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ cd ..

┌──(srinivas㉿mrcat)-[~/Desktop]
└─$ ls
210303126190  Vasu
```

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

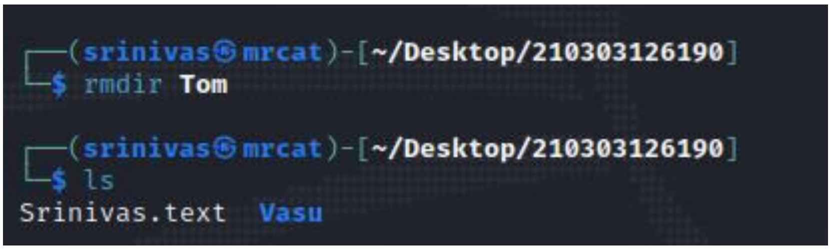


```
┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ cp Vasu Tom -r

┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ ls
Srinivas.text  Tom  Vasu
```

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.

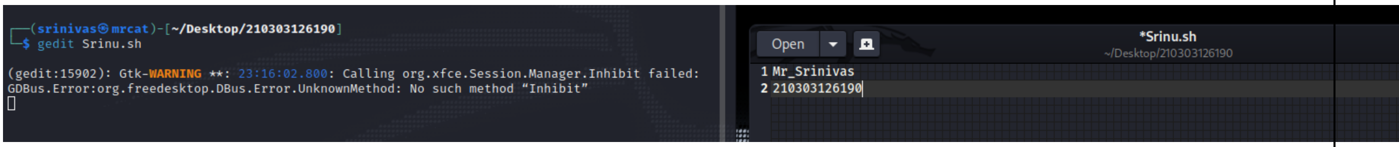
- Syntax: rmdir [directory name]



```
┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ rmdir Tom

┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ ls
Srinivas.text   Vasu
```

12.GEDIT

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

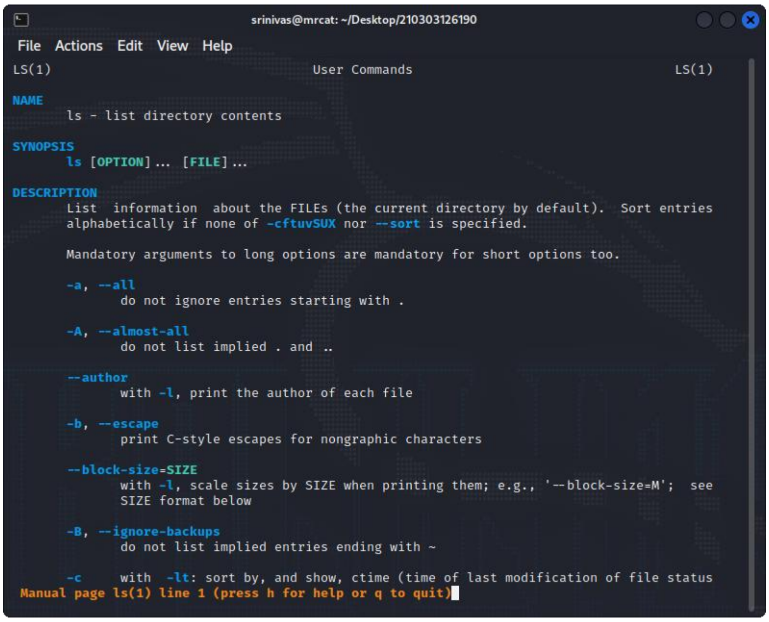


```
┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ gedit Srinu.sh

(gedit:15902): Gtk-WARNING **: 23:16:02.800: Calling org.xfce.Session.Manager.Inhibit failed:
GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: No such method “Inhibit”
▯
```

```
Mr_Srinivas
210303126190
```

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

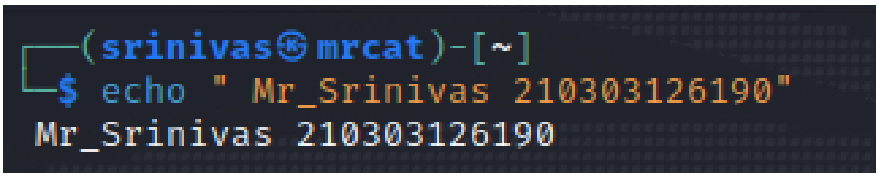


srinivas@mrcat: ~/Desktop/210303126190
File Actions Edit View Help

```
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
 Manual page ls(1) line 1 (press h for help or q to quit)
```

14.ECHO

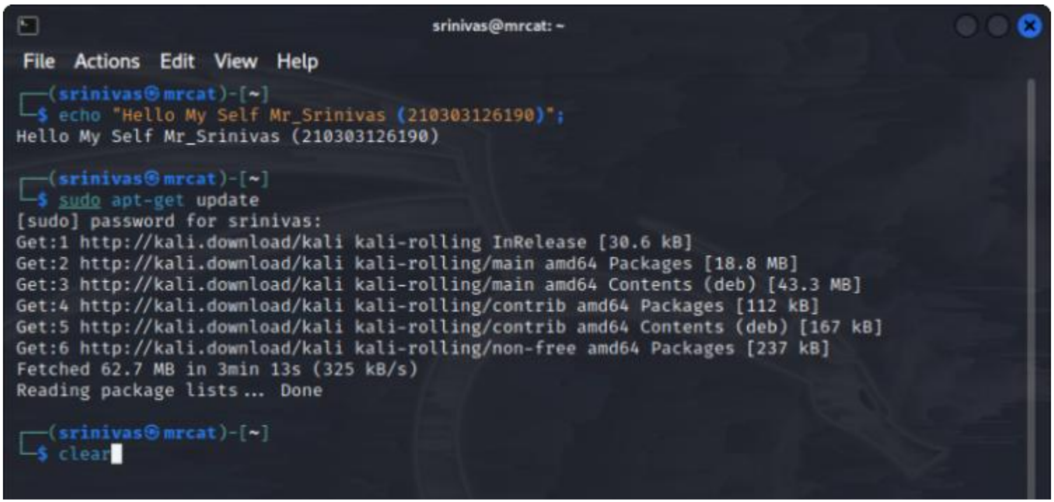
- Description: Display text on the screen
- Syntax: Display text on the screen



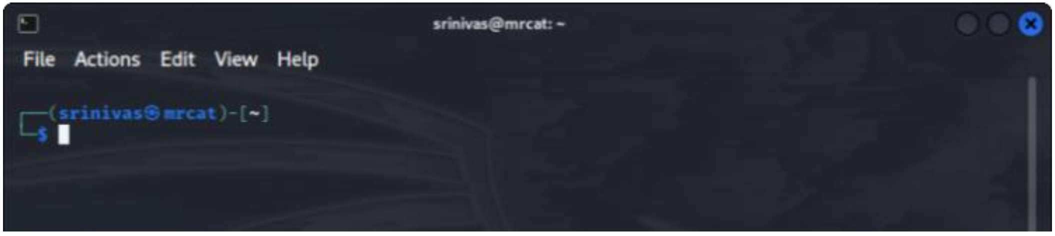
```
┌──(srinivas㉿mrcat)-[~]
└─$ echo " Mr_Srinivas 210303126190"
Mr_Srinivas 210303126190
```

15.CLEAR

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



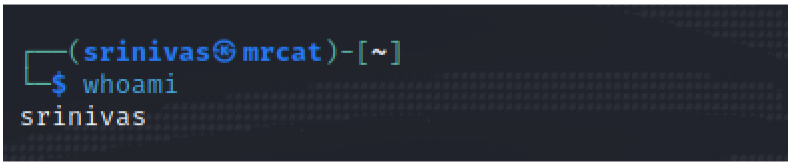
```
┌──(srinivas㉿mrcat)-[~]
└─$ echo "Hello My Self Mr_Srinivas (210303126190)";
Hello My Self Mr_Srinivas (210303126190)

┌──(srinivas㉿mrcat)-[~]
└─$ sudo apt-get update
[sudo] password for srinivas:
Get:1 http://kali.download/kali kali-rolling InRelease [30.6 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [18.8 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [43.3 MB]
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [112 kB]
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [167 kB]
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [237 kB]
Fetched 62.7 MB in 3min 13s (325 kB/s)
Reading package lists ... Done

┌──(srinivas㉿mrcat)-[~]
└─$ clear
```



```
┌──(srinivas㉿mrcat)-[~]
└─$ 
```

16.WHOAMI

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



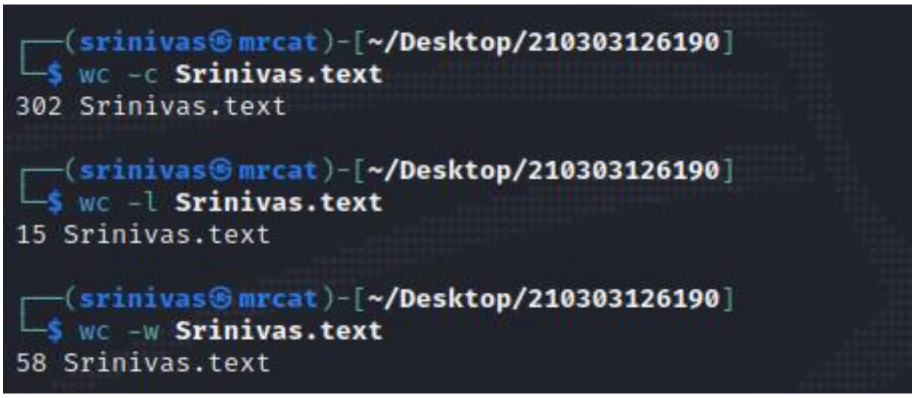
```
┌──(srinivas㉿mrcat)-[~]
└─$ whoami
srinivas
```

17.WC

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

Example:

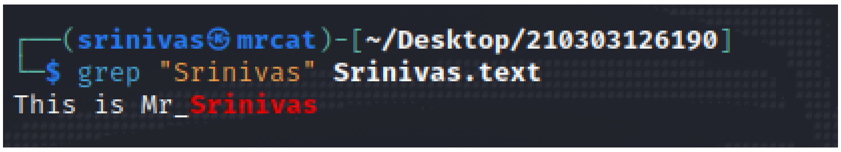
- ✓ Print the byte counts of file myfile.txt
  wc -c myfile.txt
- ✓ Print the line counts of file myfile.tx
  wc -l myfile.txt
- ✓ Print the word counts of file myfile.txt
  wc -w myfile.txt



```
┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ wc -c Srinivas.text
302 Srinivas.text

┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ wc -l Srinivas.text
15 Srinivas.text

┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ wc -w Srinivas.text
58 Srinivas.text
```

18.GREP

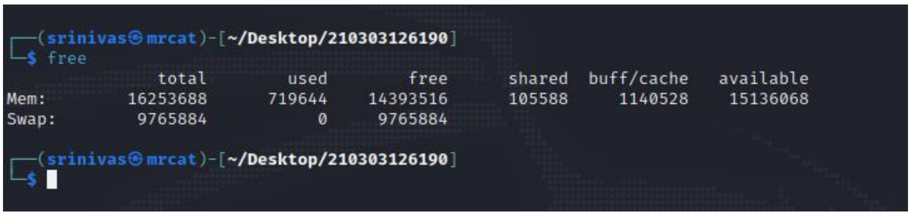
- Description: grep command uses a search term to look through a file
- Syntax: grep [option]… Pattern [file]



```
┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ grep "Srinivas" Srinivas.text
This is Mr_Srinivas
```

19.FREE

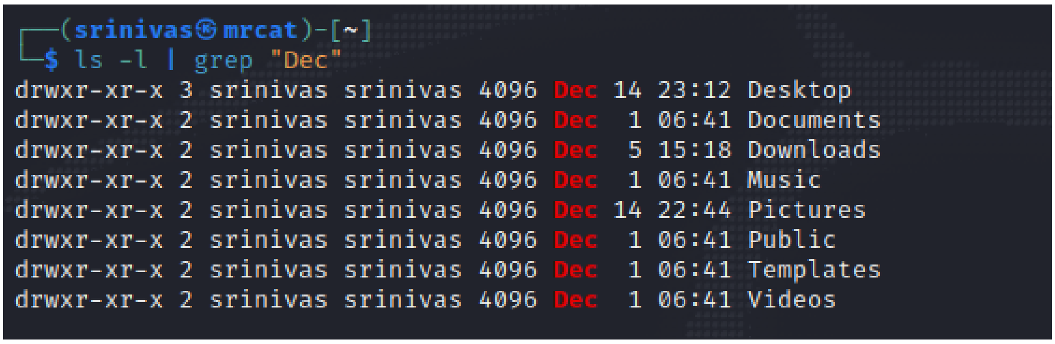
- Description: To display the RAM details in Linux machine need to write following command.
- Syntax: free



```
┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ free
               total        used        free      shared  buff/cache   available
Mem:        16253688      719644    14393516      105588     1140528    15136068
Swap:        9765884           0     9765884

┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ █
```

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



```
┌──(srinivas㉿mrcat)-[~]
└─$ ls -l | grep "Dec"
drwxr-xr-x 3 srinivas srinivas 4096 Dec 14 23:12 Desktop
drwxr-xr-x 2 srinivas srinivas 4096 Dec  1 06:41 Documents
drwxr-xr-x 2 srinivas srinivas 4096 Dec  5 15:18 Downloads
drwxr-xr-x 2 srinivas srinivas 4096 Dec  1 06:41 Music
drwxr-xr-x 2 srinivas srinivas 4096 Dec 14 22:44 Pictures
drwxr-xr-x 2 srinivas srinivas 4096 Dec  1 06:41 Public
drwxr-xr-x 2 srinivas srinivas 4096 Dec  1 06:41 Templates
drwxr-xr-x 2 srinivas srinivas 4096 Dec  1 06:41 Videos
```

PRACTICAL – 2

Aim: Study the basics of Shell programming.

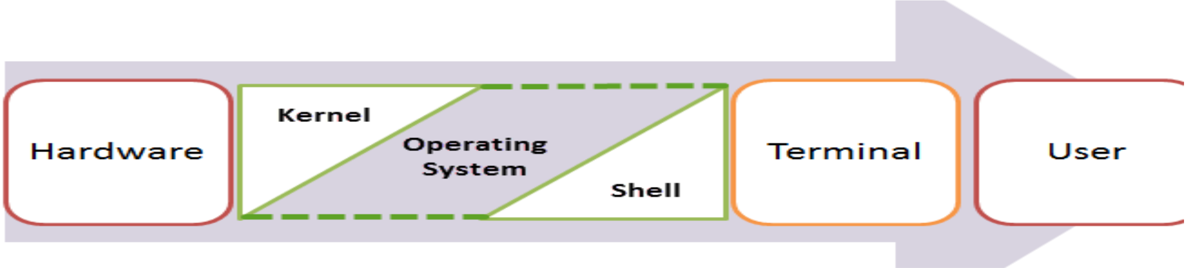
What is a Shell?

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:

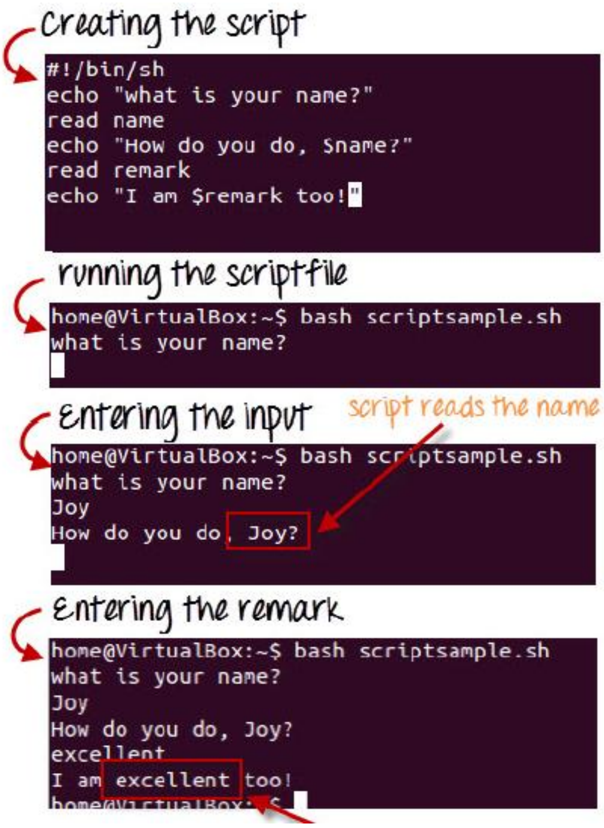
1. **Bournee shell:** This is default shell for version 7 unix. The character $ is the default prompt for the bourne shell.
2. **C shell:** This is a unix shell and a command processor that is run in a text window . The character % is the default prompt for the C shell. File commands can also be read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



```
#!/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:~$
```

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

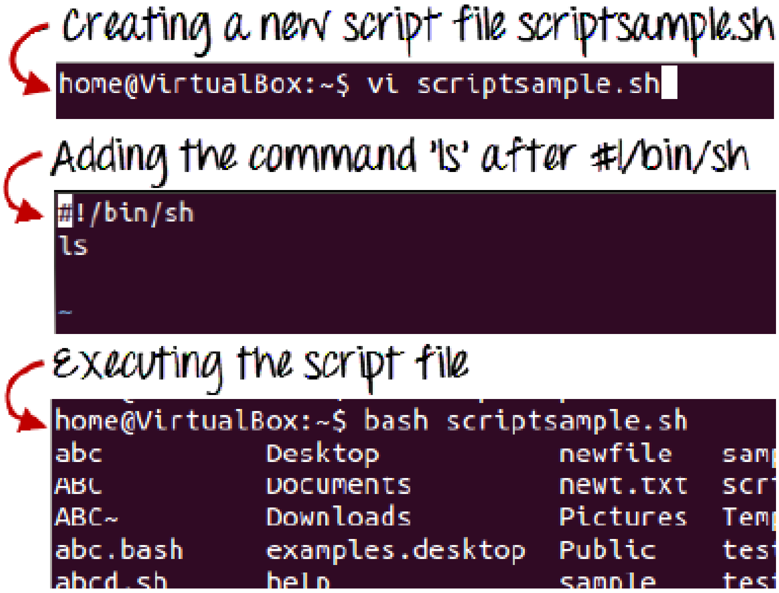
3. Write some code.

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    samp
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 –

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.

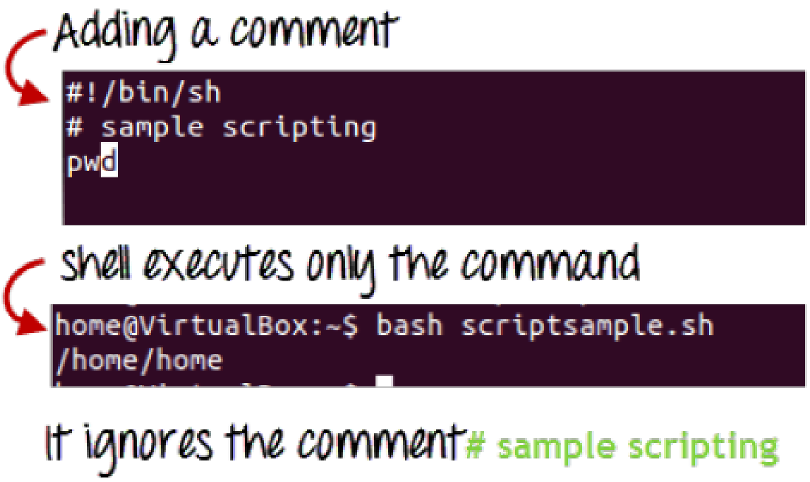
For example, the following creates a shell variable and then prints it:

```
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



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

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

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

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:

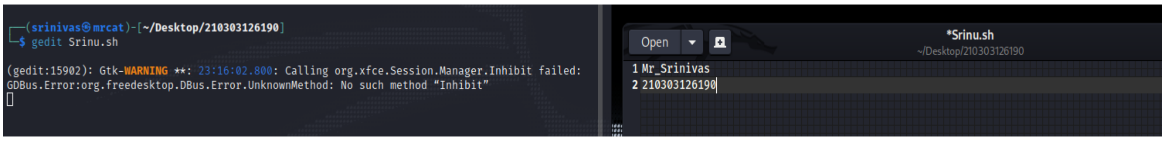
- Kernel is the nucleus of the operating systems, and it communicates between hardware and software
- Shell is a program which interprets user commands through CLI like Terminal
- The Bourne shell and the C shell are the most used shells in Linux
- Shell scripting is writing a series of command for the 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)-[~/Desktop/210303126190]
└─$ gedit Srinu.sh

(gedit:15902): Gtk-WARNING **: 23:16:02.800: Calling org.xfce.Session.Manager.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: No such method “Inhibit”
```

*Srinu.sh
~/Desktop/210303126190

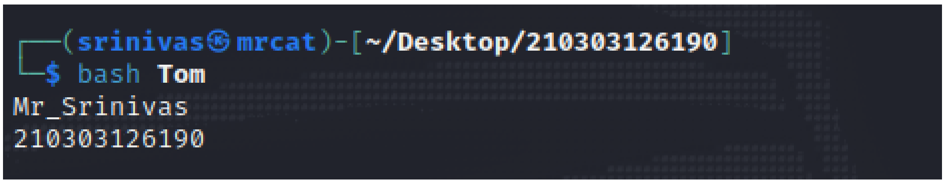
```
1 Mr_Srinivas
2 210303126190
```

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.

2. Bash:

Syntax : bash prac11.sh

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



```
┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ bash Tom
Mr_Srinivas
210303126190
```

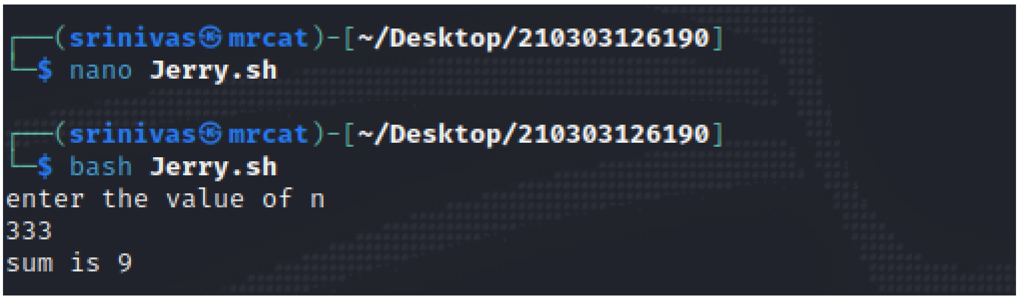
PRACTICAL –3

Aim: Write a Shell script to print given numbers sum of all digits.

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";
```

Output:



```
┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ nano Jerry.sh

┌──(srinivas㉿mrcat)-[~/Desktop/210303126190]
└─$ bash Jerry.sh
enter the value of n
333
sum is 9
```

PRACTICAL – 4

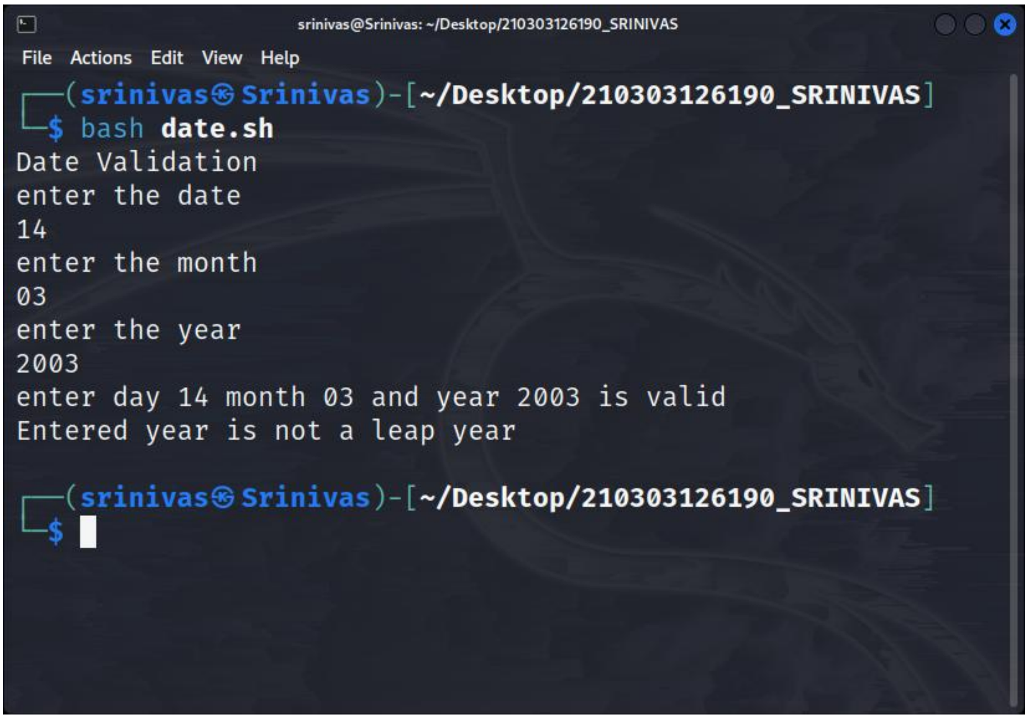
Aim: Write a shell script to validate the entered date.

(eg. Date format is: dd-mm-yyyy).

Sample Code:

```
echo "date"
read d
echo "month"
read m
echo "year"
read y
n=`expr $y % 4`
echo $d"/"$m"/"$y
if  [ $m -eq 4 ] || [ $m -eq 6 ] || [ $m -eq 9 ] || [ $m -eq 11 ]
then
if [ $d -gt 0 ] && [ $d -lt 31 ]
then
echo "valid"
else
echo "not valid"
fi
elif  [ $m -eq 2 ]
then
if [ $n -eq 0 ] && [ $d -gt 0 ] && [ $d -lt 30 ]
then
echo "valid"
elif [ $n -gt 0 ] && [ $d -gt 0 ] && [ $d -lt 29 ]
then
echo "not valid"
else
echo "not valid"
fi
else
if [ $d -gt 0 ] && [ $d -lt 32 ]
then
echo "valid"
else
echo "not valid"
fi
fi
```

Output:



srinivas@Srinivas: ~/Desktop/210303126190_SRINIVAS
File Actions Edit View Help

```
┌──(srinivas㉿Srinivas)-[~/Desktop/210303126190_SRINIVAS]
└─$ bash date.sh
Date Validation
enter the date
14
enter the month
03
enter the year
2003
enter day 14 month 03 and year 2003 is valid
Entered year is not a leap year

┌──(srinivas㉿Srinivas)-[~/Desktop/210303126190_SRINIVAS]
└─$ 
```

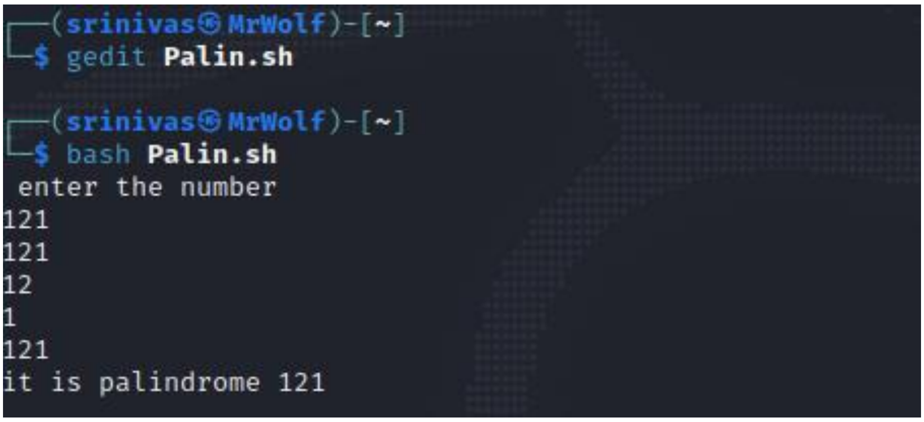
PRACTICAL – 5

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

Sample Code:

```
echo " enter the number";
read a;
c=$a;
sum=0;
while [ $a -ne 0 ]
do
        b=`expr $a % 10`;
        echo "$a";
        sum=`expr $sum \* 10`;
        sum=`expr $sum + $b`;
        a=`expr $a \/ 10`;
done
echo "$sum";
if [ $c -eq $sum ]
then {
echo "it is palindrome $sum";
} fi;
```

Output:



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

┌──(srinivas㉿MrWolf)-[~]
└─$ bash Palin.sh
 enter the number
121
121
12
1
121
it is palindrome 121
```

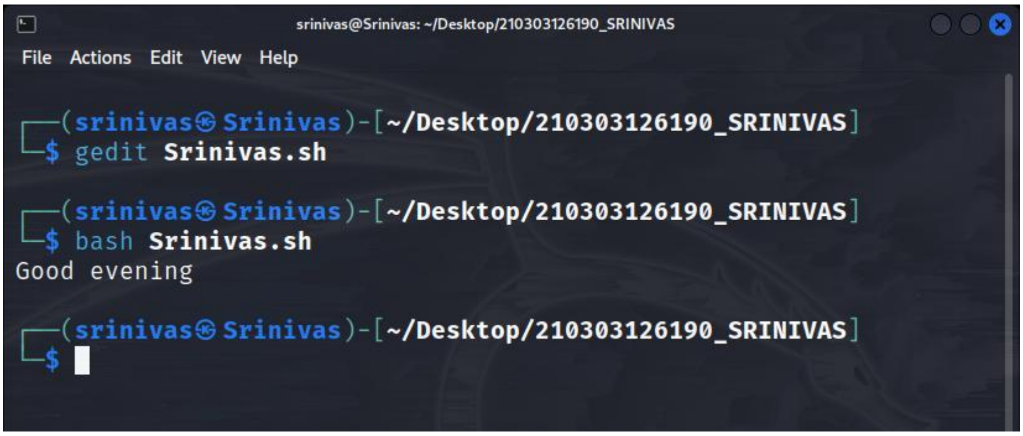
PRACTICAL – 6

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
```

Output:



```
┌──(srinivas㉿Srinivas)-[~/Desktop/210303126190_SRINIVAS]
└─$ gedit Srinivas.sh

┌──(srinivas㉿Srinivas)-[~/Desktop/210303126190_SRINIVAS]
└─$ bash Srinivas.sh
Good evening

┌──(srinivas㉿Srinivas)-[~/Desktop/210303126190_SRINIVAS]
└─$ 
```