Parul® University
VADODARA, GUJARAT

NAAC A++
ACCREDITED UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

BACHELOR OF TECHNOLOGY

Operating System
(303105251)

SEMESTER IV
Computer Science & Engineering Department

योगः कर्मसु कौशलम्
PARUL UNIVERSITY

Laboratory Manual

Session:2024-25

OPERATING SYSTEM PRACTICAL BOOK
COMPUTER SCIENCE & ENGINEERING DEPARTMENT
PREFACE

It gives us immense pleasure to present the first edition of the **OPERATING SYSTEM** Practical Book for the B.Tech . **4th semester** students for **PARUL UNIVERSITY.**
The **OS** theory and laboratory courses at **PARUL UNIVERSITY, WAGHODIA, VADODARA** are designed in such a way that students develop the basic understanding of the subject in the theory classes and then try their hands on the experiments to realize the various implementations of problems learnt during the theoretical sessions. The main objective of the **OS** laboratory course is: Learning **OS** through Experimentations. All the experiments are designed to illustrate various problems in different areas of **OS** and also to expose the students to various uses.

The objective of this **OS** Practical Book is to provide a comprehensive source for all the experiments included in the **OS** laboratory course. It explains all the aspects related to every experiment such as: basic underlying concept and how to analyze a problem. It also gives sufficient information on how to interpret and discuss the obtained results.

We acknowledge the authors and publishers of all the books which we have consulted while developing this Practical book. Hopefully this **OS** Practical Book will serve the purpose for which it has been developed.

INSTRUCTIONS TO STUDENTS

1. The main objective of the **OS** laboratory is: Learning through the Experimentation. All the experiments are designed to illustrate various problems in different areas of **OS** and also to expose the students to various problems and their uses.
2. Be prompt in arriving at the laboratory and always come well prepared for the practical.
3. Every student should have his/her individual copy of the **OS** Practical Book.
4. Every student have to prepare the notebooks specifically reserved for the **OS** practical work: ”**OS** Practical Book”
5. Every student has to necessarily bring his/her **OS** Practical Book, **OS** Practical Class Notebook and **OS** Practical Final Notebook.
6. Finally find the output of the experiments along with the problem and note results in the **OS** Practical Notebook.
7. The grades for the **OS** practical coursework will be awarded based on our performance in the laboratory, regularity, recording of experiments in the **OS** Practical Final Notebook, lab quiz, regular viva-voce and end-term examination

CERTIFICATE

This is to certify that

Mr. Bharat sinh Rathod, Enrolment No

***2303051051232**, has successfully Completed his/her*

laboratory experiments in the **OPERATING SYSTEM**

(303105251)

*from the department of **CSE** during The academic year **2024-25***

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

Head Of Department:.....

TABLE OF CONTENTS

PRACTICAL 1

AIM:-Study of Basic commands of Linux.

Command shell: A program that interprets commands is Command shell.

Shell Script: Allows a user to execute commands by typing them manually at a terminal, or automatically in programs called shell scripts.A shell is not an operating system. It is a way to interface with the operating system and run Commands.

BASH (Bourne Again Shell)

- Bash is a shell written as a free replacement to the standard Bourne Shell (/bin/sh) originally written by Steve Bourne for UNIX systems.
- It has all of the features of the original Bourne Shell, plus additions that make it easier to program with and use from the command line.
- Since it is Free Software, it has been adopted as the default shell on most Linux systems.

BASIC LINUX COMMANDS:

1) Pwd : print working

directory DESCRIPTION:

pwd prints the full pathname of the current working directory.

SYNTAX:

Pwd

```
saptarishi-pc@saptarishi-Pc:~$ pwd
/home/saptarishi-pc
```

2) cd: Change

Directory

DESCRIPTION:

It allows you to change your working directory. You use it to move around within the hierarchy of your file system.

SYNTAX:

cd directory_name

```
saptarishi-pc@saptarishi-Pc:~$ cd Desktop
saptarishi-pc@saptarishi-Pc:~/Desktop$
```

3) cd ..

DESCRIPTION:

Move up one directory.

SYNTAX:

cd ..

```
saptarishi-pc@saptarishi-Pc:~$ cd Desktop
saptarishi-pc@saptarishi-Pc:~/Desktop$ cd ..
saptarishi-pc@saptarishi-Pc:~$
```

4) ls : list all the files and

directories DESCRIPTION:

List all files and folders in the current directory in the column format.

SYNTAX:

ls [options]

```
cprocess.c  Documents  flask  Music    Public  Templates
Desktop     Downloads  g3.c   Pictures snap    Videos
saptarishi-pc@saptarishi-Pc:~$
```

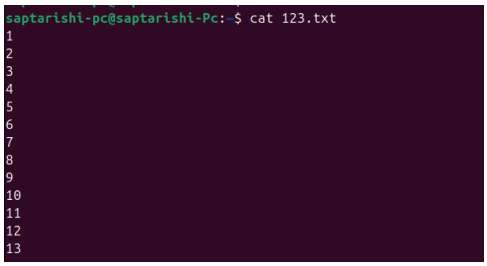
5) cat

DESCRIPTION:

cat stands for "catenate". It reads data from files, and outputs their contents. It is the simplest way to display the contents of a file at the command line.

SYNTAX:

cat filename



6) head

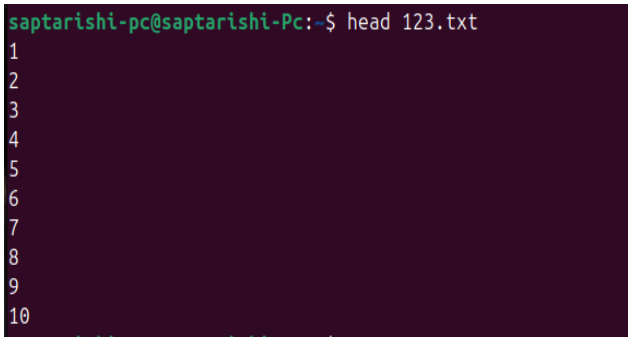
DESCRIPTION:

head, by default, prints the first 10 lines of each FILE to standard output. With more than one FILE, it precedes each set of output with a header identifying the file name.

If no FILE is specified, or when FILE is specified as a dash ("-"), head reads from standard input.

SYNTAX:

head [option]...[file/directory]



7)Tail

DESCRIPTION:

tail is a command which prints the last few number of lines (10 lines by default) of a certain file, then terminates.

SYNTAX:

tail [option]...[file/directory]

```
saptarishi-pc@saptarishi-Pc:~$ tail 123.txt
5
6
7
8
9
10
11
12
13
```

8) mv : Moving (and Renaming) Files

DESCRIPTION:

The mv command lets you move a file from one directory location to another. It also lets you rename a file (there is no separate rename command).

SYNTAX:

mv [option] source directory

```
saptarishi-pc@saptarishi-Pc:~$ ls
1234.txt  cprocess.c  Documents  flask  Music     Public  Templates
123.txt   Desktop     Downloads  g3.c   Pictures  snap    Videos
saptarishi-pc@saptarishi-Pc:~$ mv 123.txt 1234.txt
saptarishi-pc@saptarishi-Pc:~$ ls
1234.txt    Desktop    Downloads  g3.c   Pictures  snap       Videos
cprocess.c  Documents  flask      Music  Public    Templates
saptarishi-pc@saptarishi-Pc:~$
```

9) mkdir : Make

Directory

DESCRIPTION:

If the specified directory does not already exist, mkdir creates it. More than one directory may be specified when calling mkdir.

SYNTAX:

mkdir [option] directory

```
saptarishi-pc@saptarishi-Pc:~$ ls
1234.txt    Desktop    Downloads  g3.c   Pictures  snap       Videos
cprocess.c  Documents  flask      Music  Public    Templates
saptarishi-pc@saptarishi-Pc:~$ mkdir hello
saptarishi-pc@saptarishi-Pc:~$ ls
1234.txt    Desktop    Downloads  g3.c   Music     Public  Templates
cprocess.c  Documents  flask      hello  Pictures  snap    Videos
saptarishi-pc@saptarishi-Pc:~$
```

10) cp : Copy Files

DESCRIPTION:

The cp command is used to make copy of files and directories.

SYNTAX:

cp [option] source directory

```
saptarishi-pc@saptarishi-Pc:~$ cp 1234.txt 123.txt
saptarishi-pc@saptarishi-Pc:~$ ls
1234.txt  cprocess.c  Documents  flask  hello  Pictures  snap      Videos
123.txt   Desktop     Downloads  g3.c   Music  Public    Templates
saptarishi-pc@saptarishi-Pc:~$
```

11) rmdir : Remove

Directory DESCRIPTION:

The rmdir command is used to remove a directory that contains other files or directories.

SYNTAX:

rm directory_name

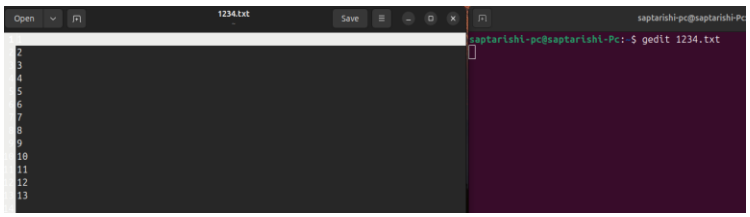
```
saptarishi-pc@saptarishi-Pc:~$ ls
1234.txt  cprocess.c  Documents  flask  hello  Pictures  snap      Videos
123.txt   Desktop     Downloads  g3.c   Music  Public    Templates
saptarishi-pc@saptarishi-Pc:~$ rmdir hello
saptarishi-pc@saptarishi-Pc:~$ ls
1234.txt  cprocess.c  Documents  flask  Music     Public  Templates
123.txt   Desktop     Downloads  g3.c   Pictures  snap    Videos
saptarishi-pc@saptarishi-Pc:~$
```

12) gedit

DESCRIPTION:

The gedit command is used to create and open a file.

SYNTAX:



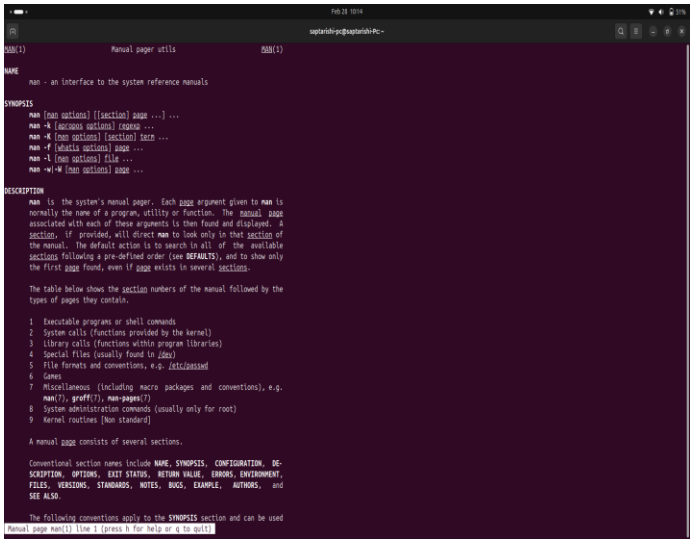
13) man

DESCRIPTION:

Displays on an online manual page or manpage.

SYNTAX:

man command



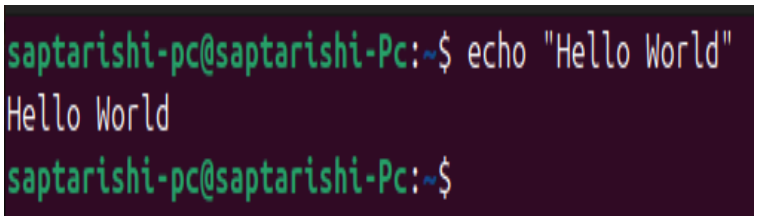
14) echo

DESCRIPTION:

Display text on the screen.

SYNTAX:

echo yourtext



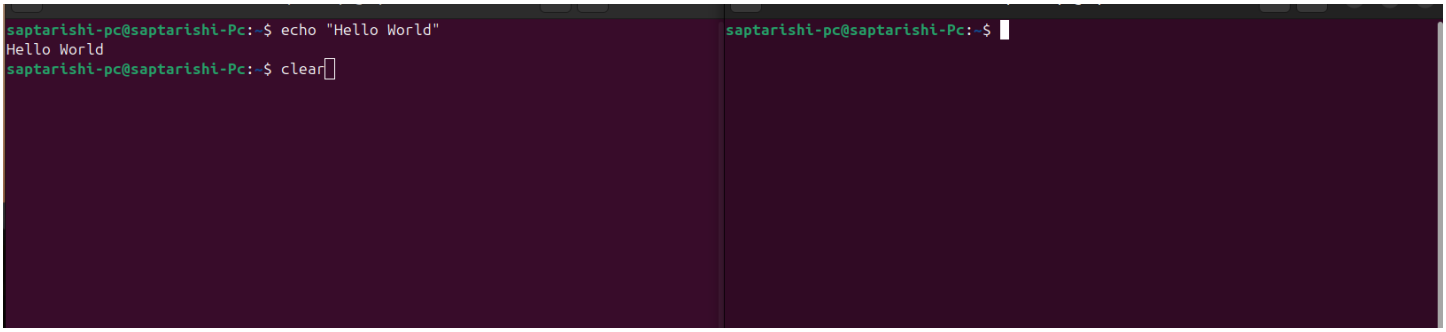
15) clear

DESCRIPTION:

Used to clear the screen

SYNTAX:

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]

```
saptarishi-pc@saptarishi-Pc:~$ whoami
saptarishi-pc
saptarishi-pc@saptarishi-Pc:~$
```

17) wc

DESCRIPTION:

wc (word count) command, can return the number of lines, words, and characters in a file.

SYNTAX:

wc [option]... [file]...

```
saptarishi-pc@saptarishi-Pc:~$ wc 1234.txt
14 13 31 1234.txt
saptarishi-pc@saptarishi-Pc:~$
```

18) **grep**

DESCRIPTION:

grep command uses a search term to look through a file.

SYNTAX:

grep [option]... Pattern [file]...

```
saptarishi-pc@saptarishi-Pc:~$ grep "2" 1234.txt
2
12
saptarishi-pc@saptarishi-Pc:~$
```

19) free

DESCRIPTION:

Display RAM details in Linux machine.

SYNTAX: Free

```
saptarishi-pc@saptarishi-Pc:~$ free
               total        used        free      shared  buff/cache   available
Mem:         7904028     2228180     2213176      421500     4189604     5675848
Swap:        4194300           0     4194300
saptarishi-pc@saptarishi-Pc:~$
```

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

```
saptarishi-pc@saptarishi-Pc:~$ sort 1234.txt | uniq

1
10
11
12
13
2
3
4
5
6
7
8
9
```

PRACTICAL 2

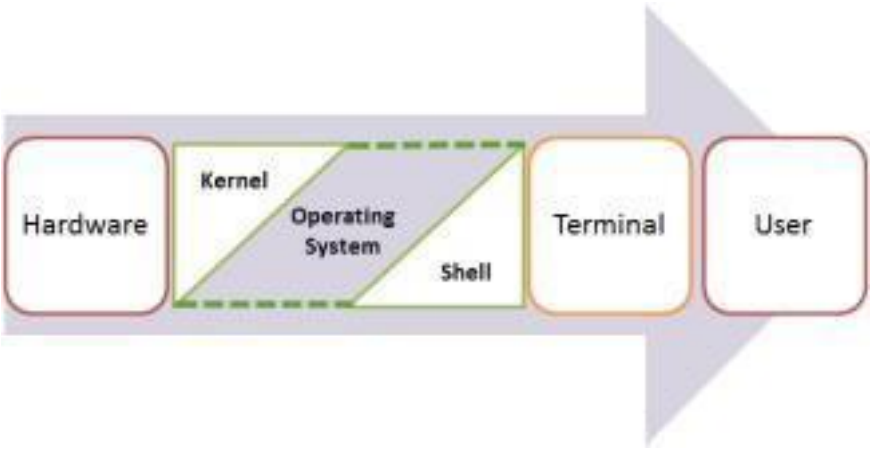
AIM:-Study the basics of shell programming.

What is a Shell?

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 Shell

There are two main shells in Linux:

1. The Bourne Shell: The prompt for this shell is $ and its derivatives are listed below:

- POSIX shell also is known as sh
- Korn Shell also knew as sh
- **B**ourne **A**gain **SH**ell also knew as bash (most popular)

2. The C shell: The prompt for this shell is %, and its subcategories are:

- C shell also is known as csh
- Tops C shell also is known as tcsh

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.

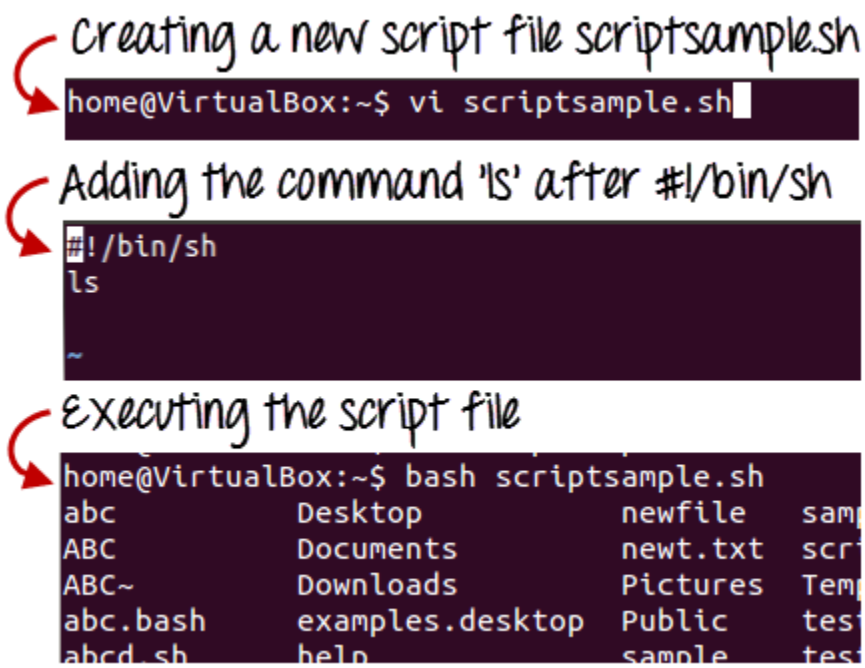
4. Save the script file as filename.sh
5. For **executing** the script type **bash filename.sh**

"#!" 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 -

```
#!/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

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

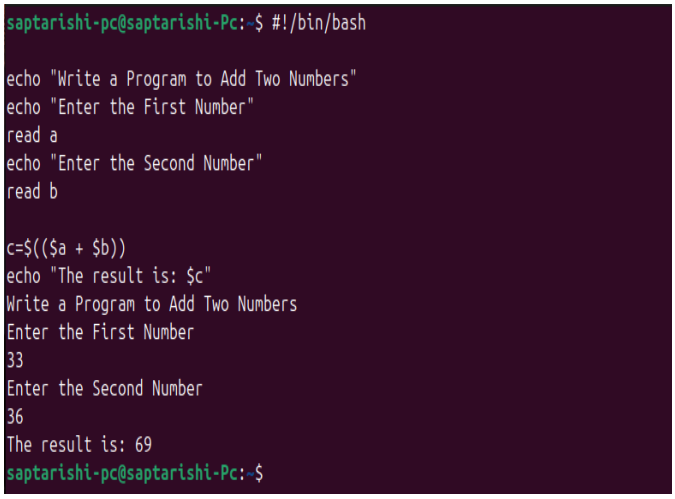
• Shell scripting can help you create complex programs containing conditional statements, loops, and functions

 # Aim: - Write a shell script to make addition of two Numbers

 # Input: -

```
#!/bin/bash
 echo " Write a Program to Addition Of two Number
" echo " Enter the First Number"
 read a
 echo " Enter the second Number "
 read b
 c=$(($a+$b))
echo $c
```

#Output:-



 # Aim: - Write a shell script to make subtraction of two Numbers

 # Input: -

```
#!/bin/bash
echo " Write a Program to subtraction of two Number "
echo " Enter the First Number"
read a
echo " Enter the second Number "
read b
c=$(($a-$b))
echo $c
```

#Output:-

```
saptarishi-pc@saptarishi-Pc:~$ #!/bin/bash

echo "Write a Program to Subtraction of two Numbers"
echo "Enter the First Number"
read a
echo "Enter the Second Number"
read b

c=$(($a - $b))
echo "The result is: $c"
Write a Program to Subtraction of two Numbers
Enter the First Number
102
Enter the Second Number
33
The result is: 69
saptarishi-pc@saptarishi-Pc:~$
```

 # Aim: - Write a shell script to make multiplication of two Numbers

 # Input: -

```
#!/bin/bash
echo " Write a Program to multiplication of two Number "
echo " Enter the First Number"
read a
echo " Enter the second Number "
read b
c=$(($a*$b))
echo $c
```

#Output:-

```
saptarishi-pc@saptarishi-Pc:~$ #!/bin/bash

echo "Write a Program for multiplication of two Numbers"
echo "Enter the First Number"
read a
echo "Enter the Second Number"
read b

c=$(($a * $b))
echo "The result is: $c"
Write a Program for multiplication of two Numbers
Enter the First Number
23
Enter the Second Number
3
The result is: 69
saptarishi-pc@saptarishi-Pc:~$
```

 # Aim: - Write a shell script to make division of two Numbers

 # Input: -

```
#!/bin/bash
echo " Write a Program to do division of two Number "
echo " Enter the First Number"
read a
echo " Enter the second Number "
```

```
read b
c=$(($a/$b))
echo $c
```

#Output:-

```
saptarishi-pc@saptarishi-Pc:~$ echo "Write a Program for division of two Numbers
"
echo "Enter the First Number"
read a
echo "Enter the Second Number"
read b

c=$(($a / $b))
echo "The result is: $c"
Write a Program for division of two Numbers
Enter the First Number
12
Enter the Second Number
4
The result is: 3
saptarishi-pc@saptarishi-Pc:~$
```

 # Aim: - Write a shell script to make modulus of two Numbers

 # Input: -

```
#!/bin/bash
echo " Write a Program to find modulus of two numbers "
echo " Enter the First Number"
read a
echo " Enter the second Number "
read b
c=$(($a%$b))
echo $c
```

#Output:-

```
saptarishi-pc@saptarishi-Pc:~$ #!/bin/bash

echo "Write a Program to find modulus of two Numbers"
echo "Enter the First Number"
read a
echo "Enter the Second Number"
read b

c=$(($a % $b))
echo "The result is: $c"
Write a Program to find modulus of two Numbers
Enter the First Number
20
Enter the Second Number
3
The result is: 2
saptarishi-pc@saptarishi-Pc:~$
```

 # Aim: - Write a shell script to type name

 # Input: -

```
#!/bin/bash
echo " Whats your Name"
read name
echo "Hello $name"
```

#Output:-

```
saptarishi-pc@saptarishi-Pc:~$ #!/bin/bash
echo "Whats your Name: "
read name
echo "Hello $name"
Whats your Name:
saptarishi
Hello saptarishi
saptarishi-pc@saptarishi-Pc:~$
```

 # Aim: - Write a shell script to find swap of 2 numbers

 # Input: -

```
#!/bin/bash
echo "Enter the First Number"
read first
echo "Enter the Second Number"
read second
temp=$first
first=$second
second=$temp
echo "After swapping, numbers are:"
echo "first = $first, second = $second"
```

#Output:-

```
saptarishi-pc@saptarishi-Pc:~$ #!/bin/bash
echo "Enter the First Number"
read first
echo "Enter the Second Number"
read second

# Swapping the numbers
temp=$first
first=$second
second=$temp

# Displaying the swapped values
echo "After swapping, numbers are:"
echo "first = $first, second = $second"
Enter the First Number
12
Enter the Second Number
21
After swapping, numbers are:
first = 21, second = 12
saptarishi-pc@saptarishi-Pc:~$
```

 # Aim: - Write a shell script to define variable

 # Input: -

PRACTICAL 3

AIM:-Write a Shell script to print the given numbers sum of all digits.

 # Input: -

```
#!/bin/bash
echo "Enter a number"
 read num
sum=0
while [ $num -gt 0 ]
do
mod=$((num % 10))
sum=$((sum + mod))
num=$((num / 10))
done
echo $sum
```

#Output:-

```
saptarishi-pc@saptarishi-Pc:~$ #!/bin/bash

echo "Enter a number"
read num

sum=0

while [ $num -gt 0 ]
do
    mod=$((num % 10))
    sum=$((sum + mod))
    num=$((num / 10))
done

echo $sum
Enter a number
59
14
saptarishi-pc@saptarishi-Pc:~$
```

PRACTICAL 4

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

 # Input: -

```
#!/bin/bash
d=`date +%m-%d-%Y`
echo $d #DD-MM-YYYY
echo " Please Enter Date "
read D
echo " Please Enter Month "
read M
echo " Please Enter Year
" read Y
if [ `expr $Y % 4` -eq 0 ]
 then
 echo "$Y is a leap year"
else
echo "$Y is not a leap year"
fi
```

#Output:-

```
saptarishi-pc@saptarishi-Pc:~$ #!/bin/bash
d=`date +%m-%d-%Y`
echo $d

echo "Please Enter Date"
read D
echo "Please Enter Month"
read M
echo "Please Enter Year"
read Y

if [ `expr $Y % 4` -eq 0 ]
then
    echo "$Y is a leap year"
else
    echo "$Y is not a leap year"
fi
02-28-2025
Please Enter Date
29
Please Enter Month
02
Please Enter Year
2024
2024 is a leap year
saptarishi-pc@saptarishi-Pc:~$
```

PRACTICAL 5

AIM:-.Write a shell script to check entered string is palindrome or not

 # Input: -

```
#!/bin/bash
# Store the string entered by the user
 echo -n "Enter a string: "
read str
# Reverse the string
revstr=$(echo $str | rev)
# Check if the string is a palindrome
if [ "$str" == "$revstr" ]
then
echo "The string is a palindrome"
 else
echo "The string is not a palindrome"
fi
```

#Output:-

```
saptarishi-pc@saptarishi-Pc:~$ #!/bin/bash
# Store the string entered by the user
echo -n "Enter a string: "
read str

# Reverse the string
revstr=$(echo $str | rev)

# Check if the string is a palindrome
if [ "$str" == "$revstr" ]
then
    echo "The string is a palindrome"
else
    echo "The string is not a palindrome"
fi
Enter a string: malayalam
The string is a palindrome
saptarishi-pc@saptarishi-Pc:~$
```

PRACTICAL 6

AIM:-.Write a Shell script to say Good morning/Afternoon/Evening as you log in to the system.

 # Input: -

```
#!/bin/bash
# Get the current hour
hour=$(date +%H)
name=" Saptarishi "
# Set the greeting message
if [ $hour -ge 6 ] && [ $hour -lt 12 ]; then
greeting="Good morning $name"
elif [ $hour -ge 12 ] && [ $hour -lt 16 ]; then
greeting="Good afternoon $name"
elif [ $hour -ge 16 ] && [ $hour -lt 20 ]; then
greeting="Good evening $name"
 else
greeting="Good Night $name"
 fi
# Print the greeting message
echo $greeting
```

#Output:-

```
saptarishi-pc@saptarishi-Pc:~$ #!/bin/bash
# Get the current hour
hour=$(date +%H)
name=" Saptarishi "

# Set the greeting message
if [ $hour -ge 6 ] && [ $hour -lt 12 ]; then
    greeting="Good morning $name"
elif [ $hour -ge 12 ] && [ $hour -lt 16 ]; then
    greeting="Good afternoon $name"
elif [ $hour -ge 16 ] && [ $hour -lt 20 ]; then
    greeting="Good evening $name"
else
    greeting="Good Night $name"
fi

# Print the greeting message
echo $greeting
Good morning Saptarishi
saptarishi-pc@saptarishi-Pc:~$
```

PRACTICAL 7

AIM:- Write a C program to create a child process.

 # Input: -

```
#include <sys/types.h>
#include <unistd.h>

void forkexample()
{
// Check if the return value of fork() is 0
if (fork() == 0)
{
printf("Hello from Child!\n");
}
// If the return value of fork() is non-zero
else
{
printf("Hello from Parent!\n");
}
}

int main()
{
forkexample();
return 0;
}
#include <stdio.h>
```

#Output:-

```
Hello from Parent!
Hello from Child!
```

PRACTICAL 8

AIM:- Finding out biggest number from given three numbers supplied as command line arguments.

 # Input: -

```
echo "Enter Num1"
 read num1
echo "Enter Num2"
read num2
echo "Enter Num3"
read num3
if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]
then
echo $num1
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]
 then
echo $num2
else
echo $num3
fi
```

#Output:-

```
saptarishi-pc@saptarishi-Pc:~$ #!/bin/bash
echo "Enter Num1"
read num1
echo "Enter Num2"
read num2
echo "Enter Num3"
read num3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]; then
    echo $num1
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]; then
    echo $num2
else
    echo $num3
fi
Enter Num1
96
Enter Num2
32
Enter Num3
14
96
```

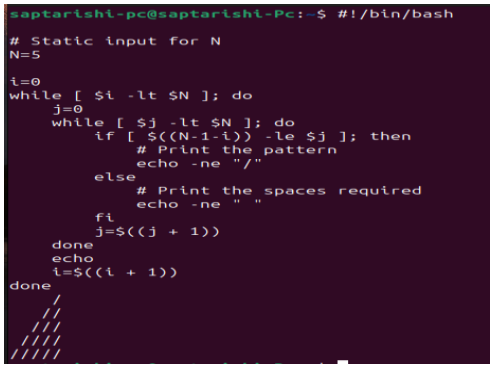
PRACTICAL 9

AIM:- Printing the patterns using for loop

 # Input: -

```
# Static input for N N=5
N=5
i=0
while [ $i -lt $N ]
do
  j=0
  while [ $j -lt $N ]
  do
    if [ $((N-1-i)) -le $j ]
    then
      # Print the pattern
      echo -ne "/"
    else
      # Print the spaces required
      echo -ne " "
    fi
    j=$((j + 1))
  done
  echo
  i=$((i + 1))
done
```

#Output:-



PRACTICAL 10

AIM:- Shell script to determine whether a given file exists or not.

 # Input: -

```
#!/bin/bash
File=dp.txt
if [ -f "$File" ]; then
echo  "$File  exists"
else
  echo "$File does not exist"
fi
```

#Output:-

```
saptarishi-pc@saptarishi-Pc:~$ #!/bin/bash

File=1234.txt

if [ -f "$File" ]; then
    echo "$File exists"
else
    echo "$File does not exist"
fi
1234.txt exists
saptarishi-pc@saptarishi-Pc:~$ #!/bin/bash

File=12345.txt

if [ -f "$File" ]; then
    echo "$File exists"
else
    echo "$File does not exist"
fi
12345.txt does not exist
saptarishi-pc@saptarishi-Pc:~$ ls
1234.txt  cprocess.c  Documents  flask  Music     Public  Templates
123.txt   Desktop     Downloads  g3.c   Pictures  snap    Videos
```

PRACTICAL 11

AIM:- Write a program for process creation using C. (Use of gcc compiler).

 # Input: -

```
#include <unistd.h>
#include <sys/wait.h>

 int main(void) {
int pid = fork();
if (pid == 0) {
        // Child process
        printf("Child process (pid=%d)\n", getpid());
  } else if (pid > 0) {
        // Parent
  process int status;
  wait(&status);
        printf("Parent process (pid=%d)\n", getpid());
  } else {
        // Error
  perror("fork");
  }
  return 0;
}
```

```
#include <stdio.h>
```

#Output:-

```
rating System/"processcreation
Child process (pid=2429)
Parent process (pid=2428)
```

PRACTICAL 12

AIM:- Implementation of FCFS &Round Robin Algorithm

#

Input: -

FCFS

```
int main()
{
   int n, count;
   int at[10], bt[10];
   printf("Enter the number of processes:
 "); scanf("%d", &n);

   for (count = 0; count < n; count++)
   {
         printf("Enter arrival time and burst time for process %d: ", count + 1);
   scanf("%d%d", &at[count], &bt[count]);
   }

   return 0;
}
```

```
#include <stdio.h>
```

#Output:-

```
Enter the number of processes: 4
Enter arrival time and burst time for process 1: 2 02
Enter arrival time and burst time for process 2: 1 05
Enter arrival time and burst time for process 3: 2 04
Enter arrival time and burst time for process 4: 3 20
```

 # Input: -

RR

```
int main() {
int n,count, tq;
int at[10],bt[10];
printf("Enter Total Process:\t "); scanf("%d",&n); for(count=0;count<n;count++) {
```

```
#include <stdio.h>
```

```
printf("Enter Arrival Time and Burst Time for Process Number %d :",count+1);
scanf("%d",&at[count]);
scanf("%d",&bt[count]);
}
printf("Enter Time Quantum:\t "); scanf("%d",&tq);
return 0;
}
```

#Output:-

```
Enter Total Process:     3
Enter Arrival Time and Burst Time for Process Number 1 :1
 3
Enter Arrival Time and Burst Time for Process Number 2 :2
4
Enter Arrival Time and Burst Time for Process Number 3 :1
2
Enter Time Quantum:      23
```

PRACTICAL 13

AIM:- Implementation of Banker's Algorithm.

 # Input: -

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  int Max[10][10], need[10][10], alloc[10][10], avail[10], completed[10];
  int p, r, i, j, process, count;
  count = 0;

  printf("Enter the number of processes: ");
  scanf("%d", &p);

  for (i = 0; i < p; i++)
  {
    completed[i] = 0;
  }

  printf("\nEnter the number of resources: ");
  scanf("%d", &r);

  printf("\nEnter the Max Matrix for each process:\n");
  for (i = 0; i < p; i++)
  {
    printf("For process %d: ", i + 1);
    for (j = 0; j < r; j++)
    {
      scanf("%d", &Max[i][j]);
    }
```

```
}

printf("\nEnter the allocation for each process:\n");
for (i = 0; i < p; i++)
{
  printf("For process %d: ", i + 1);
  for (j = 0; j < r; j++)
  {
    scanf("%d", &alloc[i][j]);
  }
}

printf("\nEnter the available resources: ");
for (i = 0; i < r; i++)
{
  scanf("%d", &avail[i]);
}

printf("\nMax Matrix\tAllocation Matrix\n");
for (i = 0; i < p; i++)
{
  for (j = 0; j < r; j++)
  {
    printf("%d ", Max[i][j]);
  }
  printf("\t\t");
  for (j = 0; j < r; j++)
  {
    printf("%d ", alloc[i][j]);
  }
  printf("\n");
}

return 0;
}
```