FACULTY OF ENGINEERING AND TECHNOLOGY

BACHELOR OF TECHNOLOGY

OPERATING SYSTEM

(303105252)

4 SEMESTER

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

CERTIFICATE

This is to certify that **Susmita Kumari** ____ with enrolment no. **2203051051156**has successfully completed his/her laboratory experiments in the

OPERATING SYSTEM (303105252) from the Department of
COMPUTER SCIENCE AND ENGINEERING during the academic year 2023-24.

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

Head Of Department: .....

Sr. No	Experiment Title	Page No		Date of Start	Date of Completion	Sign	Marks (out of 10)
		From	To				
1.	Study of Basic Commands of Linux.						
2.	Study the Basics of shell programming.						
3.	Write a shell script to print given numbers sum of all digits.						
4.	Write a shell script to validate the entered data. (eg. Date format is: dd-mm-yyyy)						
5.	Write a shell script to check entered string in palindrome or not.						
6.	Write a shell script to say Good morning/Afternoon/Evening as you log in to system.						
7.	Write a C program to create a child process.						
8.	Finding out biggest number from given three numbers supplied as command line argument.						
9.	Printing the patterns using loop.						
10.	Shell script to determine whether given file exist or not.						
11.	Write a program for process creation using C. (Use of gcc compiler).						
12.	Implementation of FCFS & Round Robin Algorithm.						
13.	Implementation of Banker's Algorithm.						

Practical – 1

AIM: STUDY OF BASIC COMMAND OF LINUX

Important Linux commands

1. PWD
2. CD
3. CP
4. R M
5. L/s
6. Mkdir
7. Touch
8. Cat
9. Grep
10. Rmdir
11. Locate command
12. Mv command
13. Sudo
14. Df command
15. Head command
16. M/V
17. Tail command
18. Tár
19. Man
20. Diff command
21. Du command
22. Echo
23. Chmod
24. Clear

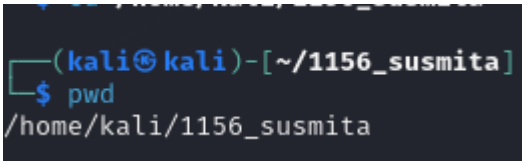
DESCRIPTION

1. **PWD**

COMMAND NAME: pwd

SYNTAX: pwd

EXAMPLE:



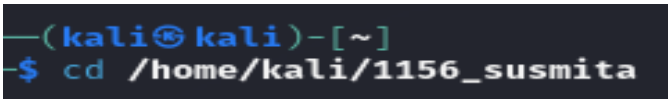
EXPLANATION: PWD stands for Print Working Directory. It writes the complete path name of the working directory to standard output in UNIX-like and other operating systems.

2. **CD**

COMMAND NAME: cd

SYNTAX: cd [directory]

EXAMPLE:



EXPLANATION:In Linux, the command line is like a super useful tool that lets people talk to the computer. One big part of using this tool is moving around the computer's folders and files, which we call the file system. The 'cd' command, which stands for "change directory," is super important for doing this. It helps you go from one folder to another. In this article, we'll talk about 'cd' in detail, explaining how it works with lots of examples. The goal is to help you to change directories in Linux or we can say help you to be good at moving around your computer using 'cd' in Linux.

3. CP

COMMAND NAME: cp

SYNTAX: cp [OPTION] Source Destination

cp [OPTION] Source Directory

cp [OPTION] Source-1 Source-2 Source-3 Source-n Directory

The first and second syntax is used to copy the Source file to the Destination file or Directory. The third syntax is used to copy multiple Sources(files) to the Directory.

EXAMPLE:

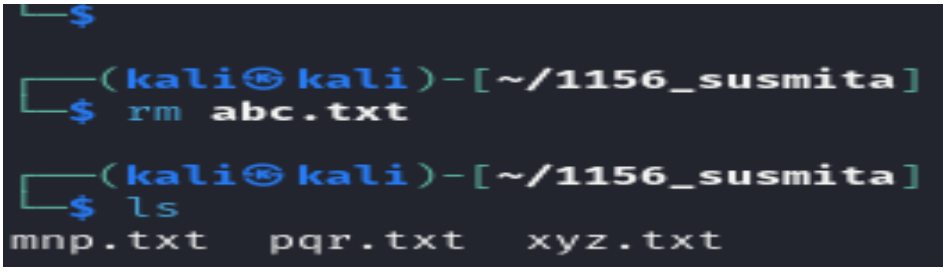
EXPLANATION: cp stands for a copy. This command is used to copy files or groups of files or directories. It creates an exact image of a file on a disk with a different file name. cp command requires at least two filenames in its arguments.

4. RM

COMMAND NAME: rm

SYNTAX: rm [OPTION]... FILE...

EXAMPLE:



EXPLANATION: rm stands for remove here. rm command is used to remove objects such as files, directories, symbolic links and so on from the file system like UNIX. To be more precise, rm removes references to objects from the file system, where those objects might have had multiple references (for example, a file with two different names). By default, it does not remove directories. This command normally works silently and you should be very careful while running rm command because once you delete the files then you are not able to recover the contents of files and directories.

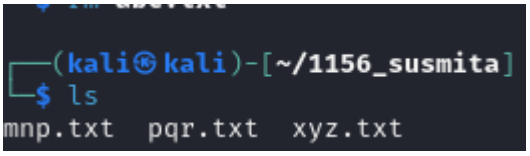
5. Ls

COMMAND NAME: Ls

SYNTAX: ls

'ls' will display the contents of the current directory. By default, 'ls' lists files and directories in alphabetical order.

EXAMPLE:



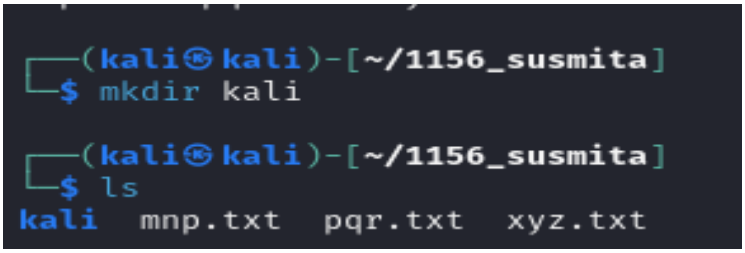
EXPLANATION: The ls is the list command in Linux. It will show the full list content of you directory. Just type ls and press the enter key. The whole content will be shown. ls is a command used to list computer directories and files in Unix-like and Unix operating systems. It is developed by the Single Unix Specification and POSIX.

6. Mkdir

COMMAND NAME: mkdir

SYNTAX: mkdir [options...] [directory_name]

EXAMPLE:



EXPLANATION: The mkdir stands for 'make directory'. With the help of mkdir command, you can create a new directory wherever you want in your system. Just type " mkdir <dir name>, in place of <dir name>type the name of new directory, you want to create and then press enter.

7. Touch

COMMAND NAME: Touch

SYNTAX: touch file name

EXAMPLE:

```
┌──(kali㉿kali)-[~/1156_susmita]
└─$ touch file1

┌──(kali㉿kali)-[~/1156_susmita]
└─$ ls
file1  kali  mnp.txt  pqr.txt  xyz.txt
```

EXPLANATION:The touch command is a standard command used in the UNIX/Linux operating system which is used to create, change and modify the timestamps of a file. Basically, there are two different commands to create a file in the Linux system which are as follows:

touch command: It is used to create a file without any content. The file created using the touch command is empty. This command can be used when the user doesn't have data to store at the time of file creation.

8. Cat

COMMAND NAME: Cat

SYNTAX: cat file name

EXAMPLE:

```
┌──(kali㉿kali)-[~/1156_susmita]
└─$ cat > pizza.txt
hii bye^C

┌──(kali㉿kali)-[~/1156_susmita]
└─$ ls
file1  kali  mnp.txt  pizza.txt  pqr.txt  xyz.txt

┌──(kali㉿kali)-[~/1156_susmita]
└─$ cat  pizza.txt
```

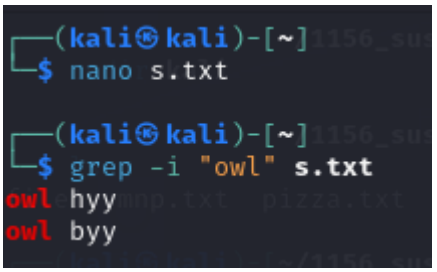
EXPLANATION: Cat(concatenate) command is very frequently used in Linux. It reads data from the file and gives its content as output. It helps us to create, view, and concatenate files. So let us see some frequently used cat commands.

9. Grep

COMMAND NAME: Grep

SYNTAX: grep [options] pattern [files]

EXAMPLE:



EXPLANATION: The grep filter searches a file for a particular pattern of characters and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression (grep stands for global search for regular expression and printout).

10. Rmdir

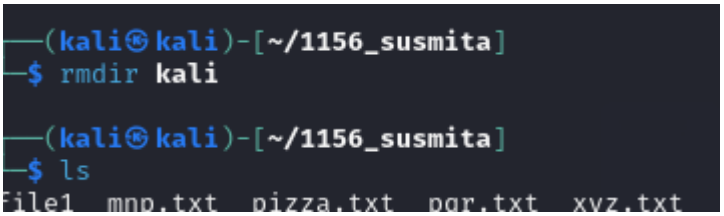
COMMAND NAME: Rmdir

SYNTAX: rmdir <options> <directory>

In the <options>, you can use various types of flags as per your

requirements while removing <directory>.

EXAMPLE:



EXPLANATION: The rmdir command is useful when you want to remove the empty directories from the filesystem in Linux. This command lets you specify the terminal to remove a particular directory right from the terminal. However, having correct knowledge of the rmdir command is essential, or you may end up deleting any important directory

11.Locate

COMMAND NAME: Locate

SYNTAX: locate [OPTION]...

EXAMPLE:

```
┌──(kali㉿kali)-[~/1156_susmita]
└─$ locate --help
Usage: plocate [OPTION]... PATTERN...

  -b, --basename         search only the file name portion of path names
  -c, --count            print number of matches instead of the matches
  -d, --database DBPATH  search for files in DBPATH
                         (default is /var/lib/plocate/plocate.db)
  -i, --ignore-case      search case-insensitively
  -l, --limit LIMIT      stop after LIMIT matches
  -0, --null             delimit matches by NUL instead of newline
  -N, --literal          do not quote filenames, even if printing to a tty
  -r, --regexp           interpret patterns as basic regexps (slow)
      --regex            interpret patterns as extended regexps (slow)
  -w, --wholename        search the entire path name (default; see -b)
      --help             print this help
      --version          print version information
```

EXPLANATION: locate command in Linux is used to find the files by name.

There are two most widely used file-searching utilities accessible users called to find and locate. The locate utility works better and faster than the find command counterpart because instead of searching the file system when a file search is initiated, it would look through a database. This database contains bits and parts of files and their corresponding paths on your system. By default, locate command does not check whether the files found in the database still exist and it never reports files created after the most recent update of the relevant database.

12. Mv

COMMAND NAME: Mv

SYNTAX: mv [options(s)] [source_file_name(s)] [Destination_file_name]

EXAMPLE:

```
┌──(kali㉿kali)-[~/1156_susmita]
└─$ mv file1 file2

┌──(kali㉿kali)-[~/1156_susmita]
└─$ ls
file2  mnp.txt  pizza.txt  pqr.txt  xyz.txt

┌──(kali㉿kali)-[~/1156_susmita]
└─$ l
file2  mnp.txt  pizza.txt  pqr.txt  xyz.txt
```

EXPLANATION: In UNIX-based operating systems like Linux and macOS, `mv` stands for "move". But in this article, we will be talking about the "mv command in Linux". As its name suggests this command is used to rename file directories and move files from one location to another within a file system.

Two Distinct Functions of `mv` Command

- 1) Renaming a file or directory.
- 2) Moving a file or directory to another location

13. Sudo

COMMAND NAME: Sudo

SYNTAX: sudo

EXAMPLE:

```
┌──(kali㉿kali)-[~]
└─$ sudo -h
sudo - execute a command as another user

usage: sudo -h | -K | -k | -V
usage: sudo -v [-ABkNnS] [-g group] [-h host] [-p prompt] [-u user]
usage: sudo -l [-ABkNnS] [-g group] [-h host] [-p prompt] [-U user]
            [-u user] [command [arg ...]]
usage: sudo [-ABbEHkNnPS] [-r role] [-t type] [-C num] [-D directory]
            [-g group] [-h host] [-p prompt] [-R directory] [-T timeout]
            [-u user] [VAR=value] [-i | -s] [command [arg ...]]
usage: sudo -e [-ABkNnS] [-r role] [-t type] [-C num] [-D directory]
            [-g group] [-h host] [-p prompt] [-R directory] [-T timeout]
            [-u user] file ...

Options:
  -A, --askpass             use a helper program for password prompting
  -b, --background          run command in the background
  -B, --bell                ring bell when prompting
  -C, --close-from=num      close all file descriptors ≥ num
  -D, --chdir=directory     change the working directory before running
                            command
  -E, --preserve-env        preserve user environment when running command
      --preserve-env=list   preserve specific environment variables
  -e, --edit                edit files instead of running a command
  -g, --group=group         run command as the specified group name or ID
  -H, --set-home            set HOME variable to target user's home dir
  -h, --help                display help message and exit
  -h, --host=host           run command on host (if supported by plugin)
  -i, --login               run login shell as the target user; a command
                            may also be specified
  -K, --remove-timestamp    remove timestamp file completely
  -k, --reset-timestamp     invalidate timestamp file
  -l, --list                list user's privileges or check a specific
                            command; use twice for longer format
  -n, --non-interactive     non-interactive mode, no prompts are used
  -P, --preserve-groups     preserve group vector instead of setting to
                            target's
  -p, --prompt=prompt       use the specified password prompt
  -R, --chroot=directory    change the root directory before running command
  -r, --role=role           create SELinux security context with specified
                            role
  -S, --stdin               read password from standard input
  -s, --shell               run shell as the target user; a command may
                            also be specified
  -t, --type=type           create SELinux security context with specified
                            type
  -T, --command-timeout=timeout terminate command after the specified time limit
  -U, --other-user=user     in list mode, display privileges for user
  -u, --user=user           run command (or edit file) as specified user
```

EXPLANATION: sudo (Super User DO) command in Linux is generally used as a prefix for some commands that only superusers are allowed to run. If you prefix any command with “sudo”, it will run that command with elevated privileges or in other words allow a user with proper permissions to execute a command as another user, such as the superuser. This is the equivalent of the “run as administrator” option in Windows. The option of sudo lets us have multiple administrators.

14. Df

COMMAND NAME: Df

SYNTAX: The basic syntax of df is:

df [options] [filesystems]

EXAMPLE:

```
┌──(kali㉿kali)-[~]
└─$ df
Filesystem     1K-blocks      Used Available Use% Mounted on
udev             1413904         0   1413904   0% /dev
tmpfs             291116      1008    290108   1% /run
/dev/sda1       28976148  13860224  13618684  51% /
tmpfs            1455560         0   1455560   0% /dev/shm
tmpfs               5120         0      5120   0% /run/lock
tmpfs             291112       124    290988   1% /run/user/1000
```

EXPLANATION: disk free also known as `df`, which is a powerful utility that provides valuable information on disk space utilization. The df command displays information about file system disk space usage on the mounted file system. This command retrieves the information from `/proc/mounts` or `/etc/mtab`. By default, df command shows disk space in Kilobytes (KB) and uses the SI unit suffixes (e.g, M for megabytes, G for gigabytes) for clarity.

15. Head

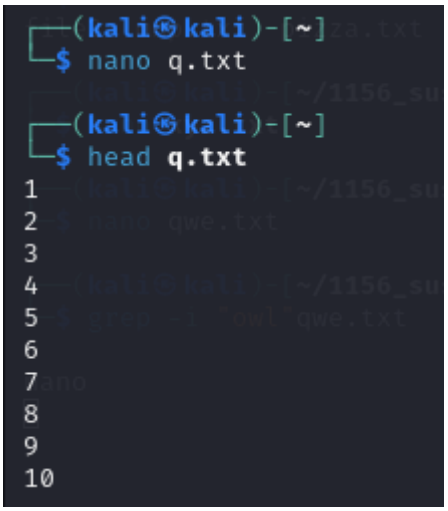
COMMAND NAME: Head

SYNTAX: head [OPTION]... [FILE]...

Let us consider two files having name state.txt and capital.txt contains all

the names of the Indian states and capitals respectively.

EXAMPLE:



EXPLANATION: 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.

16. Tail

COMMAND NAME: Tail

SYNTAX: tail [OPTION]... [FILE]...

EXAMPLE:

EXPLANATION: It is the complementary of head command. The tail command, as the name implies, prints the last N number of data of the given input. By default, it prints the last 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.

17. Tar

COMMAND NAME: Tar

SYNTAX: tar [options] [archive-file] [file or directory to be archived]

EXAMPLE:

```
┌──(kali㉿kali)-[~]
└─$ tar -cf qwl.tar
tar: Cowardly refusing to create an empty archive
Try 'tar --help' or 'tar --usage' for more information.
```

EXPLANATION: The Linux 'tar' stands for tape archive, which is used to create Archive and extract the Archive files. tar command in Linux is one of the important commands that provides archiving functionality in Linux. We can use the Linux tar command to create compressed or uncompressed Archive files and also maintain and modify them.

18. Man

COMMAND NAME: Man

SYNTAX: $man [OPTION]... [COMMAND NAME]...

EXAMPLE:

```
┌──(kali㉿kali)-[~]
└─$ man -C ls
What manual page do you want?
For example, try 'man man'.

┌──(kali㉿kali)-[~]
└─$ man man
```

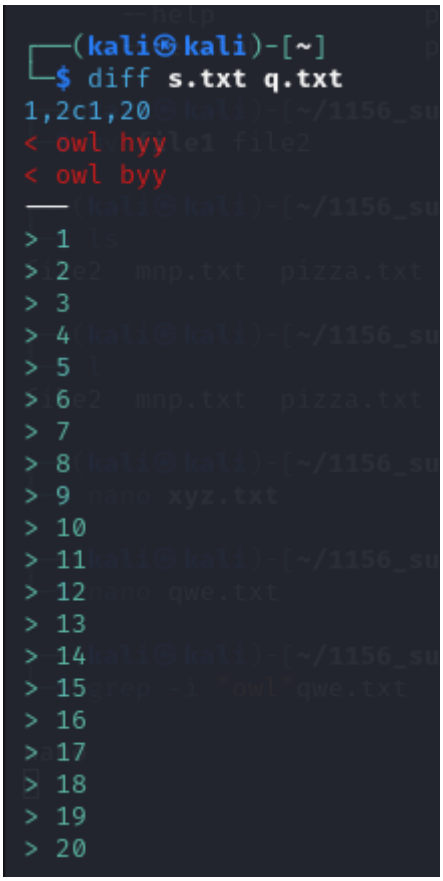
EXPLANATION: man command in Linux is used to display the user manual of any command that we can run on the terminal. It provides a detailed view of the command which includes NAME, SYNOPSIS, DESCRIPTION, OPTIONS, EXIT STATUS, RETURN VALUES, ERRORS, FILES, VERSIONS, EXAMPLES, AUTHORS.

19. Diff

COMMAND NAME: Diff

SYNTAX: diff [options] File1 File2

EXAMPLE:



EXPLANATION: diff stands for difference. This command is used to display the differences in the files by comparing the files line by line. Unlike its fellow members, cmp and comm, it tells us which lines in one file have is to be changed to make the two files identical.

20. DU

COMMAND NAME: Du

SYNTAX: du [options] [directory/file]

EXAMPLE:

EXPLANATION: The du (disk usage) command measures the disk space occupied by files or directories. By default, it measures the current directory and all its subdirectories, printing totals in blocks for each, with a grand total at the bottom.

21. Echo

COMMAND NAME: Echo

SYNTAX: echo [option] [string]

EXAMPLE:

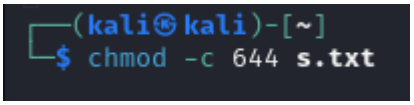
EXPLANATION: `Echo` command in Linux which is a powerful and versatile tool and allows users to display lines of the text or strings on the terminal. Overall, we can say that by understanding the `echo command experimenting with its features, we can effectively display message, variables, or any desired text in our Linux.

22. Chmod

COMMAND NAME: Chmod

SYNTAX: chmod [options] [mode] [File_name]

EXAMPLE:



EXPLANATION: The `chmod` command in Linux is used to modify the permissions and access mode of files and directories. These are the permissions that control who can read, write and execute the file. We have discussed two types of modes for specifying permission: symbolic and octal mode. In symbolic mode, one uses letters and operators to specify the permission. Whereas octal has a three-digit number for specifying the permission. The `chmod` command also provides some options for bulk modifications, for example: `-R` for recursive and `-v` and `-c` for displaying message. The overall conclusion is that `chmod` command in Linux is a very essential tool for managing file and directories permissions.

23. Clear

COMMAND NAME: Clear

SYNTAX: Clear

EXAMPLE:

EXPLANATION: The clear command clears your screen, if possible. The clear command first checks the TERM environment variable for the terminal type. Next, the /usr/share/lib/terminfo directory, which contains terminal definition files, is checked to determine how to clear the screen.

Practical -2

AIM - STUDY OF BASICS OF SHELL PROGRAMMING.

- **What is shell script ?**

A shell script is a text file that contains a sequence of commands for a UNIX-based operating system. It is called a shell script because it combines a sequence of commands, that would otherwise have to be typed into the keyboard one at a time, into a single script.

- **What are the advantages of shell script ?**

Advantages:

1. run sequence of commands as a single command
2. Easy to use
3. Portable (It can be executed in any Unix-like operating systems without any modifications)
4. To automate the frequently performed operations

- **What are disadvantages of shell script ?**

Disadvantages

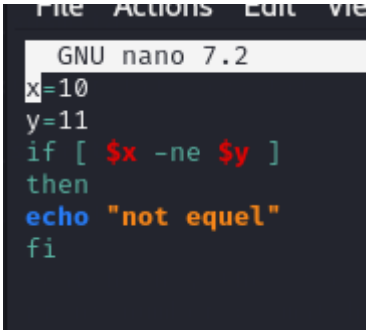
1. There may be errors in shell scripting that prove to be quite costly.
2. The programs in shell script are quite slow while executing and a new process is required for every shell command executed.
3. Different platforms in shell scripting may also have compatibility problems.

- **What are the application of shell scripting?**

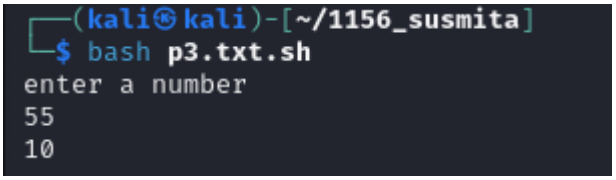
1. Automating the code compiling process.
2. Running a program or creating a program environment.
3. Completing batch.
4. Manipulating files.
5. Linking existing programs together.
6. Executing routine backups.
7. Monitoring a system.

- **BASIC PROGRAM:**

INPUT:



OUTPUT :



PRACTICAL - 3

AIM- WRITE A SHELL SCRIPT TO PRINT GIVEN NUMBERS SUM OF ALL DIGITS.

```
 GNU nano 7.2
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 -

```
┌──(kali㉿kali)-[~/1156_susmita]
└─$ bash p3.txt.sh
enter a number
55
10
```

Even or Odd -

```
 GNU nano 7.2
echo "enter a number "
read num

if  (($num % 2 == 0))

then
echo "num is even"
else
echo "num is odd"
fi
```

Output –

```
┌──(kali㉿kali)-[~/1156_susmita]
└─$ bash p4.txt.sh
enter a number
4
num is even

┌──(kali㉿kali)-[~/1156_susmita]
└─$ bash p4.txt.sh
enter a number
19
num is odd
```

Smallest / Greater Value –

```
 GNU nano 7.2
echo "enter num1"
read num1
echo "enter num2"
read num2
if ((num1>num2))
then
echo "num1 is greater"
else
echo "num2 is greater"
fi
```

Output

```
┌──(kali㉿kali)-[~/1156_susmita]
└─$ bash p3.1.txt.sh
enter num1
50
enter num2
45
num1 is greater
```

PRACTICAL - 4

AIM- WRITE A SHELL SCRIPT TO VALIDATE THE ENTERED DATA. (eg. Date format is: dd-mm-yyyy)

```
 GNU nano 7.2
echo "date validator"
dd=0
mm=0
yy=0
read -p "enter day (dd) :" dd
read -p "enter month (mm) :" mm
read -p "enter year (yyyy) :" yy
if [ $mm -le 0 -o $mm  -gt 12 ]
then
echo "$mm is inalid month."
exit 1
fi
case $mm in
1 | 3| 5|7| 8| 10|12)
days=31
;;
2)
days=28
;;
4 | 6| 9| 11)
days=30
;;
*)
days=-1
;;
esac
if [ $mm -eq 2 ]
then
a=`expr $yy % 4`
b=`expr $yy % 100`
c=`expr $yy % 400`
if [ $a -eq 0 -a $b -ne 0 -o $c -eq 0 ]
then
days=29
fi
fi
if [ $dd -le 0 -o $dd -gt $days ]
then
echo "$dd day is invalid"
exit 3
fi
echo "$dd/$mm/$yy is a valid date"
```

Output –

```
┌──(kali㉿kali)-[~/1156_susmita]
└─$ bash  p4.1.txt.sh
date validator
enter day (dd) :12
enter month (mm) :12
enter year (yyyy) :1003
12/12/1003 is a valid date
```

PRACTICAL-5

AIM- WRITE A SHELL SCRIPT TO CHECK ENTERED STRING IN PALINDROME OR NOT.

```
echo "enter a string"
read str
reversed_string=""
len=${#str}
for (( i=$len-1; i>=0; i-- ))
do
reversed_string="$reversed_string${str:$i:1}"
done
if [ $str == $reversed_string ]
then
echo "pallindrome"
else
echo "not pallindrome"
fi
```

Output –

```
┌──(kali㉿kali)-[~/1156_susmita]
└─$ bash  p5.1.txt.sh
enter a string
xyz
not pallindrome

┌──(kali㉿kali)-[~/1156_susmita]
└─$ bash  p5.1.txt.sh
enter a string
qqqq
pallindrome
```

PRACTICAL - 6

AIM- WRITE A SHELL SCRIPT TO SAY GOOD MORNING /AFTERNOON/ EVENING AS YOU LOG INTO SYSTEM.

```
GNU nano 7.2                                    p6
#!/bin/bash
hour=`date +%I`
min=`date +%M`
ampm=`date +%p`
echo "$hour : $min $ampm"
if [ $hour -It 12] #if hour is less than 12
then
echo "GOOD MORNING WORLD"
elif[$hour -le 16] # if hour is less than equal to 16
then
echo"GOOF AFTERNOON WORLD"
elif[$hour -le 20] # if hour is less than equal to 20
then
echo "GOOD EVENING WORLD"
else
echo "GOOD NIGHT WORLD"
fi
```

Output –

```
┌──(kali㉿kali)-[~/1156_susmita]
└─$ bash  p6.1.txt.sh
02 : 28 PM
```

PRACTICAL - 7

AIM– WRITE A C PROGRAM TO CREATE A CHILD PROCESS.

main.c Save Run

```c
#include <stdio.h>

int main()
{
    for(int i=0;i<5;i++) // loop will run n times (n=5)
    {
        if(fork() == 0)
        {
            printf("[son] pid %d from [parent] pid %d\n",getpid(),getppid());
            exit(0);
        }
    }
    for(int i=0;i<5;i++) // loop will run n times (n=5)
    wait(NULL);

}
```

Output –

Output

```
/tmp/GHYAXcoUHq.o
[son] pid 22522 from [parent] pid 22521
[son] pid 22523 from [parent] pid 22521
[son] pid 22524 from [parent] pid 22521
[son] pid 22525 from [parent] pid 22521
[son] pid 22526 from [parent] pid 22521
```

PRACTICAL - 8

AIM– FINDING OUT BIGGEST NUMBER FROM GIVEN THREE NUMBERS SUPPLIED AS COMMAND LINE ARGUMENT.

```
num1=15
num2=20
num3=10
max=$num1
if [ $num2 -gt $max ];then
max=$num2
fi
if [ $num3 -gt $max ];then
max=$num3
fi
echo "the largest num is : $max"
```

Output –

```
┌──(kali㉿kali)-[~/1156_susmita]
└─$ bash  p8.1.txt.sh
the largest num is : 20
```