

## LAB NO. 1

22-09-2022

1. Investigate the fundamental Unix/Linux commands.
2. Obtaining the OS system data file and its associated information.
4. Create utility programs that use I/O system calls to simulate operations such as ls, cp, grep, and others.

## Linux commands

### 1. cd :- Change to new directory

cd command in linux known as change directory command. It is used to change current working directory.

#### Syntax :-

```
$ cd [directory]
```

To move inside a subdirectory : to move inside a subdirectory in linux we use

```
$ cd [directory_name]
```

#### Description :-

In the below example, we have checked number of directories in our home directory and moved inside the “ nidhi ” directory by using “ cd nidhi ” command.



A screenshot of a terminal window titled "daksdutta@dakshita: ~/Desktop". The window shows the command \$ cd Desktop being typed and then executed, changing the directory to ~/Desktop. The terminal has a dark theme with light-colored text.

```
daksdutta@dakshita:~$ cd Desktop
daksdutta@dakshita:~/Desktop$
```

### 2. mkdir :- create new directory

mkdir command in Linux allows the user to create directories (also referred to as folders in some operating systems ). This command can create multiple directories at once as well as set the permissions for the directories.

#### Syntax :-

1. mkdir [options...] [directories ...]
2. mkdir directory name
3. mkdir <directory name>

#### Description :-

Create a two directories “nidhi1” and “nidhi2” at the current directory and display the directories by ls command.

```
daksdutta@dakshita:~/Desktop$ cd Desktop
daksdutta@dakshita:~/Desktop$ mkdir OS_LAB
daksdutta@dakshita:~/Desktop$ mkdir OS_LAB1
daksdutta@dakshita:~/Desktop$ mkdir OS_LAB2
daksdutta@dakshita:~/Desktop$ ls
nidhi  OS_LAB  OS_LAB1  OS_LAB2  xampp-control-panel.desktop
daksdutta@dakshita:~/Desktop$
```

### 3. rmdir :- remove empty directory (remove files first)

rmdir 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 :-**

1. rmdir <directory name>
2. *rmdir [-p] [-v / -verbose] [-ignore-fail-on-non-empty] directories ...*

**Description :-**

In the below example, We have two directories nidhi1 and nidhi2. We execute rmdir command to remove the directory nidhi2. Remove the directory “nidhi2” if it is empty. After executing the rmdir command nidhi2 directory is removed.

```
daksdutta@dakshita:~/Desktop$ ls
nidhi  OS_LAB  OS_LAB1  OS_LAB2  xampp-control-panel.desktop
daksdutta@dakshita:~/Desktop$ rmdir nidhi
daksdutta@dakshita:~/Desktop$ ls
OS_LAB  OS_LAB1  OS_LAB2  xampp-control-panel.desktop
daksdutta@dakshita:~/Desktop$
```

### 4. mv : - change name of directory

The mv command is used to move a file or a directory form one location to another location.

**Syntax :-**

mv <file name> <directory path>

### Description :-

```
daksdutta@dakshita:~/Desktop$ ls
OS_LAB OS_LAB1 OS_LAB2 xampp-control-panel.desktop
daksdutta@dakshita:~/Desktop$ mv OS_LAB OS_LAB3
daksdutta@dakshita:~/Desktop$ ls
OS_LAB1 OS_LAB2 OS_LAB3 xampp-control-panel.desktop
daksdutta@dakshita:~/Desktop$ █
```

### 5. pwd :- show current directory

The pwd command is used to display the location of the current working directory.

#### Syntax :-

```
pwd
```

### Description :-

```
daksdutta@dakshita:~/Desktop$ pwd
/home/daksdutta/Desktop
daksdutta@dakshita:~/Desktop$
```

### 6. date :- show date and time

The date command is used to display date, time, time zone, and more.

#### Syntax :-

```
date
```

### Description :-

```
daksdutta@dakshita:~/Desktop$ date
Saturday 15 October 2022 09:34:51 PM IST
daksdutta@dakshita:~/Desktop$ █
```

### 7. history :- list of previously executed commands

history command is used to view the previously executed command. This feature was not available in the Bourne shell. Bash and Korn support this feature in which every command executed is treated as the event and is associated with an event number using

which they can be recalled and changed if required. These commands are saved in a history file. In Bash shell history command shows the whole list of the command.

### Syntax :-

```
$ history
```

### Description :-

```
daksdutta@dakshita:~/Desktop$ history
 1 truncate -s 0 /var/log/syslog-
 2 sudo thunar
 3 journalctl --disk-usage
 4 sudo journalctl --rotate
 5 sudo journalctl --vacuum-time=2days
 6 sudo journalctl --vacuum-size=100M
 7 sudo journalctl --vacuum-files=5
 8 sudo -H gedit /etc/systemd/journald.conf
 9 python -u "/media/daksdutta/New Volume/input.py"
10 /bin/python3 "/media/daksdutta/New Volume/input.py"
11 python
12 python3
13 /bin/python3 "/media/daksdutta/New Volume1/input.py"
14 sudo apt-get update
15 sudo apt-get upgrade -y
16 sudo apt-get install python3
17 pip3 install jupyter
18 jupyter notebook --allow-root
19 pip install numpy
20 pip install pandas
21 pip install matplotlib.pyplot
22 pip3 install matplotlib
23 install seaborn
24 pip install seaborn
25 auto-update-vscode
26 auto-update-vscode
27 sudo apt install code
28 sudo apt update
29 sudo add-apt-repository -y "deb https://packages.microsoft.com/repos/vscode stable main "
30 sudo apt update
31 sudo apt -y install code
32 echo "# SpotifyAnalysis" >> README.md
33 git init
34 git add README.md
35 git commit -m "first commit"
36 git branch -M main
37 git remote add origin https://github.com/daksdutta/SpotifyAnalysis.git
38 git push -u origin main
39 echo "# SpotifyAnalysis" >> README.md
40 git init
41 git add README.md
42 git commit -m "first commit"
43 git config --global user.email "you@example.com"
44 git config --global user.name "you@example.com"
45 daksdutta@gmail.com
46 git config --user.email "daksdutta@gmail.com"
47 git commit -m "first commit"
48 git branch -M main
49 git remote add origin https://github.com/daksdutta/SpotifyAnalysis.git
50 git push -u origin main
51 git remote add origin https://github.com/daksdutta/SpotifyAnalysis.git
52 git branch -M main
53 git push -u origin main-
54 echo "# SpotifyAnalysis" >> README.md-
```

```
84 sudo apt-get install r-base
85 lsb_release -a
86 apt install r-base
87 sudo apt install r-base
88 cd "/media/daksdutta/New Volume/MCA/Data Structure/" && g++ 22MCA0313DA1.cpp -o 22MCA0313DA1 && "/media/daksdutta/New Volume/MCA/Data Structure/"22MCA0313DA1
89 cd "/media/daksdutta/New Volume/MCA/Data Structure/" && g++ testing.cpp -o testing && "/media/daksdutta/New Volume/MCA/Data Structure/"testing
90 cd "/media/daksdutta/New Volume/MCA/Data Structure/" && g++ 22MCA0313DA1.cpp -o 22MCA0313DA1 && "/media/daksdutta/New Volume/MCA/Data Structure/"22MCA0313DA1
91 cd "/media/daksdutta/New Volume/MCA/Data Structure/" && g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile && "/media/daksdutta/New Volume/MCA/Data Structure/"tempCodeRu
92 cd "/media/daksdutta/New Volume/MCA/Data Structure/" && g++ 22MCA0313DA1.cpp -o 22MCA0313DA1 && "/media/daksdutta/New Volume/MCA/Data Structure/"22MCA0313DA1
93 cd "/media/daksdutta/New Volume/MCA/DS Practice/" && g++ Stack.cpp -o Stack && "/media/daksdutta/New Volume/MCA/DS Practice/"Stack
94 unzip oracle-xe-11.2.0-1.0.x86_64.rpm.zip
95 sudo apt-get install alien libaio1 unixodbc
96 sudo alien --scripts -d oracle-xe-11.2.0-1.0.x86_64.rpm
97 cd Disk1
98 sudo apt-get install alien libaio1 unixodbc
99 sudo alien --scripts -d oracle-xe-11.2.0-1.0.x86_64.rpm
100 sudo apt get alien
101 sudo apt-get install alien
102 sudo apt insstall unrar
103 sudo apt install unrar
104 unzip oracle-xe-11.2.0-1.0.x86_64.rpm.zip
105 unzip oracle-xe-11.2.0-1.0.x86_64.rpm.zip
106 unzip oracle-xe-11.2.0-1.0.x86_64.rpm.zip
107 sudo dpkg --install oracle-xe_11.2.0-2_amd64.deb
108 sqlplus
109 sudo apt-get install alien libaio1 unixodbc
110 tty
111 finger
112 sudo apt install finger
113 finfer
114 sudo apt install
115 finger
116 sudo apt install finger
117 finger
118 sudo apt install finger
119 finger
120 sudo apt install finger
121 --fix-missing
122 apt-get update
123 finger
124 cd Desktop
125 mkdir nidhi
126 cd Desktop
127 mkdir OS_LAB
128 mkdir OS_LAB1
129 mkdir OS_LAB2
130 ls
131 rmdir nidhi
132 ls
133 mv OS_LAB OS_LAB3
134 ls
135 pwd
136 date
137 history
aksdutta@dakshita:~/Desktop$ finger
```

## 8. cal month year :- Prints a calendar for the specified month of the specified year.

The cal command displays a well-presented calendar on the terminal. Just enter the word cal on your terminal prompt. The cal command is used to display the current month's calendar with the current date highlighted.

### Syntax :-

Cal < month year>

### Description :-

```
$ cal 11 2022
November 2022
Su Mo Tu We Th Fr Sa
      1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
$
```

## 9. Man:- show online documentation by program name

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 [OPTION]... [COMMAND NAME]...
```

### Description :-

```
MAN(1)                                Manual pager utils                               MAN(1)

NAME
    man - an interface to the system reference manuals

SYNOPSIS
    man [man options] [[section] page ...] ...
    man -k [apropos options] regexp ...
    man -K [man options] [section] term ...
    man -f [whatis options] page ...
    man -l [man options] file ...
    man -w|-W [man options] page ...

DESCRIPTION
    man is the system's manual pager. Each page argument given to man is normally the name of
    a program, utility or function. The manual page associated with each of these arguments is
    then found and displayed. A section, if provided, will direct man to look only in that
    section of the manual. The default action is to search in all of the available sections
    following a pre-defined order (see DEFAULTS), and to show only the first page found, even
    if page exists in several sections.

    The table below shows the section numbers of the manual followed by the types of pages they
    contain.

    1 Executable programs or shell commands
    2 System calls (functions provided by the kernel)
    3 Library calls (functions within program libraries)
    4 Special files (usually found in /dev)
    5 File formats and conventions, e.g. /etc/passwd
    6 Games
    7 Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
    8 System administration commands (usually only for root)
    9 Kernel routines [Non standard]

    A manual page consists of several sections.

    Conventional section names include NAME, SYNOPSIS, CONFIGURATION, DESCRIPTION, OPTIONS,
    EXIT STATUS, RETURN VALUE, ERRORS, ENVIRONMENT, FILES, VERSIONS, CONFORMING TO, NOTES,
    BUGS, EXAMPLE, AUTHORS, and SEE ALSO.

    The following conventions apply to the SYNOPSIS section and can be used as a guide in other
    sections.

    bold text          type exactly as shown.
    italic text        replace with appropriate argument.
    [-abc]            any or all arguments within [ ] are optional.
    -a|-b             options delimited by | cannot be used together.
    argument ...      argument is repeatable.
    [expression] ...  entire expression within [ ] is repeatable.

    Exact rendering may vary depending on the output device. For instance, man will usually
    not be able to render italics when running in a terminal, and will typically use underlined
    or coloured text instead.

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

## 10. w, who :- who is on the system and what they are doing

w command in Linux is used to show who is logged on and what they are doing. This command shows the information about the users currently on the machine and their processes.

### Syntax :-

w [options] user [...]

### Description :-

```
$ w -i
18:17:33 up 4 days, 23:45, 32 users, load average: 2.90, 2.58, 1.75
USER      TTY      FROM          LOGIN@      IDLE      JCPU      PCPU WHAT
$ w -o
18:17:40 up 4 days, 23:45, 32 users, load average: 2.67, 2.54, 1.74
USER      TTY      FROM          LOGIN@      IDLE      JCPU      PCPU WHAT
$ w -f
18:17:55 up 4 days, 23:45, 32 users, load average: 2.23, 2.45, 1.72
USER      TTY          LOGIN@      IDLE      JCPU      PCPU WHAT
$ w phoenixnap
18:18:15 up 4 days, 23:46, 32 users, load average: 1.81, 2.34, 1.70
USER      TTY          FROM          LOGIN@      IDLE      JCPU      PCPU WHAT
$ w --help
Usage:
```

```
$ w --help
Usage:
w [options]

Options:
-h, --no-header      do not print header
-u, --no-current    ignore current process username
-s, --short         short format
-f, --from           show remote hostname field
-o, --old-style     old style output
-i, --ip-addr        display IP address instead of hostname (if possible)

    | --help      display this help and exit
-V, --version        output version information and exit
```

```
$ w -i
18:17:33 up 4 days, 23:45, 32 users, load average: 2.90, 2.58, 1.75
USER      TTY      FROM          LOGIN@      IDLE      JCPU      PCPU WHAT
```

```
$ w -u
18:17:13 up 4 days, 23:45, 32 users, load average: 3.56, 2.68, 1.77
USER      TTY      FROM          LOGIN@      IDLE      JCPU      PCPU WHAT
```

## 11. who am i :- who is logged onto this terminal

whoami command is used both in *Unix Operating System* and as well as in *Windows Operating System*.

- It is basically the concatenation of the strings “who”, “am”, “i” as whoami.

- It displays the username of the current user when this command is invoked.
- It is similar as running the id command with the options -un.

#### Syntax :-

```
$ whoami
```

#### Description :-

```
daksdutta@dakshita:~/Desktop$ whoami
daksdutta
daksdutta@dakshita:~/Desktop$
```

## 12. uptime :- show one line summary of system status

It is used to find out how long the system is active (running). This command returns set of values that involve, the current time, and the amount of time system is in running state, number of users currently logged into, and the load time for the past 1, 5 and 15 minutes respectively.

#### Syntax :-

```
uptime [-options]
```

#### Description :-

```
daksdutta@dakshita:~/Desktop$ uptime
21:40:52 up 21 min, 1 user,  load average: 0.10, 0.13, 0.10
daksdutta@dakshita:~/Desktop$
```

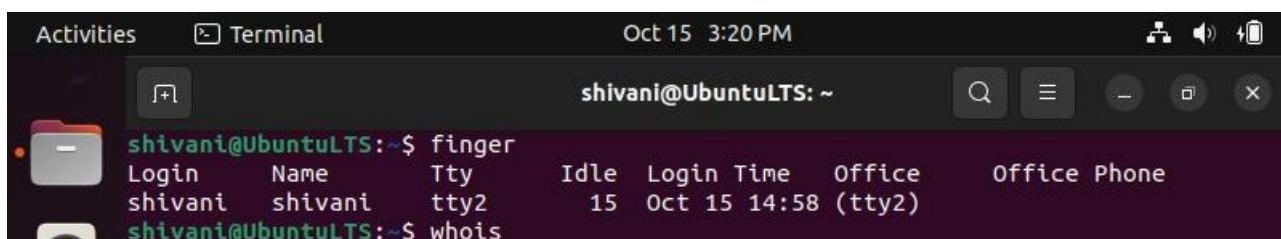
## 13. finger :- find out info about a user@system

The finger command looks up and displays information about system users.

#### Syntax :-

```
Finger [-Imsp] [user ...][user@host ...]
```

#### Description :-



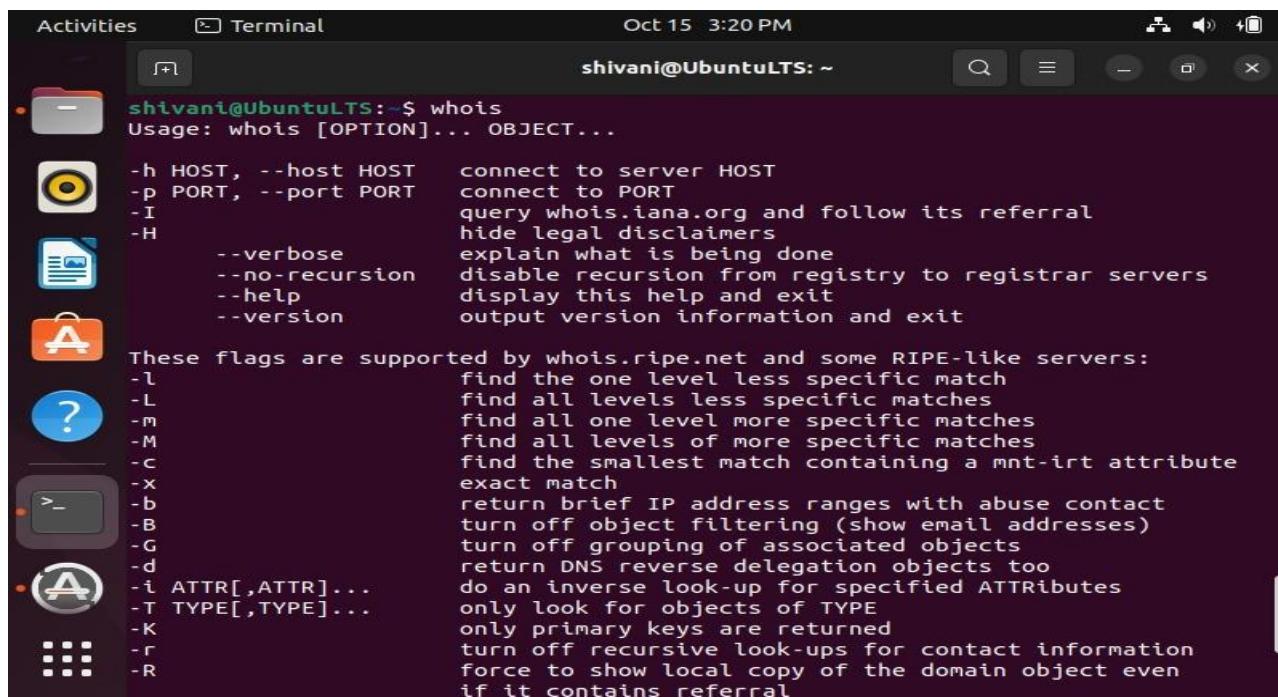
## 14. whois :- look up information in the Stanford Directory

The whois command displays information about a website's record. You may get all the information about a website regarding its registration and owner's information.

Syntax :-

whois <websiteName>

Description :-



A screenshot of a Linux terminal window titled "Terminal". The window shows the command "shivani@UbuntuLTS: ~\$ whois" followed by its usage information. The usage information includes options for connecting to hosts and ports, querying IANA, hiding disclaimers, being verbose, disabling recursion, displaying help, and outputting version information. It also lists flags supported by whois.ripe.net, such as -l, -L, -m, -M, -c, -x, -b, -B, -G, -d, -i, -T, -K, -r, and -R, each with a brief description of their function.

```
shivani@UbuntuLTS: ~$ whois
Usage: whois [OPTION]... OBJECT...
-h HOST, --host HOST      connect to server HOST
-p PORT, --port PORT     connect to PORT
-I                         query whois.iana.org and follow its referral
-H                         hide legal disclaimers
--verbose                  explain what is being done
--no-recursion             disable recursion from registry to registrar servers
--help                      display this help and exit
--version                  output version information and exit

These flags are supported by whois.ripe.net and some RIPE-like servers:
-l                         find the one level less specific match
-L                         find all levels less specific matches
-m                         find all one level more specific matches
-M                         find all levels of more specific matches
-c                         find the smallest match containing a mnt-irt attribute
-x                         exact match
-b                         return brief IP address ranges with abuse contact
-B                         turn off object filtering (show email addresses)
-G                         turn off grouping of associated objects
-d                         return DNS reverse delegation objects too
-i ATTR[,ATTR]...          do an inverse look-up for specified ATTRibutes
-T TYPE[,TYPE]...          only look for objects of TYPE
-K                         only primary keys are returned
-r                         turn off recursive look-ups for contact information
-R                         force to show local copy of the domain object even
if it contains referral
```

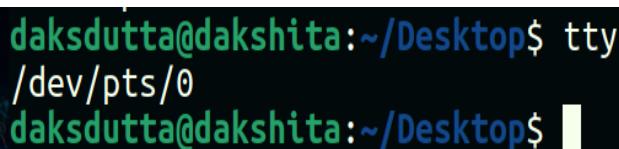
## 15. tty :- know the terminal name.

The **tty** command of terminal basically prints the file name of the terminal connected to standard input. **tty** is short of teletype, but popularly known as a terminal it allows you to interact with the system by passing on the data (you input) to the system, and displaying the output produced by the system.

Syntax :-

tty [OPTION]....

Description :-



A screenshot of a Linux terminal window showing the command "daksdutta@dakshita: ~/Desktop\$ tty" followed by the output "/dev/pts/0".

```
daksdutta@dakshita: ~/Desktop$ tty
/dev/pts/0
daksdutta@dakshita: ~/Desktop$
```

## 16. uname :- print system information

Linux command to get basic information about the OS. The uname commands allow you to know some basic information which comes really handy when you work on multiple systems. In general, if you're working with a single computer, you won't really need it as often as someone who is a network administrator.

### Syntax :-

```
        uname
```

### Description :-

```
daksdutta@dakshita:~/Desktop$ uname  
Linux  
daksdutta@dakshita:~/Desktop$
```

## 17. cat :- view files

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

### Syntax :-

```
$cat filename
```

### Description :-

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ touch ntant1.txt,ntant2.txt,ntant3.txt  
daksdutta@dakshita:~/Desktop/OS_LAB1$ ls  
nidhi1.txt,nidhi2.txt,nidhi3.txt  nidhi.txt  
daksdutta@dakshita:~/Desktop/OS_LAB1$ cat nidhi.txt  
daksdutta@dakshita:~/Desktop/OS_LAB1$ cat > nidhi.txt  
this is operating system lab 1 work assignment given by scope faculty.  
os works as interface between user and application program.  
os manages all the resources of system.  
os is also known as resource manager.  
^C  
daksdutta@dakshita:~/Desktop/OS_LAB1$ cat nidhi.txt  
this is operating system lab 1 work assignment given by scope faculty.  
os works as interface between user and application program.  
os manages all the resources of system.  
os is also known as resource manager.  
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

## 18. cp :- copy files

The cp command is used to copy a file or directory.

### Syntax :-

To copy in the same directory:

```
cp <existing file name> <new file name>
```

## Description :-

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ cp nidhi.txt nidhi1.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ cat nidhi1.txt
this is operating system lab 1 work assignment given by scope faculty.
os works as interface between user and application program.
os manages all the resources of system.
os is also known as resource manager.
daksdutta@dakshita:~/Desktop/OS_LAB1$ ls
```

## 19. ls :- list files in a directory and their attributes

The ls command is used to display a list of content of a directory.

### Syntax :-

```
ls
```

### Description :-

```
daksdutta@dakshita:~/Desktop$ cd OS_LAB1
daksdutta@dakshita:~/Desktop/OS_LAB1$ touch nidhi.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ touch nidhi.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ cat nidhi1.txt
cat: nidhi1.txt: No such file or directory
daksdutta@dakshita:~/Desktop/OS_LAB1$ touch nidhi1.txt,nidhi2.txt,nidhi3.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ ls
nidhi1.txt,nidhi2.txt,nidhi3.txt  nidhi.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

## 20. rm :- remove files

The rm command is used to remove a file.

### Syntax :-

```
rm <file name>
```

### Description :-

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ ls
nidhi1.txt  nidhi1.txt,nidhi2.txt,nidhi3.txt  nidhi4.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ rm nidhi4.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ ls
nidhi1.txt  nidhi1.txt,nidhi2.txt,nidhi3.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

## 21. head :- show first few lines of a file(s)

The head command is used to display the content of a file. It displays the first 10 lines of a file.

### Syntax :-

```
head <file name>
```

### Description :-

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ head nidhi1.txt
1
2
3
4
5
6
6
77
```

## 22. tail :- show last few lines of a file; or reverse line order

The tail command is similar to the head command. The difference between both commands is that it displays the last ten lines of the file content. It is useful for reading the error message.

### Syntax :-

```
tail <file name>
```

### Description :-

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ tail nidhi1.txt
3
4
4
5
5
55
6
6
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

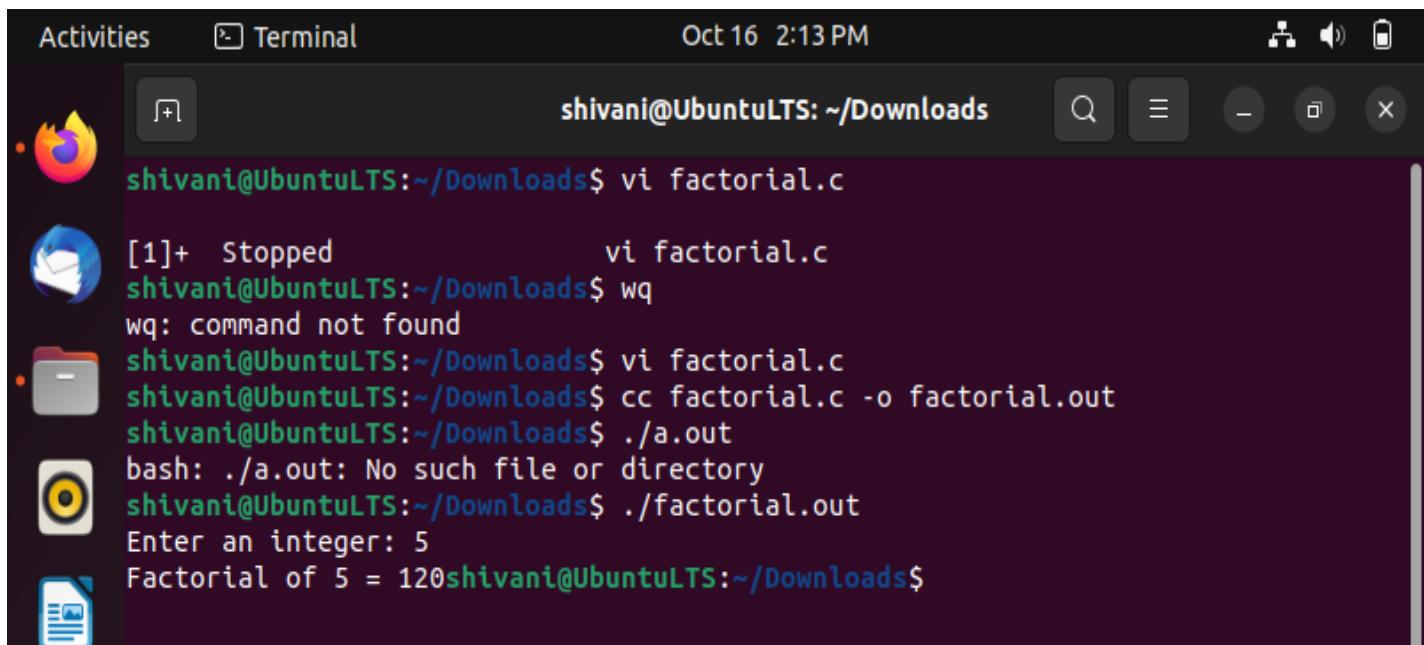
## 23. vi :- full-featured screen editor for modifying text files

The default editor that comes with the UNIX operating system is called vi (visual editor). Using vi editor, we can edit an existing file or create a new file from scratch. we can also use this editor to just read a text file.

### Syntax :-

vi filename

### Description :-



A screenshot of a Linux desktop environment showing a terminal window. The terminal title is "Terminal" and the date and time are "Oct 16 2:13 PM". The terminal window shows the following session:

```
shivani@UbuntuLTS:~/Downloads$ vi factorial.c
[1]+  Stopped                  vi factorial.c
shivani@UbuntuLTS:~/Downloads$ wq
wq: command not found
shivani@UbuntuLTS:~/Downloads$ vi factorial.c
shivani@UbuntuLTS:~/Downloads$ cc factorial.c -o factorial.out
shivani@UbuntuLTS:~/Downloads$ ./a.out
bash: ./a.out: No such file or directory
shivani@UbuntuLTS:~/Downloads$ ./factorial.out
Enter an integer: 5
Factorial of 5 = 120shivani@UbuntuLTS:~/Downloads$
```

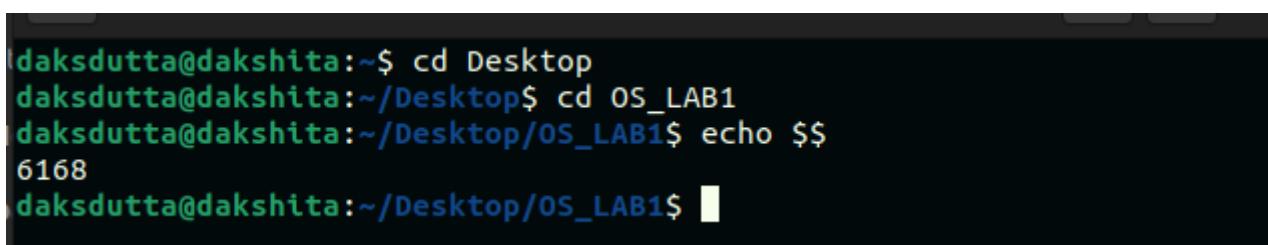
## 24. echo \$\$ :- process id of current shell

It shows the process id of the current shell. The \$\$ variable holds the process ID of the script in which it appears.

### Syntax :-

echo \$\$

### Description :-



A screenshot of a Linux terminal window showing the use of the echo command to print the process ID of the current shell. The terminal session is as follows:

```
daksdutta@dakshita:~$ cd Desktop
daksdutta@dakshita:~/Desktop$ cd OS_LAB1
daksdutta@dakshita:~/Desktop/OS_LAB1$ echo $$
6168
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

## 25. ps :- process status

Linux provides us a utility called **ps** for viewing information related with the processes on a system which stands as abbreviation for “**Process Status**”. ps command is used to list the currently running processes and their PIDs along with some other information depends on different options.

**Syntax :-**

ps [options]

**Description :-**

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ ps
  PID  TTY      TIME CMD
  6168 pts/0    00:00:00 bash
  6191 pts/0    00:00:00 ps
daksdutta@dakshita:~/Desktop/OS_LAB1$ █
```

## 26. Kill :- kill background job or previous process...

kill command in Linux (located in /bin/kill), is a built-in command which is used to terminate processes manually. kill command sends a signal to a process which terminates the process. If the user doesn't specify any signal which is to be sent along with kill command then default TERM signal is sent that terminates the process.

**Syntax :-**

\$kill

**Description :-**

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ kill
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill
-l [sigspec]
daksdutta@dakshita:~/Desktop/OS_LAB1$ █
```

## 27. touch Command

The touch command is used to create empty files. We can create multiple empty files by executing it once.

**Syntax:**

1. touch <file name>
2. touch <file1> <file2> ....

**Description :-**

```
daksdutta@dakshita:~/Desktop$ cd OS_LAB1
daksdutta@dakshita:~/Desktop/OS_LAB1$ touch nidhi.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ touch nidhi.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ cat nidhi1.txt
cat: nidhi1.txt: No such file or directory
daksdutta@dakshita:~/Desktop/OS_LAB1$ touch nidhi1.txt,nidhi2.txt,nidhi3.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ ls
nidhi1.txt,nidhi2.txt,nidhi3.txt  nidhi.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

## Unix commands

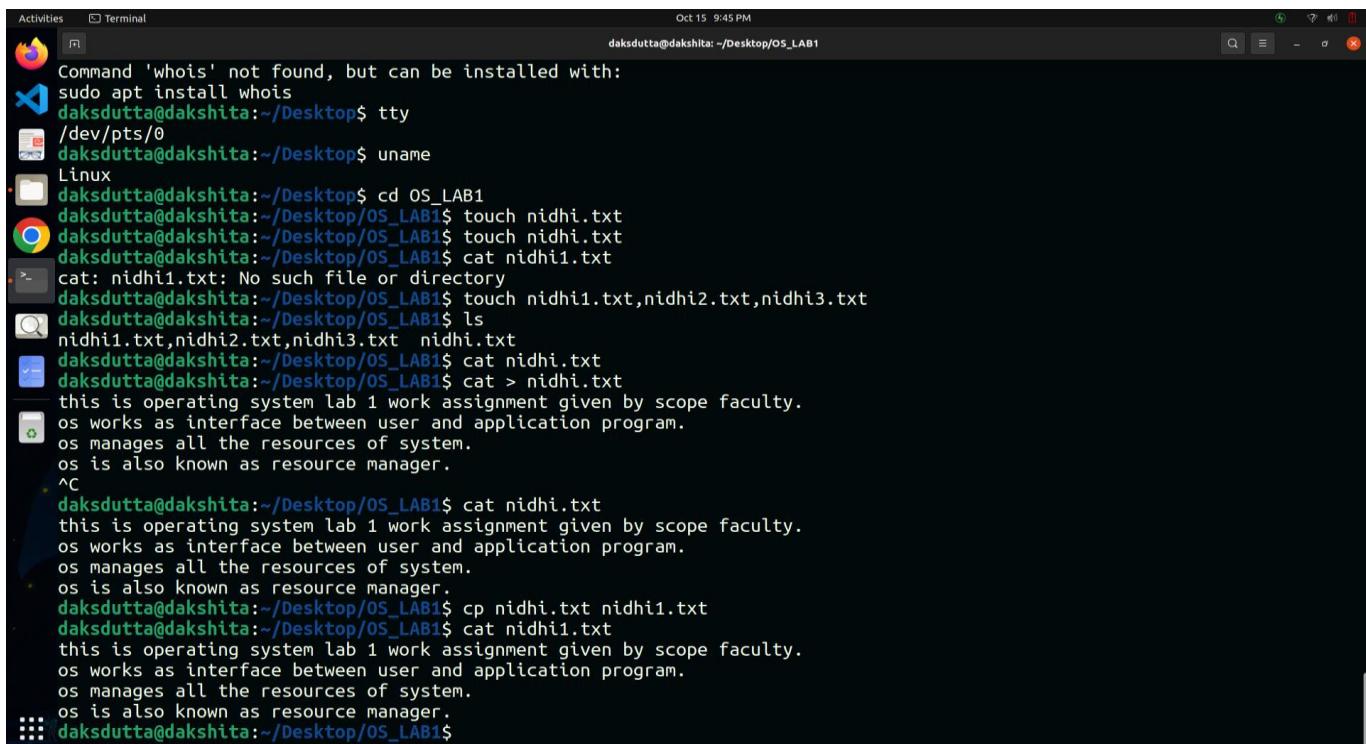
### 1. cat - display or concatenate files

**cat** takes a copy of a file and sends it to the standard output (i.e. to be displayed on your terminal, unless redirected elsewhere), so it is generally used either to read files, or to string together copies of several files, writing the output to a new file.

#### Syntax:-

```
cat ex
cat ex1 ex2 > newex
```

#### Description :-



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "Terminal" and the command line shows the user's session. The user has run several commands related to file creation and concatenation using the "cat" command, and also provided some descriptive text about the operating system's role.

```
Activities Terminal Oct 15 9:45 PM
daksdutta@dakshita: ~/Desktop/OS_LAB1
Command 'whois' not found, but can be installed with:
sudo apt install whois
daksdutta@dakshita:~/Desktop$ tty
/dev/pts/0
daksdutta@dakshita:~/Desktop$ uname
Linux
daksdutta@dakshita:~/Desktop$ cd OS_LAB1
daksdutta@dakshita:~/Desktop/OS_LAB1$ touch nidhi.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ touch nidhi.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ cat nidhi1.txt
cat: nidhi1.txt: No such file or directory
daksdutta@dakshita:~/Desktop/OS_LAB1$ touch nidhi1.txt,nidhi2.txt,nidhi3.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ ls
nidhi1.txt,nidhi2.txt,nidhi3.txt  nidhi.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ cat nidhi.txt
this is operating system lab 1 work assignment given by scope faculty.
os works as interface between user and application program.
os manages all the resources of system.
os is also known as resource manager.
^C
daksdutta@dakshita:~/Desktop/OS_LAB1$ cat nidhi.txt
this is operating system lab 1 work assignment given by scope faculty.
os works as interface between user and application program.
os manages all the resources of system.
os is also known as resource manager.
daksdutta@dakshita:~/Desktop/OS_LAB1$ cp nidhi.txt nidhi1.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ cat nidhi1.txt
this is operating system lab 1 work assignment given by scope faculty.
os works as interface between user and application program.
os manages all the resources of system.
os is also known as resource manager.
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

## 2. cd - change directory

**cd** is used to change from one directory to another.

### Syntax :-

**cd dir1**

changes directory so that dir1 is your new current directory. dir1 may be either the full pathname of the directory, or its pathname relative to the current directory.

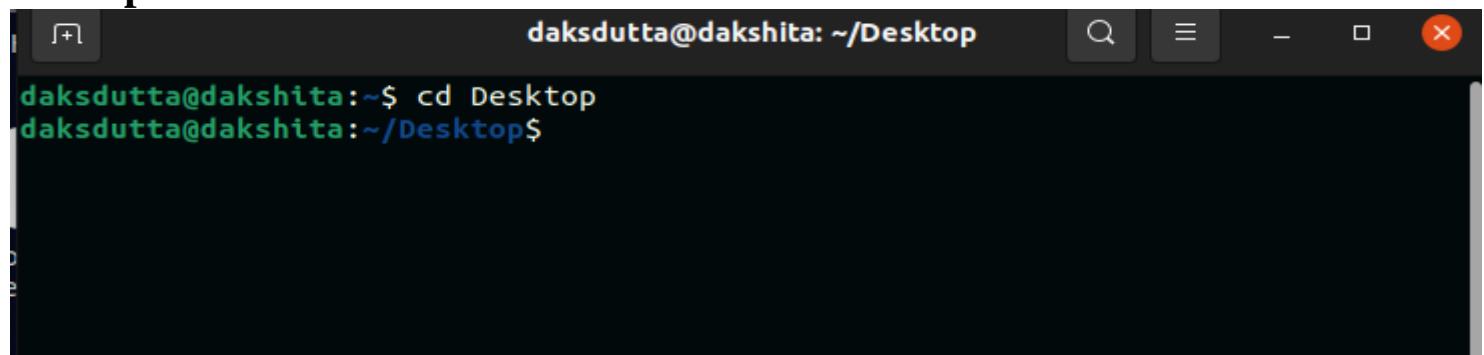
**cd**

changes directory to your home directory.

**cd ..**

moves to the parent directory of your current directory.

### Description :-



A screenshot of a terminal window titled "daksdutta@dakshita: ~/Desktop". The window shows the command "cd Desktop" being typed and then executed, resulting in the directory being changed to the user's desktop folder. The terminal has a dark theme with white text and a black background.

```
daksdutta@dakshita:~$ cd Desktop
daksdutta@dakshita:~/Desktop$
```

## 3. chmod - change the permissions on a file or directory

**chmod** alters the permissions on files and directories using either symbolic or octal numeric codes. The symbolic codes are given here:-

**u** user + to add a permission **r** read

**g** group - to remove a permission **w** write

**o** other = to assign a permission explicitly **x** execute (for files),access (for directories)

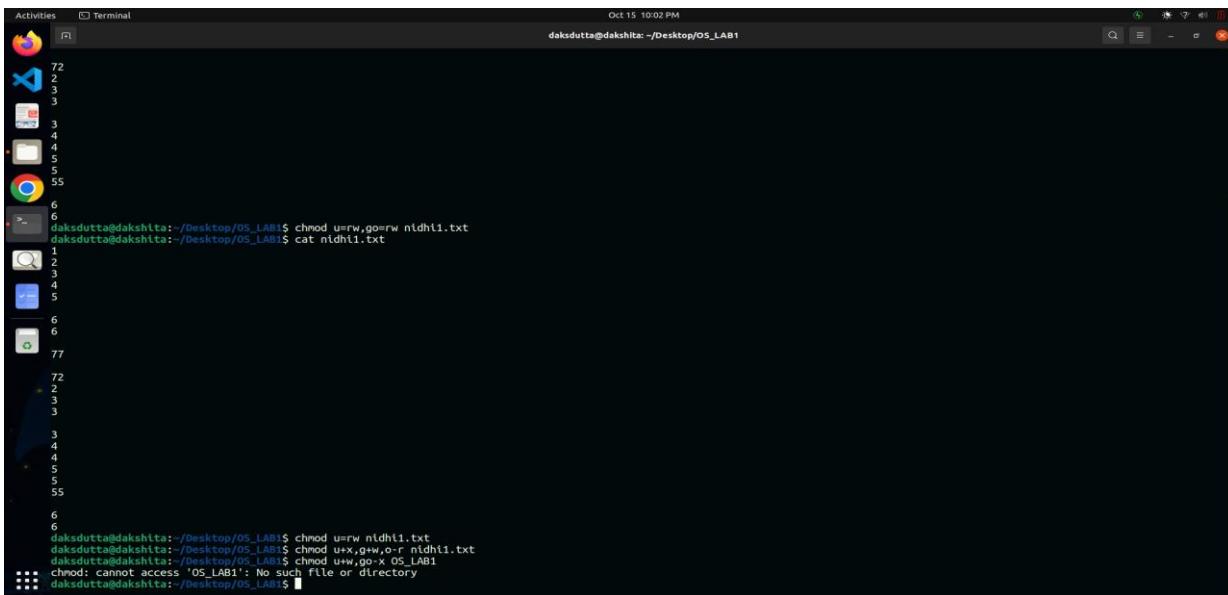
### Syntax :-

**chmod u=rw file1**

**chmod u+x,g+w,o-r file1**

**chmod u+w,go-x dir1**

### Description :-



The screenshot shows a terminal window with a dark background. At the top, it displays the date and time as Oct 15, 10:02 PM. The prompt indicates the user is daksdutta@dakshita: ~/Desktop/OS\_LAB1. The terminal output shows several commands being run:

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ chmod u=rw,g=rw nidl1.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ cat nidl1.txt
1
2
3
4
5
55
6
6
daksdutta@dakshita:~/Desktop/OS_LAB1$ chmod u=rw nidl1.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ chmod u+x,g+w,o-r nidl1.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ chmod u+w,g=r OS_LAB1
chmod: cannot access 'OS_LAB1': No such file or directory
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

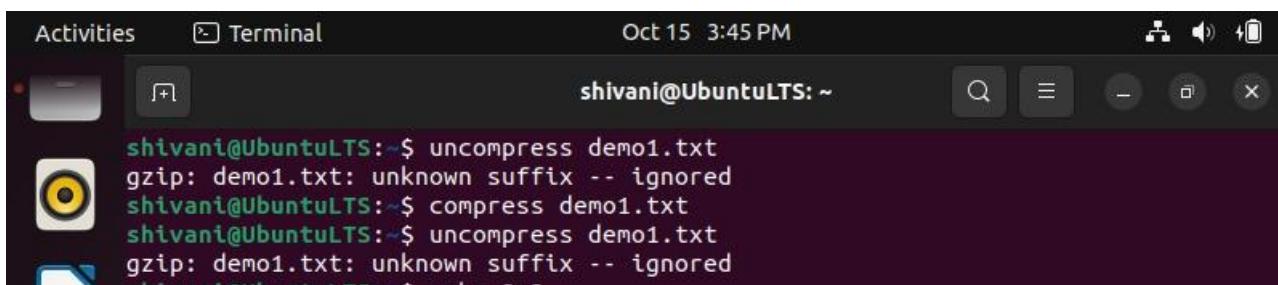
#### 4. compress - compress a file

**compress** reduces the size of named files, replacing them with files of the same name extended by **.Z**. The amount of space saved by compression varies. If no saving of space would occur, then the file will not be altered.

##### Syntax :-

```
compress file1
compress -v file2
```

##### Description :-



The screenshot shows a terminal window with a dark background. At the top, it displays the date and time as Oct 15 3:45 PM. The prompt indicates the user is shivani@UbuntuLTS: ~. The terminal output shows the following sequence of commands:

```
shivani@UbuntuLTS:~$ uncompress demo1.txt
gzip: demo1.txt: unknown suffix -- ignored
shivani@UbuntuLTS:~$ compress demo1.txt
shivani@UbuntuLTS:~$ uncompress demo1.txt
gzip: demo1.txt: unknown suffix -- ignored
```

#### 5. cp - copy a file

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

##### Syntax :-

```
cp file1 file2
cp file3 file4 dir1
cp -r dir2 dir3
```

## Description :-

```
os is also known as resource manager.  
^C  
daksdutta@dakshita:~/Desktop/OS_LAB1$ cat nidhi.txt  
this is operating system lab 1 work assignment given by scope faculty.  
os works as interface between user and application program.  
os manages all the resources of system.  
os is also known as resource manager.  
daksdutta@dakshita:~/Desktop/OS_LAB1$ cp nidhi.txt nidhi1.txt  
daksdutta@dakshita:~/Desktop/OS_LAB1$ cat nidhi1.txt  
this is operating system lab 1 work assignment given by scope faculty.  
os works as interface between user and application program.  
os manages all the resources of system.  
os is also known as resource manager.  
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

## 6. date - display the current date and time

**date** returns information on the current date and time in the format shown below:-

Tue Mar 25 15:21:16 GMT 1997

It is possible to alter the format of the output from date. For example, using the command line  
**date 'The date is d/m/y, and the time is H:M:S.'**

at exactly 3.10pm on 14th December 1997, would produce the output

The date is 14/12/97, and the time is 15:10:00.

### Syntax :-

Date

### Description :-

```
daksdutta@dakshita:~/Desktop$ date  
Saturday 15 October 2022 09:34:51 PM IST  
daksdutta@dakshita:~/Desktop$
```

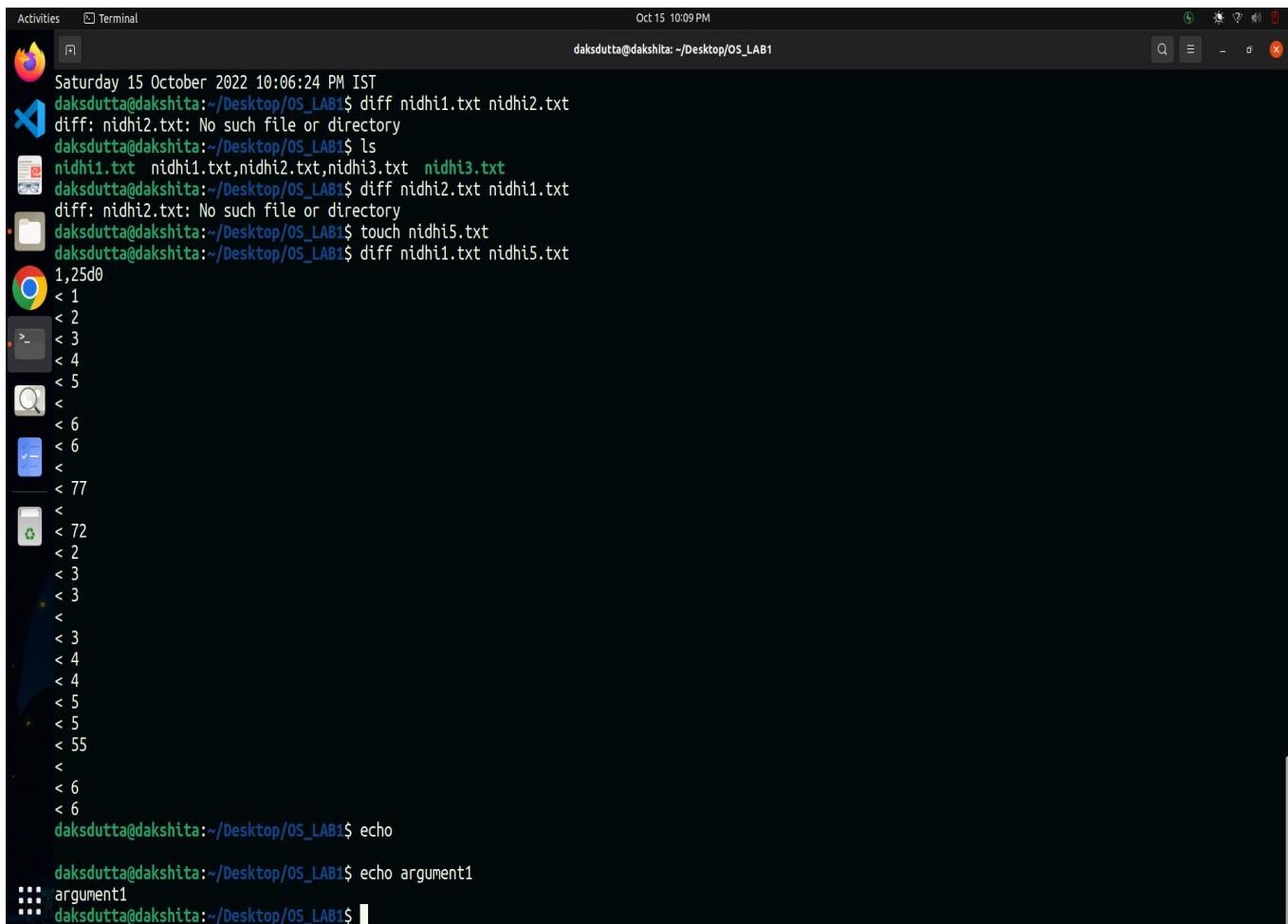
## 7. diff - display differences between text files

**diff** file1 file2 reports line-by-line differences between the text files file1 and file2. The default output will contain lines such as **n1 a n2,n3** and **n4,n5 c n6,n7**, (where n1 a n2,n3 means that file2 has the extra lines n2 to n3 following the line that has the number n1 in file1, and **n4,n5 c n6,n7** means that lines n4 to n5 in **file1** differ from lines n6 to n7 in file2). After each such line, **diff** prints the relevant lines from the text files, with < in front of each line from file1 and > in front of each line from file2.

### Syntax :-

**diff** file1 file2

## Description :-



The screenshot shows a terminal window titled "Terminal" with the command-line interface. The session starts with the date and time: Saturday 15 October 2022 10:06:24 PM IST. The user is at the prompt "daksdutta@dakshita:~/Desktop/OS\_LAB1\$". The user runs several commands: "diff nidhi1.txt nidhi2.txt", which outputs "diff: nidhi2.txt: No such file or directory"; "ls", which lists files: "nidhi1.txt" and "nidhi2.txt,nidhi3.txt"; "diff nidhi2.txt nidhi1.txt", which outputs "diff: nidhi2.txt: No such file or directory"; "touch nidhi5.txt", which creates the file "nidhi5.txt"; and "diff nidhi1.txt nidhi5.txt", which outputs "1,25d0< 1< 2< 3< 4< 5< 6< 6< 77< < 72< 2< 3< 3< 3< 3< 4< 4< 4< 5< 5< 55< < 6< 6". Finally, the user runs "echo argument1", which outputs "argument1". The terminal window has a dark theme and includes a dock with various icons.

## 8. echo - echo arguments to the standard output

**echo** echoes given arguments to the standard output, and is generally used in shell programs.

### Syntax :-

**echo** argument1

writes argument1 to the standard output.

## Description :-

```
< 6
< 6
daksdutta@dakshita:~/Desktop/OS_LAB1$ echo

daksdutta@dakshita:~/Desktop/OS_LAB1$ echo argument1
argument1
daksdutta@dakshita:~/Desktop/OS_LAB1$ file nidhi1.txt
```

## 9. file - determine the type of a file

**file** tests named files to determine the categories their contents belong to.

**Syntax :-**

```
    file file1
```

**Description :-**

```
argument1
daksdutta@dakshita:~/Desktop/OS_LAB1$ file nidhi1.txt
nidhi1.txt: ASCII text
daksdutta@dakshita:~/Desktop/OS_LAB1$ find
./.sw0
```

## 10. find - find files of a specified name or type

**find** searches for files in a named directory and all its subdirectories.

**find . -name \*.f -print**

searches the current directory and all its subdirectories for files ending in .f, and writes their names to the standard output. In some versions of Unix the names of the files will only be written out if the **-print** option is used.

**Syntax :-**

```
find /local -name core -user user1 -print
```

**Description :-**

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ find
./.sw0
./nidhi3.txt
./nidhi1.txt
./.swp
./nidhi1.txt,nidhi2.txt,nidhi3.txt
./nidhi5.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

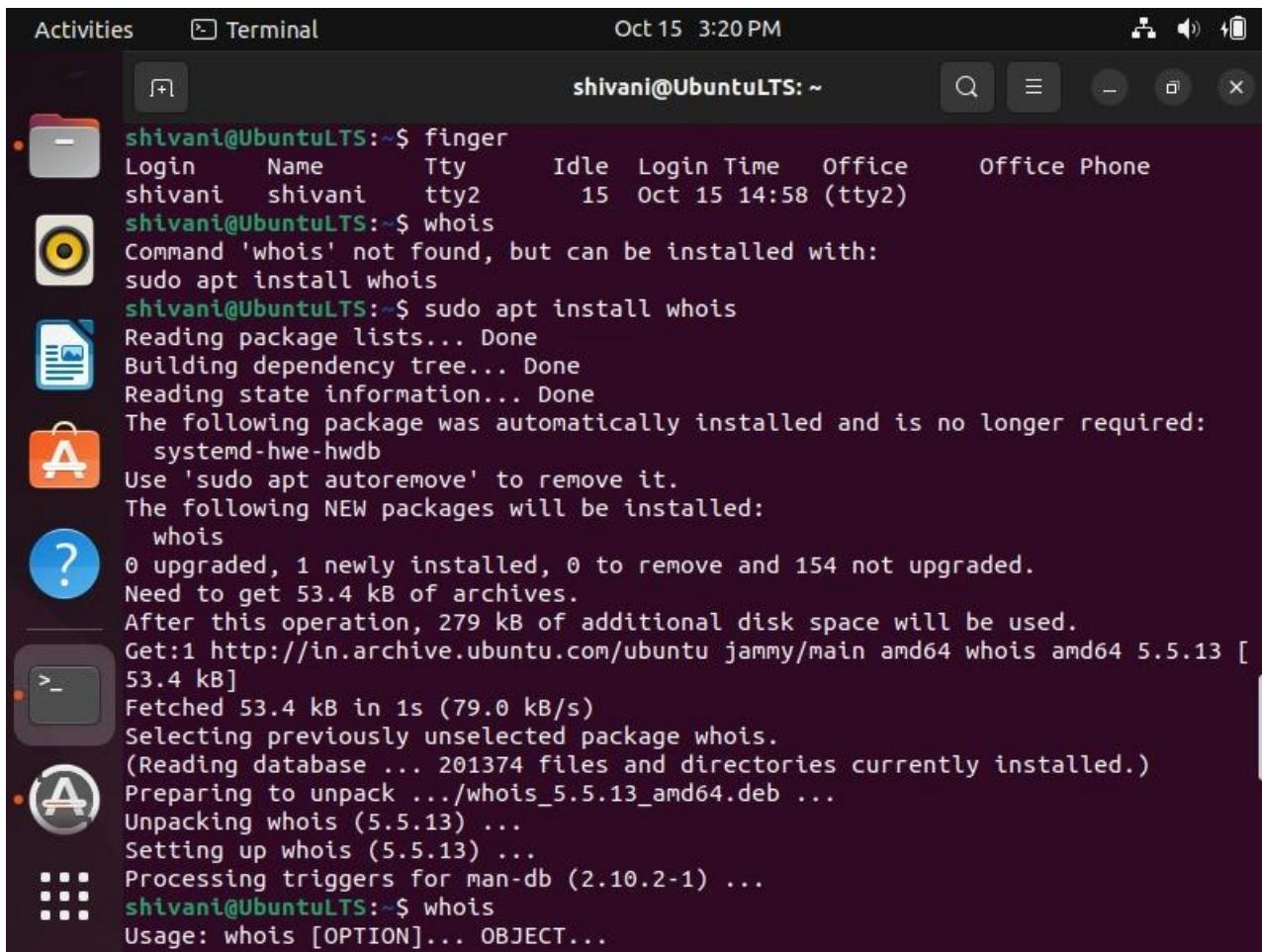
## 11. finger - display information about a user

**finger** can be used to obtain information on users on your own and other machines. **Finger** on its own will give information on all users currently logged onto your machine: their user names, real names, the terminal they are using and its idle time, the time they logged on, and the name of the machine from which they logged on.

**Syntax :-**

```
    finger
```

**Description :-**

A screenshot of the Ubuntu LTS desktop environment. The terminal window is open and shows a session of the Linux command-line interface. The user, shivani, is running several commands: 'finger' (showing login information), 'whois' (which is not installed), 'sudo apt install whois' (installing the package), and finally 'whois' again after installation. The terminal also shows the progress of the package download and unpacking.

```
Activities Terminal Oct 15 3:20 PM shivani@UbuntuLTS: ~
shivani@UbuntuLTS:~$ finger
Login      Name      Tty      Idle  Login Time   Office      Office Phone
shivani    shivani   tty2        15  Oct 15 14:58 (tty2)
shivani@UbuntuLTS:~$ whois
Command 'whois' not found, but can be installed with:
sudo apt install whois
shivani@UbuntuLTS:~$ sudo apt install whois
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  systemd-hwe-hwdb
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  whois
0 upgraded, 1 newly installed, 0 to remove and 154 not upgraded.
Need to get 53.4 kB of archives.
After this operation, 279 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 whois amd64 5.5.13 [53.4 kB]
Fetched 53.4 kB in 1s (79.0 kB/s)
Selecting previously unselected package whois.
(Reading database ... 201374 files and directories currently installed.)
Preparing to unpack .../whois_5.5.13_amd64.deb ...
Unpacking whois (5.5.13) ...
Setting up whois (5.5.13) ...
Processing triggers for man-db (2.10.2-1) ...
shivani@UbuntuLTS:~$ whois
Usage: whois [OPTION]... OBJECT...
```

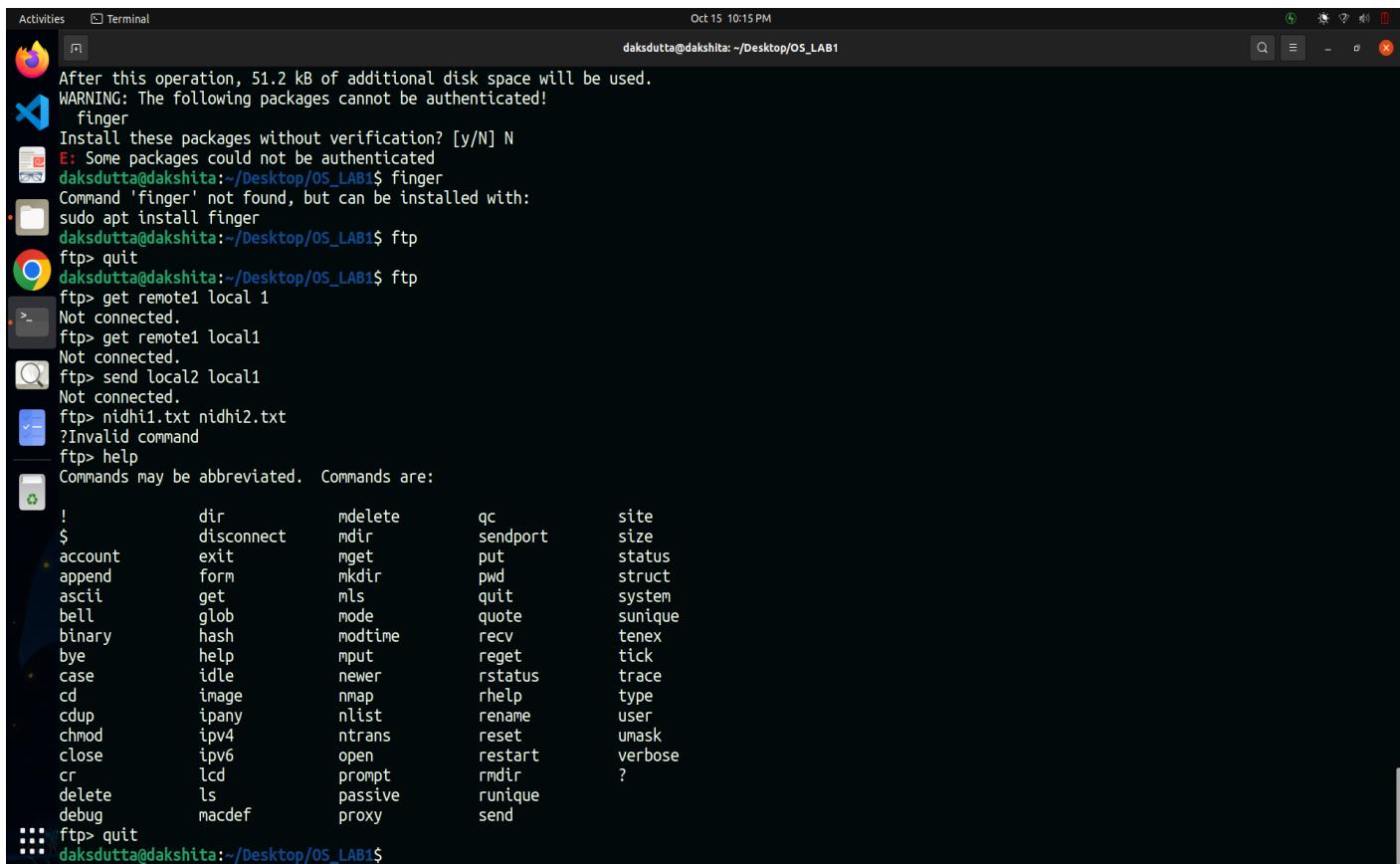
## 12. ftp - file transfer program

ftp is an interactive file transfer program. While logged on to one machine (described as the local machine), **ftp** is used to logon to another machine (described as the remote machine) that files are to be transferred to or from. As well as file transfers, it allows the inspection of directory contents on the remote machine. There are numerous options and commands associated with **ftp**, and **man** **ftp** will give details of those.

### Syntax :-

**ftp**

### Description :-



A screenshot of a Linux desktop environment, specifically Ubuntu, showing a terminal window titled "Terminal". The terminal window has a dark theme and displays a command-line session. The session starts with a warning about installing unsigned packages, followed by a finger command which fails because the package is not found. It then shows an ftp session where files are transferred between local and remote hosts. Finally, it lists various commands available in the current context.

```
After this operation, 51.2 kB of additional disk space will be used.
WARNING: The following packages cannot be authenticated!
  finger
Install these packages without verification? [y/N] N
E: Some packages could not be authenticated
daksdutta@dakshita:~/Desktop/OS_LAB1$ finger
Command 'finger' not found, but can be installed with:
sudo apt install finger
daksdutta@dakshita:~/Desktop/OS_LAB1$ ftp
ftp> quit
daksdutta@dakshita:~/Desktop/OS_LAB1$ ftp
ftp> get remote1 local 1
Not connected.
ftp> get remote1 local1
Not connected.
ftp> send local2 local1
Not connected.
ftp> nihil1.txt nihil2.txt
?Invalid command
ftp> help
Commands may be abbreviated. Commands are:
!
$      dir      mdelete    qc      site
account  disconnect mdir      sendport  size
append   exit      mget      put      status
ascii    get       mkdir     pwd      struct
bell     glob      mode      quit     system
binary   hash      modtime   quote    sunique
bye     help      mput      recv    tenex
case    idle      newer     rstatus   trace
cd      image     nmap      rhelp    type
cdup   ipany     nlist     rename   user
chnod  ipv4      ntrans    reset    umask
close   ipv6      open      restart  verbose
cr      lcd       prompt   passive  runique
delete  ls        proxy    send
debug   macdef
ftp> quit
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

### 13. grep - searches files for a specified string or expression

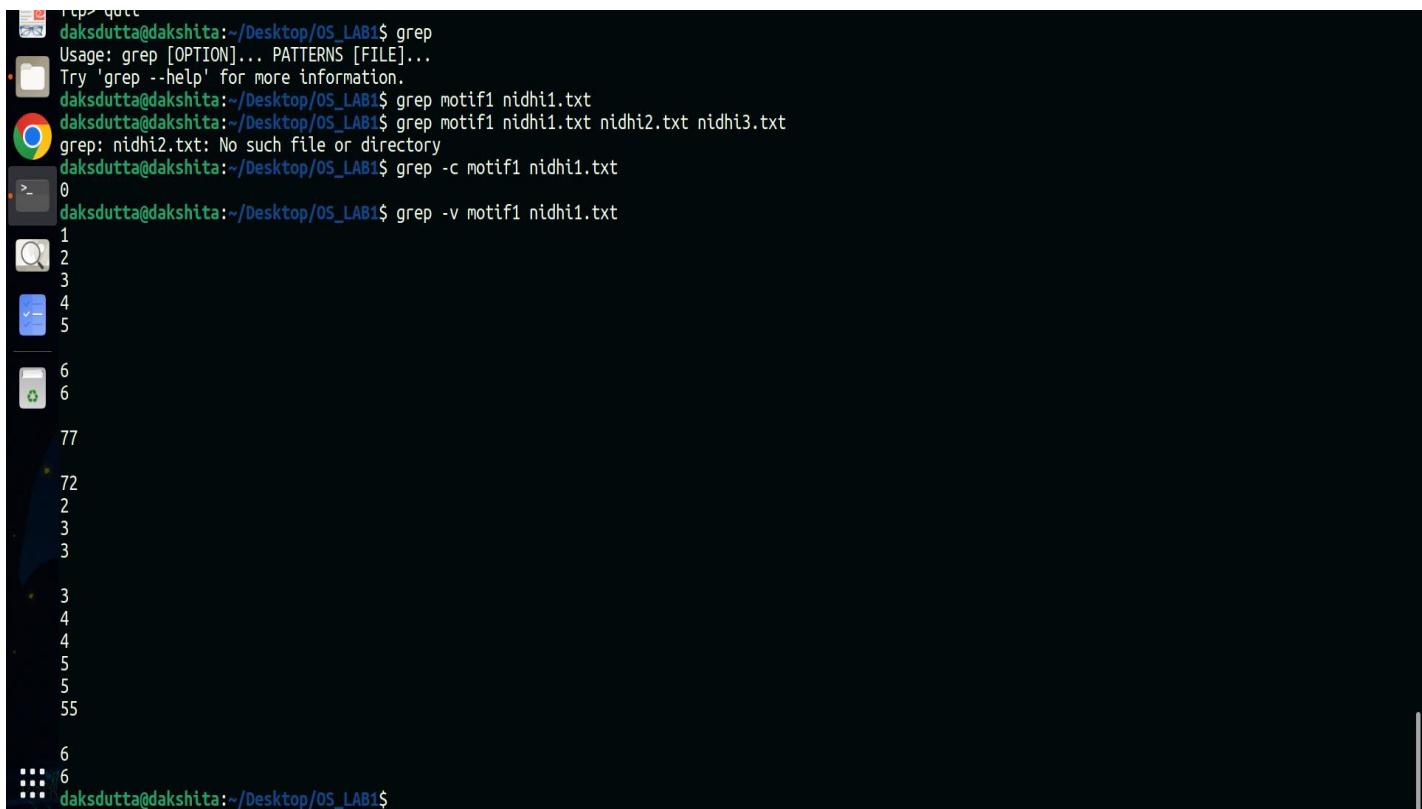
grep searches for lines containing a specified pattern and, by default, writes them to the standard output.

**Syntax :-**

```
grep motif1 file1
grep motif1 file1 file2 ... filen
grep -c motif1 file1

grep -v motif1 file1
```

**Description :-**



```
dakshita@dakshita:~/Desktop/OS_LAB1$ grep
Usage: grep [OPTION]... PATTERN [FILE]...
Try 'grep --help' for more information.
dakshita@dakshita:~/Desktop/OS_LAB1$ grep motif1 nidhi1.txt
dakshita@dakshita:~/Desktop/OS_LAB1$ grep motif1 nidhi1.txt nidhi2.txt nidhi3.txt
grep: nidhi2.txt: No such file or directory
dakshita@dakshita:~/Desktop/OS_LAB1$ grep -c motif1 nidhi1.txt
0
dakshita@dakshita:~/Desktop/OS_LAB1$ grep -v motif1 nidhi1.txt
1
2
3
4
5
6
6
77
72
2
3
3
3
4
4
5
5
55
6
6
dakshita@dakshita:~/Desktop/OS_LAB1$
```

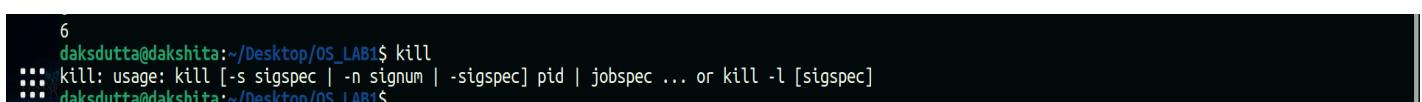
## 14. kill - kill a process

To kill a process using **kill** requires the process id (PID). This can be found by using **ps**. Suppose the PID is 3429, then **kill 3429** should kill the process.

**Syntax :-**

```
kill
```

**Description :-**



```
6
dakshita@dakshita:~/Desktop/OS_LAB1$ kill
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill -l [sigspec]
dakshita@dakshita:~/Desktop/OS_LAB1$
```

## 15. lpr - print out a file

**lpr** is used to send the contents of a file to a printer. If the printer is a laserwriter, and the file contains PostScript, then the PostScript will be interpreted and the results of that printed out.

**Syntax :-**

```
lpr -Pprinter1 file1
lpq -Pprinter1
```

**Description :-**

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ lpr -Pprinter1 nidhi1.txt
lpr: Error - The printer or class does not exist.
daksdutta@dakshita:~/Desktop/OS_LAB1$ lpq -Pprinter1
lpq: Unknown destination "printer1".
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

## 16. ls - list names of files in a directory

ls lists the contents of a directory, and can be used to obtain information on the files and directories within it.

### Syntax :-

```
ls dir1
ls -a dir1
ls -l file1
ls -l dir1
ls -ld dir1
```

### Description :-

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ ls
nidhi1.txt  nidhi1.txt,nidhi2.txt,nidhi3.txt  nidhi3.txt  nidhi5.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

## 17. man - display an on-line manual page

man displays on-line reference manual pages.

### Syntax :-

```
man command1
man -k keyword
man -Mpath command1
```

### Description

:-

```
nidhi1.txt  nidhi1.txt,nidhi2.txt,nidhi3.txt  nidhi3.txt  nidhi5.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ man
What manual page do you want?
For example, try 'man man'.
daksdutta@dakshita:~/Desktop/OS_LAB1$ man man
```

The screenshot shows a terminal window titled 'Terminal' with the command 'man(1)' entered. The output is as follows:

```
daksdutta@dakshita: ~/Desktop/OS_LAB1
Oct 15 10:22 PM
Manual pager utils
NAME
man - an interface to the system reference manuals
SYNOPSIS
man [man options] [[section] page ...] ...
man -k [apropos options] regexp ...
man -K [man options] [section] term ...
man -f [whatis options] page ...
man -l [man options] file ...
man -w|-W [man options] page ...
DESCRIPTION
man is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed. A section, if provided, will direct man to look only in that section of the manual. The default action is to search in all of the available sections following a pre-defined order (see DEFAULTS), and to show only the first page found, even if page exists in several sections.

The table below shows the section numbers of the manual followed by the types of pages they contain.

1 Executable programs or shell commands
2 System calls (functions provided by the kernel)
Manual page man(1) line 1 (press h for help or q to quit)
```

## 18. mkdir - make a directory

**mkdir** is used to create new directories. In order to do this you must have write permission in the parent directory of the new directory.

### Syntax :-

```
mkdir newdir
mkdir -p
mkdir -p dir1/dir2/newdir
```

### Description :-

```
daksdutta@dakshita:~$ cd Desktop
daksdutta@dakshita:~/Desktop$ mkdir OS_LAB
daksdutta@dakshita:~/Desktop$ mkdir OS_LAB1
daksdutta@dakshita:~/Desktop$ mkdir OS_LAB2
daksdutta@dakshita:~/Desktop$ ls
nidhi  OS_LAB  OS_LAB1  OS_LAB2  xampp-control-panel.desktop
daksdutta@dakshita:~/Desktop$ █
```

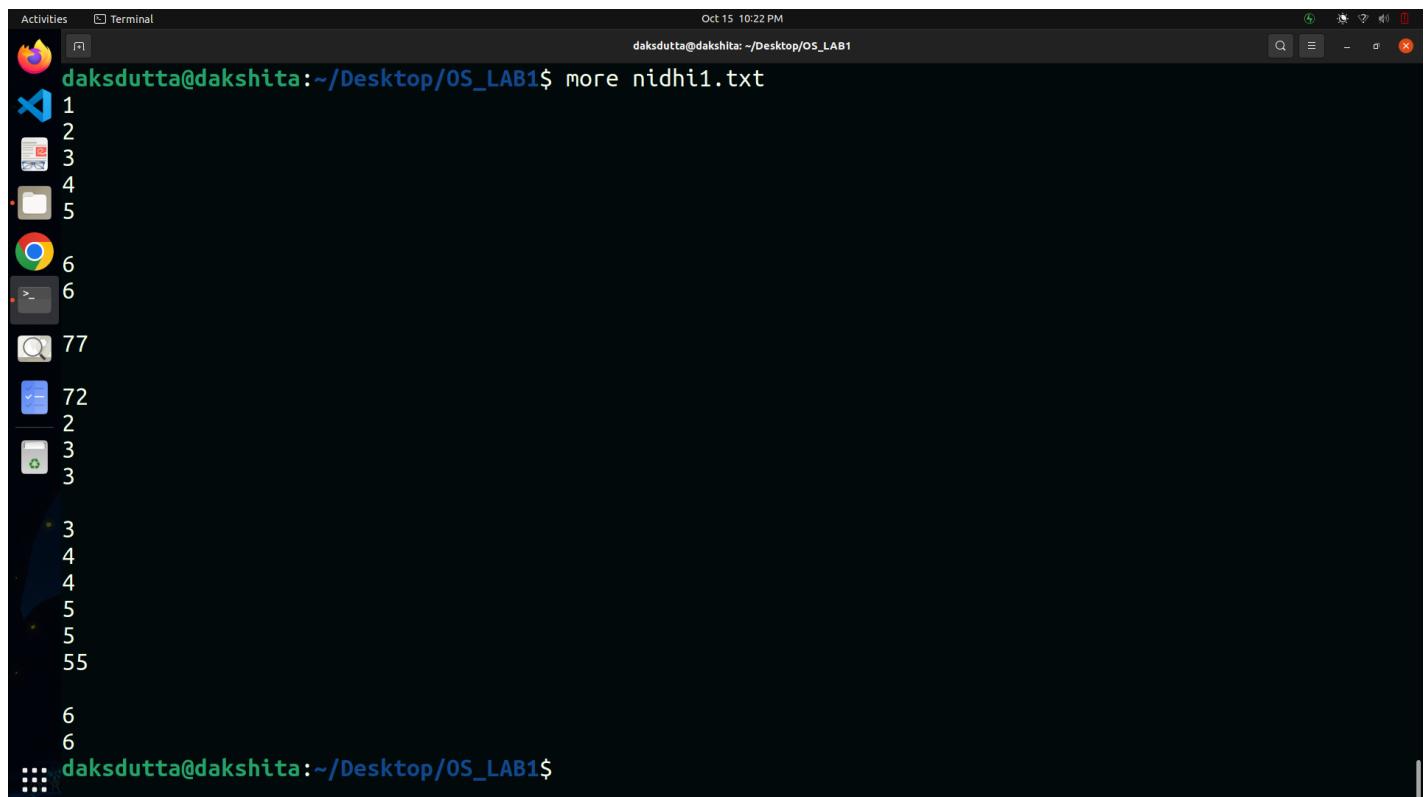
## 19. more - scan through a text file page by page

more displays the contents of a file on a terminal one screenful at a time.

**Syntax :-**

```
more file1
more -n file1
```

**Description :-**



The screenshot shows a terminal window titled 'Terminal' with the command 'more nidhi1.txt' entered. The output displays the contents of the file 'nidhi1.txt' in a paginated format. The terminal window is part of a desktop environment with a dark theme. The desktop background shows various icons for applications like a browser, file manager, and terminal. The terminal window has a scroll bar on the right side.

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ more nidhi1.txt
1
2
3
4
5
6
6
77
72
2
3
3
3
4
4
5
5
55
6
6
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

## 20. mv - move or rename files or directories

**mv** is used to change the name of files or directories, or to move them into other directories. **mv** cannot move directories from one file-system to another, so, if it is necessary to do that, use **cp** instead.

### Syntax :-

```
mv file1 file2
mv dir1 dir2
mv file1 file2 dir3
```

### Description :-

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ ls
nidhi1.txt nidhi1.txt,nidhi2.txt,nidhi3.txt nidhi.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ mv nidhi.txt nidhi4.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ ls
nidhi1.txt nidhi1.txt,nidhi2.txt,nidhi3.txt nidhi4.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

## 21. nice - change the priority at which a job is being run

**nice** causes a command to be run at a lower than usual priority. **nice** can be particularly useful when running a long program that could cause annoyance if it slowed down the execution of other users' commands. An example of the use of **nice** is

### Syntax :-

```
nice compress file1
```

### Description :-

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ nice
0
daksdutta@dakshita:~/Desktop/OS_LAB1$ nice compress nidhi1.txt
nice: 'compress': No such file or directory
daksdutta@dakshita:~/Desktop/OS_LAB1$ nice nidhi1.txt
nice: 'nidhi1.txt': No such file or directory
daksdutta@dakshita:~/Desktop/OS_LAB1$ nice
0
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

## 22. passwd - change your password

Use **passwd** when you wish to change your password. You will be prompted once for your current password, and twice for your new password. Neither password will be displayed on the screen.

### Syntax :-

```
passwd
```

### Description :-

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ passwd
Changing password for daksdutta.
Current password:
New password:
Retype new password:
passwd: password updated successfully
daksdutta@dakshita:~/Desktop/OS_LAB1$ passwd
Changing password for daksdutta.
Current password:
New password:
Retype new password:
passwd: password updated successfully
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

### 23. ps - list processes

**ps** displays information on processes currently running on your machine. This information includes the process id, the controlling terminal (if there is one), the cpu time used so far, and the name of the command being run.

**Syntax :-**

```
ps
ps -a
```

**Description :-**

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ ps
  PID TTY          TIME CMD
 6233 pts/0    00:00:00 bash
 7210 pts/0    00:00:00 ps
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

### 24. pwd - display the name of your current directory

The command **pwd** gives the full pathname of your current directory.

### Syntax :-

pwd

### Description :-

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ pwd
/home/daksdutta/Desktop/OS_LAB1
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

## 25. quota - disk quota and usage

**quota** gives information on a user's disk space quota and usage.

### Syntax :-

quota

## 26. rm - remove files or directories

**rm** is used to remove files. In order to remove a file you must have write permission in its directory, but it is not necessary to have read or write permission on the file itself.

### Syntax :-

```
rm file1
rm -i file1
rm -r dir1
```

### Description :-

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ ls
nidhi1.txt  nidhi2.txt  nidhi3.txt  nidhi5.txt  os
daksdutta@dakshita:~/Desktop/OS_LAB1$ rm os
rm: cannot remove 'os': Is a directory
daksdutta@dakshita:~/Desktop/OS_LAB1$ rm nidhi5.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ ls
nidhi1.txt  nidhi2.txt  nidhi3.txt  os
daksdutta@dakshita:~/Desktop/OS_LAB1$
```

## 27. rmdir - remove a directory

**rmdir** removes named empty directories. If you need to delete a non-empty directory **rm -r** can be used instead.

### Syntax :-

rmdir exdir

## Description :-

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ ls
nidhi1.txt  nidhi2.txt  nidhi3.txt  os
daksdutta@dakshita:~/Desktop/OS_LAB1$ rmdir os
daksdutta@dakshita:~/Desktop/OS_LAB1$ ls
nidhi1.txt  nidhi2.txt  nidhi3.txt
```

## 28. sort - sort and collate lines

The command **sort** sorts and collates lines in files, sending the results to the standard output. If no file names are given, **sort** acts on the standard input. By default, **sort** sorts lines using a character bycharacter comparison, working from left to right, and using the order of the ASCII character set.

### Syntax :-

```
sort -d
sort -r
sort -n
```

## Description :-

```
daksdutta@dakshita:~/Desktop/OS_LAB1$ sort
-d
^C
daksdutta@dakshita:~/Desktop/OS_LAB1$ sort -d
1
2
3^C
daksdutta@dakshita:~/Desktop/OS_LAB1$ sort -n
hjkll
^C
daksdutta@dakshita:~/Desktop/OS_LAB1$ █
```

## 29. talk - talk to another user

**talk** copies lines from your terminal to another user's terminal.

### Syntax :-

```
talk user1
talk user1@sole
talk user2@carp
```

## 30. wc - display a count of lines, words and characters

**wc** counts the number of lines, words, and characters in files. If no filename is given, **wc** will count the standard input instead.

**Syntax :-**

**wc** file1

**Description :-**

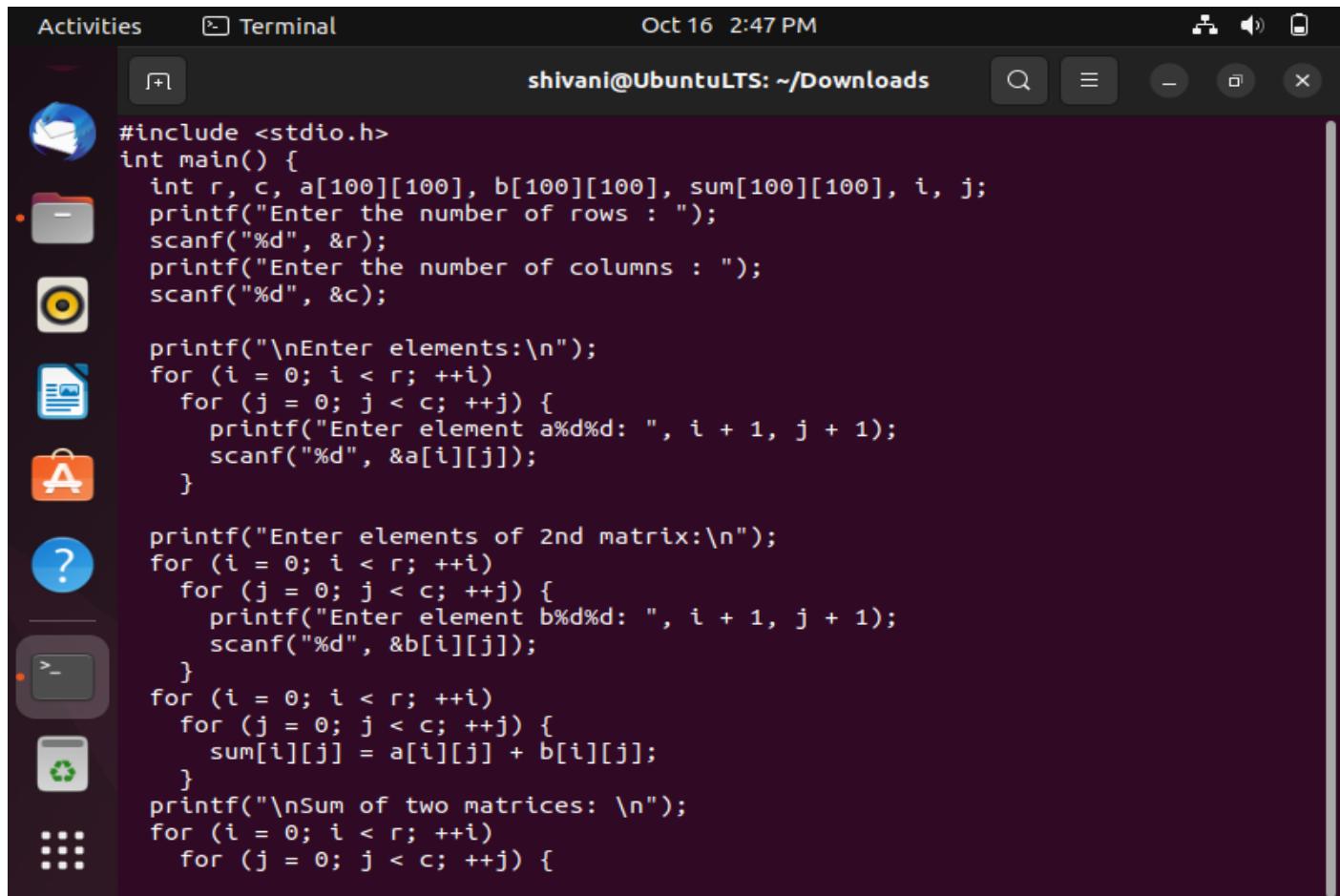
```
daksdutta@dakshita:~/Desktop/OS_LAB1$ wc nidhi1.txt
25 20 48 nidhi1.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ wc nidhi2.txt
0 0 0 nidhi2.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ wc nidhi3.txt
25 20 48 nidhi3.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ wc nidhi4.txt
wc: nidhi4.txt: No such file or directory
daksdutta@dakshita:~/Desktop/OS_LAB1$ wc -l nidhi1.txt
25 nidhi1.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ wc -w nidhi1.txt
20 nidhi1.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ wc -c nidhi1.txt
48 nidhi1.txt
daksdutta@dakshita:~/Desktop/OS_LAB1$ █
```

## LAB – 2

29-09-2022

### Writing 5 c program in LINUX using vi editor.

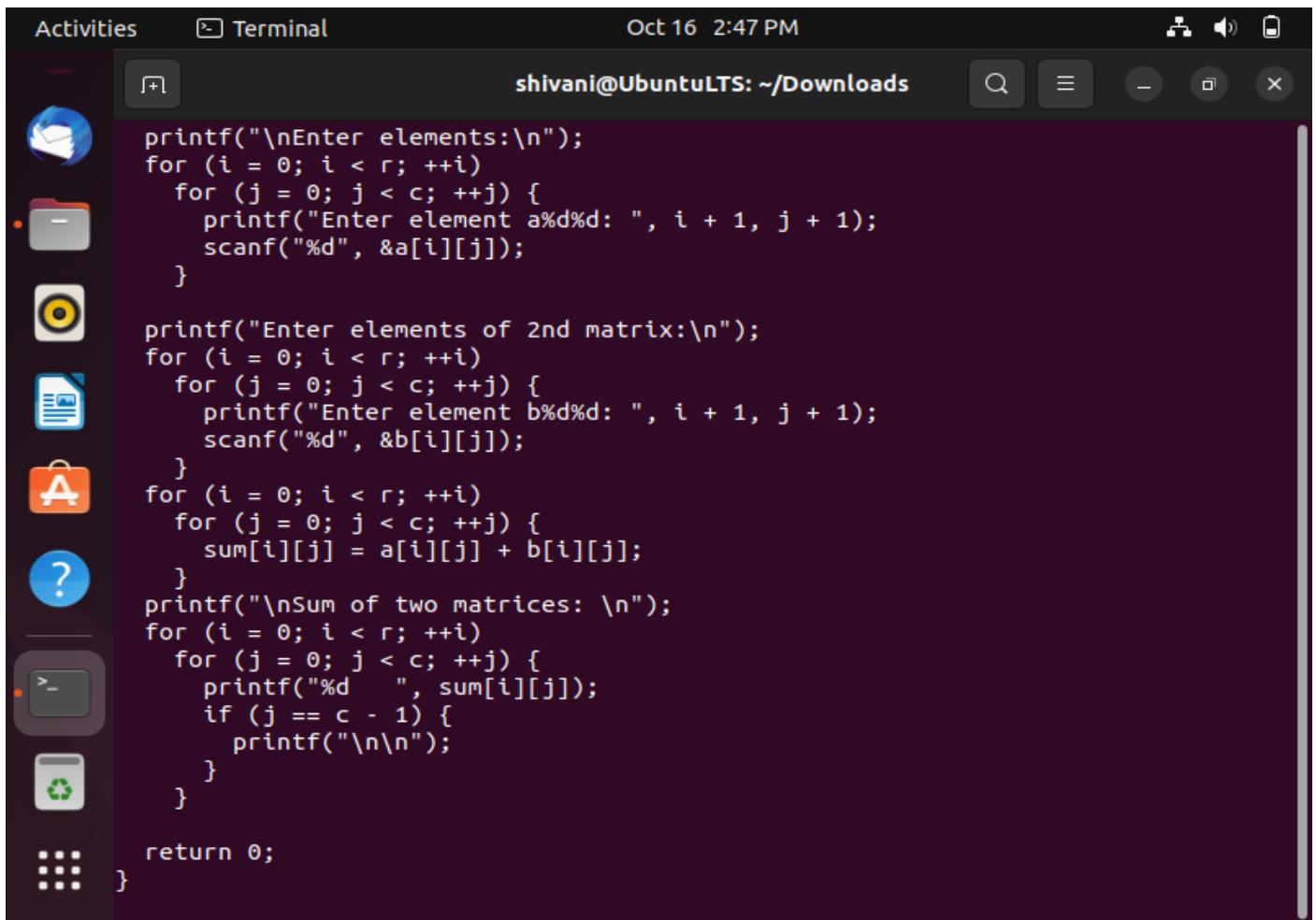
#### 1. write a c program for addition of two matrices.



```
#include <stdio.h>
int main() {
    int r, c, a[100][100], b[100][100], sum[100][100], i, j;
    printf("Enter the number of rows : ");
    scanf("%d", &r);
    printf("Enter the number of columns : ");
    scanf("%d", &c);

    printf("\nEnter elements:\n");
    for (i = 0; i < r; ++i)
        for (j = 0; j < c; ++j) {
            printf("Enter element a%d%d: ", i + 1, j + 1);
            scanf("%d", &a[i][j]);
        }

    printf("Enter elements of 2nd matrix:\n");
    for (i = 0; i < r; ++i)
        for (j = 0; j < c; ++j) {
            printf("Enter element b%d%d: ", i + 1, j + 1);
            scanf("%d", &b[i][j]);
        }
    for (i = 0; i < r; ++i)
        for (j = 0; j < c; ++j)
            sum[i][j] = a[i][j] + b[i][j];
    printf("\nSum of two matrices: \n");
    for (i = 0; i < r; ++i)
        for (j = 0; j < c; ++j) {
```



A screenshot of a Linux desktop environment showing a terminal window. The terminal window title is "Terminal" and the command line shows "shivani@UbuntuLTS: ~/Downloads". The terminal content displays a C program for matrix addition. The program prompts the user to enter the number of rows and columns for two matrices, then asks for elements of both matrices. It calculates the sum and prints it. The terminal window has a dark theme with light-colored text.

```
printf("\nEnter elements:\n");
for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
        printf("Enter element a%d%d: ", i + 1, j + 1);
        scanf("%d", &a[i][j]);
    }

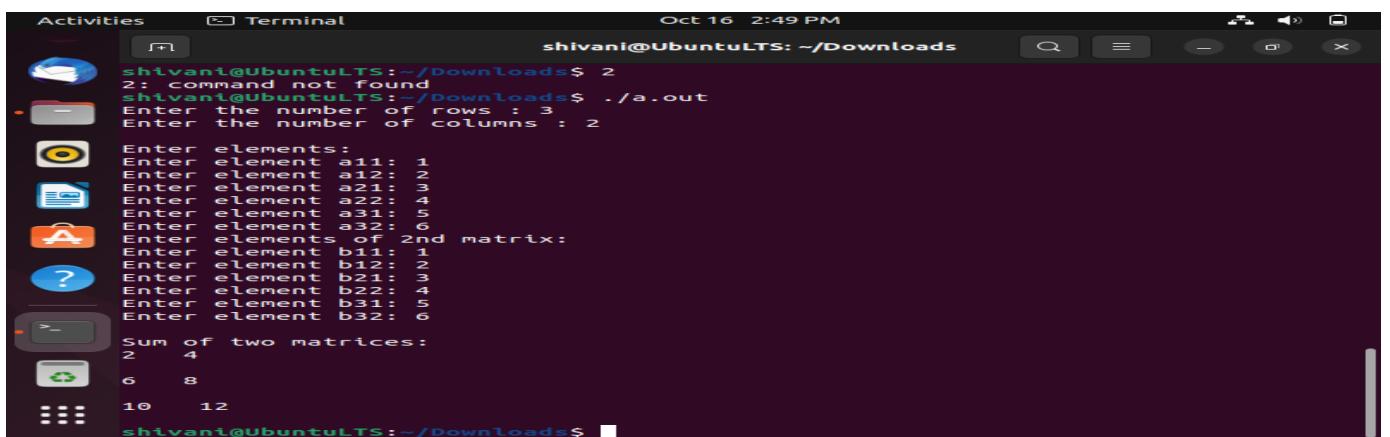
printf("Enter elements of 2nd matrix:\n");
for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
        printf("Enter element b%d%d: ", i + 1, j + 1);
        scanf("%d", &b[i][j]);
    }

for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
        sum[i][j] = a[i][j] + b[i][j];
    }

printf("\nSum of two matrices: \n");
for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
        printf("%d ", sum[i][j]);
        if (j == c - 1) {
            printf("\n\n");
        }
    }

return 0;
}
```

## Output :-



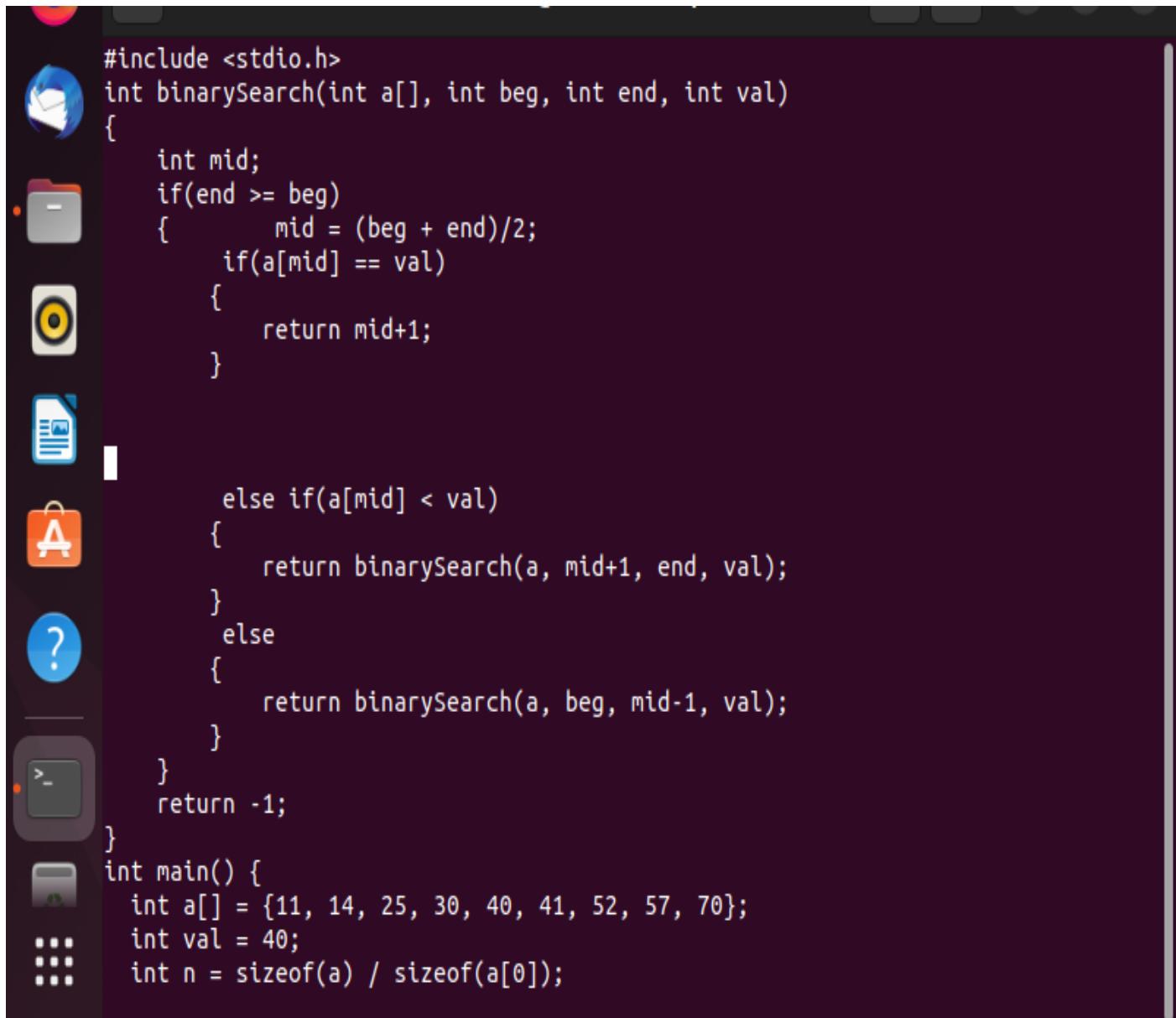
A screenshot of a Linux terminal window titled "Terminal" showing the command line "shivani@UbuntuLTS: ~/Downloads". The terminal output shows the execution of the program. It first asks for the number of rows and columns, then prompts for matrix elements. Finally, it prints the sum of the two matrices. The terminal window has a dark theme with light-colored text.

```
2: command not found
shivani@UbuntuLTS:~/Downloads$ ./a.out
Enter the number of rows : 3
Enter the number of columns : 2

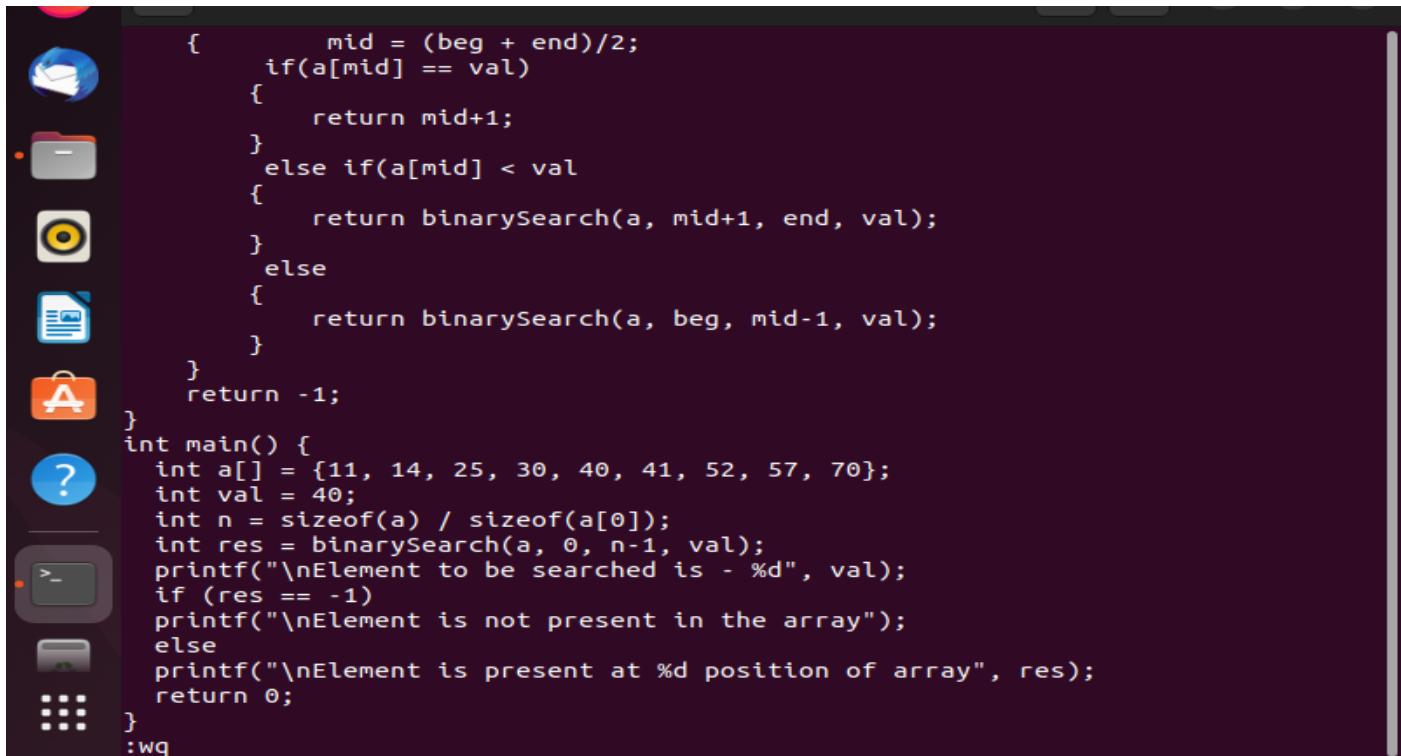
Enter elements:
Enter element a11: 1
Enter element a12: 2
Enter element a21: 3
Enter element a22: 4
Enter element a31: 5
Enter element a32: 6
Enter elements of 2nd matrix:
Enter element b11: 1
Enter element b12: 2
Enter element b21: 3
Enter element b22: 4
Enter element b31: 5
Enter element b32: 6

Sum of two matrices:
2   4
6   8
10  12
```

## 2. write a program to implement binary search using c language.

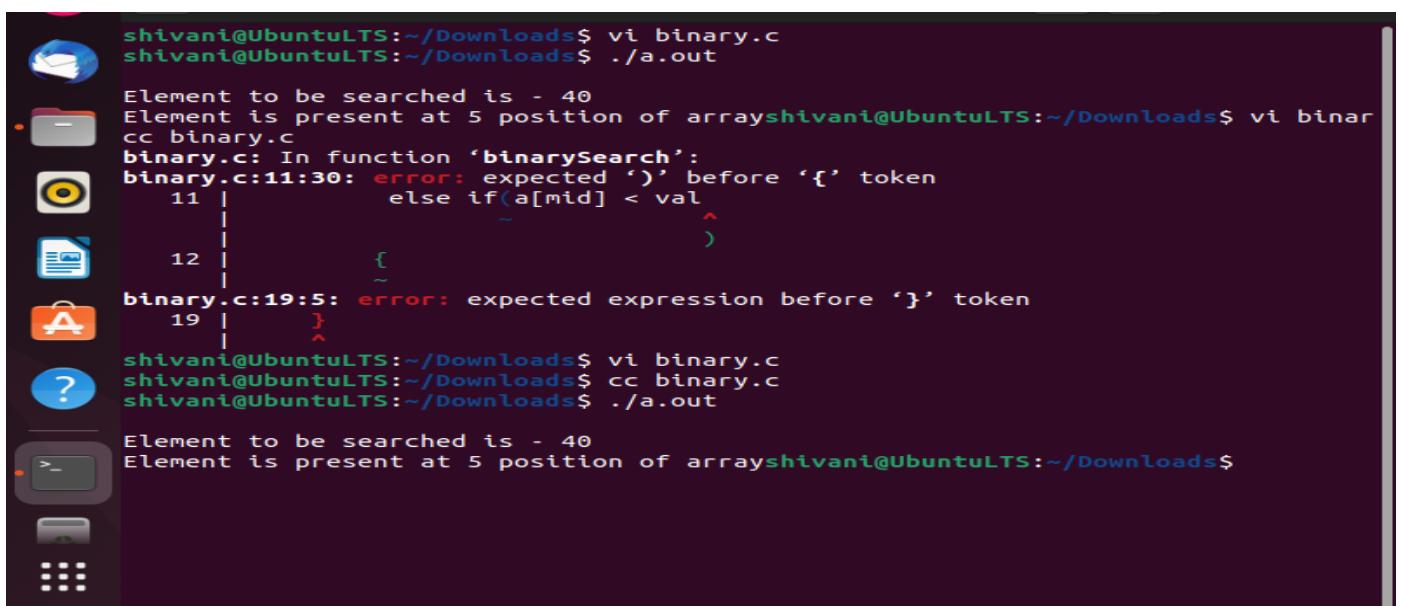


```
#include <stdio.h>
int binarySearch(int a[], int beg, int end, int val)
{
    int mid;
    if(end >= beg)
    {
        mid = (beg + end)/2;
        if(a[mid] == val)
        {
            return mid+1;
        }
        else if(a[mid] < val)
        {
            return binarySearch(a, mid+1, end, val);
        }
        else
        {
            return binarySearch(a, beg, mid-1, val);
        }
    }
    return -1;
}
int main()
{
    int a[] = {11, 14, 25, 30, 40, 41, 52, 57, 70};
    int val = 40;
    int n = sizeof(a) / sizeof(a[0]);
```



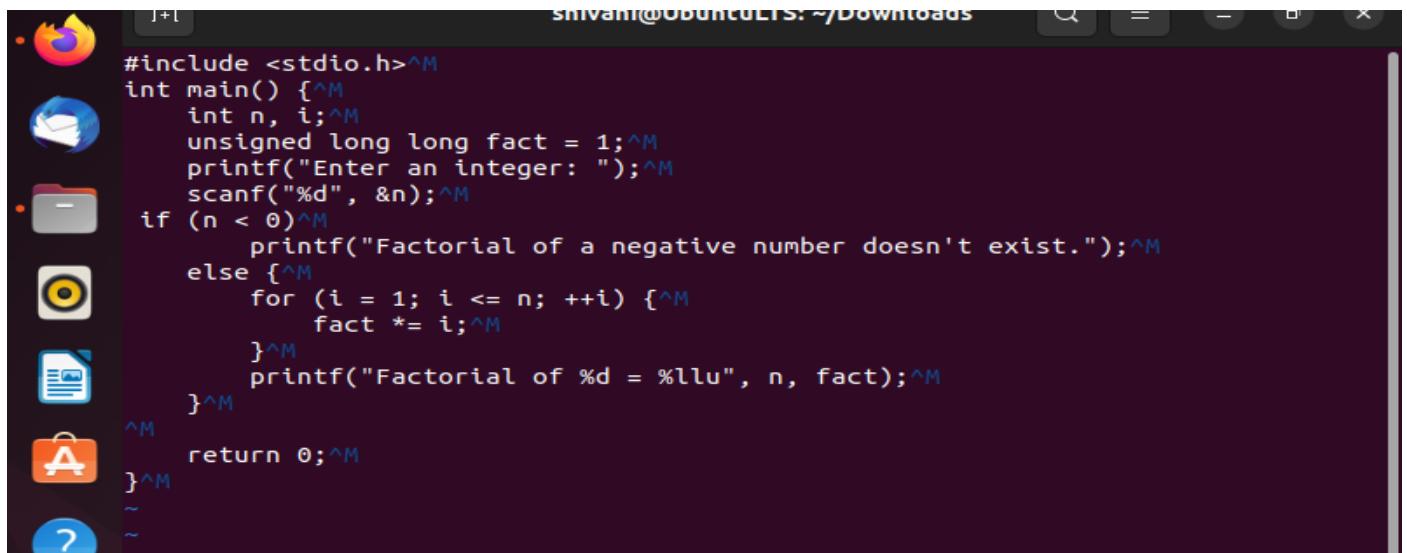
```
{      mid = (beg + end)/2;
    if(a[mid] == val)
    {
        return mid+1;
    }
    else if(a[mid] < val)
    {
        return binarySearch(a, mid+1, end, val);
    }
    else
    {
        return binarySearch(a, beg, mid-1, val);
    }
}
return -1;
}
int main()
{
    int a[] = {11, 14, 25, 30, 40, 41, 52, 57, 70};
    int val = 40;
    int n = sizeof(a) / sizeof(a[0]);
    int res = binarySearch(a, 0, n-1, val);
    printf("\nElement to be searched is - %d", val);
    if (res == -1)
        printf("\nElement is not present in the array");
    else
        printf("\nElement is present at %d position of array", res);
    return 0;
}
:wq
```

## Outout:-



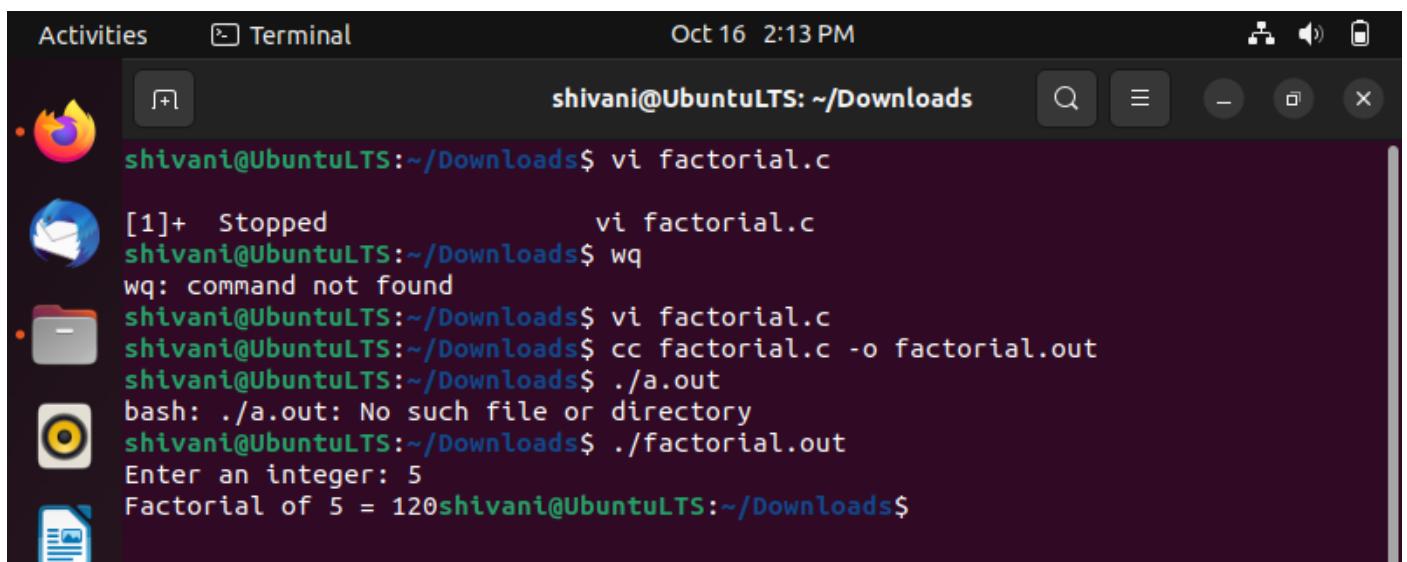
```
shivani@UbuntuLTS:~/Downloads$ vi binary.c
shivani@UbuntuLTS:~/Downloads$ ./a.out
Element to be searched is - 40
Element is present at 5 position of array
cc binary.c
binary.c: In function ‘binarySearch’:
binary.c:11:30: error: expected ‘)’ before ‘{’ token
    11 |         else if(a[mid] < val
                  ^
                  )
    12 |             {
binary.c:19:5: error: expected expression before ‘}’ token
    19 |     }
shivani@UbuntuLTS:~/Downloads$ vi binary.c
shivani@UbuntuLTS:~/Downloads$ cc binary.c
shivani@UbuntuLTS:~/Downloads$ ./a.out
Element to be searched is - 40
Element is present at 5 position of array
shivani@UbuntuLTS:~/Downloads$
```

3. write a program to find factorial of number.



```
#include <stdio.h>
int main() {
    int n, i;
    unsigned long long fact = 1;
    printf("Enter an integer: ");
    scanf("%d", &n);
    if (n < 0)
        printf("Factorial of a negative number doesn't exist.");
    else {
        for (i = 1; i <= n; ++i) {
            fact *= i;
        }
        printf("Factorial of %d = %llu", n, fact);
    }
    return 0;
}
```

## Output :-



```
Activities Terminal Oct 16 2:13 PM
shivani@UbuntuLTS: ~/Downloads$ vi factorial.c
[1]+  Stopped                  vi factorial.c
shivani@UbuntuLTS:~/Downloads$ wq
wq: command not found
shivani@UbuntuLTS:~/Downloads$ vi factorial.c
shivani@UbuntuLTS:~/Downloads$ cc factorial.c -o factorial.out
shivani@UbuntuLTS:~/Downloads$ ./a.out
bash: ./a.out: No such file or directory
shivani@UbuntuLTS:~/Downloads$ ./factorial.out
Enter an integer: 5
Factorial of 5 = 120
shivani@UbuntuLTS:~/Downloads$
```

#### 4. write a program to print Fibonacci series in c language.

The screenshot shows a terminal window titled "Terminal" with the command "Oct 16 2:50 PM". The user is at the prompt "shivani@UbuntuLTS: ~/Downloads". The code displayed is:

```
#include <stdio.h>
int main() {
    int i, n;
    int t1 = 0, t2 = 1;
    int next = t1 + t2;
    printf("Enter the number of terms: ");
    scanf("%d", &n);
    printf("Fibonacci Series: %d, %d, ", t1, t2);

    for (i = 3; i <= n; ++i) {
        printf("%d, ", next);
        t1 = t2;
        t2 = next;
        next = t1+ t2;
    }

    return 0;
}
```

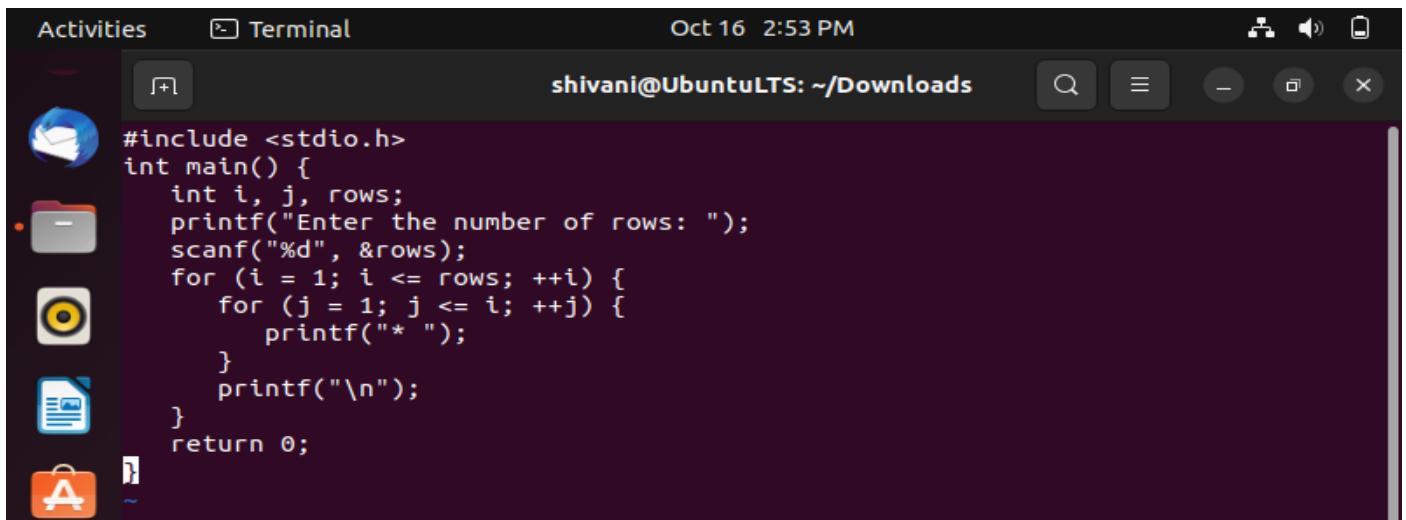
At the bottom of the terminal, it says "fabinocci.c" 18 lines, 337 bytes.

Output :-

The screenshot shows a terminal window titled "Terminal" with the command "Oct 16 2:54 PM". The user is at the prompt "shivani@UbuntuLTS: ~/Downloads". The session shows the following steps:

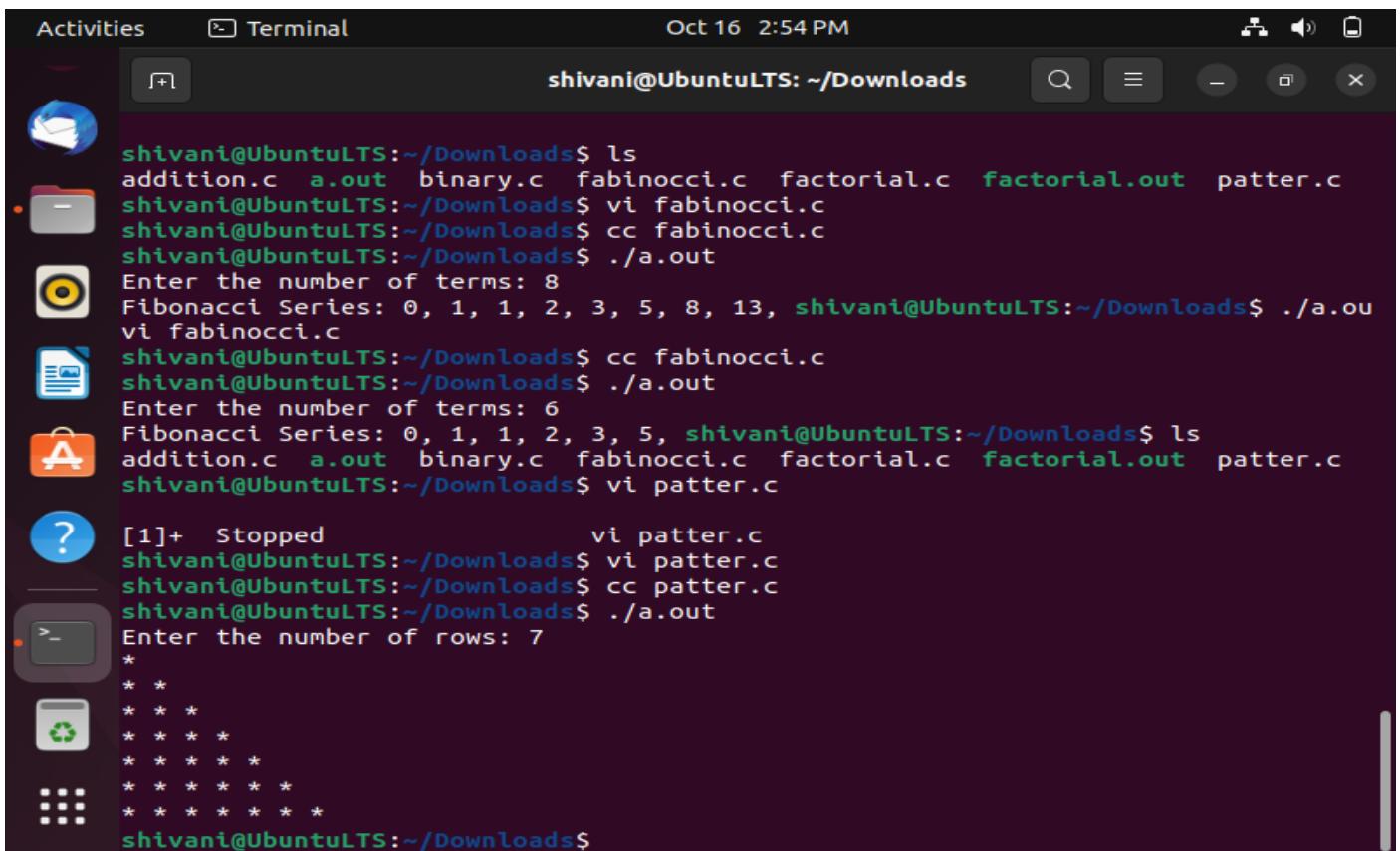
- "ls" command showing files: addition.c, a.out, binary.c, fabinocci.c, factorial.c, factorial.out, patter.c
- "vi fabinocci.c" command to edit the file
- "cc fabinocci.c" command to compile the file
- "./a.out" command to run the program, prompting "Enter the number of terms: 8" and displaying the Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13
- "vi fabinocci.c" command to edit the file again
- "cc fabinocci.c" command to compile the file
- "./a.out" command to run the program, prompting "Enter the number of terms: 6" and displaying the Fibonacci Series: 0, 1, 1, 2, 3, 5
- "ls" command showing files: addition.c, a.out, binary.c, fabinocci.c, factorial.c, factorial.out, patter.c

## 5. write a program to print a specific pattern in c language.



```
#include <stdio.h>
int main() {
    int i, j, rows;
    printf("Enter the number of rows: ");
    scanf("%d", &rows);
    for (i = 1; i <= rows; ++i) {
        for (j = 1; j <= i; ++j) {
            printf("* ");
        }
        printf("\n");
    }
    return 0;
}
```

Output :-



```
shivani@UbuntuLTS:~/Downloads$ ls
addition.c a.out binary.c fabinocci.c factorial.c factorial.out patter.c
shivani@UbuntuLTS:~/Downloads$ vi fabinocci.c
shivani@UbuntuLTS:~/Downloads$ cc fabinocci.c
shivani@UbuntuLTS:~/Downloads$ ./a.out
Enter the number of terms: 8
Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, shivani@UbuntuLTS:~/Downloads$ ./a.out
vi fabinocci.c
shivani@UbuntuLTS:~/Downloads$ cc fabinocci.c
shivani@UbuntuLTS:~/Downloads$ ./a.out
Enter the number of terms: 6
Fibonacci Series: 0, 1, 1, 2, 3, 5, shivani@UbuntuLTS:~/Downloads$ ls
addition.c a.out binary.c fabinocci.c factorial.c factorial.out patter.c
shivani@UbuntuLTS:~/Downloads$ vi patter.c
[1]+  Stopped                  vi patter.c
shivani@UbuntuLTS:~/Downloads$ vi patter.c
shivani@UbuntuLTS:~/Downloads$ cc patter.c
shivani@UbuntuLTS:~/Downloads$ ./a.out
Enter the number of rows: 7
*
* *
* * *
* * * *
* * * * *
* * * * * *
```

## LAB 3:-

06/10/2022

**Create child, Orphan and Zombie processes using suitable system calls such as fork(), exec(), wait(), kill(), sleep() and exit() system calls.**

### Child Process:-

Fork system call is used for creating a new process, which is called **child process**, which runs concurrently with the process that makes the fork() call (parent process). After a new child process is created, both processes will execute the next instruction following the fork() system call. A child process uses the same pc(program counter), same CPU registers, same open files which use in the parent process.

It takes no parameters and returns an integer value. Below are different values returned by fork().

**Negative Value:** creation of a child process was unsuccessful.

**Zero:** Returned to the newly created child process.

**Positive value:** Returned to parent or caller. The value contains process ID of newly created child process.

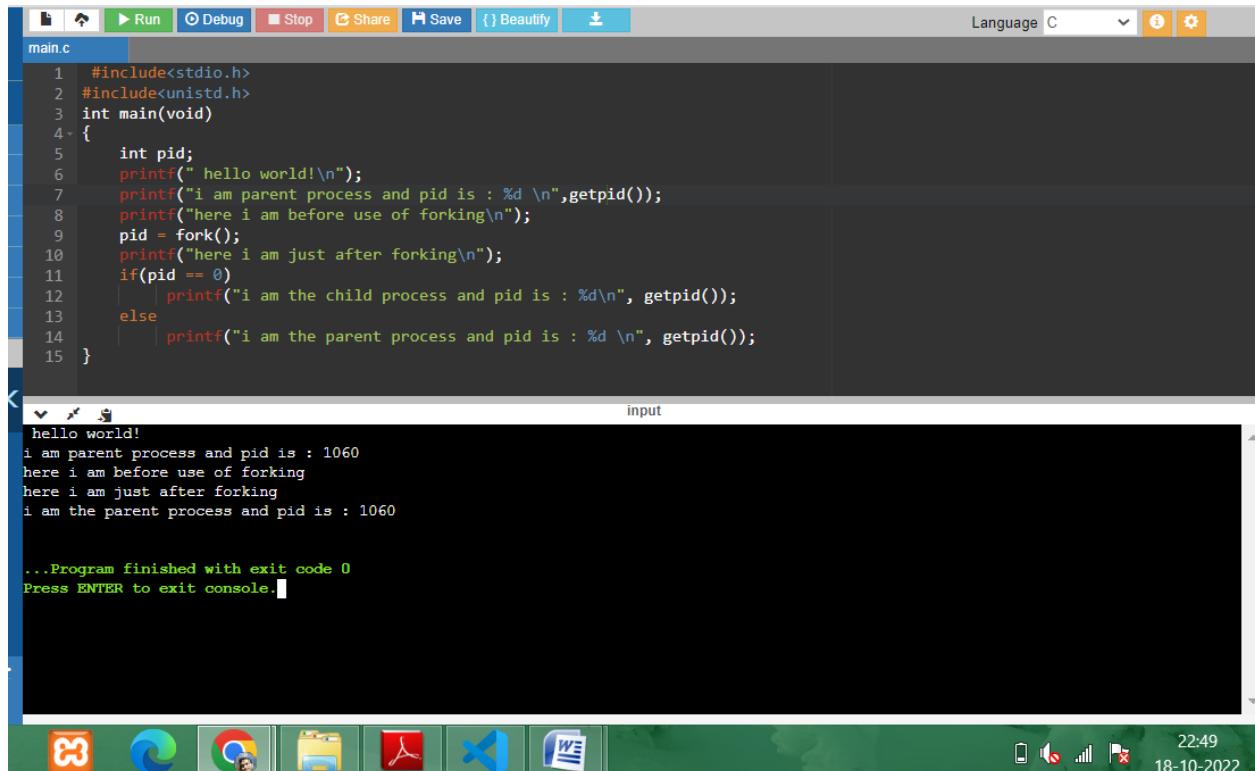
## LAB 4.1

```
main.c
1 #include <stdio.h>
2 #include <unistd.h>
3 int main(void)
4 {
5     printf("Hello world!\n");
6     fork();
7     printf("i am after forking\n");
8     printf("\t I am process %d.\n", getpid());
9 }
```

Hello world!  
i am after forking  
    I am process 1779.

...Program finished with exit code 0  
Press ENTER to exit console.[]

## LAB 4.2

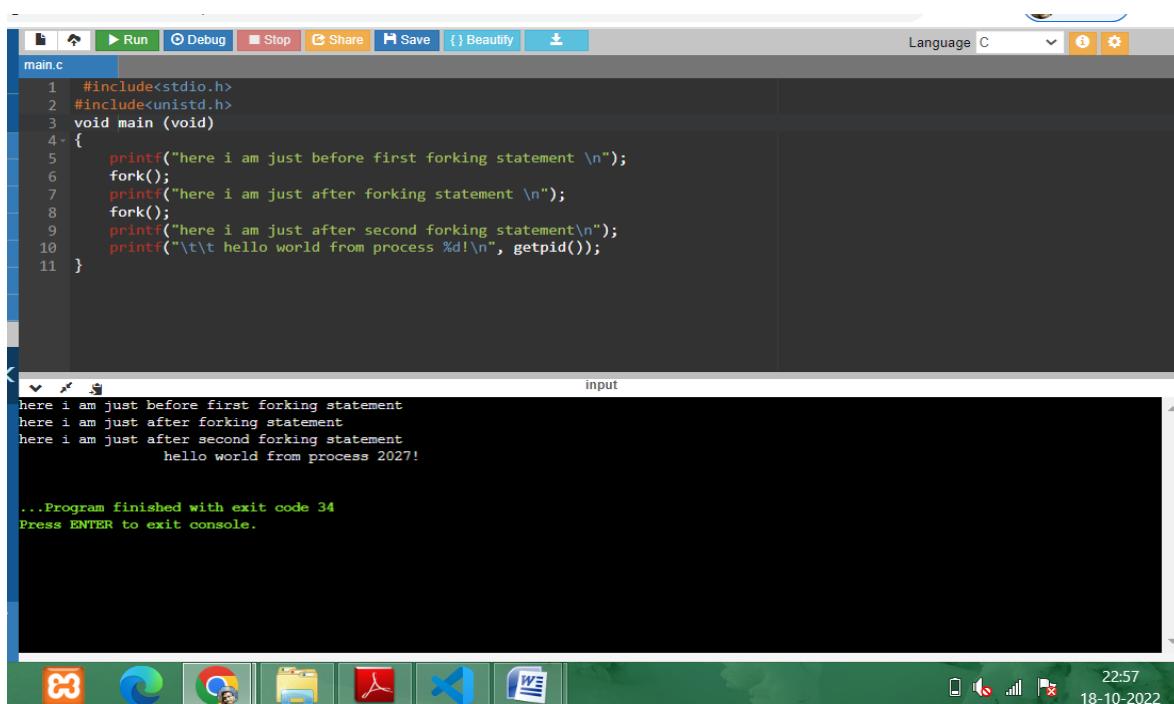


```
main.c
1 #include<stdio.h>
2 #include<unistd.h>
3 int main(void)
4 {
5     int pid;
6     printf(" hello world!\n");
7     printf("i am parent process and pid is : %d \n",getpid());
8     printf("here i am before use of forking\n");
9     pid = fork();
10    printf("here i am just after forking\n");
11    if(pid == 0)
12        printf("i am the child process and pid is : %d\n", getpid());
13    else
14        printf("i am the parent process and pid is : %d \n", getpid());
15 }
```

```
hello world!
i am parent process and pid is : 1060
here i am before use of forking
here i am just after forking
i am the parent process and pid is : 1060

...Program finished with exit code 0
Press ENTER to exit console.
```

## LAB 4.3

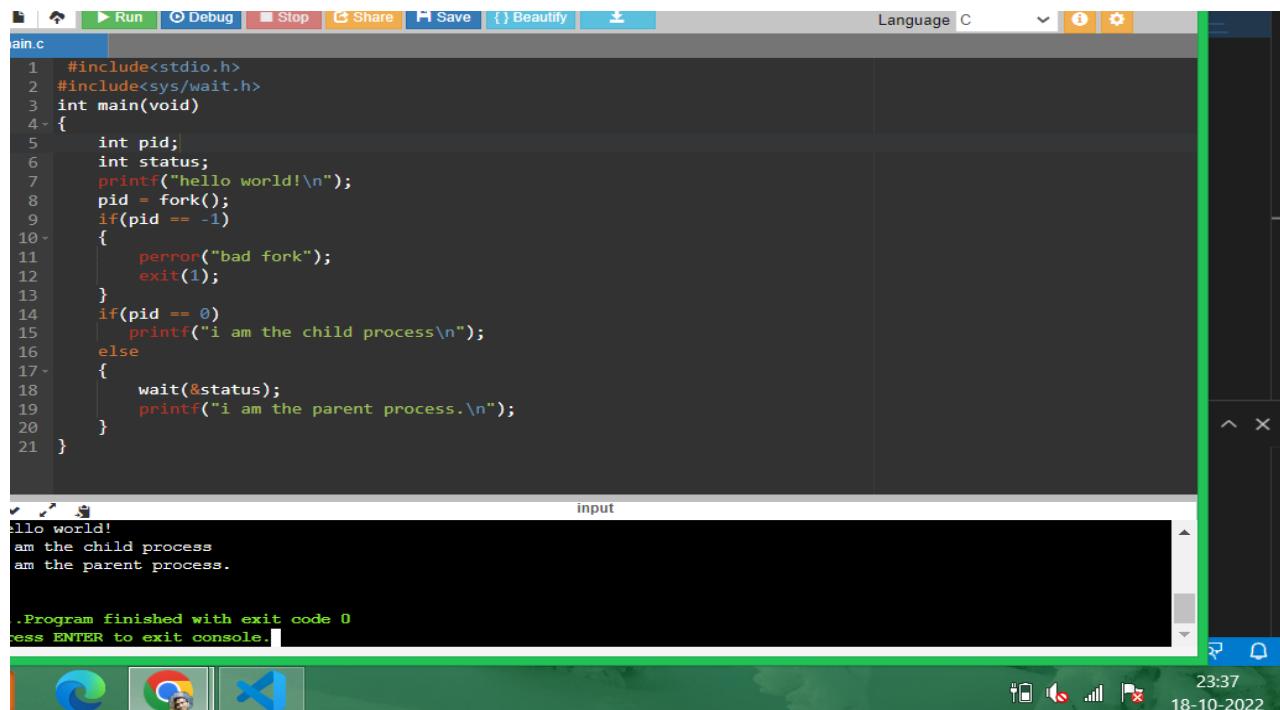


```
main.c
1 #include<stdio.h>
2 #include<unistd.h>
3 void main (void)
4 {
5     printf("here i am just before first forking statement \n");
6     fork();
7     printf("here i am just after forking statement \n");
8     fork();
9     printf("here i am just after second forking statement\n");
10    printf("\t\t hello world from process %d!\n", getpid());
11 }
```

```
here i am just before first forking statement
here i am just after forking statement
here i am just after second forking statement
    hello world from process 2027!

...Program finished with exit code 34
Press ENTER to exit console.
```

#### Lab 4.4 : Guarantees the child process will print its message before the parent process.

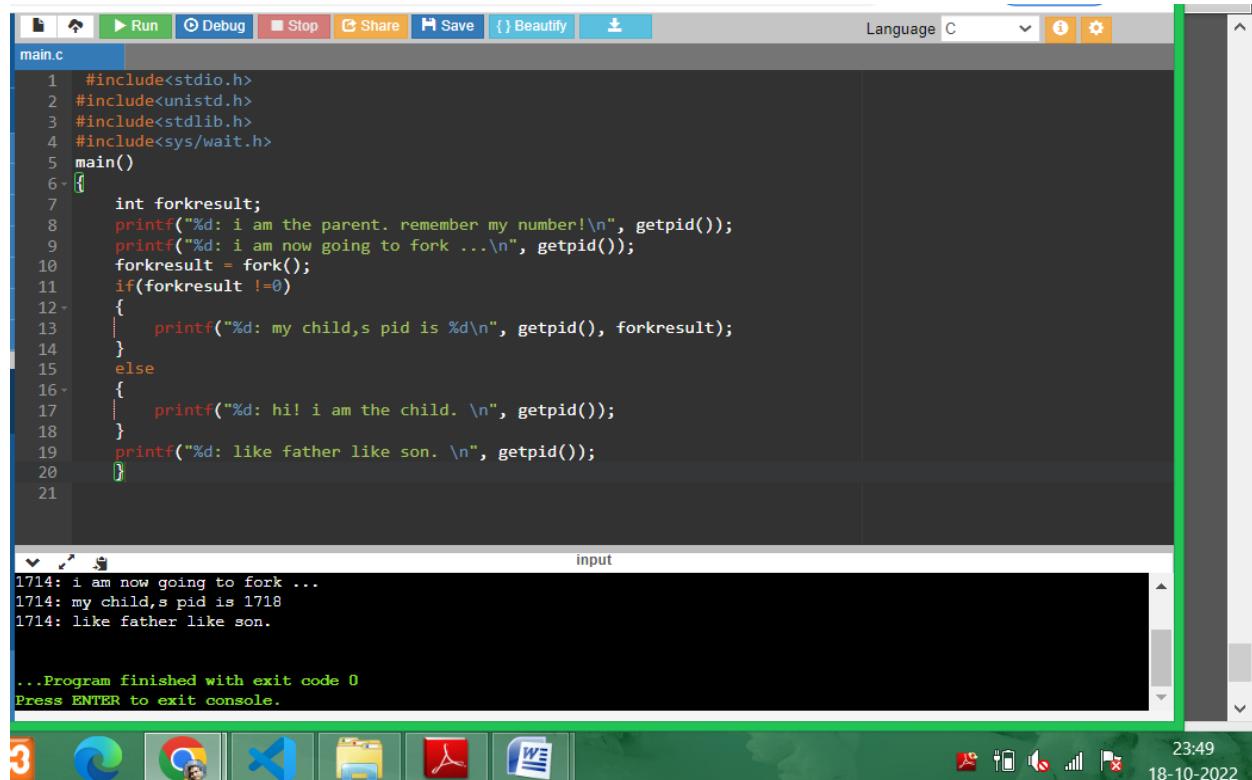


```
main.c
1 #include<stdio.h>
2 #include<sys/wait.h>
3 int main(void)
4 {
5     int pid;
6     int status;
7     printf("Hello world!\n");
8     pid = fork();
9     if(pid == -1)
10    {
11        perror("bad fork");
12        exit(1);
13    }
14    if(pid == 0)
15        printf("I am the child process\n");
16    else
17    {
18        wait(&status);
19        printf("I am the parent process.\n");
20    }
21 }
```

```
Hello world!
I am the child process
I am the parent process.

Program finished with exit code 0
Press ENTER to exit console.
```

#### LAB 4.5



```
main.c
1 #include<stdio.h>
2 #include<unistd.h>
3 #include<stdlib.h>
4 #include<sys/wait.h>
5 main()
6 {
7     int forkresult;
8     printf("%d: I am the parent. remember my number!\n", getpid());
9     printf("%d: I am now going to fork ... \n", getpid());
10    forkresult = fork();
11    if(forkresult !=0)
12    {
13        printf("%d: my child,s pid is %d\n", getpid(), forkresult);
14    }
15    else
16    {
17        printf("%d: hi! I am the child. \n", getpid());
18    }
19    printf("%d: like father like son. \n", getpid());
20 }
21 
```

```
1714: I am now going to fork ...
1714: my child,s pid is 1718
1714: like father like son.

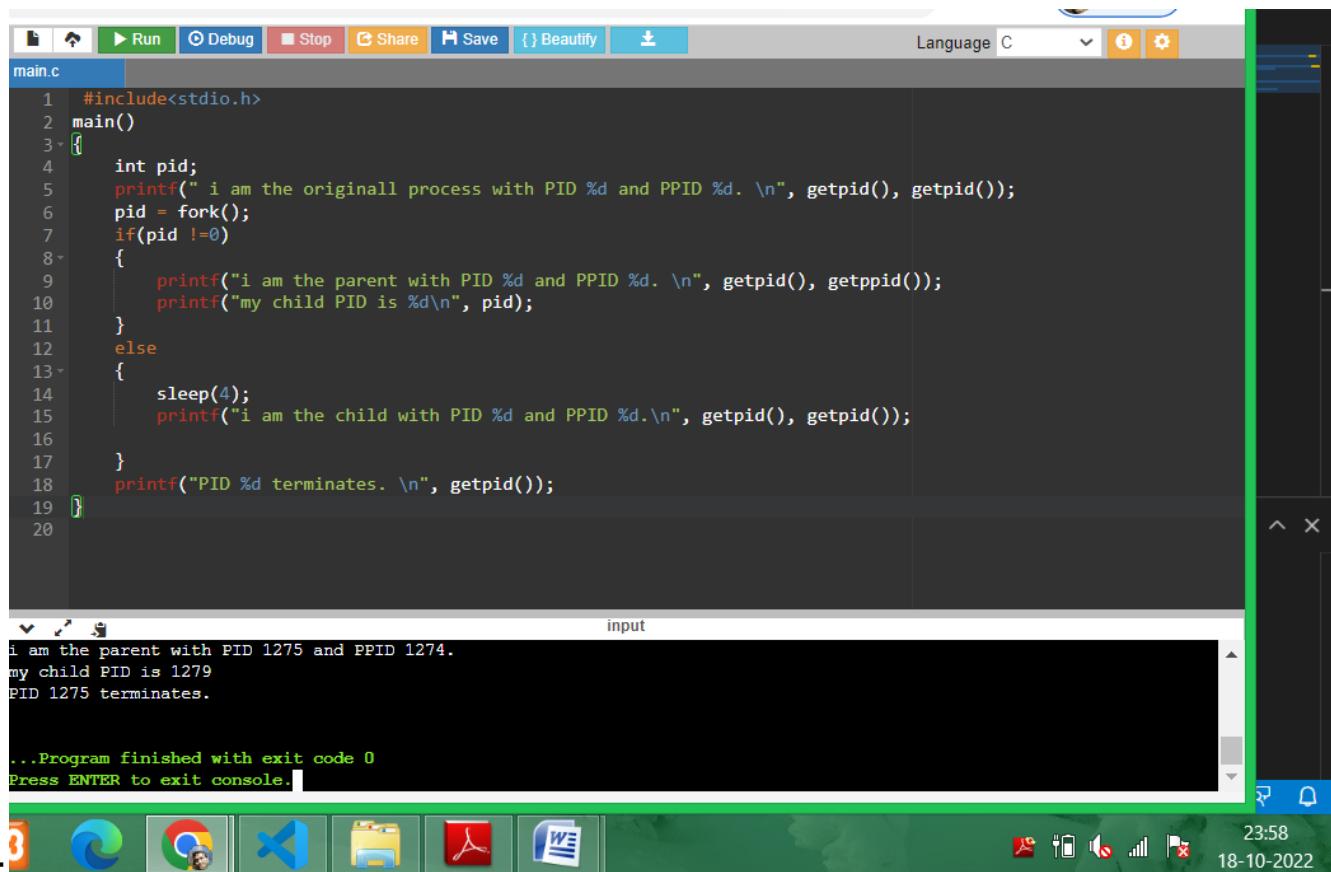
Program finished with exit code 0
Press ENTER to exit console.
```

## 4.6 Orphan Process :

A process whose parent process no more exists i.e. either finished or terminated without waiting for its child process to terminate is called an orphan process.

In the following code, parent finishes execution and exits while the child process is still executing and is called an orphan process now.

However, the orphan process is soon adopted by init process, once its parent process dies.



```
main.c
1 #include<stdio.h>
2 main()
3 {
4     int pid;
5     printf(" i am the original process with PID %d and PPID %d. \n", getpid(), getpid());
6     pid = fork();
7     if(pid !=0)
8     {
9         printf("i am the parent with PID %d and PPID %d. \n", getpid(), getppid());
10        printf("my child PID is %d\n", pid);
11    }
12    else
13    {
14        sleep(4);
15        printf("i am the child with PID %d and PPID %d.\n", getpid(), getpid());
16    }
17 }
18 printf("PID %d terminates. \n", getpid());
19 }
20 
```

i am the parent with PID 1275 and PPID 1274.  
my child PID is 1279  
PID 1275 terminates.

...Program finished with exit code 0  
Press ENTER to exit console.

The screenshot shows the Code::Blocks IDE interface. The title bar reads "\*orphan.c - Code::Blocks 20.03". The menu bar includes File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, Plugins, DoxyBlocks, Settings, and Help. The toolbar has various icons for file operations like Open, Save, Find, and Build. The project manager on the left shows files like fcfsc.c, child.c, ch.c, orphan.c, zom.c, zombi.c, zombiii.c, and chlld.c. The main code editor window displays the following C code:

```
2 #include <sys/types.h>
3 #include <unistd.h>
4 int main()
5 {
6     int pid = fork();
7     if (pid > 0) {
8         printf("Parent process\n");
9         printf("ID : %d\n\n", getpid());
10    } else if (pid == 0) {
11        printf("Child process\n");
12        printf("ID: %d\n", getpid());
13        printf("Parent -ID: %d\n\n", getppid());
14        sleep(10);
15        printf("\nChild process \n");
16        printf("ID: %d\n", getpid());
17        printf("Parent -ID: %d\n", getppid());
18    } else {
19        printf("Failed to create child process");
20    }
21 }
22 }
```

The status bar at the bottom shows the path C:\Users\User\Desktop\c program\... and the file type C/C++. The system tray icons include the Start button, Task View, Edge, File Explorer, Google Chrome, FileZilla, and DEV. The date and time are 16-10-2022 18:53.

The screenshot shows a terminal window titled "Terminal". The output of the program is displayed:

```
Parent process
ID : 56195

Child process
ID: 56196
Parent -ID: 56195

Child process
ID: 56196
Parent -ID: 1
```

## 4.7. Zombie process

A process which has finished the execution but still has entry in the process table to report to its parent process is known as a zombie process. A child process always first becomes a zombie before being removed from the process table. The parent process reads the exit status of the child process which reaps off the child process entry from the process table.

The screenshot shows a terminal window with the following content:

```
main.c
1 #include<stdio.h>
2 #include<stdlib.h>
3 int main()
4 {
5     int pid;
6     pid = fork();
7     if(pid != 0)
8     {
9         while(1)
10        sleep(100);
11    }
12    else
13    {
14        exit(42);
15    }
16    return 0;
17 }
```

input

```
main.c:6:11: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
   6 |     pid = fork();
   |           ^~~~~
main.c:10:9: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
  10 |     sleep(100);
   |           ^~~~~~
```

The terminal window is part of a desktop environment with a green taskbar at the bottom. The taskbar icons include a file explorer, a browser, and other application icons. The system tray shows the date and time as 19-10-2022 00:08.