LAB REPORT NO 4,5

Submitted by: **Muhammad Ali**
Registration No: **19PWCSE1801**
Class Section: **A**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Submitted to:
Engr.Mian Ibad Ali shah

Data:(05,05,2021)
Department of Computer Systems Engineering
University of Engineering and Technology, Peshawar

SHELL Programming (Part I)

Objectives: -

- To understand shell and shell script
- To understand how to make gedit file for command
- To learn different command in c
- To execute shell script in terminal window

Introduction to Linux Shell and Shell Scripting

If we are using any major operating system we are indirectly interacting to shell. If we are running Ubuntu, Linux Mint or any other Linux distribution, we are interacting to shell every time we use terminal. In this article I will discuss about linux shells and shell scripting so before understanding shell scripting we have to get familiar with following terminologies –

- Kernel
- Shell
- Terminal

Kernel: -

The kernel is a computer program that is the core of a computer's operating system, with complete control over everything in the system. It manages following resources of the Linux system –

- File management
- Process management
- I/O management
- Memory management
- Device management etc.

It is often mistaken that Linus Torvalds has developed Linux OS, but actually he is only responsible for development of Linux kernel.
Complete Linux system = Kernel + GNU system utilities and libraries + other management scripts + installation scripts.

Shell: -

A shell is special user program which provide an interface to user to use operating system services. Shell accept human readable commands from user and convert them into something which kernel can understand. It is a command language interpreter that execute commands read from input devices such as keyboards or from files. The shell gets started when the user logs in or start the terminal.

Shell is broadly classified into two categories:

- Command Line Shell
- Graphical shell

Command Line Shell: -

Shell can be accessed by user using a command line interface. A special program called Terminal in linux/macOS or Command Prompt in Windows OS is provided to type in the human readable commands such as “cat”, “ls” etc. and then it is being execute. The result is then displayed on the terminal to the user. A terminal in Ubuntu 16.4 system.

Shell Scripting: -

Usually shells are interactive that mean, they accept command as input from users and execute them. However some time we want to execute a bunch of commands routinely, so we have type in all commands each time in terminal.
As shell can also take commands as input from file we can write these commands in a file and can execute them in shell to avoid this repetitive work. These files are called Shell Scripts or Shell Programs. Shell scripts are similar to the batch file in MS-DOS. Each shell script is saved with .sh file extension eg. myscript.sh
A shell script have syntax just like any other programming language. If we have any prior experience with any programming language like Python, C/C++ etc. it would be very easy to get started with it.

A shell script comprises following elements:

- Shell Keywords – if, else, break etc.
- Shell commands – cd, ls, echo, pwd, touch etc.
- Functions
- Control flow – if..then..else, case and shell loops etc.

Graphical Shells: -

Graphical shells provide means for manipulating programs based on graphical user interface (GUI), by allowing for operations such as opening, closing, moving and resizing windows, as well as switching focus between windows.

Arithmetic operation: -

The oldest command for doing arithmetic operations in bash is 'expr'. This command can work with integer values only and prints the output directly in the terminal. We have to use space with each operand when we want to use 'expr' command to do any mathematical operations.

Note: All practices screenshot of terminal window is inserted at the end of lab report.

SHELL Programming (Part II)

Objectives: -

- To understand program as shell script learn and practice SHELL scripts by writing small SHELL programs.
- To understand special variable and user input
- To check file type, permission detail, and directory search
- And lastly some conditional and loops understanding.

variables in script: -

A shell script allows us to set and use our own variables within the script.
Setting variables allows we to temporarily store data and use it throughout the script, making the shell script more like a real computer program. User variables can be any text string of up to 20 letters, digits, or an underscore character.

Passing argument: -

Arguments can be passed to the script when it is executed, by writing them as a space-delimited list following the script file name.

Inside the script, the $1 variable references the first argument in the command line, $2 the second argument and so forth. The variable $0 references to the current script. In the following example, the script name is followed by 6 arguments.

If there are more than 9 arguments, then tenth or onwards arguments can't be assigned as $10 or $11.

1. $* – Store all command line arguments.
2. $@ – Store all command line arguments.
3. $# – Store count of command line arguments.
4. $0 – Store name of script itself.
5. $1 – Store first command line argument.
6. $2 – Store second command line argument.
7. $3 – Store third command line argument.

If then and if else statement: -

The if statement allows we to specify courses of action to be taken in a shell script, depending on the success or failure of some command.
In a conditional, we frequently have tasks to perform when the tested condition succeeds or fails. The shell can accommodate this with the if/then/else syntax. In the if/then/else form of the if statement, the block of statements after the then statement is executed if the condition succeeds. Execution continues with the statement following the fi statement. If the condition in the if statement fails, then the block of statements after the then statement is skipped, and statements following the else are executed. Execution continues with the statement following the fi statement. The syntax for if/then/else is:

Shell Scripting for loop: -

The for loop moves through a specified list of values until the list is exhausted.

1**) Syntax:**

Syntax of for loop using in and list of values is shown below. This for loop contains a number of variables in the list and will execute for each item in the list. For example, if there are 10 variables in the list, then loop will execute ten times and value will be stored in varname.

Example: -

```
For(i=10; i>=0; i--);

do

echo ”$i ”

done
```

Shell Scripting while loop: -

Linux scripting while loop is similar to C language while loop. There is a condition in while. And commands are executed till the condition is valid. Once condition becomes false, loop terminates.

Example: -

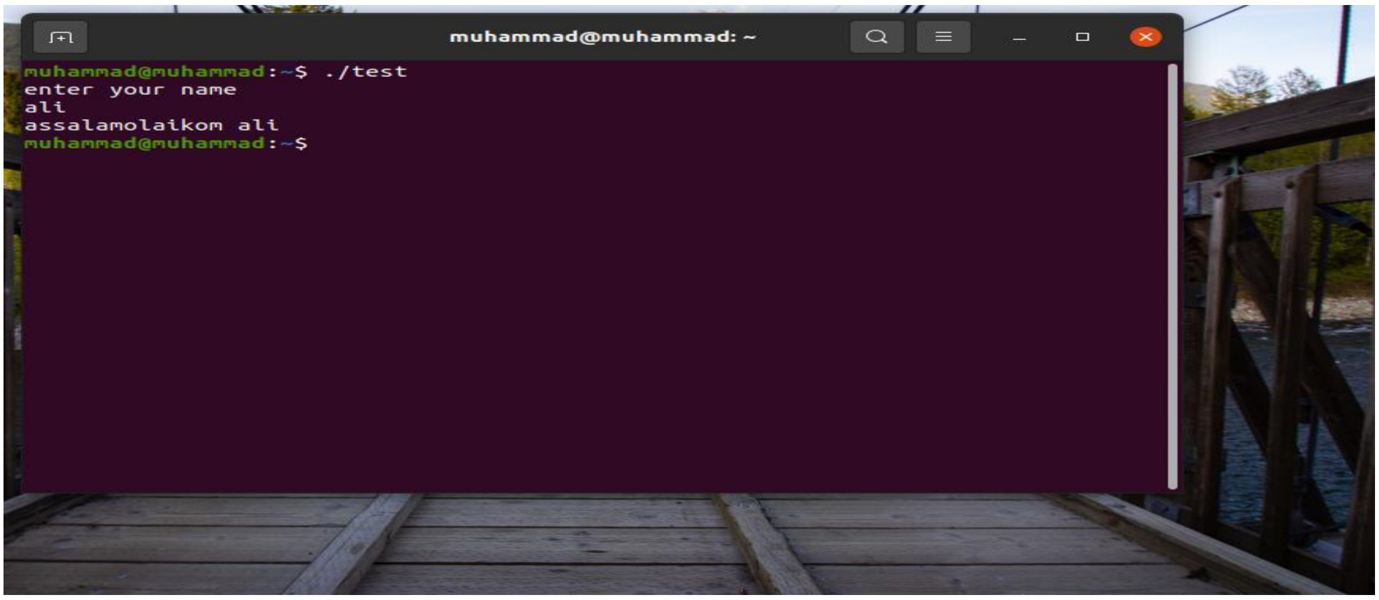
```
i=10;
while [$i -ge 0];
do
echo “number in reverse order $i”;
i-- ;
done
```

The break and continue statements:
It is often necessary to handle exception conditions within loops. The statements break and continue are used for this.
The break command terminates the execution of the innermost enclosing loop, causing execution to resume after the nearest done statement.
To exit from *n* levels, use the command:
break *n*
This will cause execution to resume after the done *n* levels up.
The continue command causes execution to resume at the while, until or for statement which begins the loop containing the continue command.

Assignment Problems on UNIX SHELL programming

1.Run all the programs given in the Lab Notes, and observe the output for each program. (Both lab 4 and 5 practice)

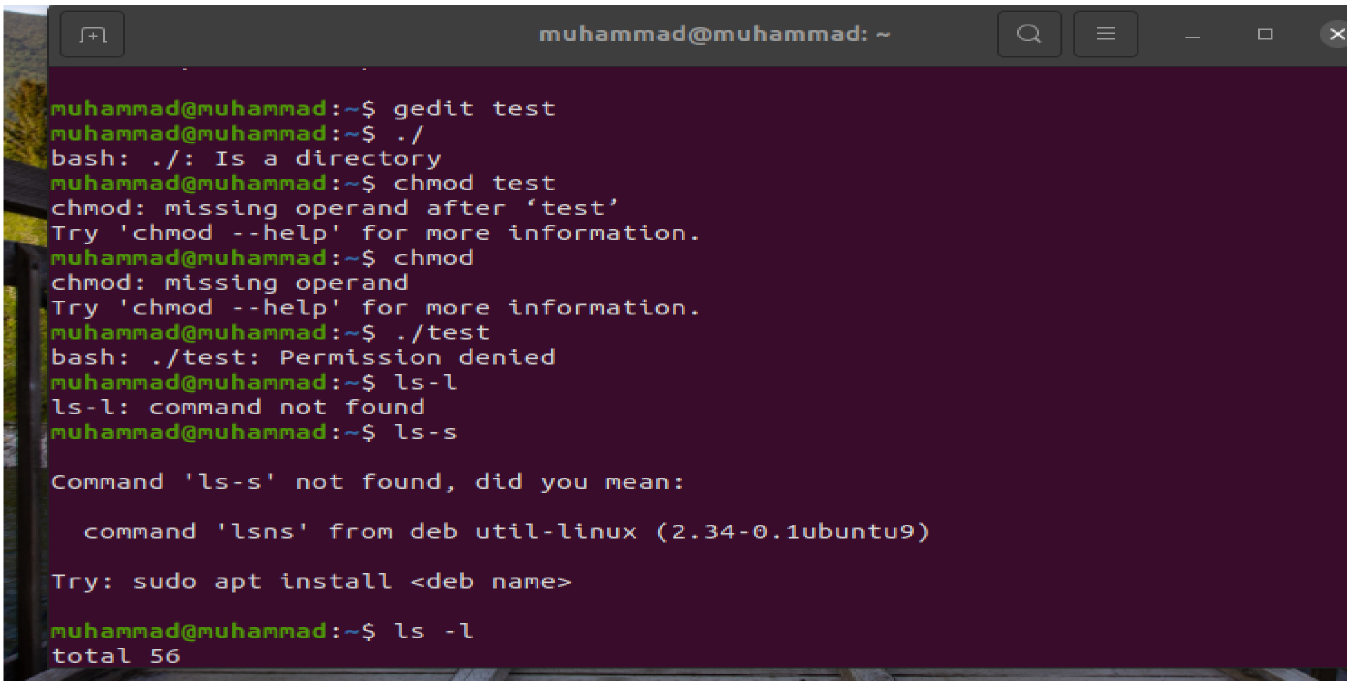
Command Output:



muhammad@muhammad: ~

```
muhammad@muhammad:~$ ./test
enter your name
ali
assalamolaikom ali
muhammad@muhammad:~$
```

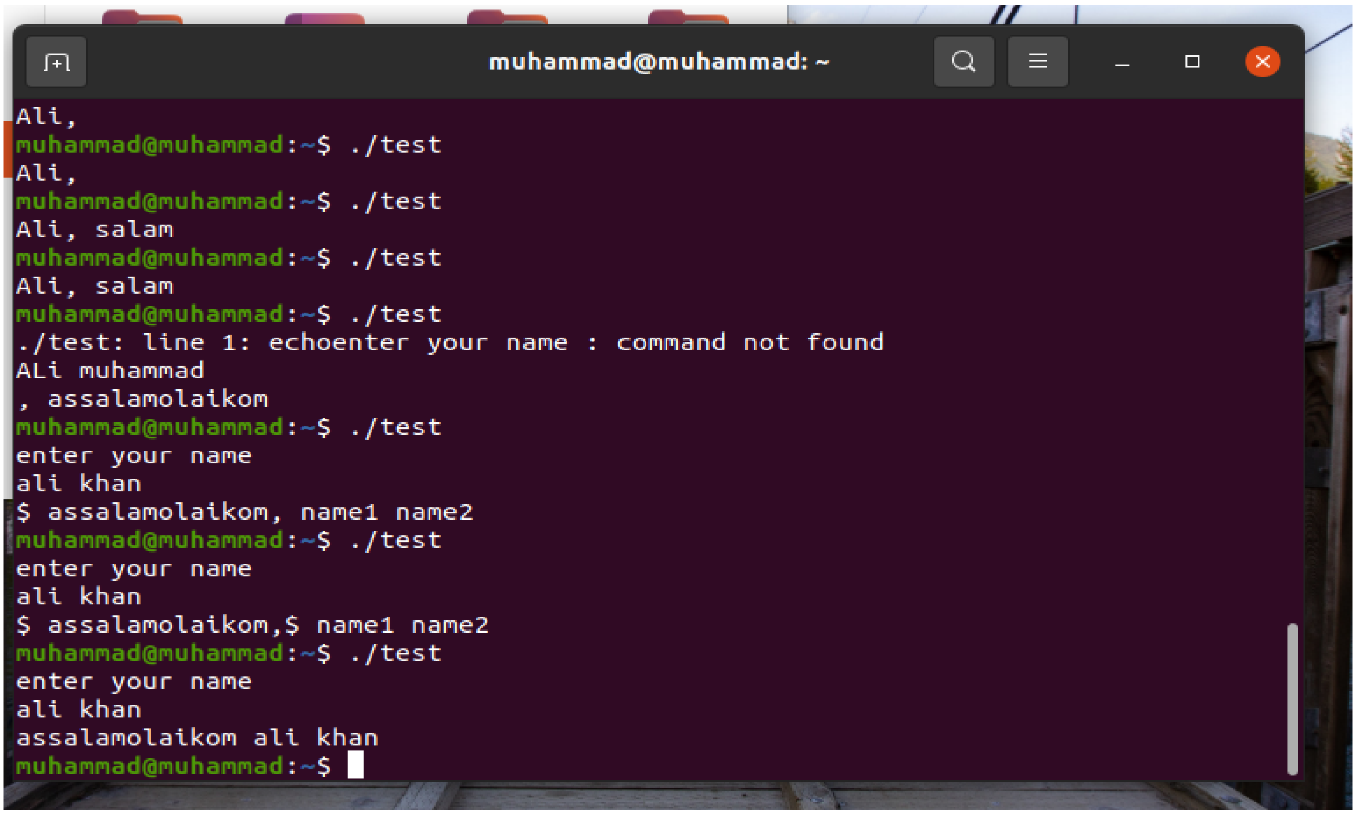
Command Output:



muhammad@muhammad: ~

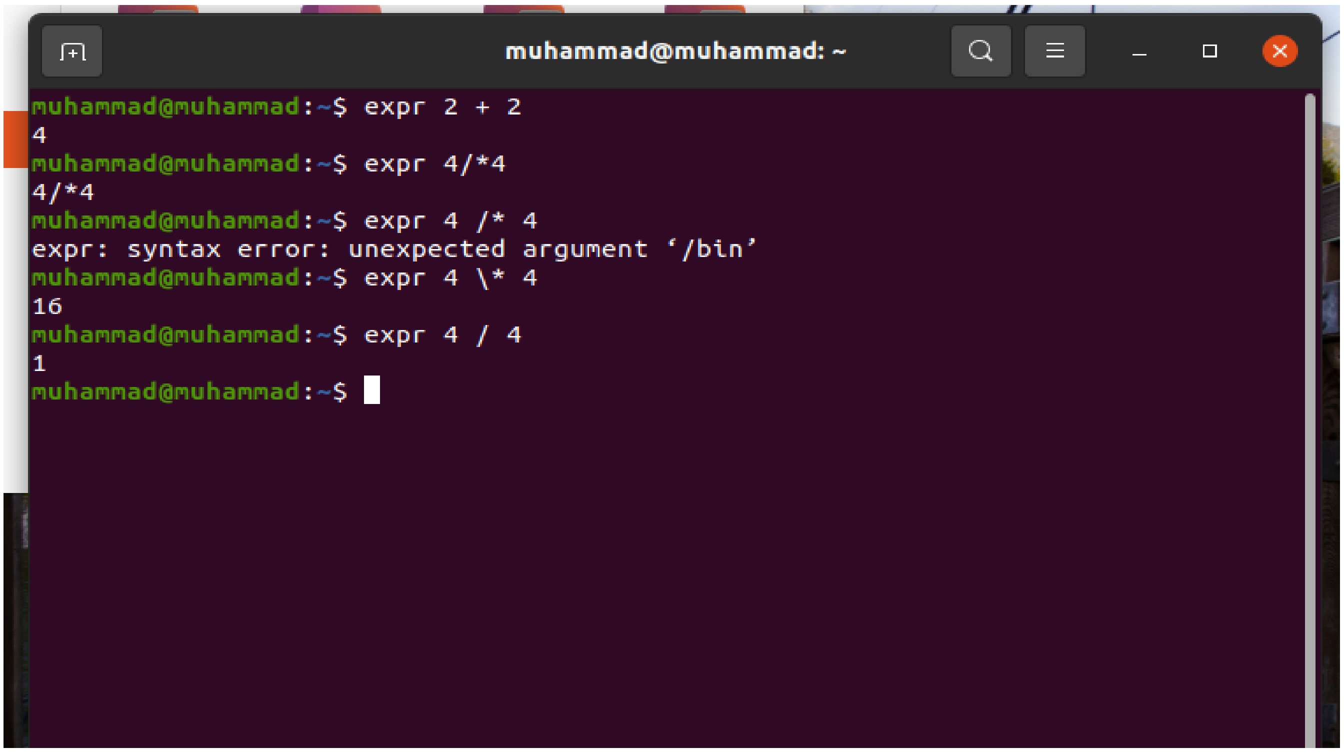
```
muhammad@muhammad:~$ gedit test
muhammad@muhammad:~$ ./
bash: ./: Is a directory
muhammad@muhammad:~$ chmod test
chmod: missing operand after ‘test’
Try 'chmod --help' for more information.
muhammad@muhammad:~$ chmod
chmod: missing operand
Try 'chmod --help' for more information.
muhammad@muhammad:~$ ./test
bash: ./test: Permission denied
muhammad@muhammad:~$ ls-l
ls-l: command not found
muhammad@muhammad:~$ ls-s

Command 'ls-s' not found, did you mean:

  command 'lsns' from deb util-linux (2.34-0.1ubuntu9)

Try: sudo apt install <deb name>

muhammad@muhammad:~$ ls -l
total 56
```

Command Output:



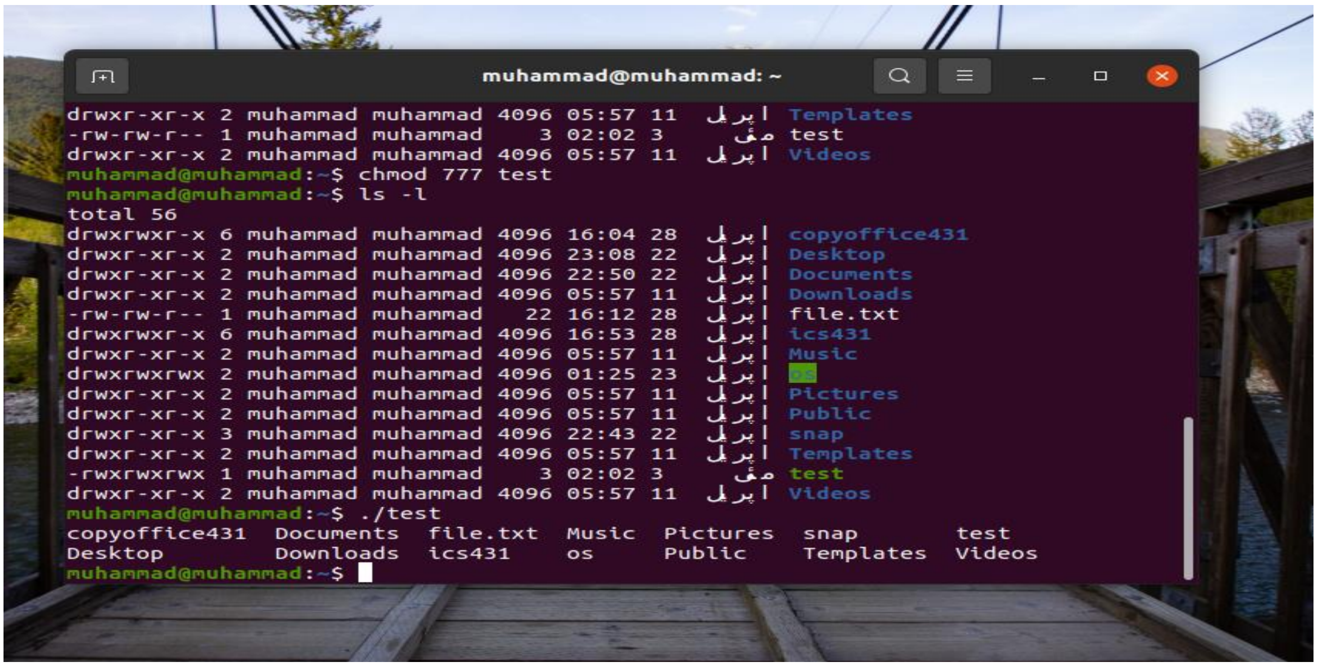
```
ALi,
muhammad@muhammad:~$ ./test
ALi,
muhammad@muhammad:~$ ./test
ALi, salam
muhammad@muhammad:~$ ./test
ALi, salam
muhammad@muhammad:~$ ./test
./test: line 1: echoenter your name : command not found
ALi muhammad
, assalamolaikom
muhammad@muhammad:~$ ./test
enter your name
ali khan
$ assalamolaikom, name1 name2
muhammad@muhammad:~$ ./test
enter your name
ali khan
$ assalamolaikom,$ name1 name2
muhammad@muhammad:~$ ./test
enter your name
ali khan
assalamolaikom ali khan
muhammad@muhammad:~$
```

Command Output:



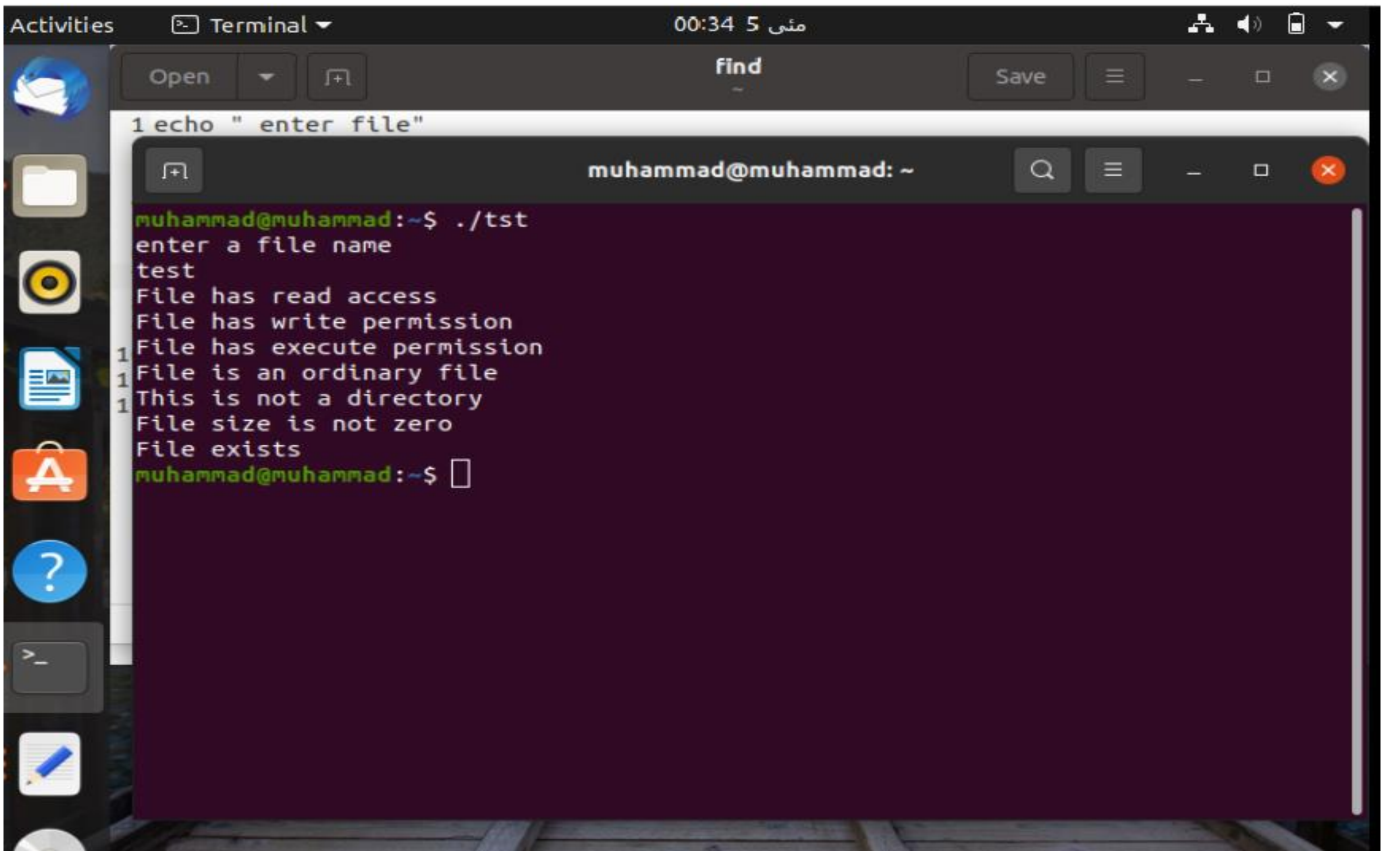
```
muhammad@muhammad:~$ expr 2 + 2
4
muhammad@muhammad:~$ expr 4/*4
4/*4
muhammad@muhammad:~$ expr 4 /* 4
expr: syntax error: unexpected argument ‘/bin’
muhammad@muhammad:~$ expr 4 \* 4
16
muhammad@muhammad:~$ expr 4 / 4
1
muhammad@muhammad:~$
```

Command Output:



```
drwxr-xr-x 2 muhammad muhammad 4096 05:57 11  ابريل Templates
-rw-rw-r-- 1 muhammad muhammad    3 02:02 3    مئى test
drwxr-xr-x 2 muhammad muhammad 4096 05:57 11  ابريل Videos
muhammad@muhammad:~$ chmod 777 test
muhammad@muhammad:~$ ls -l
total 56
drwxrwxr-x 6 muhammad muhammad 4096 16:04 28  ابريل copyoffice431
drwxr-xr-x 2 muhammad muhammad 4096 23:08 22  ابريل Desktop
drwxr-xr-x 2 muhammad muhammad 4096 22:50 22  ابريل Documents
drwxr-xr-x 2 muhammad muhammad 4096 05:57 11  ابريل Downloads
-rw-rw-r-- 1 muhammad muhammad   22 16:12 28  ابريل file.txt
drwxrwxr-x 6 muhammad muhammad 4096 16:53 28  ابريل ics431
drwxr-xr-x 2 muhammad muhammad 4096 05:57 11  ابريل Music
drwxrwxrwx 2 muhammad muhammad 4096 01:25 23  ابريل os
drwxr-xr-x 2 muhammad muhammad 4096 05:57 11  ابريل Pictures
drwxr-xr-x 2 muhammad muhammad 4096 05:57 11  ابريل Public
drwxr-xr-x 3 muhammad muhammad 4096 22:43 22  ابريل snap
drwxr-xr-x 2 muhammad muhammad 4096 05:57 11  ابريل Templates
-rwxrwxrwx 1 muhammad muhammad    3 02:02 3    مئى test
drwxr-xr-x 2 muhammad muhammad 4096 05:57 11  ابريل Videos
muhammad@muhammad:~$ ./test
copyoffice431  Documents  file.txt  Music  Pictures  snap       test
Desktop        Downloads  ics431    os     Public    Templates  Videos
muhammad@muhammad:~$
```

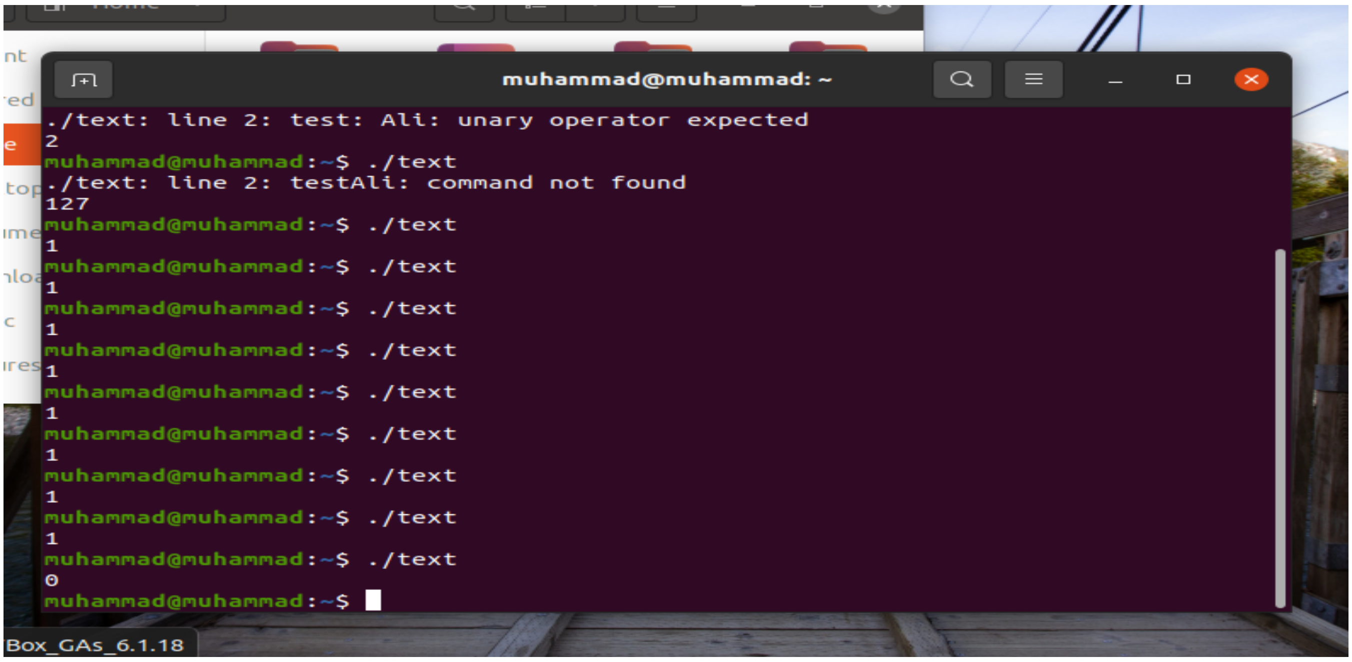
Command Output: program to check the permission detail of a various files.



```
echo " enter file"
```

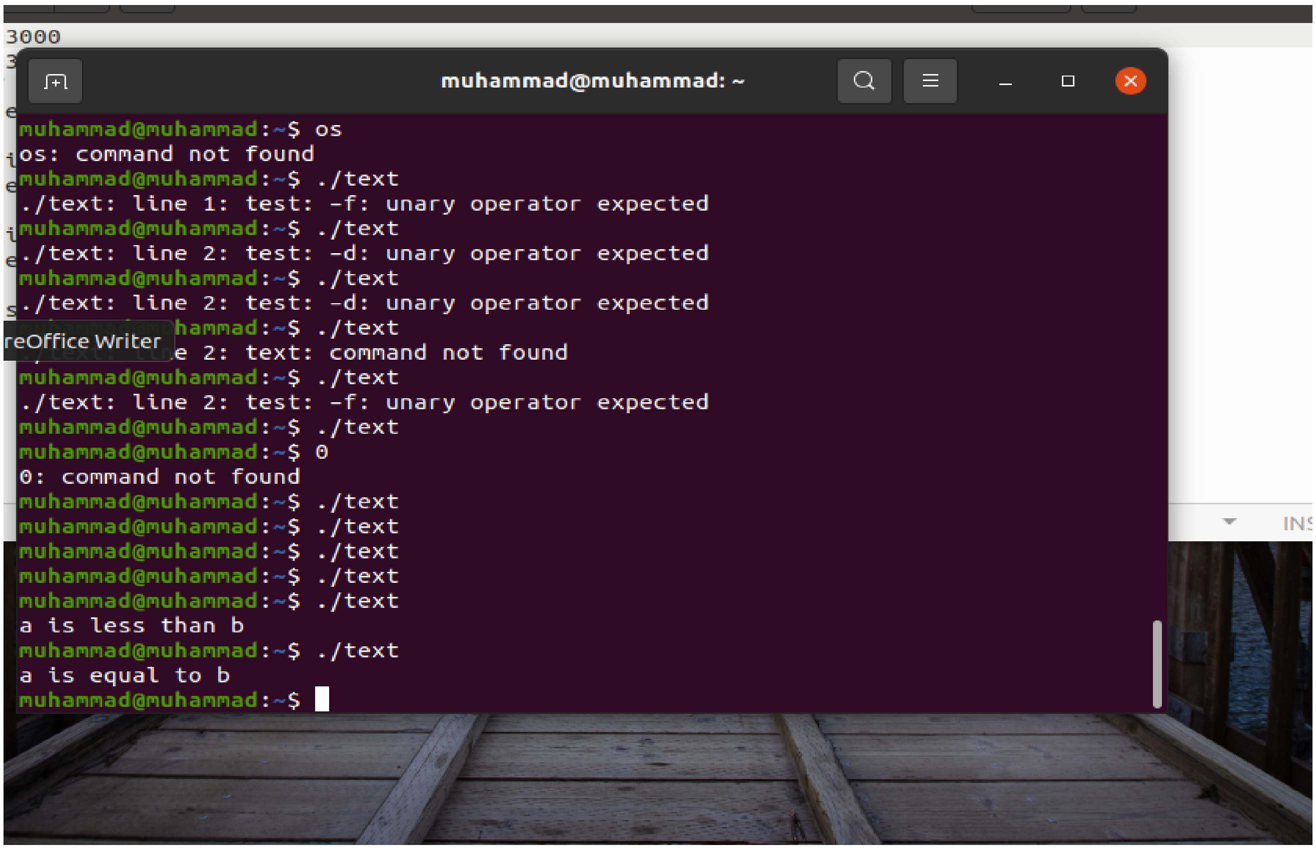
```
muhammad@muhammad:~$ ./tst
enter a file name
test
File has read access
File has write permission
File has execute permission
File is an ordinary file
This is not a directory
File size is not zero
File exists
muhammad@muhammad:~$
```

Command output: test -n $filename



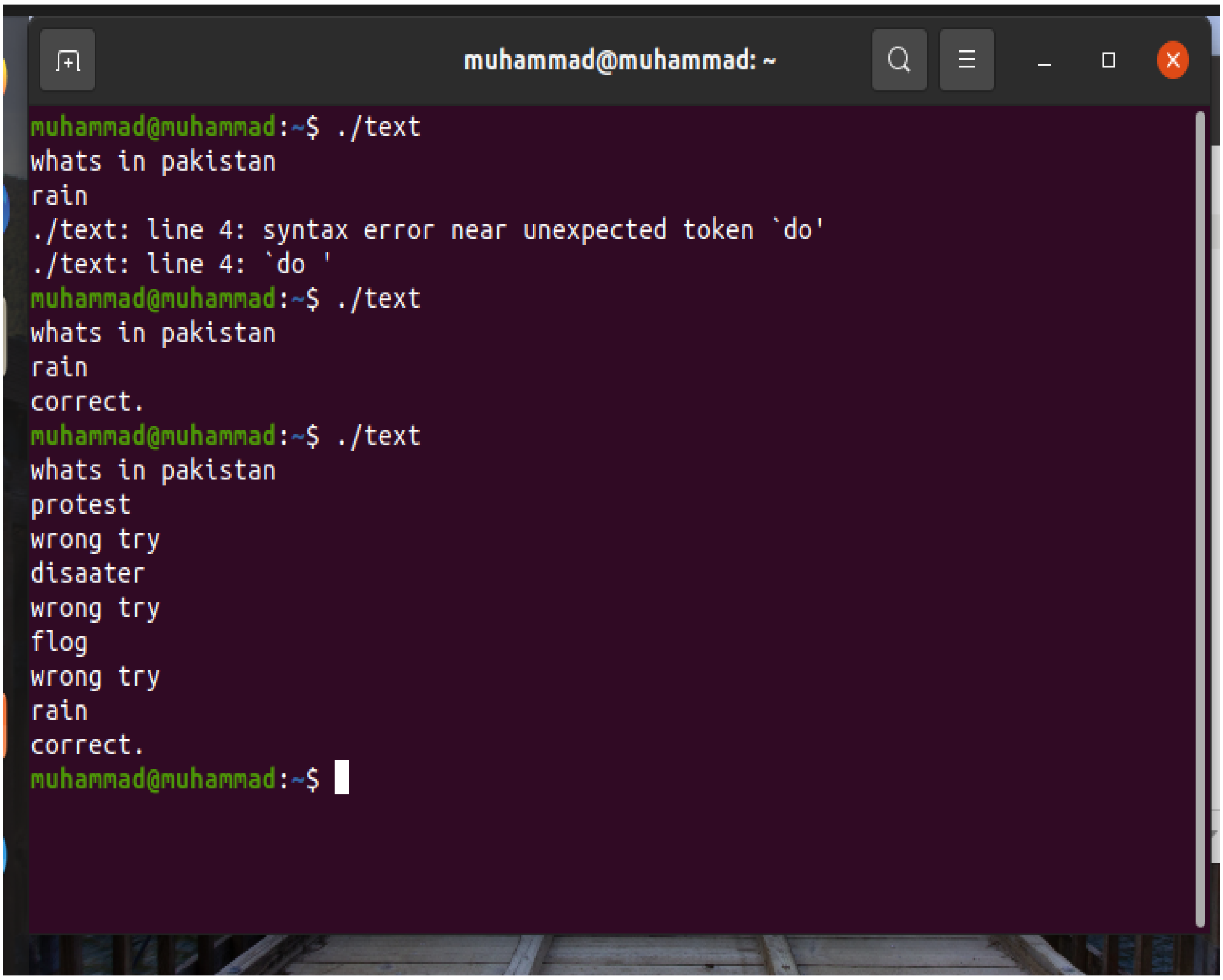
```
./text: line 2: test: Ali: unary operator expected
2
muhammad@muhammad:~$ ./text
./text: line 2: testAli: command not found
127
muhammad@muhammad:~$ ./text
1
muhammad@muhammad:~$ ./text
1
muhammad@muhammad:~$ ./text
1
muhammad@muhammad:~$ ./text
1
muhammad@muhammad:~$ ./text
1
muhammad@muhammad:~$ ./text
1
muhammad@muhammad:~$ ./text
1
muhammad@muhammad:~$ ./text
1
muhammad@muhammad:~$ ./text
0
muhammad@muhammad:~$ 
```

Command Output: if else conditional program examining comparison between two integer values.



```
muhammad@muhammad:~$ os
os: command not found
muhammad@muhammad:~$ ./text
./text: line 1: test: -f: unary operator expected
muhammad@muhammad:~$ ./text
./text: line 2: test: -d: unary operator expected
muhammad@muhammad:~$ ./text
./text: line 2: test: -d: unary operator expected
muhammad@muhammad:~$ ./text
./text: line 2: text: command not found
muhammad@muhammad:~$ ./text
./text: line 2: test: -f: unary operator expected
muhammad@muhammad:~$ ./text
muhammad@muhammad:~$ 0
0: command not found
muhammad@muhammad:~$ ./text
muhammad@muhammad:~$ ./text
muhammad@muhammad:~$ ./text
muhammad@muhammad:~$ ./text
muhammad@muhammad:~$ ./text
a is less than b
muhammad@muhammad:~$ ./text
a is equal to b
muhammad@muhammad:~$ 
```

Command Output: while loop, while when the user enter “rain”



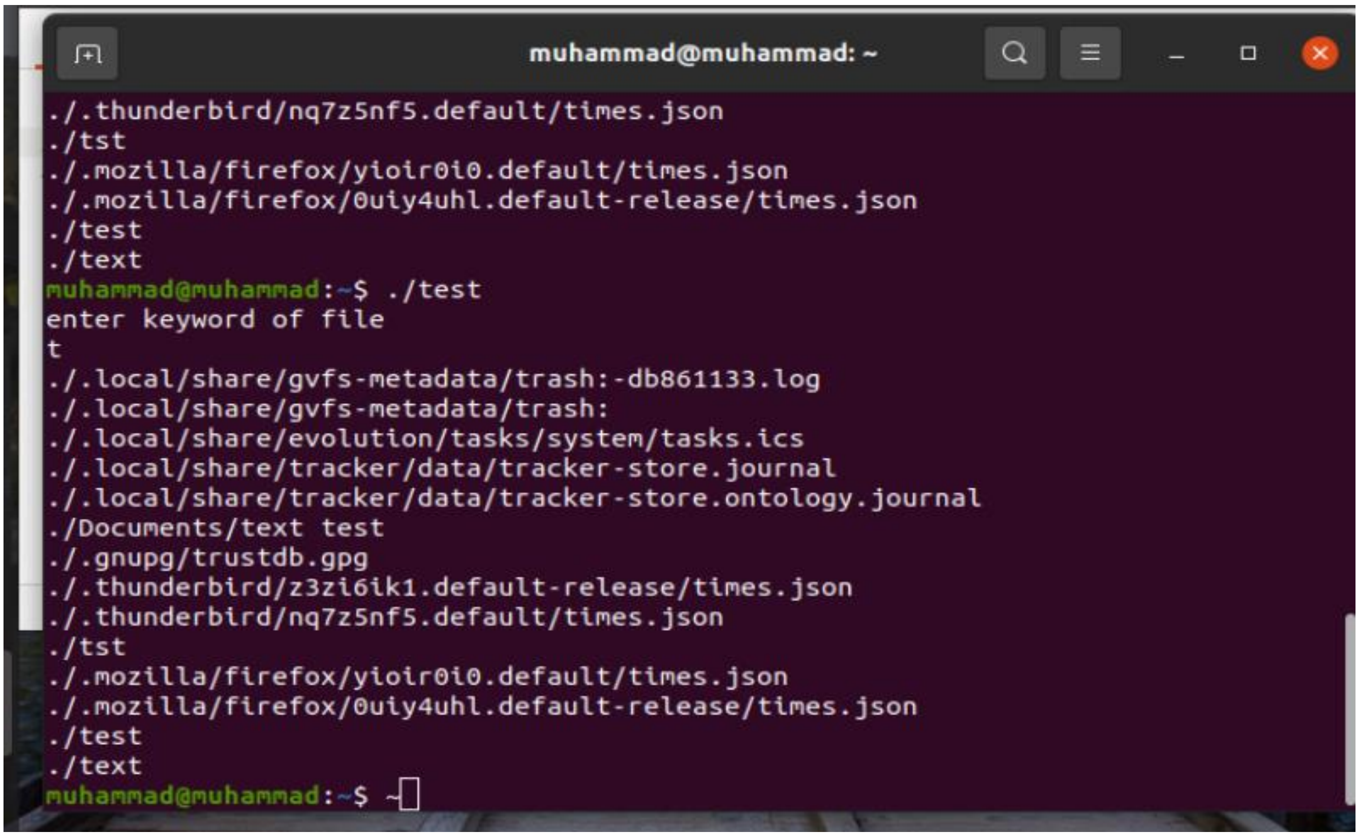
```
muhammad@muhammad:~$ ./text
whats in pakistan
rain
./text: line 4: syntax error near unexpected token `do'
./text: line 4: `do '
muhammad@muhammad:~$ ./text
whats in pakistan
rain
correct.
muhammad@muhammad:~$ ./text
whats in pakistan
protest
wrong try
disaater
wrong try
flog
wrong try
rain
correct.
muhammad@muhammad:~$
```

2. Write a shell script that takes a keyword as a command line argument and lists the filenames containing the keyword

Command:

```
echo "enter keyword of file"

read t

find . -type f -name "t*"
```

Output:



muhammad@muhammad: ~

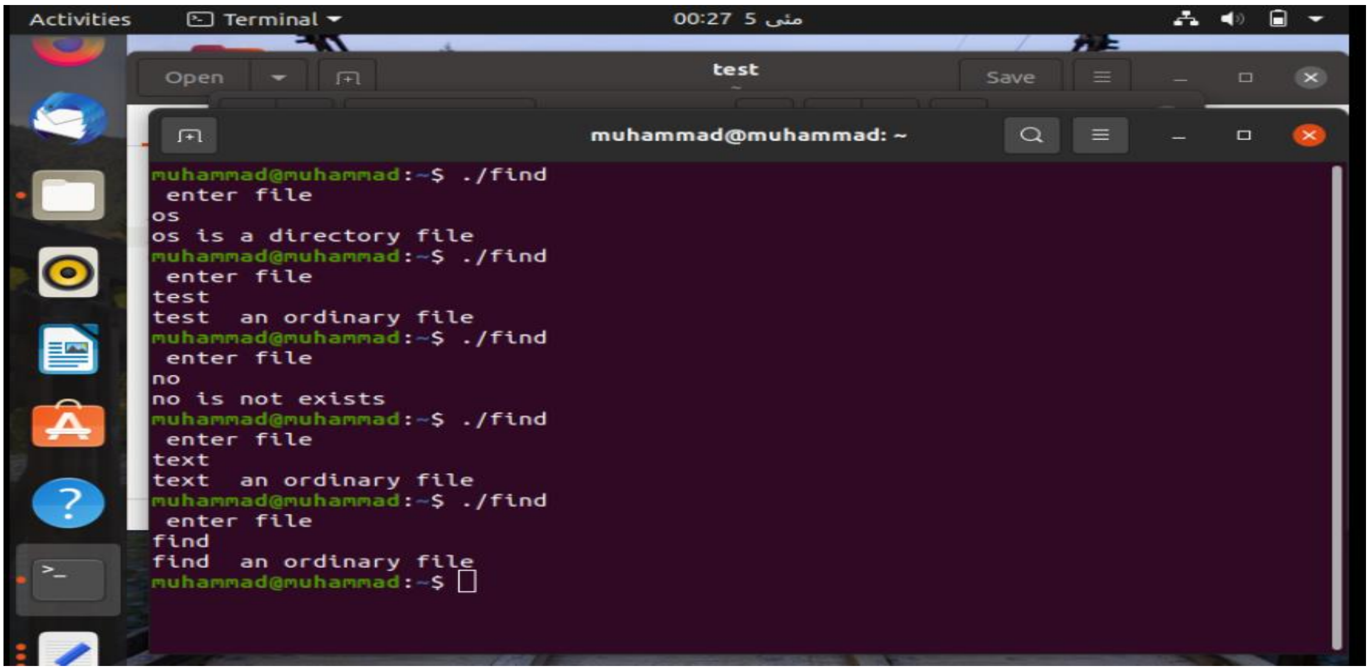
```
./.thunderbird/nq7z5nf5.default/times.json
./tst
./.mozilla/firefox/yioir0i0.default/times.json
./.mozilla/firefox/0uiy4uhl.default-release/times.json
./test
./text
muhammad@muhammad:~$ ./test
enter keyword of file
t
./.local/share/gvfs-metadata/trash:-db861133.log
./.local/share/gvfs-metadata/trash:
./.local/share/evolution/tasks/system/tasks.ics
./.local/share/tracker/data/tracker-store.journal
./.local/share/tracker/data/tracker-store.ontology.journal
./Documents/text test
./.gnupg/trustdb.gpg
./.thunderbird/z3zi6ik1.default-release/times.json
./.thunderbird/nq7z5nf5.default/times.json
./tst
./.mozilla/firefox/yioir0i0.default/times.json
./.mozilla/firefox/0uiy4uhl.default-release/times.json
./test
./text
muhammad@muhammad:~$ 
```

3. Write a shell script that takes a command line argument and reports whether it is a directory, or a file or a link.

Command:

```
echo " enter file"

read str

if test -f $str

then echo $str " an ordinary file"

elif test -d $str

then echo $str "is a directory file"

else

echo $str "is not exists"

fi

if test -c $str

then echo $str " is a character device files"

fi
```

Output:



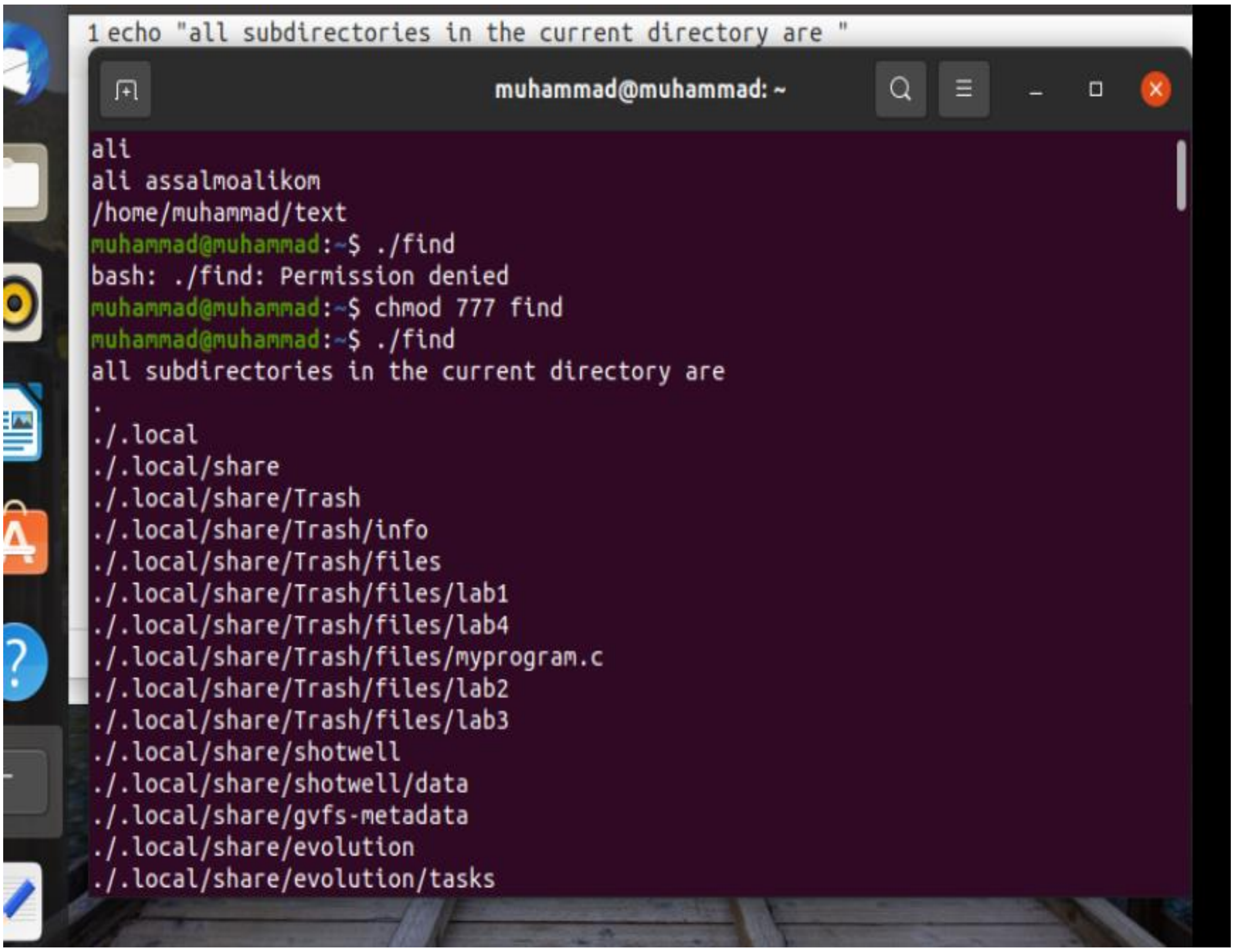
```
muhammad@muhammad:~$ ./find
 enter file
os
os is a directory file
muhammad@muhammad:~$ ./find
 enter file
test
test  an ordinary file
muhammad@muhammad:~$ ./find
 enter file
no
no is not exists
muhammad@muhammad:~$ ./find
 enter file
text
text  an ordinary file
muhammad@muhammad:~$ ./find
 enter file
find
find  an ordinary file
muhammad@muhammad:~$
```

4. Write a script to find the number of sub directories in a given directory.

Command:

```
ls -d */.
```

Output:



```
echo "all subdirectories in the current directory are "
```

```
ali
ali assalmoalikom
/home/muhammad/text
muhammad@muhammad:~$ ./find
bash: ./find: Permission denied
muhammad@muhammad:~$ chmod 777 find
muhammad@muhammad:~$ ./find
all subdirectories in the current directory are
.
./.local
./.local/share
./.local/share/Trash
./.local/share/Trash/info
./.local/share/Trash/files
./.local/share/Trash/files/lab1
./.local/share/Trash/files/lab4
./.local/share/Trash/files/myprogram.c
./.local/share/Trash/files/lab2
./.local/share/Trash/files/lab3
./.local/share/shotwell
./.local/share/shotwell/data
./.local/share/gvfs-metadata
./.local/share/evolution
./.local/share/evolution/tasks
```

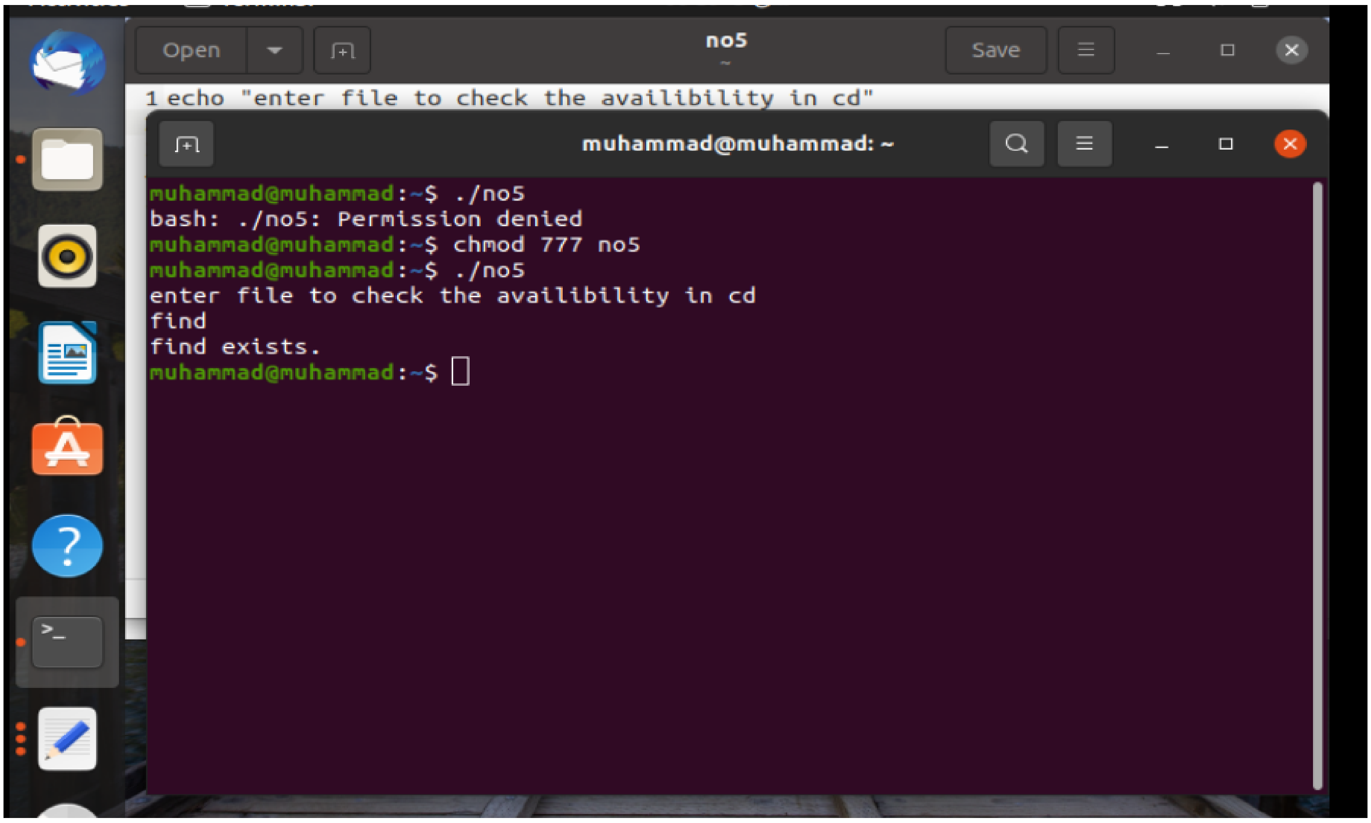
5. Write a menu driven program that has the following options.

5.1. Search a given file is in the directory or not.

Command:

```
echo "enter file to check the availibility in cd"

read FILE

if test -f "$FILE"; then

   echo "$FILE exists."

   else

      echo "$FILE dosen't not exists."

fi
```

Output:



```
echo "enter file to check the availibility in cd"
```

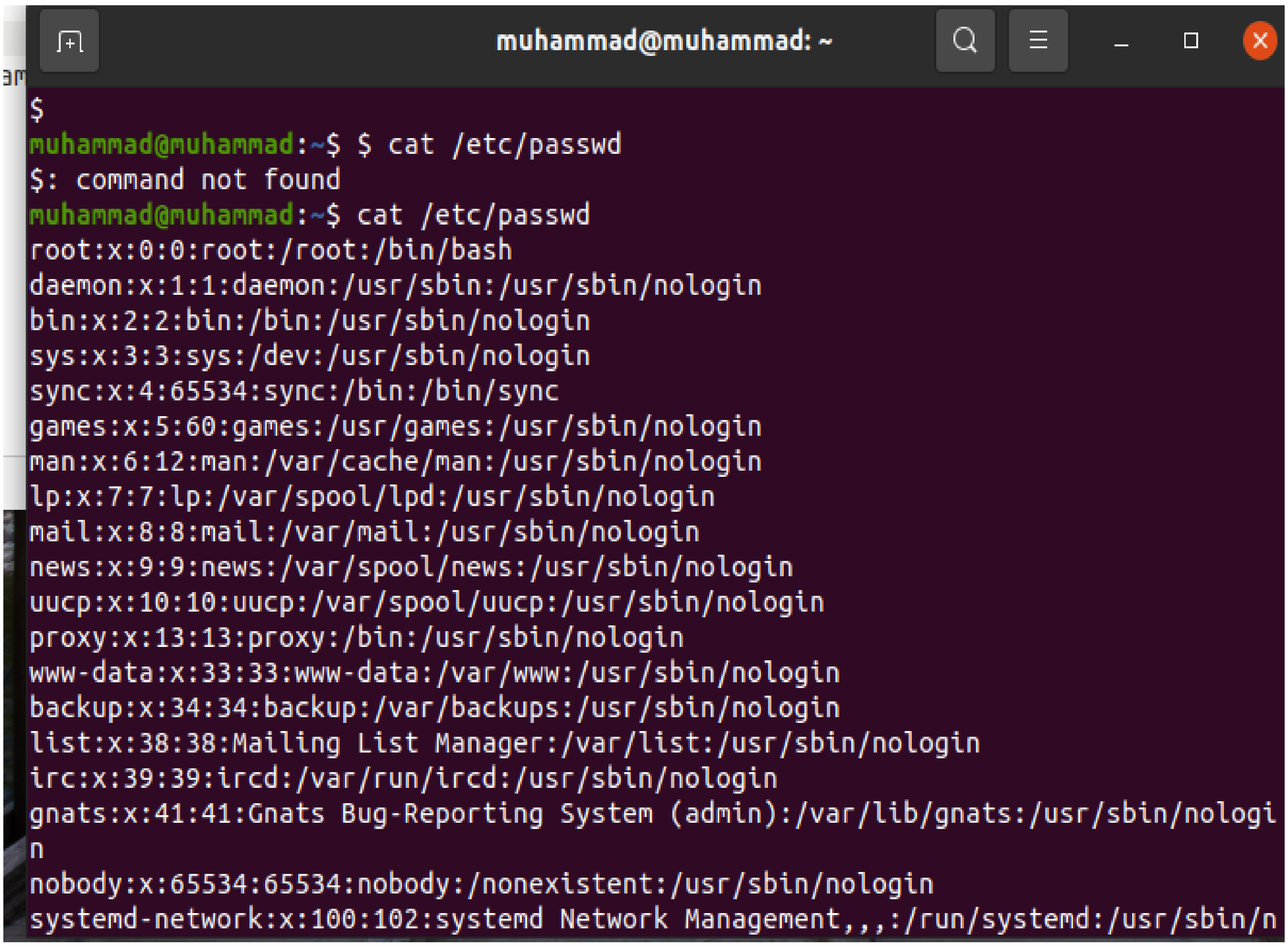
```
muhammad@muhammad:~$ ./no5
bash: ./no5: Permission denied
muhammad@muhammad:~$ chmod 777 no5
muhammad@muhammad:~$ ./no5
enter file to check the availibility in cd
find
find exists.
muhammad@muhammad:~$
```

5.2. Display the names of the users logged in.

Command:

# w home/muhammad

Output:



muhammad@muhammad: ~

```
$
muhammad@muhammad:~$ $ cat /etc/passwd
$: command not found
muhammad@muhammad:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologi
n
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/n
```