INDEX

Que(1):Execution of various file/directory handling commands.

=>

Directories

File and directory paths in UNIX use the forward slash "/" to separate directory names in a path.

examples of the directory strcucture:

directory	explanation
`/`	"root" directory
`/usr`	directory usr (sub-directory of / "root" directory)
`/usr/local`	local is a subdirectory of /usr

Creating a new Directory

mkdir command creates a new directory. The command below creates a new directory named "newDir" under the current directory.

```
$ mkdir newDir
```

This command creates a new directory in user's home directory.

```
$ mkdir ~/newDir
```

The next command creates a the target directory and all the non-existing directories in the path. The command will create samtools directory, and will create "opt" directory if it does not exist. All of this will be done in user's home directory as indicated by "~/" in that path.

```
$ mkdir -p ~/opt/samtools
```

Moving around the file system

CD :- ***cd*** command stands for "change directory" lets you move around the file system.

Type variants of these to your shell to move around your file system.

Command arguments	+ explanation
`pwd`	Show the "present working directory", or current directory.
`cd`	Change current directory to your HOME directory.
`cd /usr/local`	Change current directory to /usr/local
`cd INIT`	change current directory to INIT which is a sub-directory of the current
`cd ..`	Change current directory to the parent directory of the current
`cd ~akalin`	Change the current directory to the user akalin's home directory (if you have permission).

Listing directory contents

ls command lists the contets of a directory. It can take multiple options.

commands	explanation
`ls`	list a directory
`ls -l`	list a directory in detailed format including **file sizes** and **permissions**
`ls -a`	List the current directory including hidden files. Hidden files start with "."
`ls -ld *`	List all the file and directory names in the current directory using long format. Without the "d" option, ls would list the contents of any sub-directory of the current. With the "d" option, ls just lists them like regular files.
`ls -lh`	list detailed format this time file sizes are human readable not in bytes

Moving, renaming and copying files

cp command **copies** the files and ***mv*** command **moves** the files. They are generally used with two main arguments. `cp target_file destination_file` or `mv target_file destination_file`.

commands	explanation
`cp file1 file2`	copy file1 as file2
`cp /data/seq_data/file1 ~/`	copy file1 at /data/seq_data to your home directory.
`mv file1 newname`	move or rename a file
`mv file1 ~/opt/`	move file1 into sub-directory opt in your home directory.

Finding files

There are a couple of ways you can find files in your file system. We will show the **find** command, it works in the following syntax `find directory -name targetfile`. It is useful when you have a rough idea about file location.

The following finds all files ending in ".html" under /home/user directory.

```
$ find /home/user -name "*.html"
```

find can also do more than just finding files. It also execute commands on the files you find via -exec option. The following command finds all files in the current directory with ".txt" ending and counts the number of lines in every text file. The '{}' is replaced by the name of each file found and the ';' ends the -exec clause.

```
$ find . -name "*.txt" -exec wc -l '{}' ';'
```

Another command that can find files is **locate**. The locate command provides a faster way of locating all files whose names match a particular search string. For example:

```
 $ locate ".txt"
```

will find all filenames in the filesystem that contain ".txt" anywhere in their full paths.

A disadvantage of locate is that it stores all filenames on the system in an index that is usually updated only once a day. This means locate will not find files that have been created very recently.

Searching the contents of a text file

Often times you would need search a file for existence of certain characters or words. Imagine that you need to find gene ids in a text file containing some scores and gene ids, you would like to get the line(s) that contains your gene id of interest. This is similar to "find" functions in modern text processors such as MS Word. This can be achieved via **grep commmand**. Syntax of the command is: `grep options pattern files`

Command	Explanation
`grep id1 genes.txt`	searches and prints lines matching "id1" in "genes.txt"
`grep id1 *.txt`	searches and prints lines matching "id1" in files ending with ".txt"
`grep -vi id1 *.txt`	similar to above, but -i option ignores the case (Id1,ID1,iD1 and id1 treated equally), -v option prints lines that don't match the pattern

using grep and find together

You can search all files in an entire directory tree for a particular pattern by combining **grep** and **find**. The following command prints lines containing "genes" string, from the files 'find . -name "*.txt" -print' found.

```
$ grep genes `find . -name "*.txt" -print`
```

Deleting files and directories

To remove a directory that contains other files or directories, use the following command.

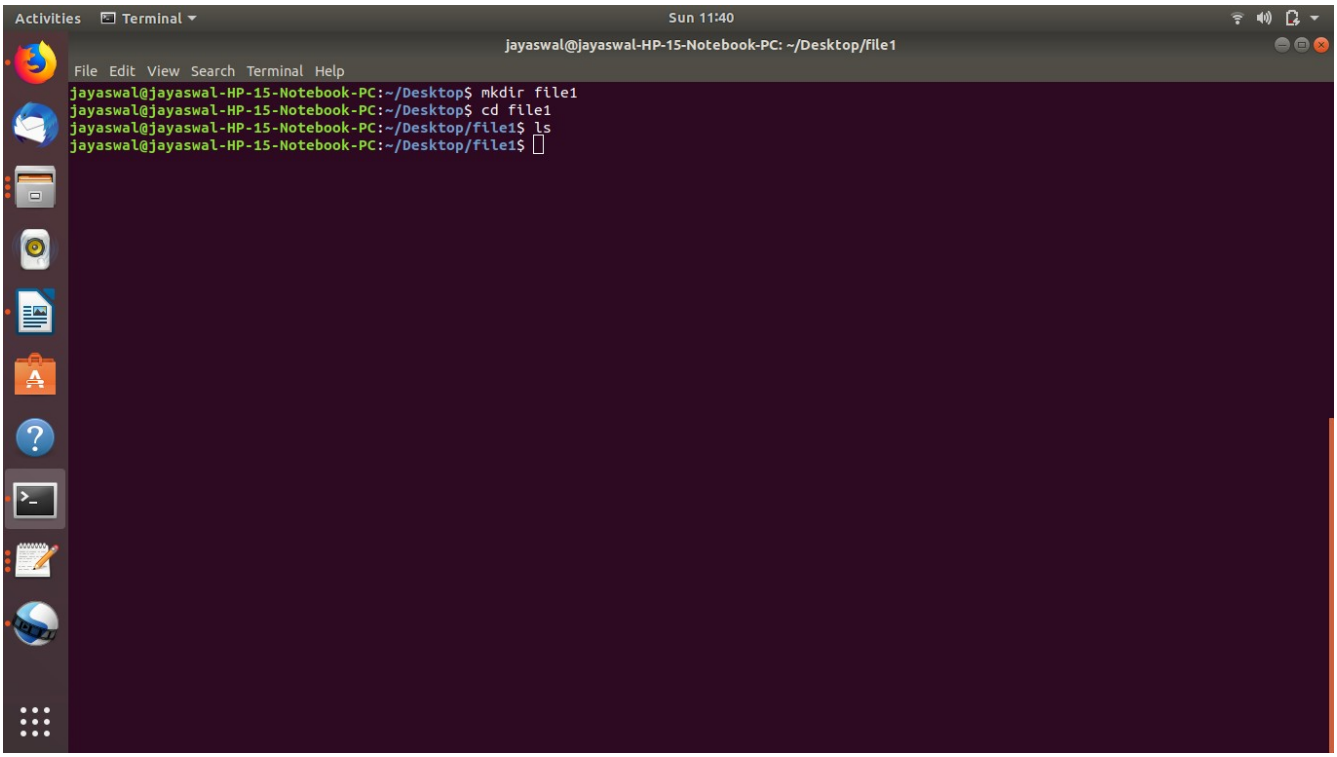
rm -r mydir

In the example above, you would replace "mydir" with the name of the directory you want to delete. For example, if the directory was named **files**, you would type **rm -r files** at the prompt.

rm -rf mydir

In the example above, the "mydir" directory, along with all files and directories in that directory, would be deleted with no prompt or message.

Output:

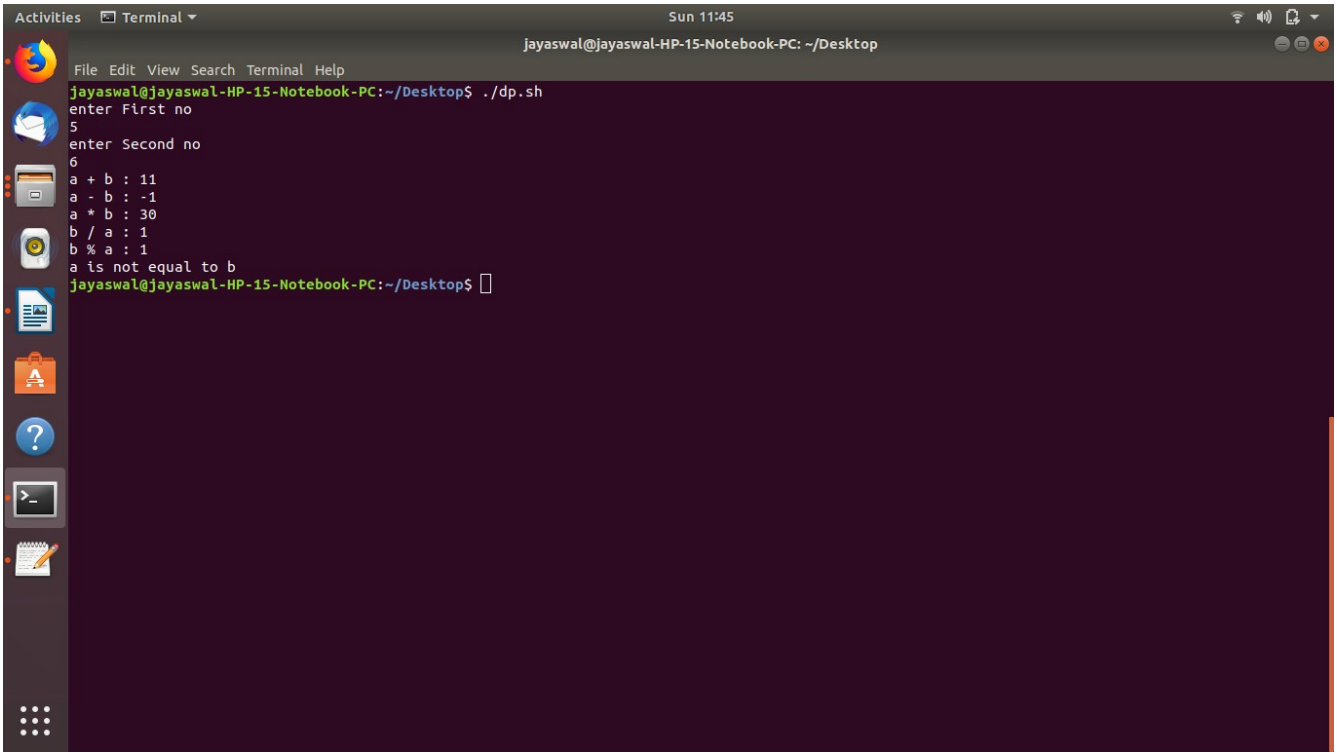


Que(2):Simple shell script for basic arithmetic and logical calculations.

=>

```
#!/bin/bash
echo "enter First no"
read a
echo "enter Second no"
read b
val=`expr $a + $b`
echo "a + b : $val"
val=`expr $a - $b`
echo "a - b : $val"
val=`expr $a \* $b`
echo "a * b : $val"
val=`expr $b / $a`
echo "b / a : $val"
val=`expr $b % $a`
echo "b % a : $val"
if [ $a == $b ]
then
   echo "a is equal to b"
fi

if [ $a != $b ]
then
   echo "a is not equal to b"
fi
```

Output:



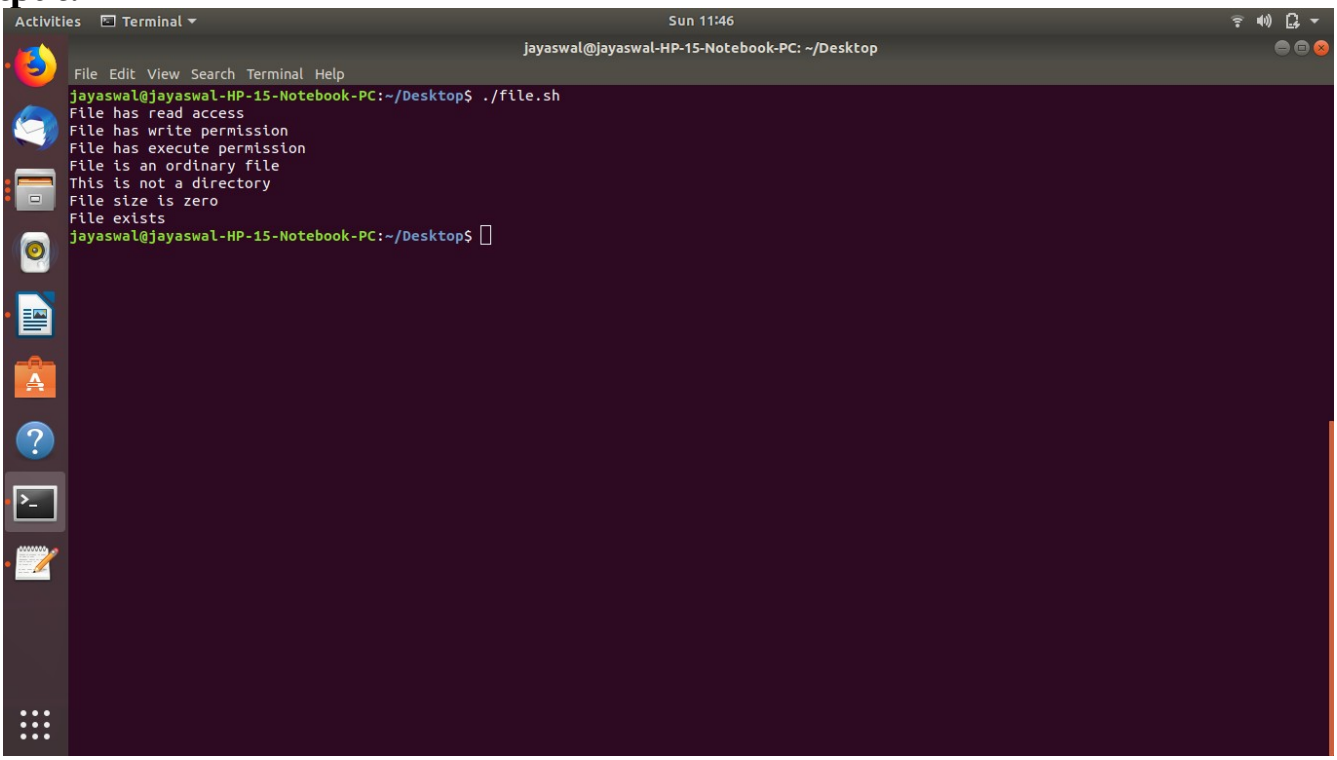
Que(3): Shell scripts to check various attributes of Files and Directories.

=>

```
#!/bin/sh

file="/home/jayaswal/Desktop/dp.sh"

if [ -r $file ]
then
  echo "File has read access"
else
  echo "File does not have read access"
fi

if [ -w $file ]
then
  echo "File has write permission"
else
  echo "File does not have write permission"
fi

if [ -x $file ]
then
  echo "File has execute permission"
else
  echo "File does not have execute permission"
fi
```

```
if [ -f $file ]
then
  echo "File is an ordinary file"
else
  echo "This is sepcial file"
fi

if [ -d $file ]
then
  echo "File is a directory"
else
  echo "This is not a directory"
fi

if [ -s $file ]
then
  echo "File size is zero"
else
  echo "File size is not zero"
fi

if [ -e $file ]
then
  echo "File exists"
else
  echo "File does not exist"
fi
```

Output:

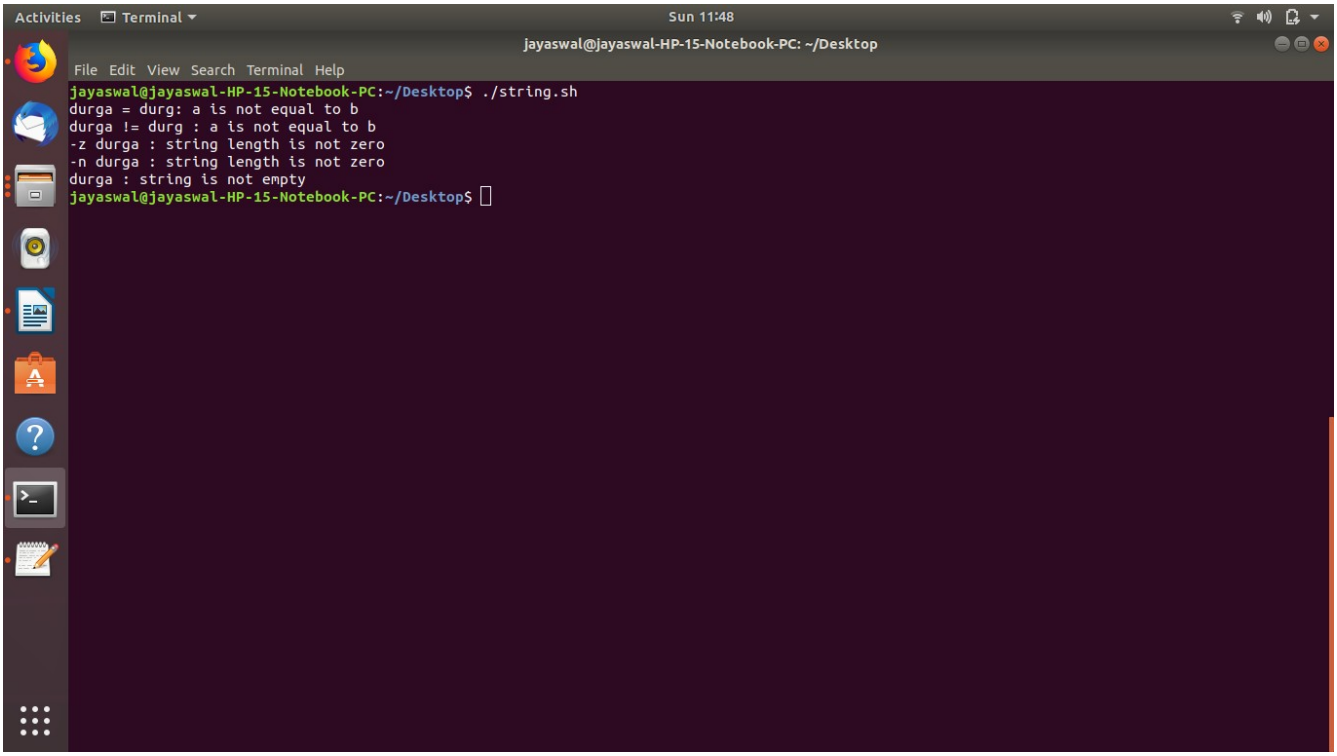


Que(4): Shell scripts to perform various operations on given strings.

=>

```
#!/bin/sh

a="durga"
b="durg"

if [ $a = $b ]
then
   echo "$a = $b : a is equal to b"
else
   echo "$a = $b: a is not equal to b"
fi

if [ $a != $b ]
then
   echo "$a != $b : a is not equal to b"
else
   echo "$a != $b: a is equal to b"
fi

if [ -z $a ]
then
   echo "-z $a : string length is zero"
else
   echo "-z $a : string length is not zero"
fi

if [ -n $a ]
then
   echo "-n $a : string length is not zero"
else
   echo "-n $a : string length is zero"
fi

if [ $a ]
then
   echo "$a : string is not empty"
else
   echo "$a : string is empty"
fi
```

Output:

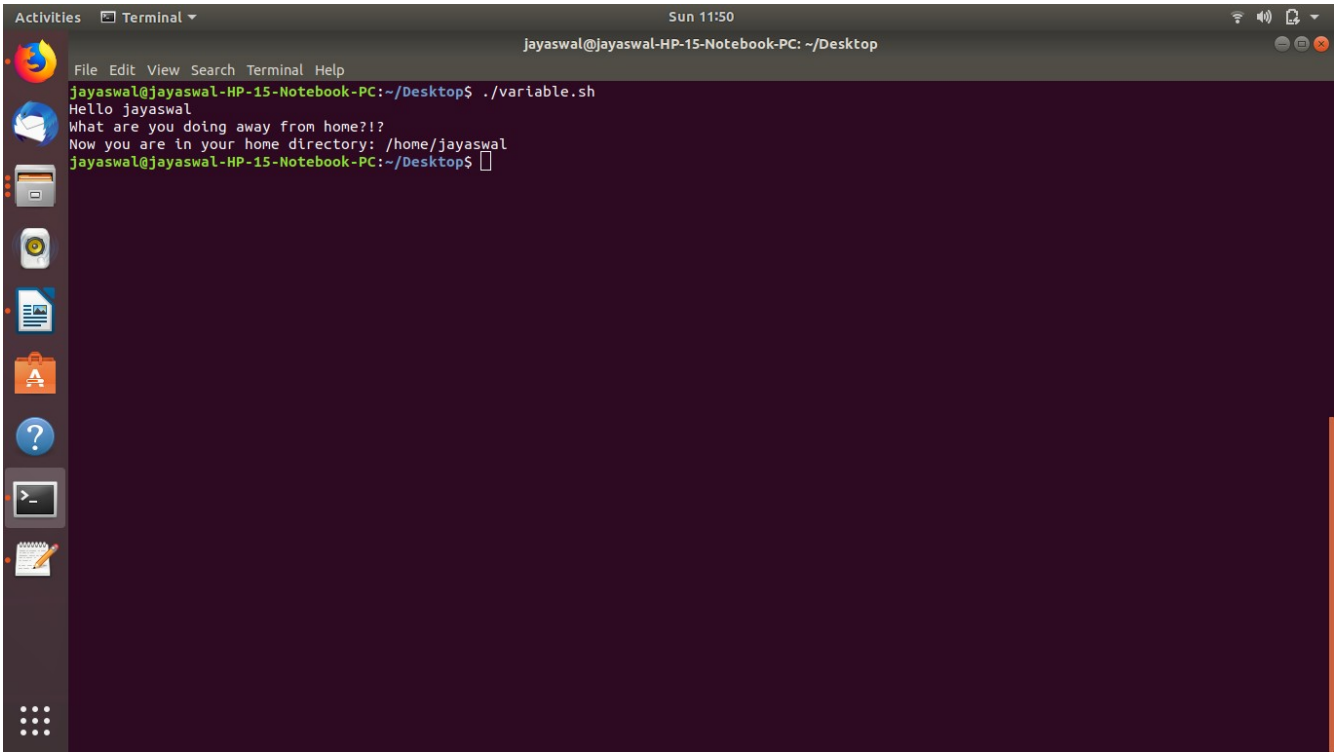


Que(5): Shell scripts to explore system variables such as PATH,HOME etc.

=>

```
#!/bin/bash

echo "Hello $USER"

if [ $HOME == $PWD ]
then
   echo "Good, you're in your home directory: $HOME"
else
   echo "What are you doing away from home?!?"

   cd $HOME

   echo "Now you are in your home directory: $PWD"
fi
```

Output:

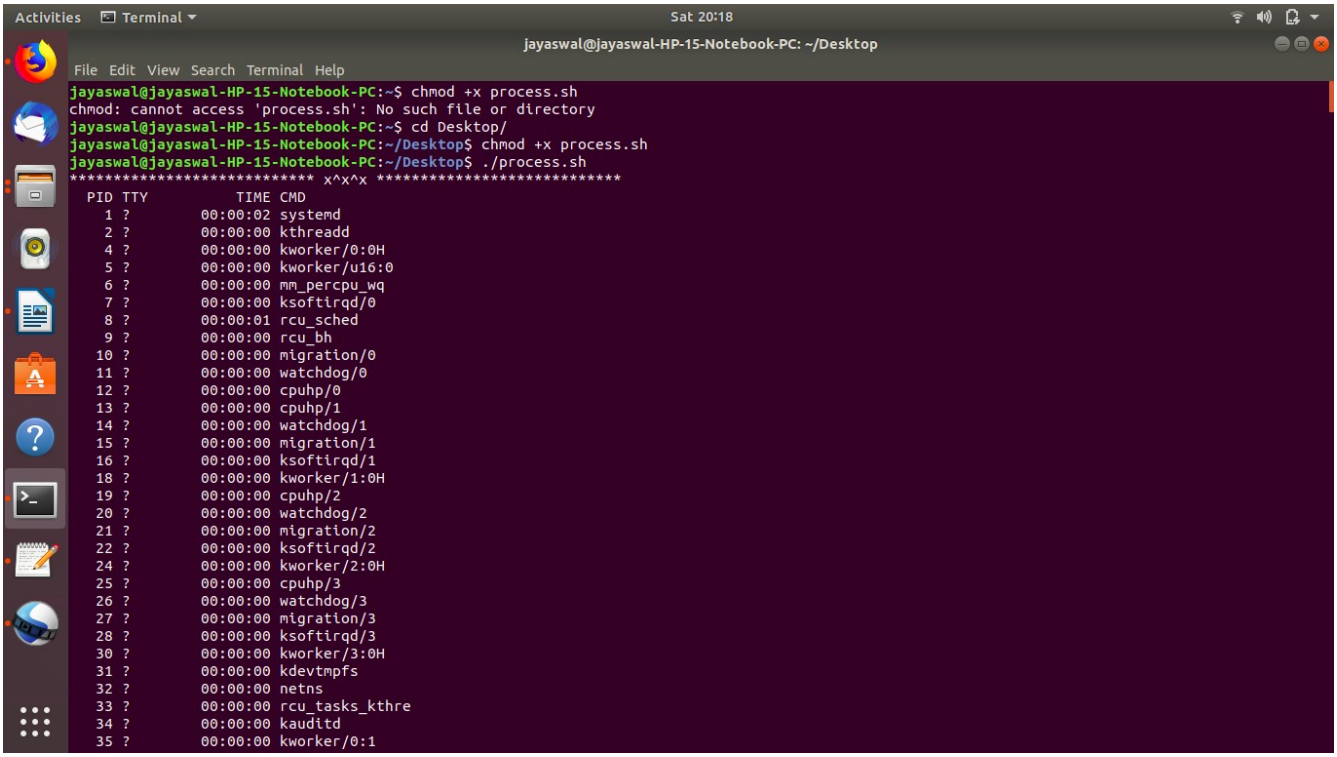


Que(6): Shell scripts to check and list attributes of processes.

=>

```
#!/bin/bash
# Write a shell script to display the process running on the system for every
# 30 seconds, but only for 3 times.
# ----------------------------------------------------------------------
# Copyright (c) 2008 nixCraft project <http://www.cyberciti.biz/fb/>
# This script is licensed under GNU GPL version 2.0 or above
# ----------------------------------------------------------------------
# This script is part of nixCraft shell script collection (NSSC)
# Visit http://bash.cyberciti.biz/ for more information.
# ----------------------------------------------------------------------
#
# for loop 3 times
for r in 1 2 3
do
 #see every process on the system
 echo "***************************** x^x^x *****************************"
 ps -e
 echo "***************************** x^x^x *****************************"
 #sleep for 30 seconds
 sleep 3
       # clean
done
```

Output :

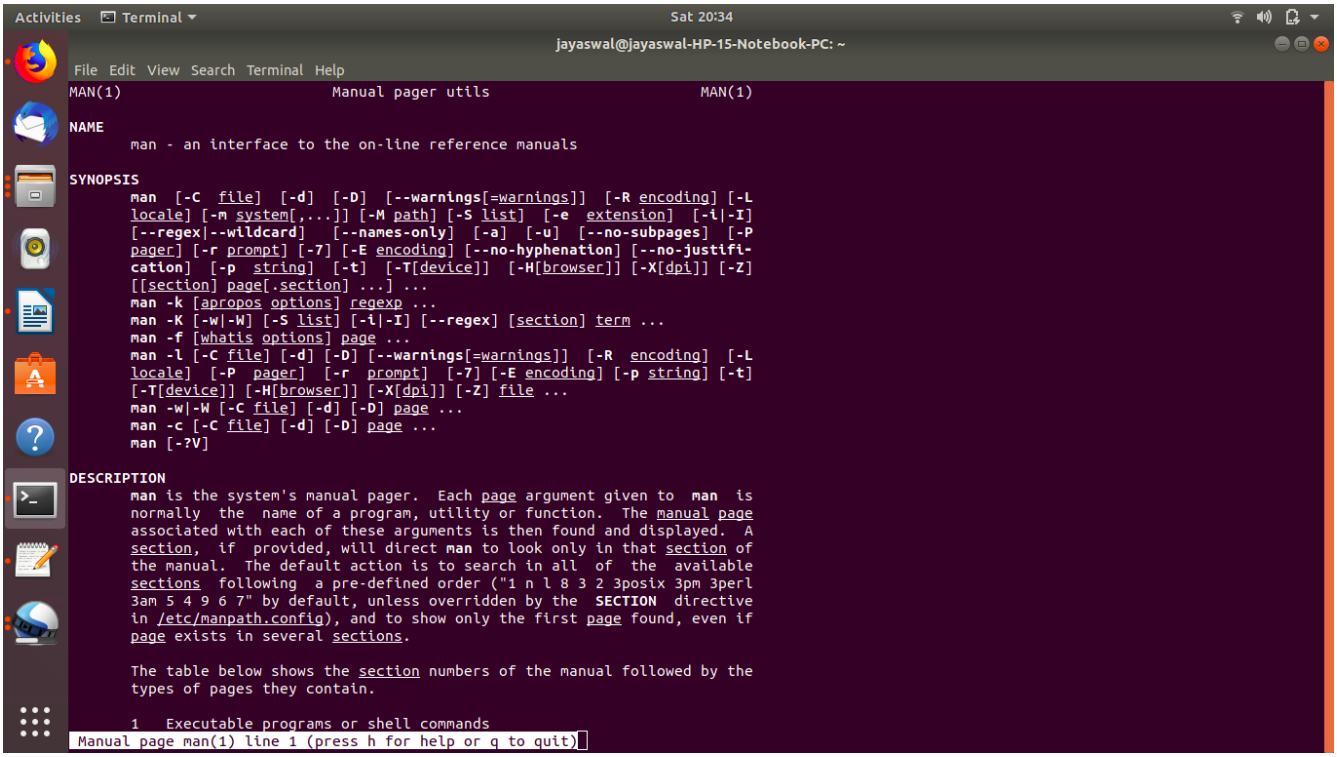


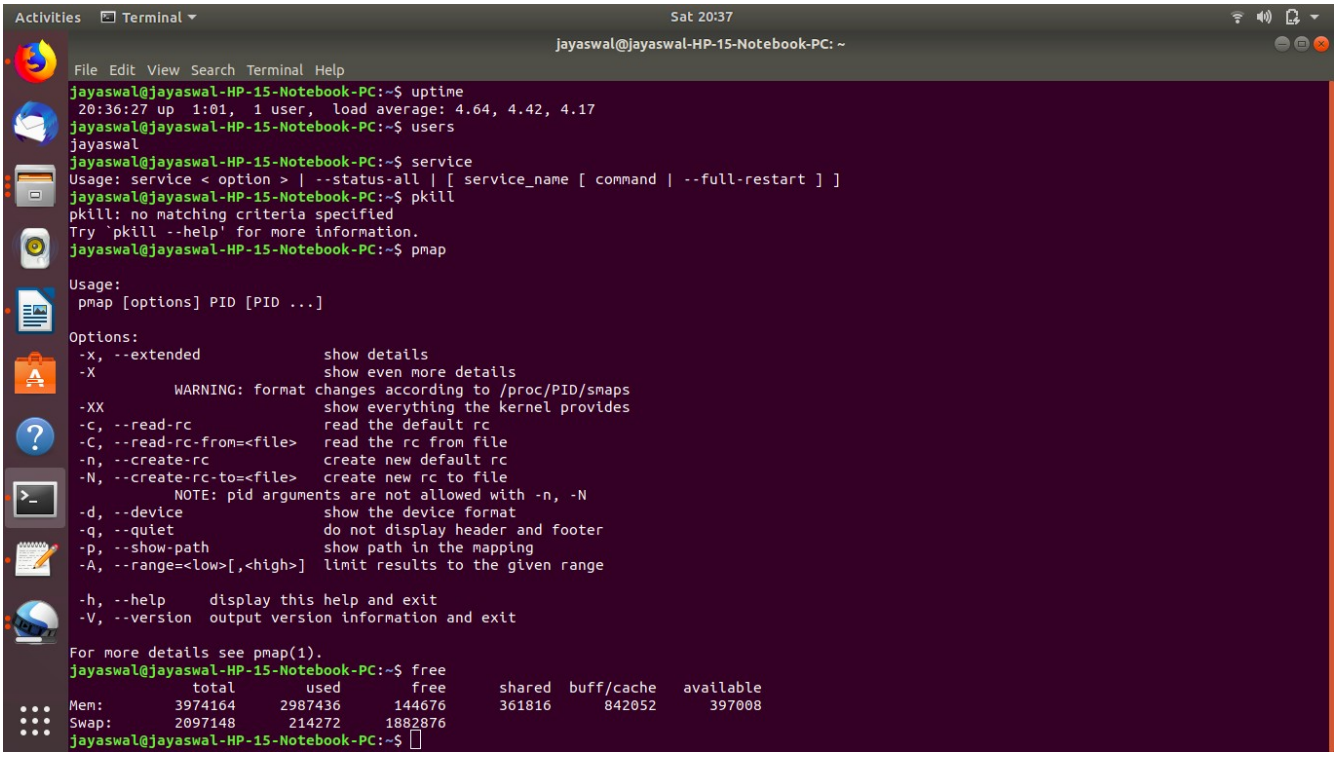
Que(7): Execution of various system administrative commands.
=>

Command	**Function**
man	Display information about all commands
uptime	Show how long system is running
users	Show username who are currently logged in
service	Call and execute script
pkill	Kill a process
pmap	Memory map of a process
wget	Download file from network
ftp or sftp	Connect remote ftp host
free	Show memory status
top	Display processor activity of system
last	Display user's activity in the system
ps	Display about processes running on the system
Shutdown commands	Shutdown and reboot system
info	Display information about given command
env	Display environment variable for currently logged-in user
netstat	Display network status
arp	Check ethernet connectivity and IP address
df	Display filesystem information
du	Display usage
init	Allow to change server bootup

nano	A command line editor
nslookup	Check domain name and IP information
shred	Delete a file by over writing its content
cat	Display, copy or combine text files
pwd>	Print path of current working directory
locate	Finding files by name on system
chown	Change ownership of a file
>alias	To short a command
echo	Display text
cmp	Compare two files byte by byte
mount	Mount a filesystem
ifconfig	Display configuration
traceroute>	Trace existing network
sudo	Run a command as a root user
route	List routing table for your server
ping	Check connection by sending packet test packet
find	Find location of files/directories
users	Show current logged in user
who	Same as w but doesn't show current process
ls	List all the files
tar	Compress directories
grep	Search for a string in a file
su	Switch from one to another user
awk	Search lines for a given pattern

Man :


Activities
Terminal
Sat 20:34
jayaswal@jayaswal-HP-15-Notebook-PC: ~
File Edit View Search Terminal Help
MAN(1) Manual pager utils MAN(1)
NAME
man - an interface to the on-line reference manuals
SYNOPSIS
man [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-m system[,...]] [-M path] [-S list] [-e extension] [-i|-I] [--regex|--wildcard] [--names-only] [-a] [-u] [--no-subpages] [-P pager] [-r prompt] [-7] [-E encoding] [--no-hyphenation] [--no-justification] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] [[section] page[.section] ...] ...
man -k [apropos options] regexp ...
man -K [-w|-W] [-S list] [-i|-I] [--regex] [section] term ...
man -f [whatis options] page ...
man -l [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-P pager] [-r prompt] [-7] [-E encoding] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] file ...
man -w|-W [-C file] [-d] [-D] page ...
man -c [-C file] [-d] [-D] page ...
man [-?V]
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 ("1 n l 8 3 2 3posix 3pm 3perl 3am 5 4 9 6 7" by default, unless overridden by the SECTION directive in /etc/manpath.config), 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
Manual page man(1) line 1 (press h for help or q to quit)


Activities
Terminal
Sat 20:37
jayaswal@jayaswal-HP-15-Notebook-PC: ~
File Edit View Search Terminal Help
jayaswal@jayaswal-HP-15-Notebook-PC:~$ uptime
20:36:27 up 1:01, 1 user, load average: 4.64, 4.42, 4.17
jayaswal@jayaswal-HP-15-Notebook-PC:~$ users
jayaswal
jayaswal@jayaswal-HP-15-Notebook-PC:~$ service
Usage: service < option > | --status-all | [service_name [command | --full-restart]]
jayaswal@jayaswal-HP-15-Notebook-PC:~$ pkill
pkill: no matching criteria specified
Try `pkill --help' for more information.
jayaswal@jayaswal-HP-15-Notebook-PC:~$ pmap
Usage:
pmap [options] PID [PID ...]
Options:
-x, --extended show details
-X show even more details
WARNING: format changes according to /proc/PID/smaps
-XX show everything the kernel provides
-c, --read-rc read the default rc
-C, --read-rc-from=<file> read the rc from file
-n, --create-rc create new default rc
-N, --create-rc-to=<file> create new rc to file
NOTE: pid arguments are not allowed with -n, -N
-d, --device show the device format
-q, --quiet do not display header and footer
-p, --show-path show path in the mapping
-A, --range=<low>[,<high>] limit results to the given range
-h, --help display this help and exit
-V, --version output version information and exit
For more details see pmap(1).
jayaswal@jayaswal-HP-15-Notebook-PC:~$ free
total used free shared buff/cache available
Mem: 3974164 2987436 144676 361816 842052 397008
Swap: 2097148 214272 1882876
jayaswal@jayaswal-HP-15-Notebook-PC:~$

Que(8): Write awk script that uses all of its feature.

=>

Syntax:

```
awk '/search pattern1/ {Actions}
     /search pattern2/ {Actions}' file
```

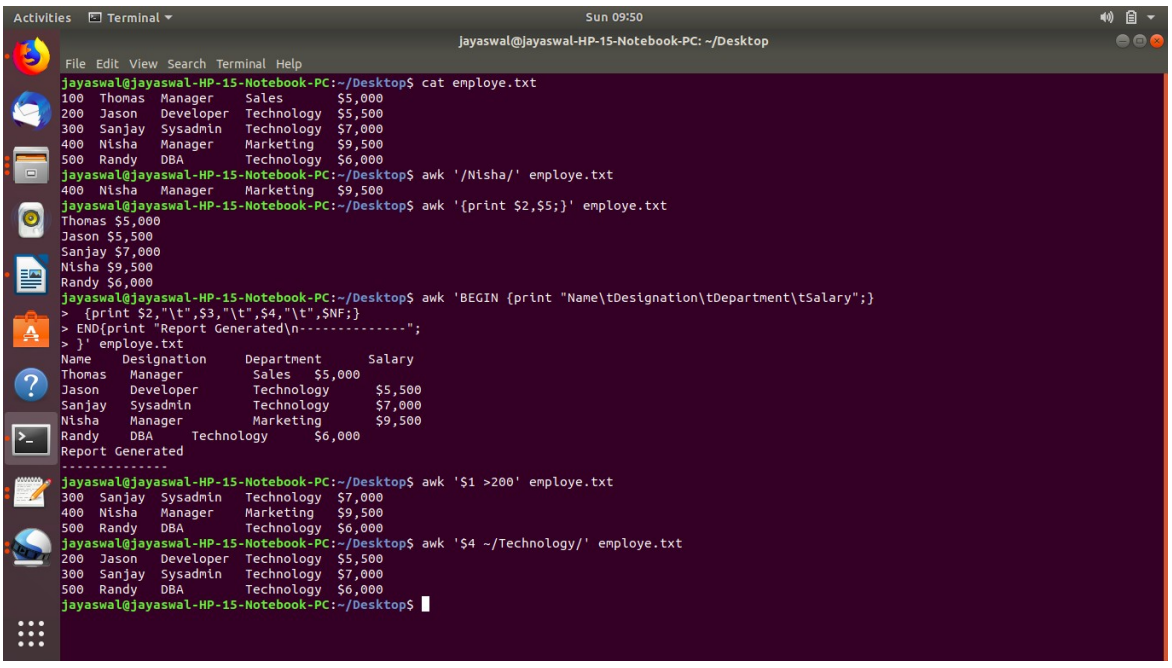
In the above awk syntax:

- search pattern is a regular expression.
- Actions – statement(s) to be performed.
- several patterns and actions are possible in Awk.
- file – Input file.
- Single quotes around program is to avoid shell not to interpret any of its special characters.

Awk Working Methodology

1. Awk reads the input files one line at a time.
2. For each line, it matches with given pattern in the given order, if matches performs the corresponding action.
3. If no pattern matches, no action will be performed.
4. In the above syntax, either search pattern or action are optional, But not both.
5. If the search pattern is not given, then Awk performs the given actions for each line of the input.
6. If the action is not given, print all that lines that matches with the given patterns which is the default action.
7. Empty braces with out any action does nothing. It wont perform default printing operation.
8. Each statement in Actions should be delimited by semicolon.

Output:



Que(9):Use sed instruction to process /*etc*/password file.

=>

SED :- Sed is a Stream Editor used for modifying the files in unix (or linux). Whenever you want to make changes to the file automatically, sed comes in handy to do this.

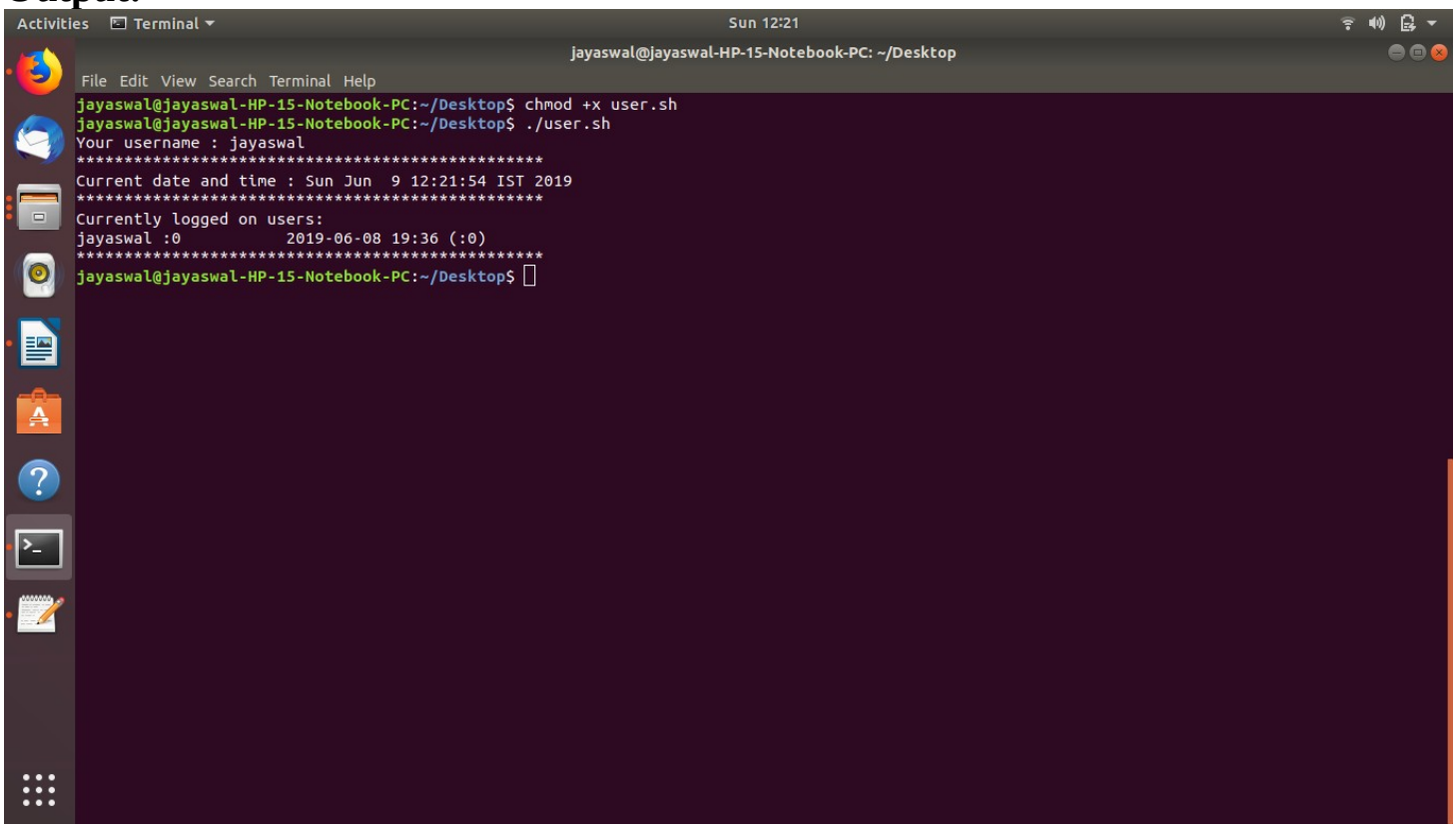
1. Replacing or substituting string
2. Replacing the nth occurrence of a pattern in a line.
3.Replacing all the occurrence of the pattern in a line.
4. Replacing from nth occurrence to all occurrences in a line.
5. Changing the slash (/) delimiter
6. Using & as the matched string
7. Using \1,\2 and so on to \9
8.Duplicating the replaced line with /p flag
9. Printing only the replaced lines
10. Running multiple sed commands.
11. Replacing string on a specific line number.
12. Replacing string on a range of lines.
13. Replace on a lines which matches a pattern.
14. Deleting lines.
15. Duplicating lines
16. Sed as grep command
17. Add a line after a match.
18. Add a line before a match
19. Change a line
20. Transform like tr command

Que(10): Write a shell scripts to display list of users currently logged in.

=>

```
#!/bin/bash
# Write a shell script called hello which output the following:
# + Your username
# + The time and date
# + Who is logged on
# + also output a line of asterices (*******) after each section

# function to display a line of asterices
function line(){
 echo "**************************************************"
}

echo "Your username : $(echo $USER)"
line # call function

echo "Current date and time : $(date)"
line

echo "Currently logged on users:"
who
line
```

Output:

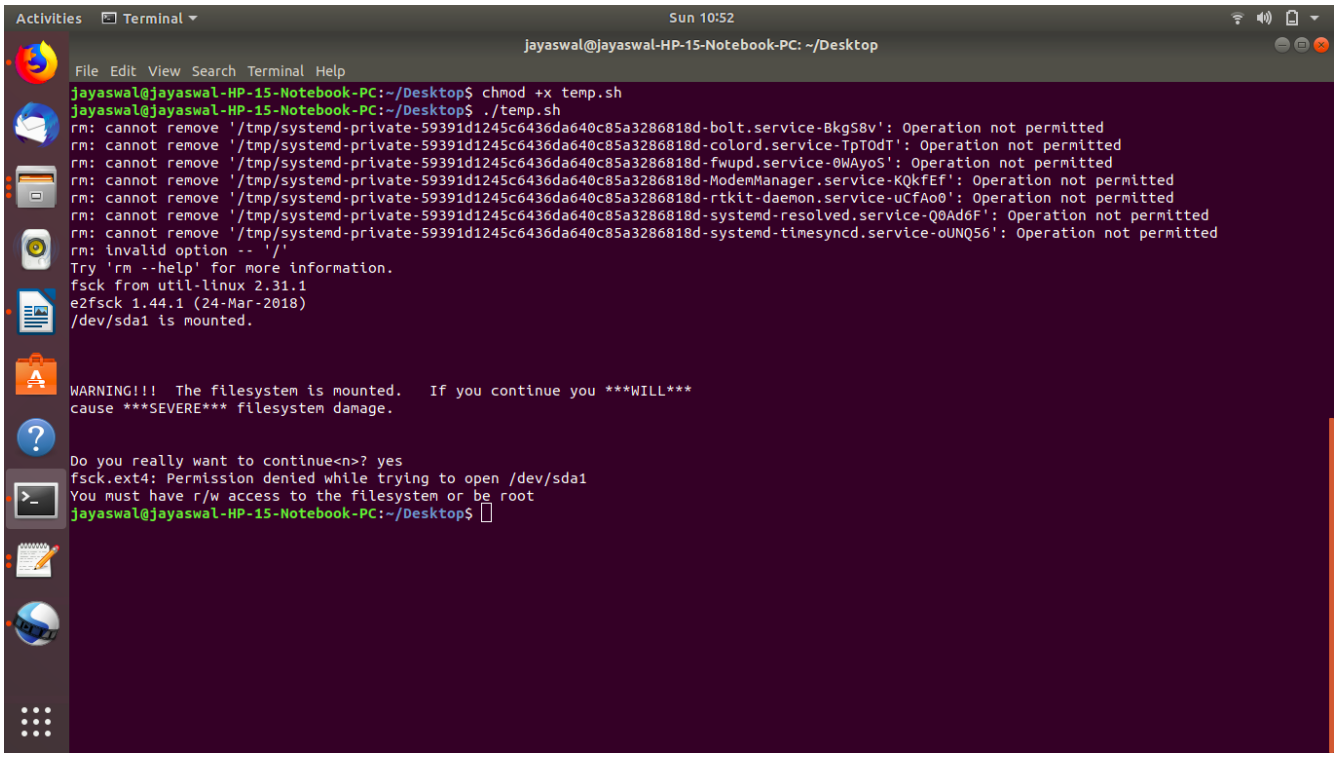


Que(11): Write a shell script to delete all the temporary files.

=>

```
#!/bin/bash
#This script will automatically delete temporary files
rm -rf /tmp/*
rm -rf/var/tmp/*
fsck -A
exit
```

Output:



Que(12):Write a shell script to search an element from an array using binary searching.

=>

echo “Enter the limit:”

read n

echo “Enter the numbers”

for((i=0 ;i<n; i++))

do

read m

a[i]=$m

done

for((i=1; i<n; i++))

do

for((j=0; j<n-i; j++))

do

```
if [ ${a[$j]} -gt ${a[$j+1]} ]
then
t=${a[$j]}
a[$j]=${a[$j+1]}
a[$j+1]=$t
fi
done
done
echo “Sorted array is”
for(( i=0; i<n; i++ ))
do
echo “${a[$i]}”
done
echo “Enter the element to be searched :”
read s
l=0
c=0
u=$(($n-1))
while [ $l -le $u ]
do
mid=$(((( $l+$u ))/2 ))
if [ $s -eq ${a[$mid]} ]
then
c=1
break
```

```
elif [ $s -lt ${a[$mid]} ]
then
u=$(($mid-1))
else
l=$(($mid+1))
fi
done
if [ $c -eq 1 ]
then
echo “Element found at position $(($mid+1))”
else
echo “Element not found”
fi
```

Output :