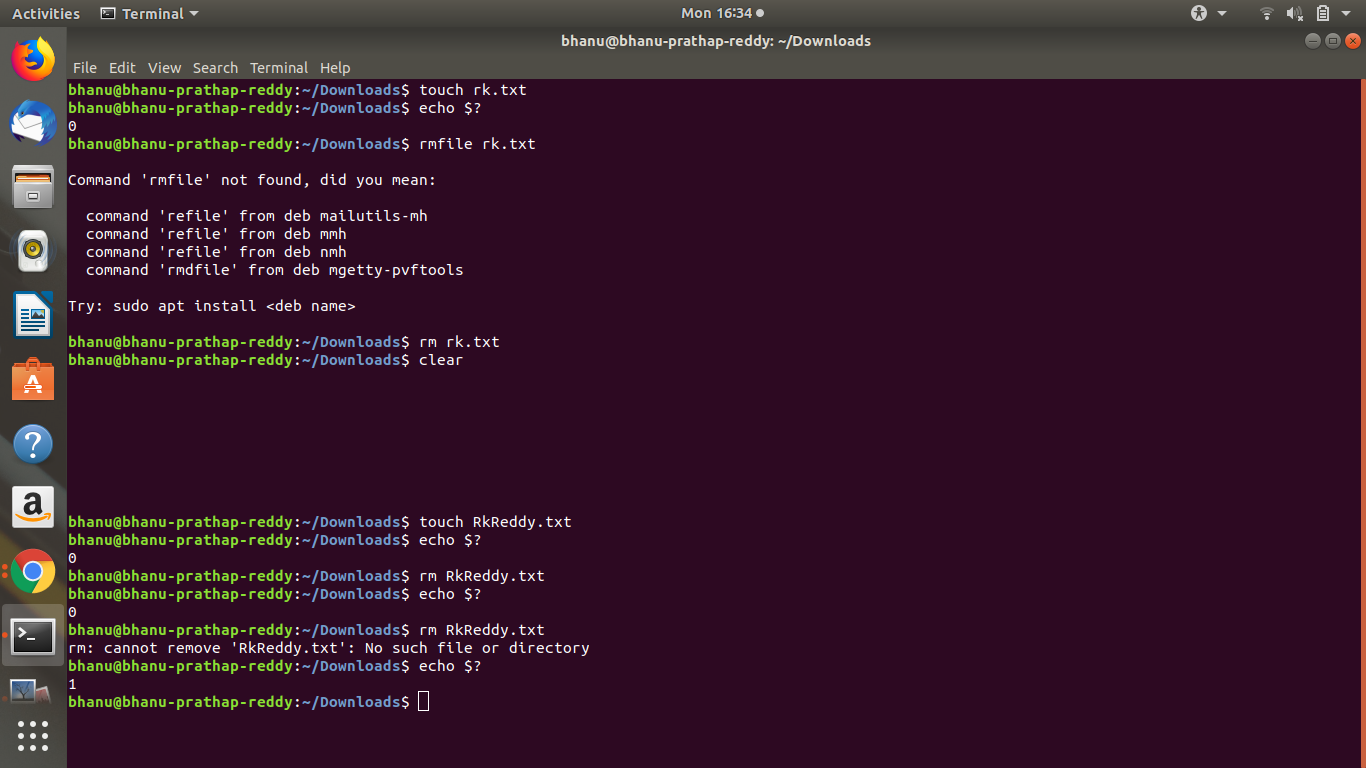
**Linux shell commands**

**1.$? dollar question mark**

The exit code of the previous command is stored in the shell variable $?. Actually $? is a

shell parameter and not a variable, since you cannot assign a value to $?.

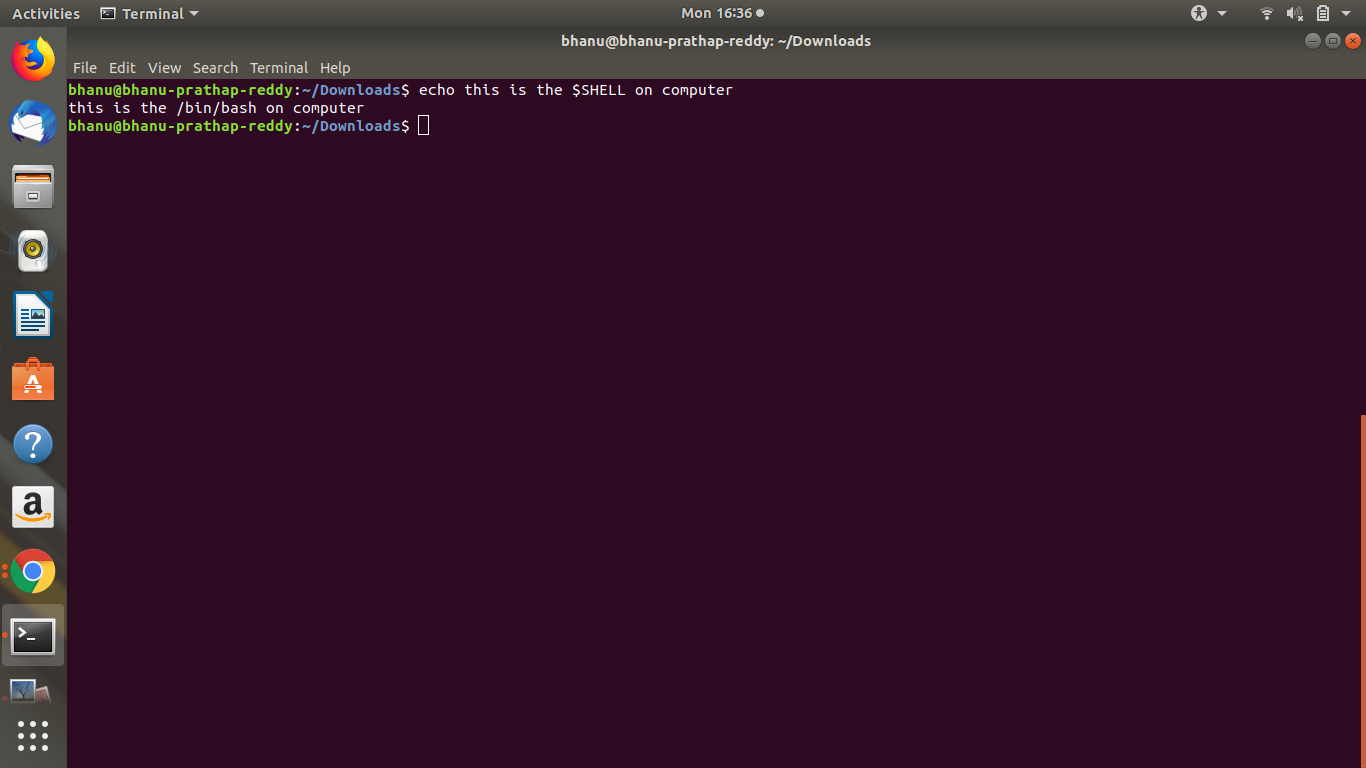


**2.$ dollar sign**

Another important character interpreted by the shell is the dollar sign $. The shell will look

for an environment variable named like the string following the dollar sign and replace it

with the value of the variable (or with nothing if the variable does not exist).

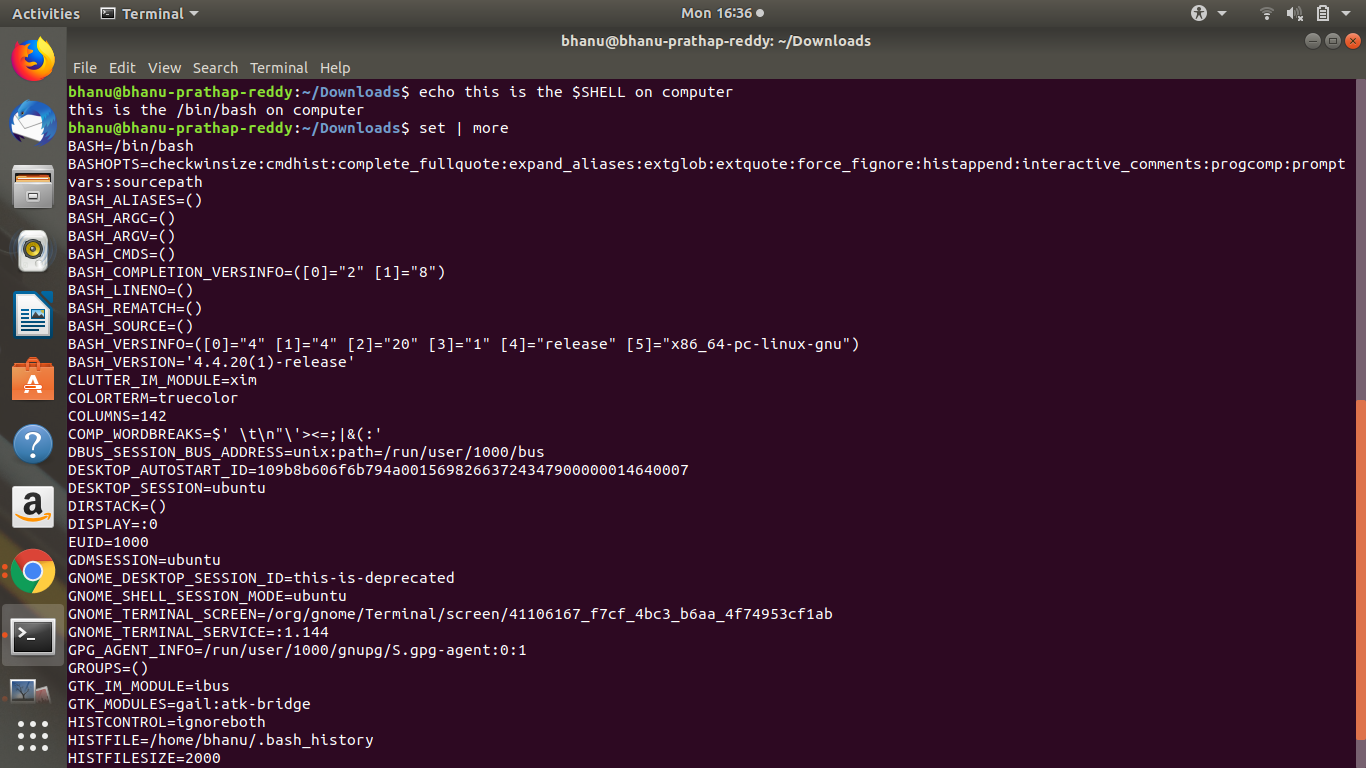


**3.set**

We can use the set command to display a list of environment variables. On Ubuntu and

Debian systems, the set command will also list shell functions after the shell variables. Use

set | more to see the variables then.



**4.unset**

Use the unset command to remove a variable from your shell environment.

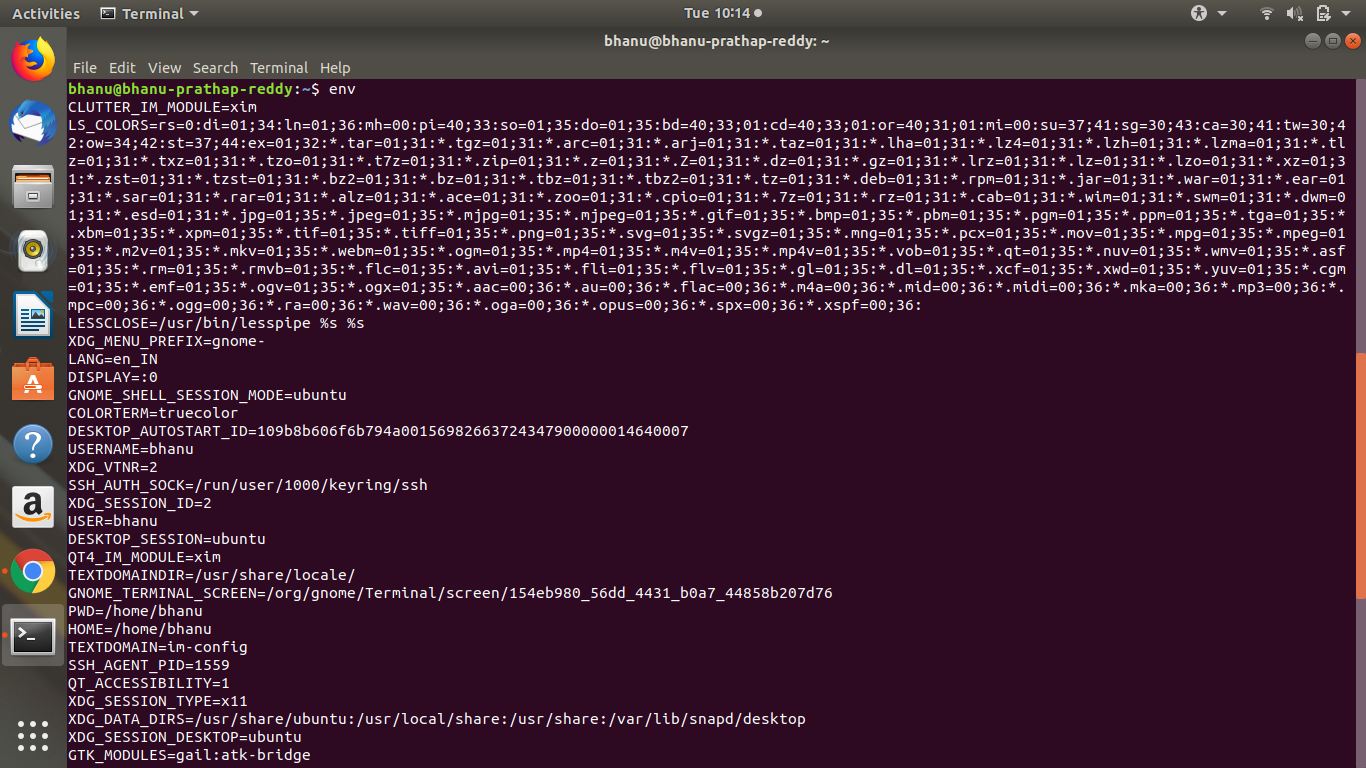
**5.env**

The env command without options will display a list of exported variables. The difference

with set with options is that set lists all variables, including those not exported to child shells.

But env can also be used to start a clean shell (a shell without any inherited environment).

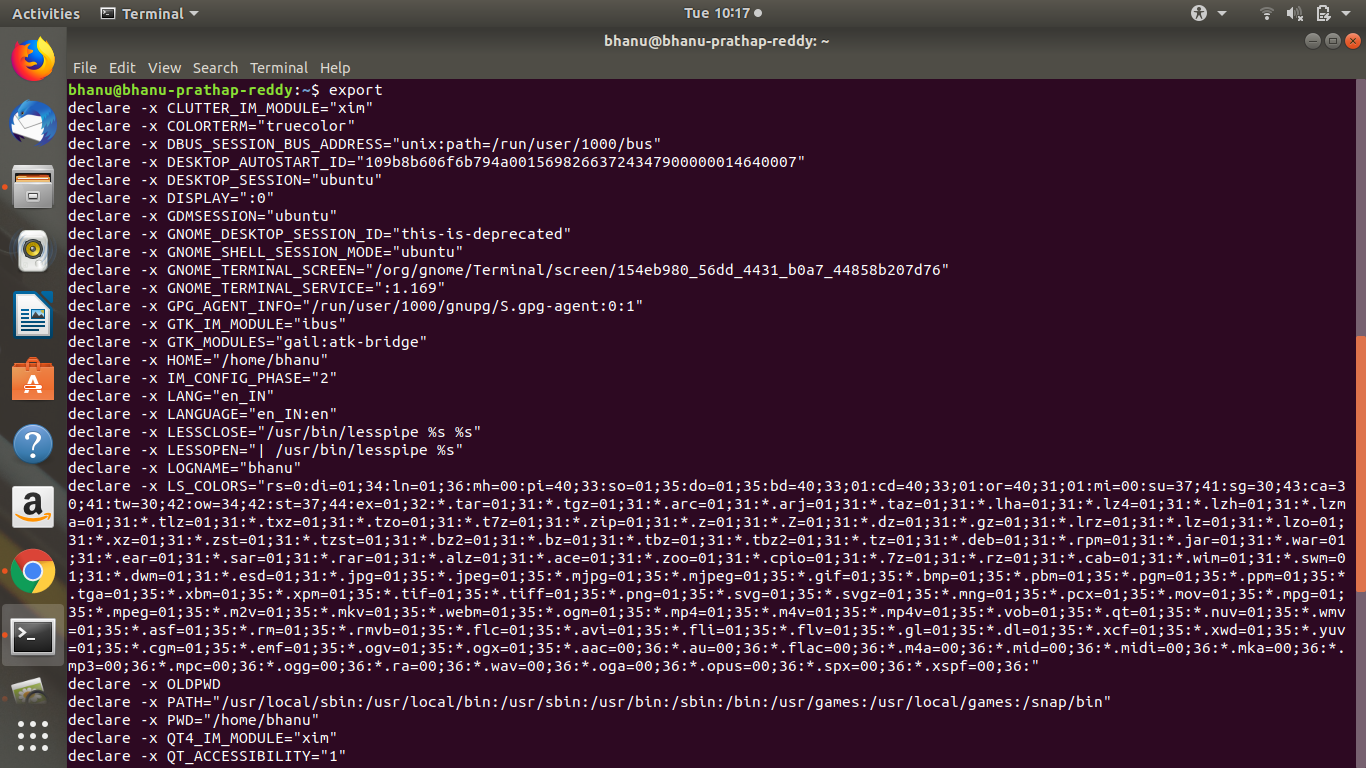
The env -i command clears the environment for the subshell.

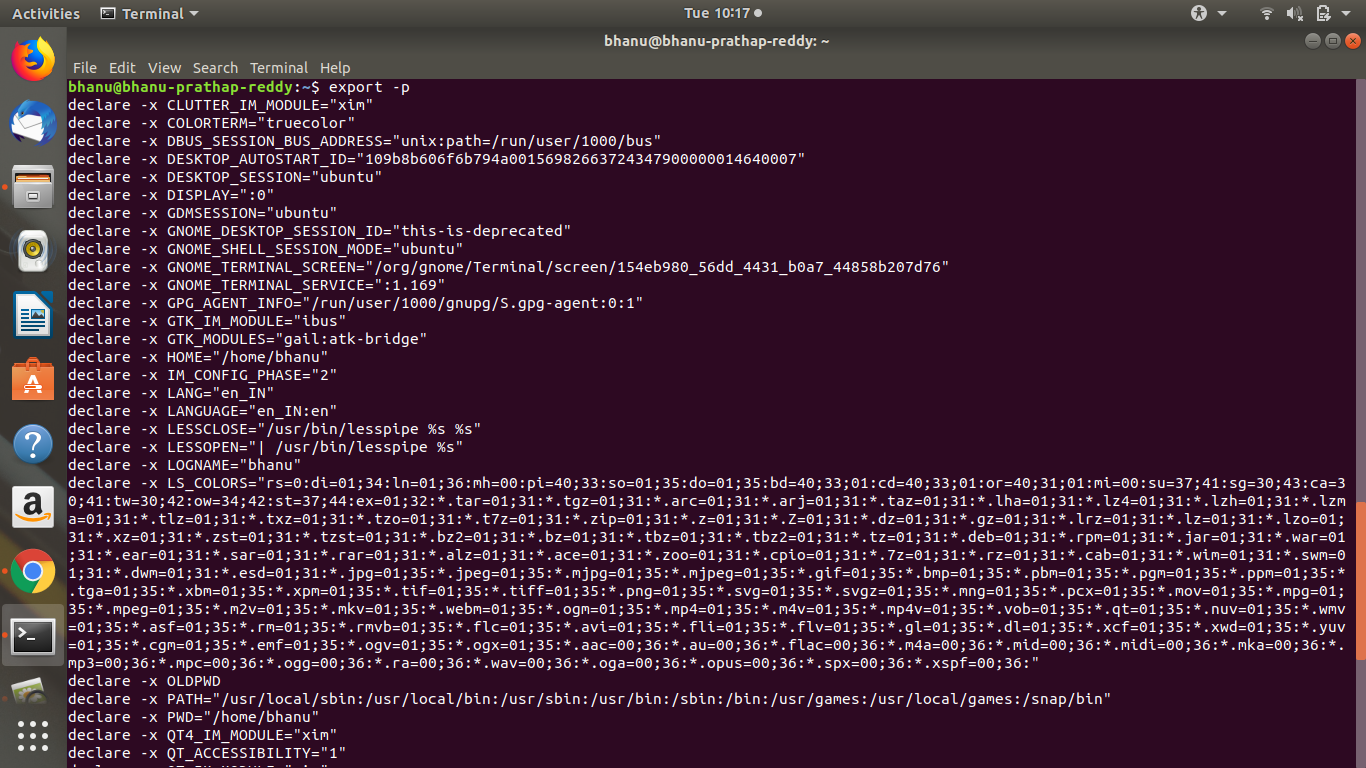


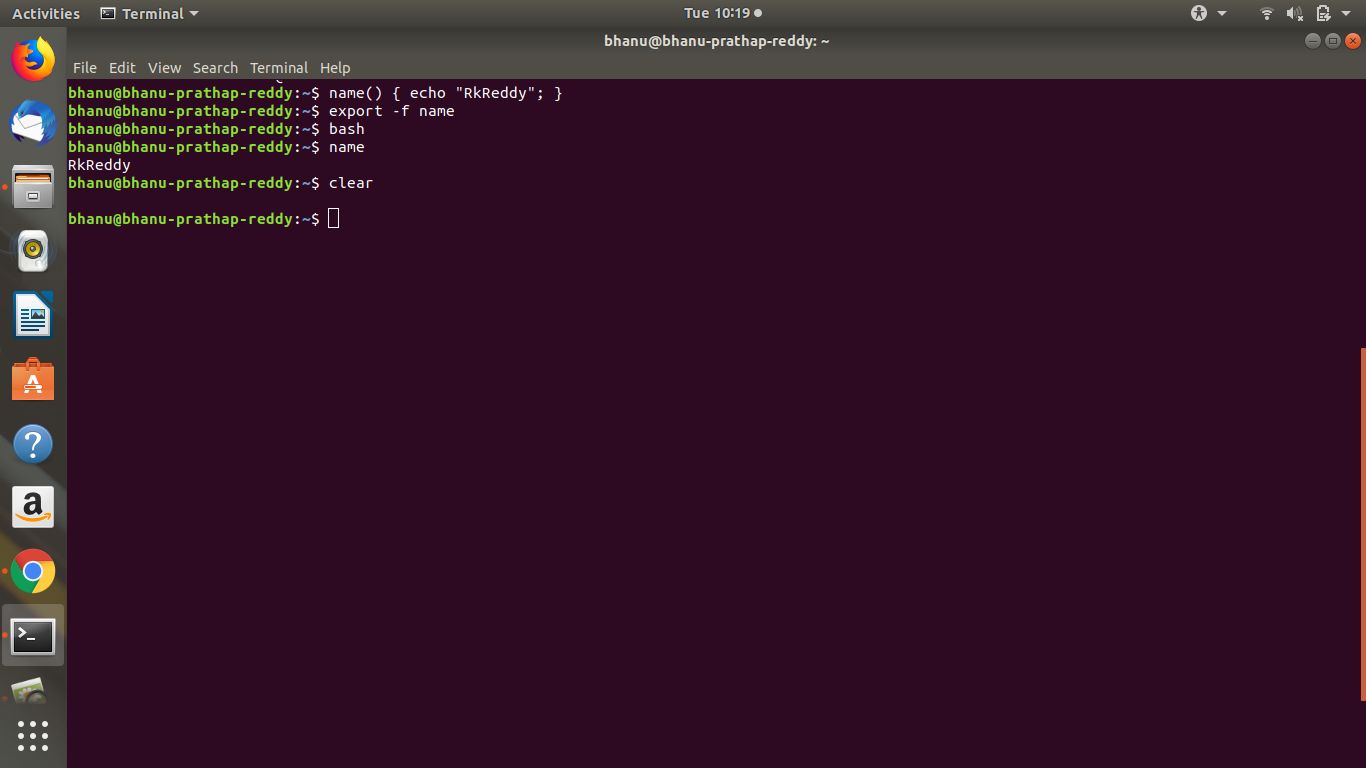
**6.export**

We can export shell variables to other shells with the export command. This will export

the variable to child shells.





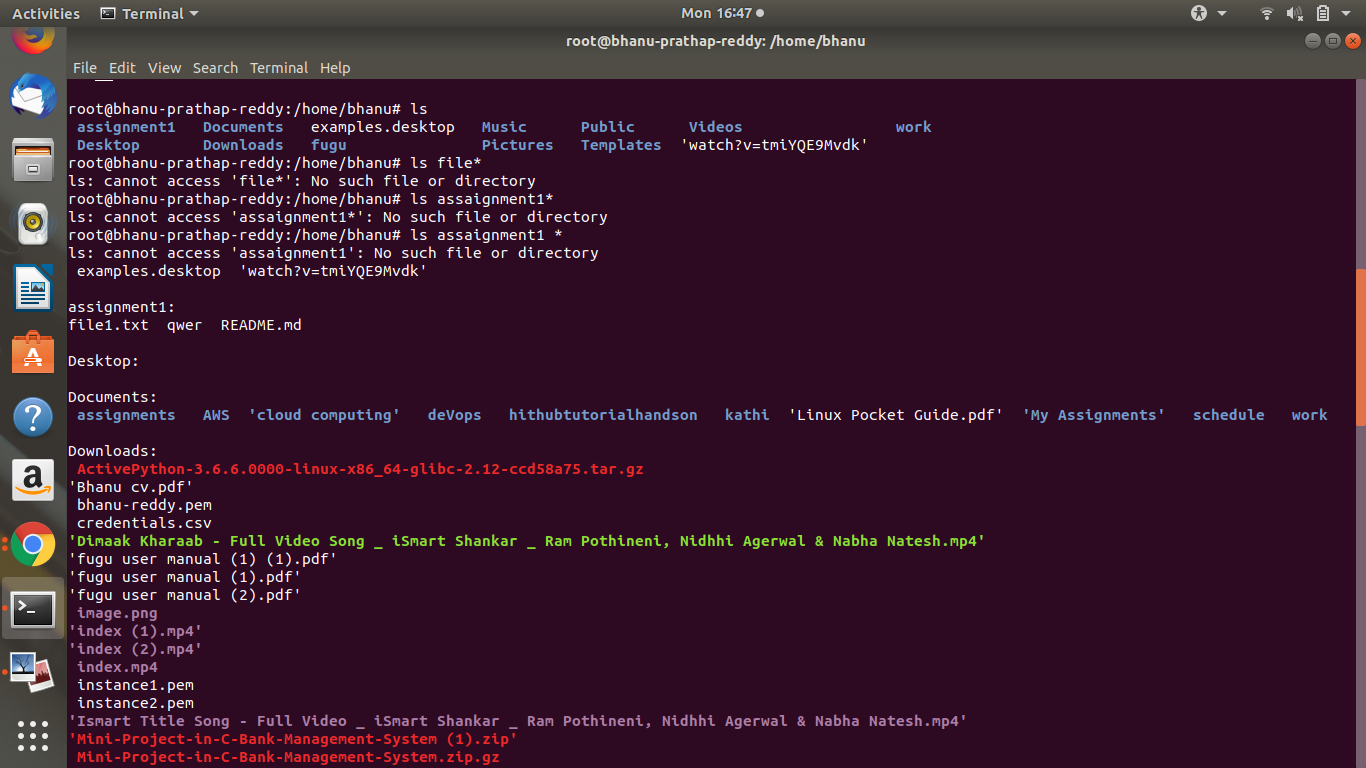


**7. \*asterisk**

The asterisk \* is interpreted by the shell as a sign to generate filenames, matching the asterisk

to any combination of characters (even none). When no path is given, the shell will use

filenames in the current directory



**8.? question mark**

Similar to the asterisk, the question mark ? is interpreted by the shell as a sign to generate

filenames, matching the question mark with exactly one character.

**9.stdin & stdout**

Generally standard input, referred to as "stdin", comes from the keyboard.

When you type stuff, you're typing it on stdin (a standard input terminal). A standard input device, which is usually the keyboard, but Linux also allows you take standard input from a file.

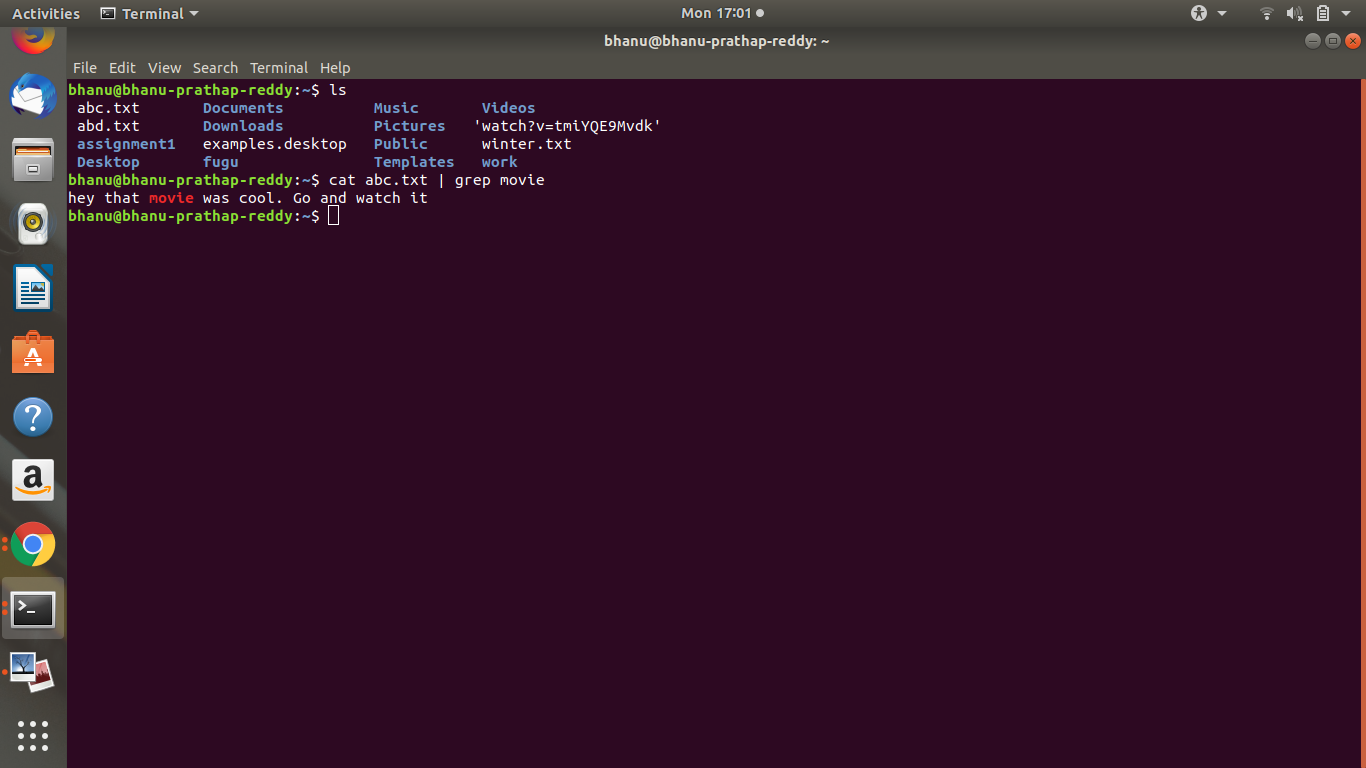
Standard output, as created at process creating time, goes to the console, your terminal or an X terminal. Exactly where output is sent clearly depends on where the process originated.

Our console or terminal should be the device that is accepting the output.

**10.grep**

The grep filter is famous among Unix users. The most common use of grep is to filter lines

of text containing (or not containing) a certain string.

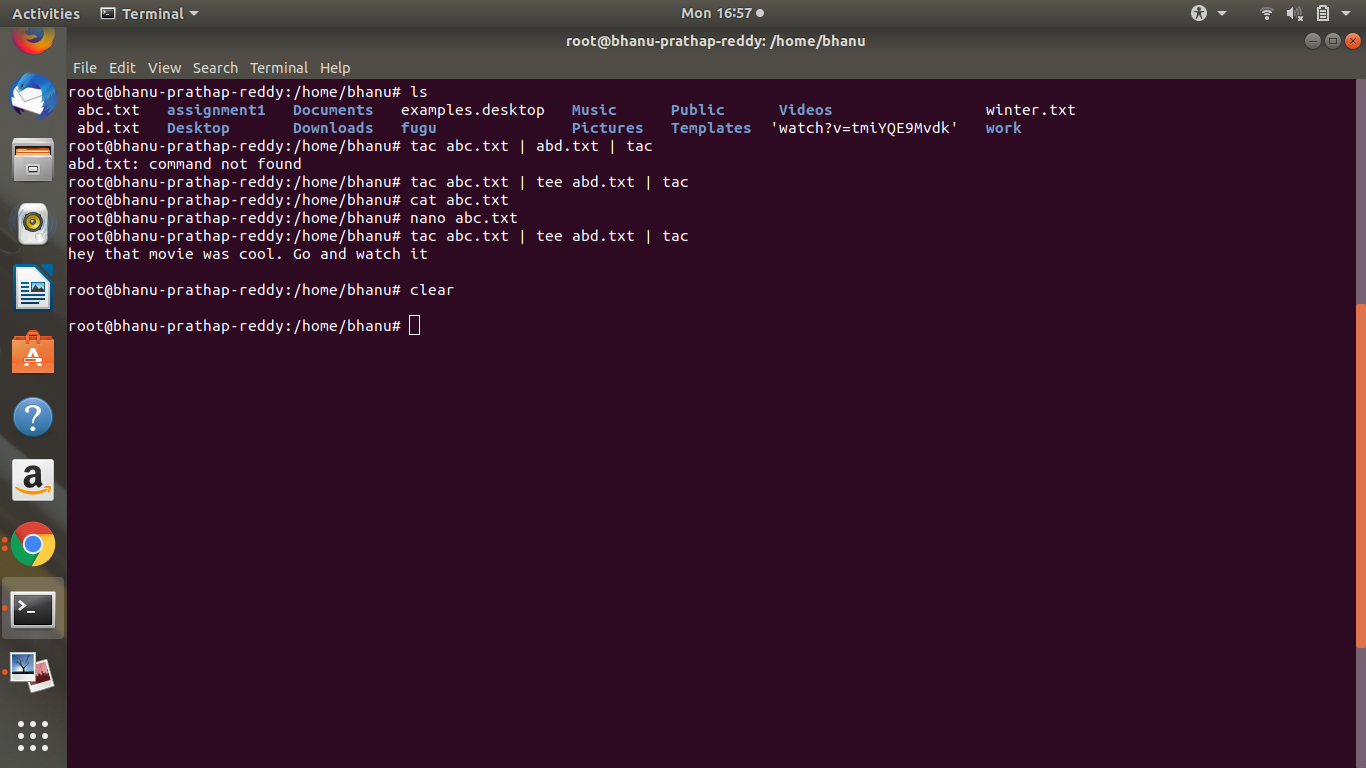


**11.tee**

Writing long pipes in Unix is fun, but sometimes you may want intermediate results. This

is were tee comes in handy. The tee filter puts stdin on stdout and also into a file. So tee is

almost the same as cat, except that it has two identical outputs.

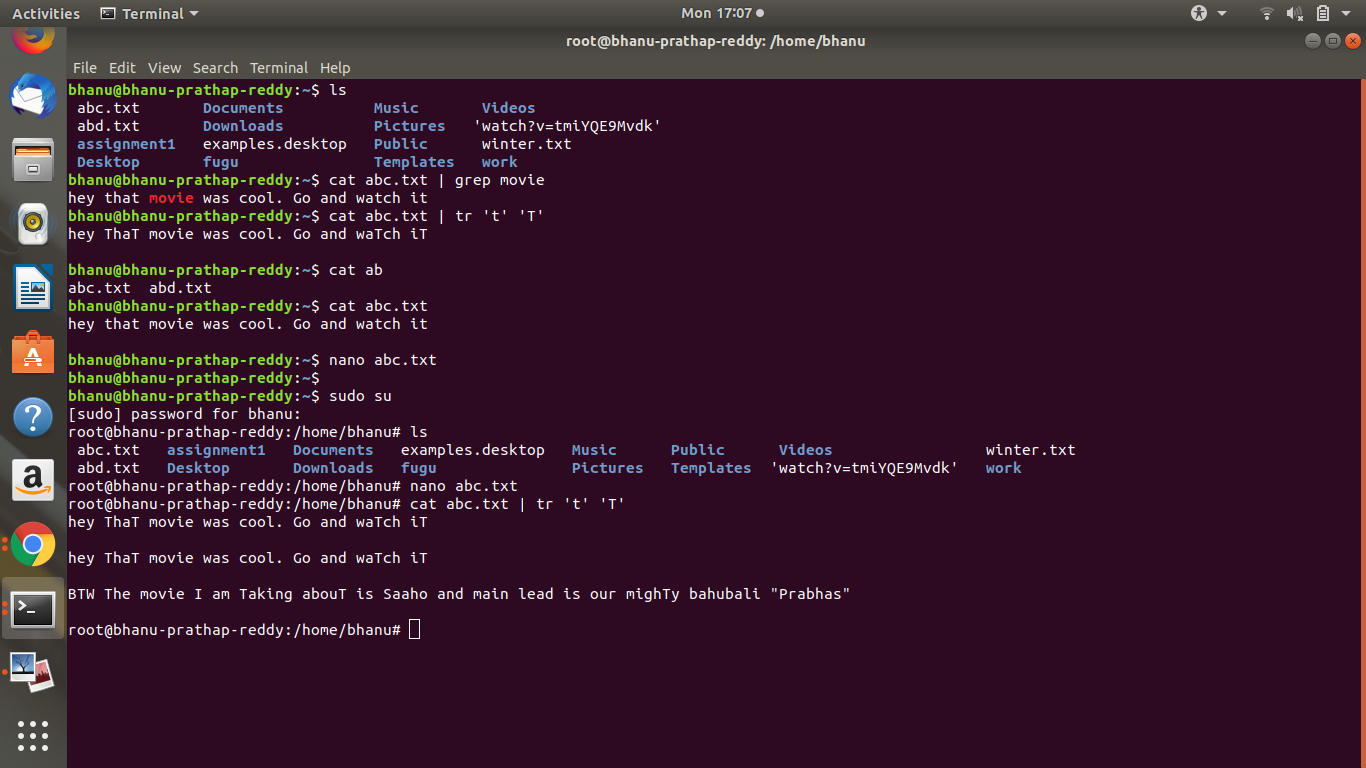


**12.tr**

You can translate characters with tr

For Example, If you type tr ‘e’ ‘E’

Then the ‘e’ letter replaced by ‘E’

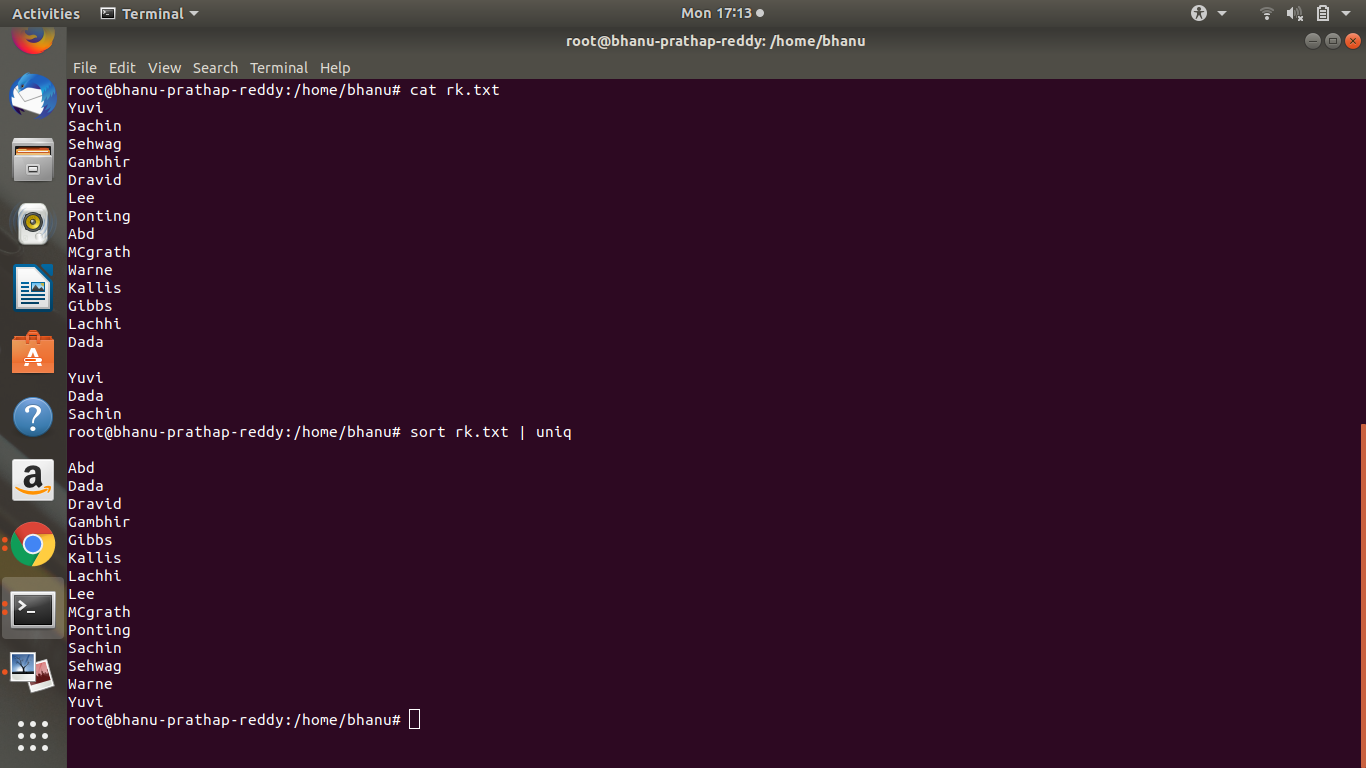


**13.sort**

The sort filter will default to an alphabetical sort.

**14.uniq**

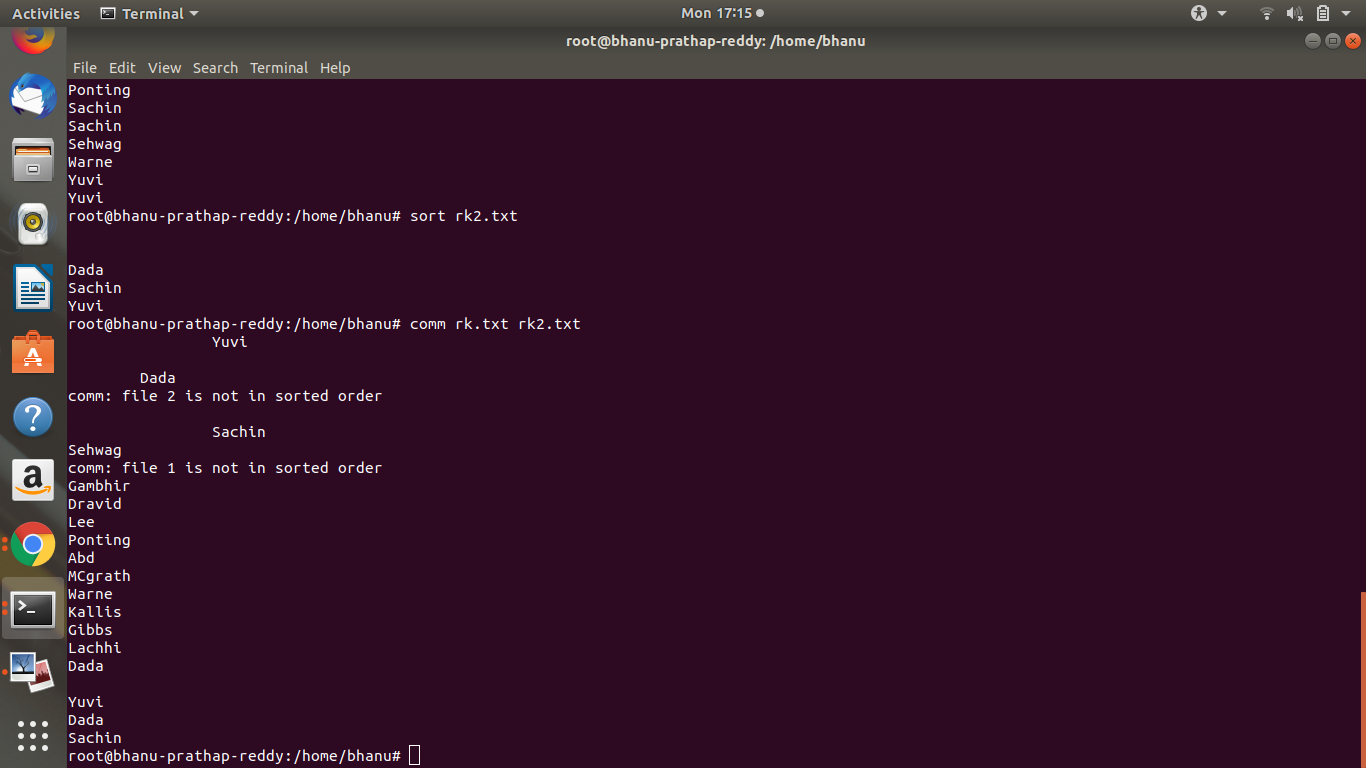
With uniq you can remove duplicates from a sorted list.



**15.comm**

Comparing streams (or files) can be done with the comm. By default comm will output

three columns.



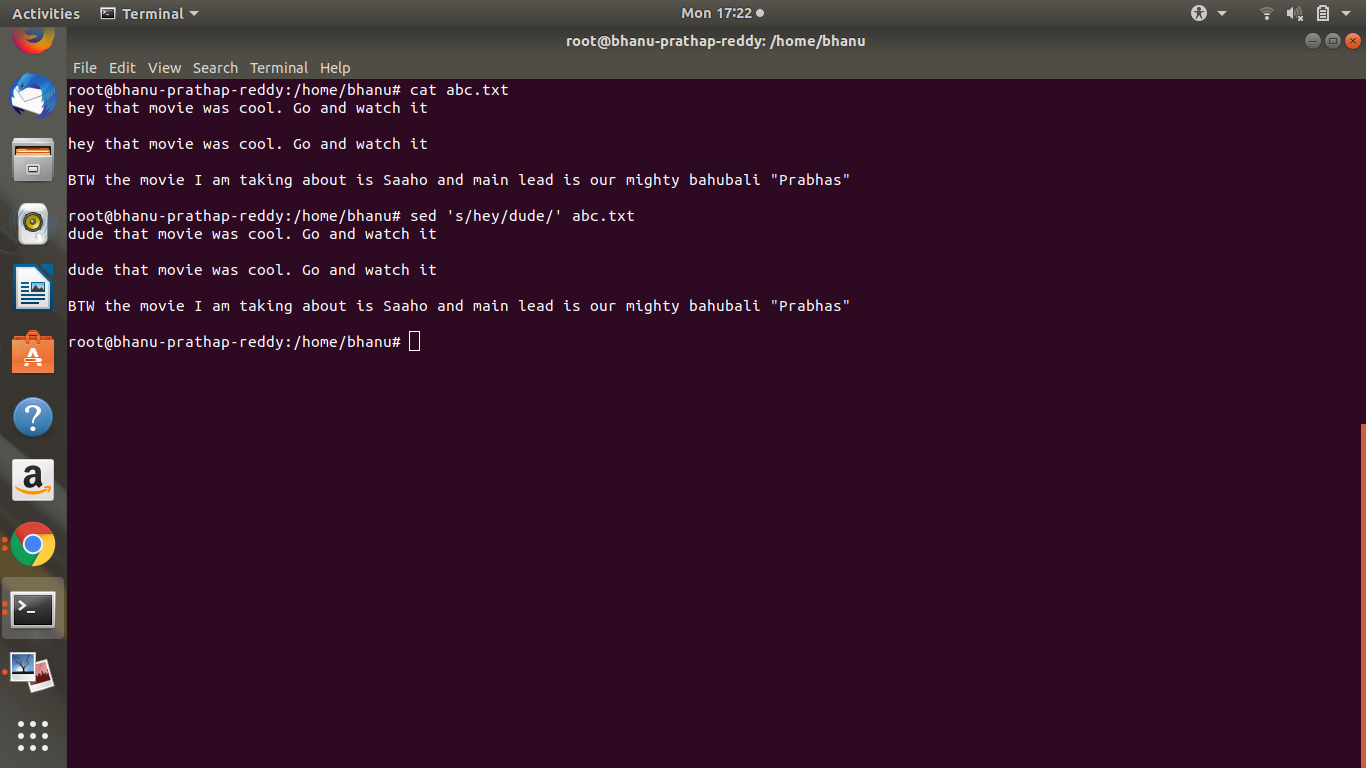
**16.sed**

The stream editor sed can perform editing functions in the stream, using regular

Expressions.

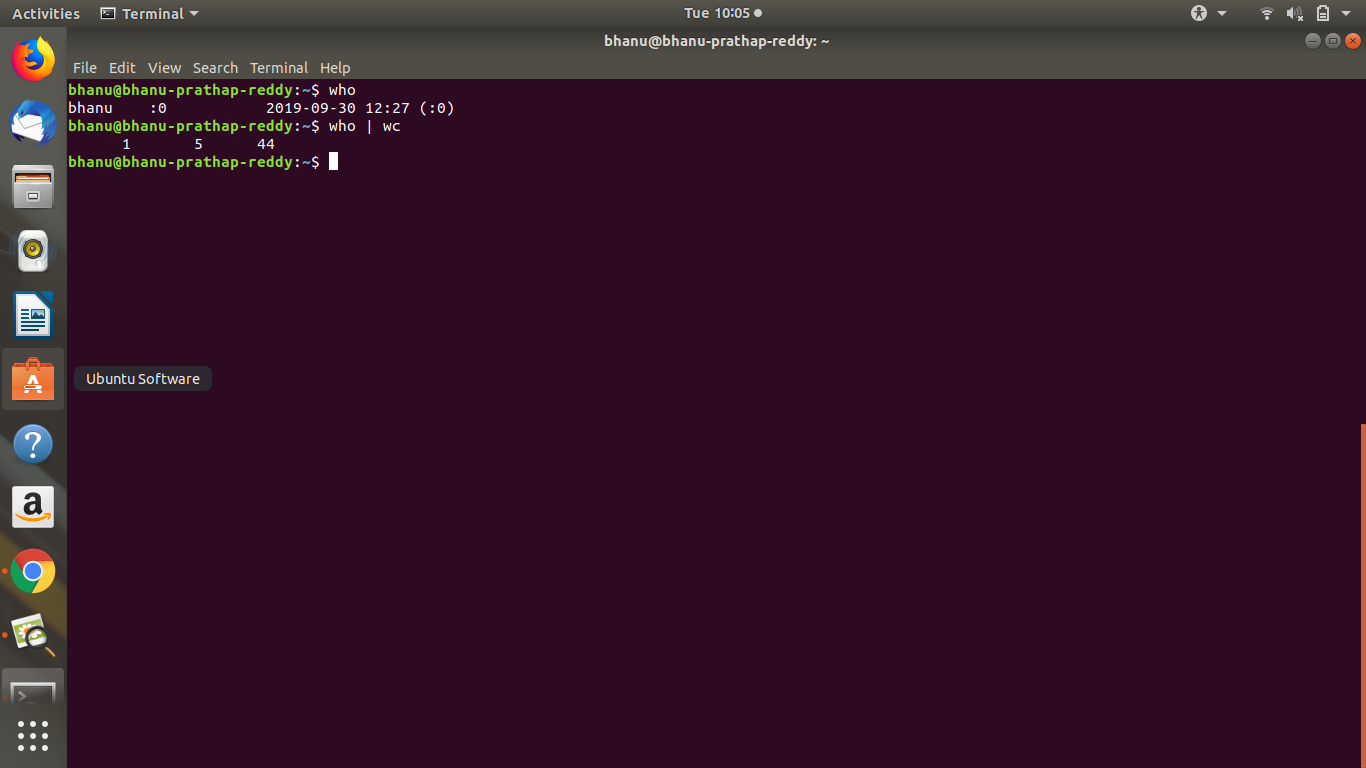
-- Add **g** for global replacements(all occurrences of the string per line).

-- with **d** you can remove lines from a stream containing a character.



**17.who | wc**

It will tell us how many users are logged on to the system



**18. grep | cut**

Display a list of all bash user accounts on this computer. Users accounts are explained in

detail later.

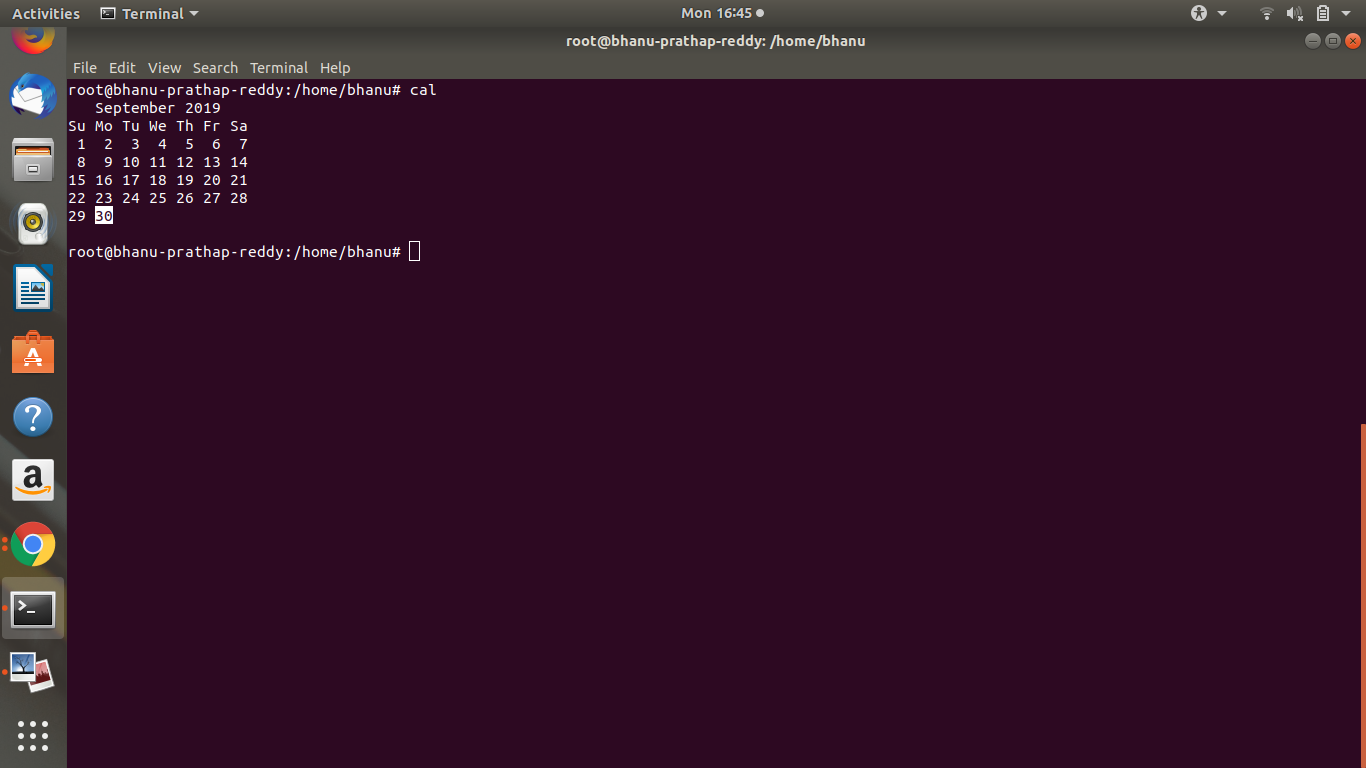
**19.sleep**

The sleep command is sometimes used in scripts to wait a number of seconds. Below example shows a five second sleep.

Sleep 5

**20.cal**

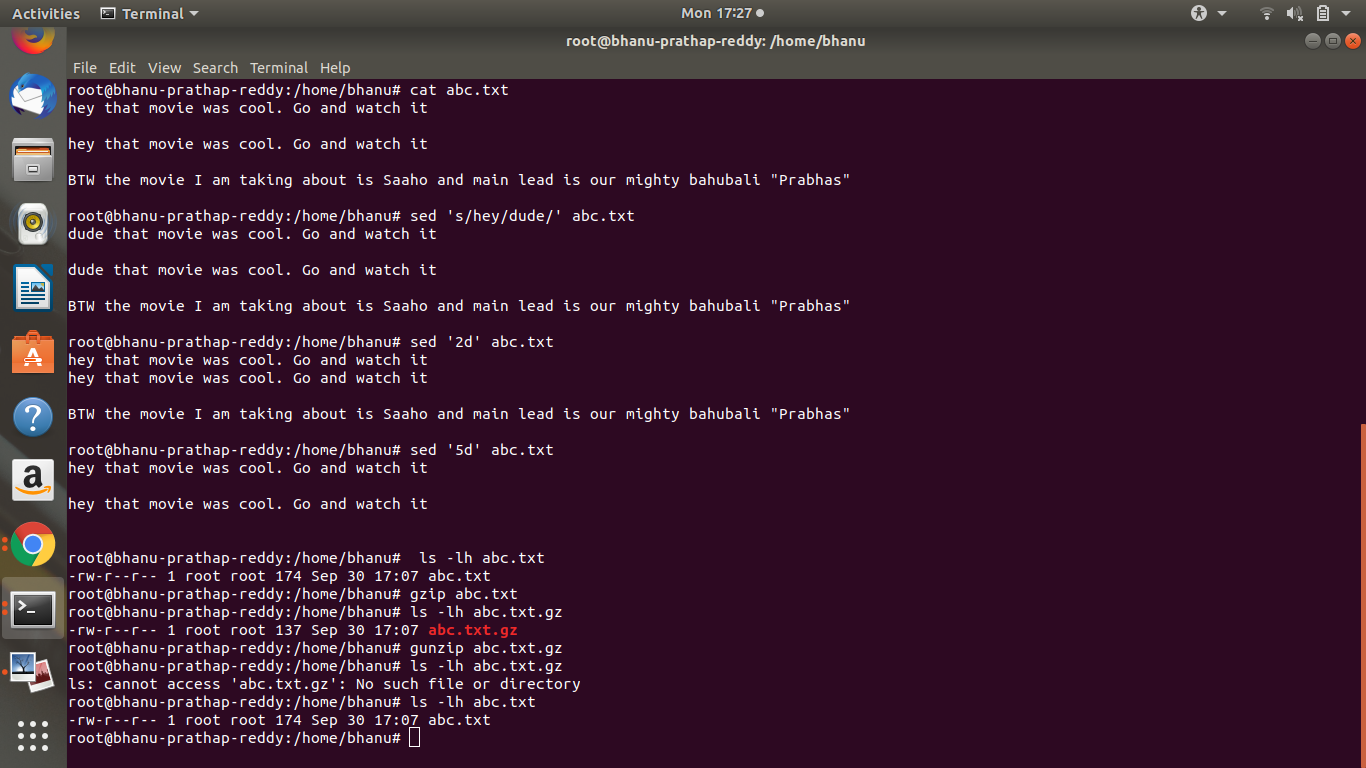
The cal command displays the current month, with the current day highlighted.



**21.gzip - gunzip**

Users never have enough disk space, so compression comes in handy. The gzip command

can make files take up less space.



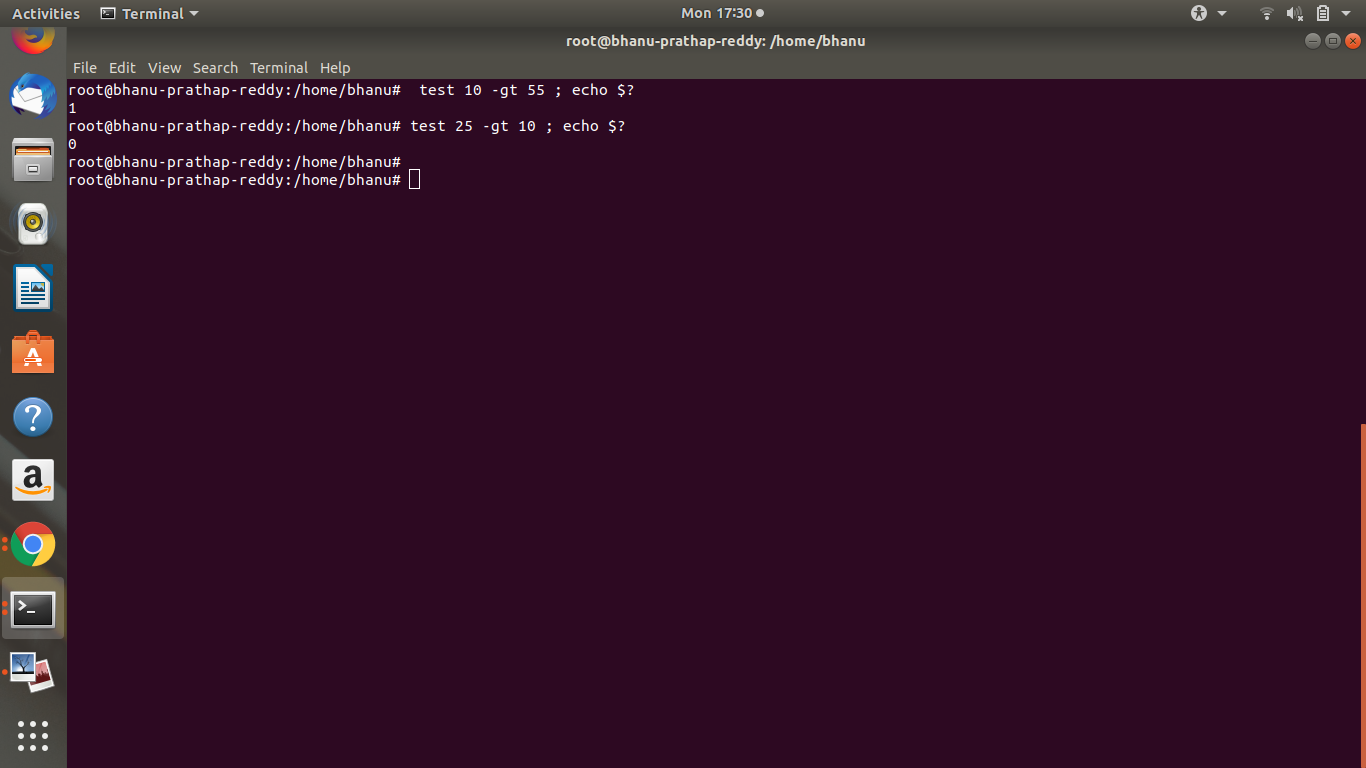
**22.zcat - zmore**

Text files that are compressed with **gzip** can be viewed with **zcat** and **zmore**

**23.test [ ]**

The test command can test whether something is true or false. Let's start by testing whether

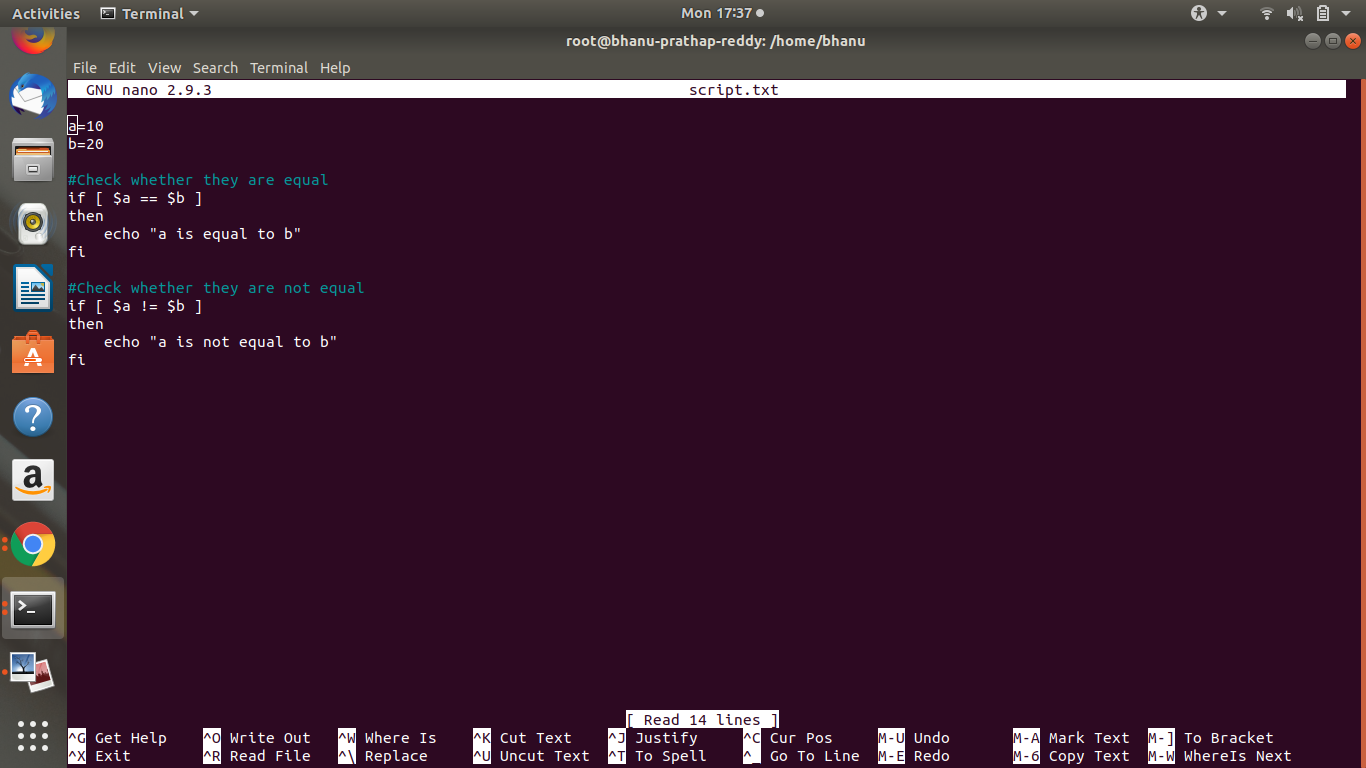
10 is greater than 55.

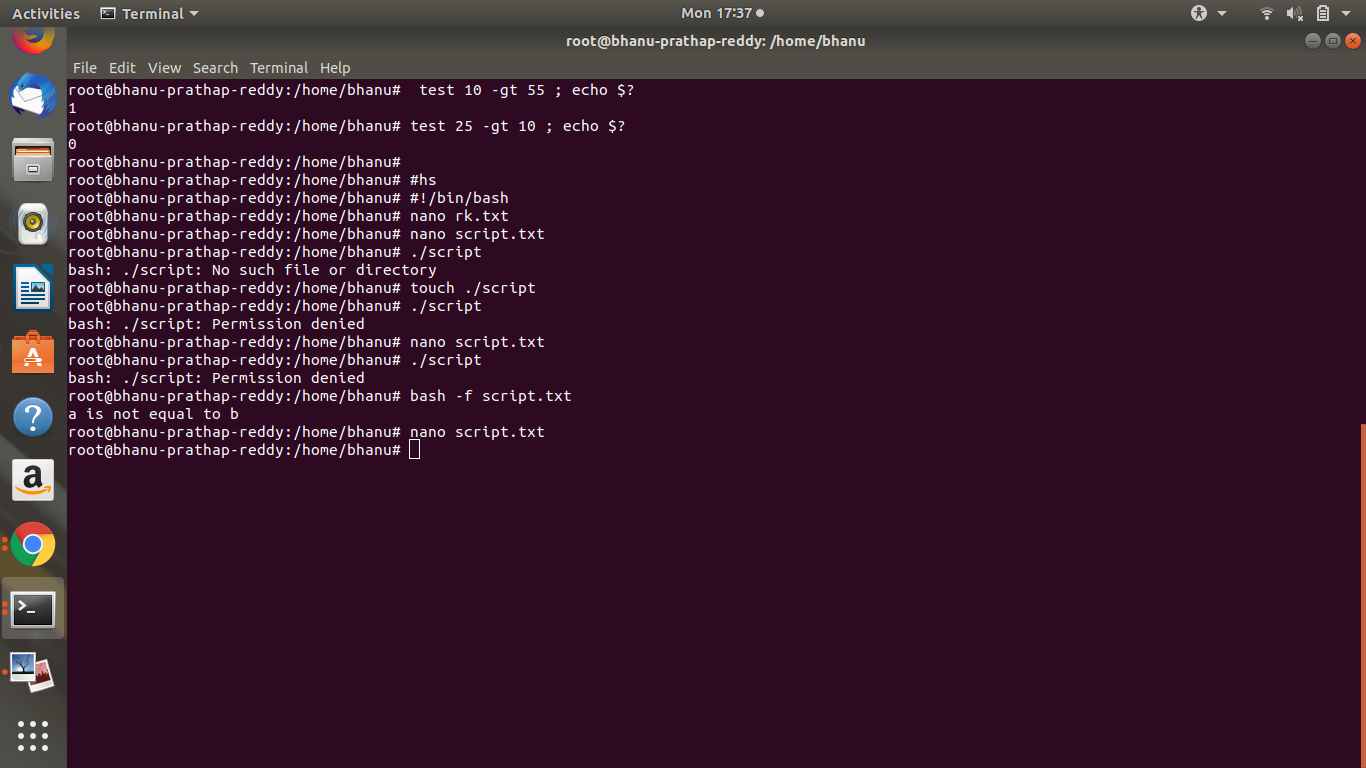


**24.if then else**

The if then else construction is about choice. If a certain condition is met, then execute

something, else execute something else



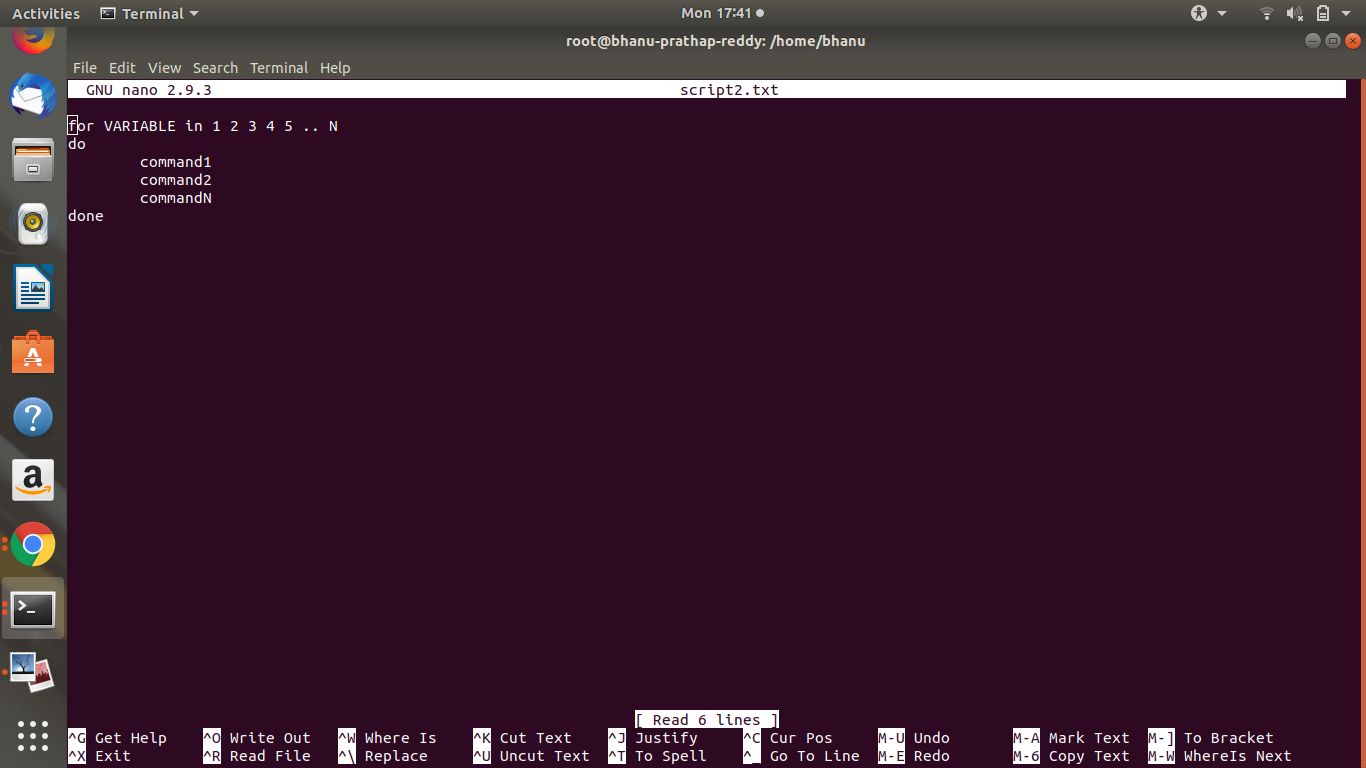


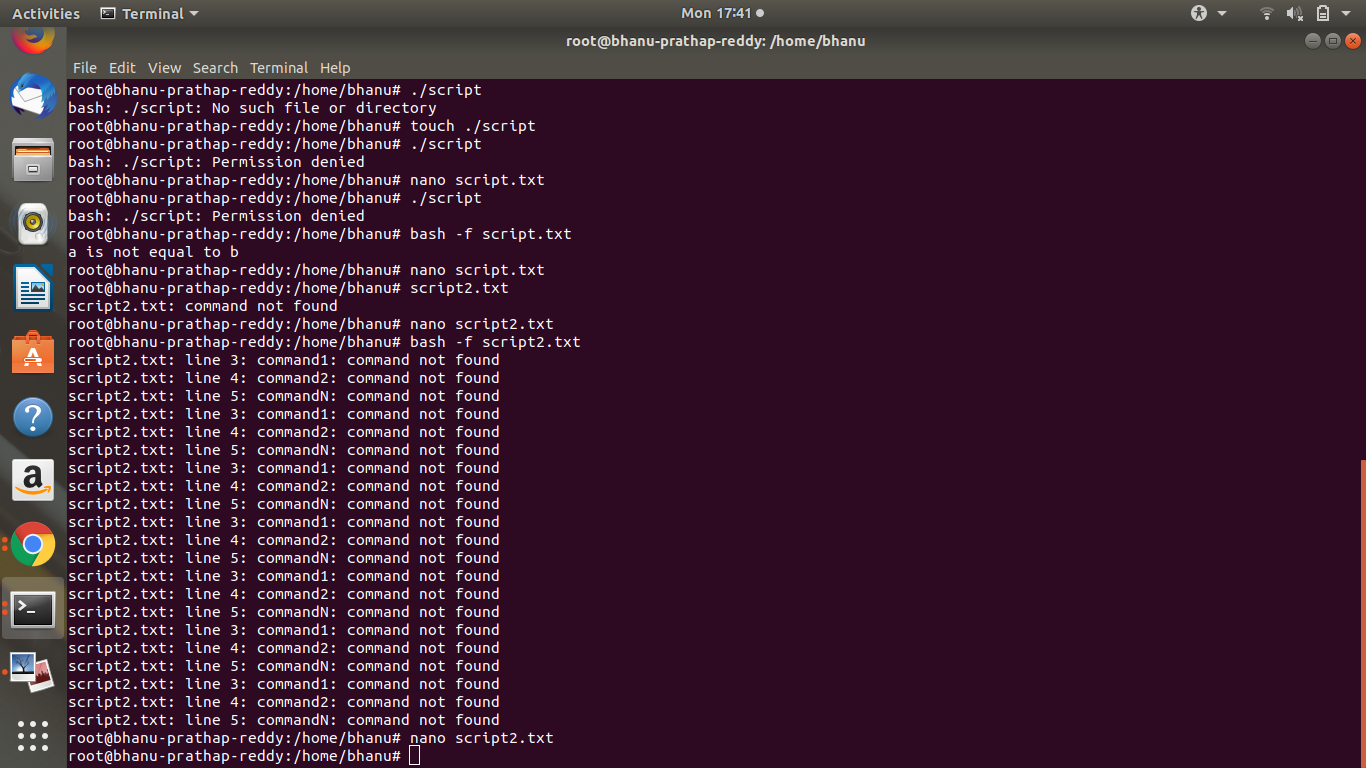
**25.if then elif**

You can nest a new if inside an else with elif

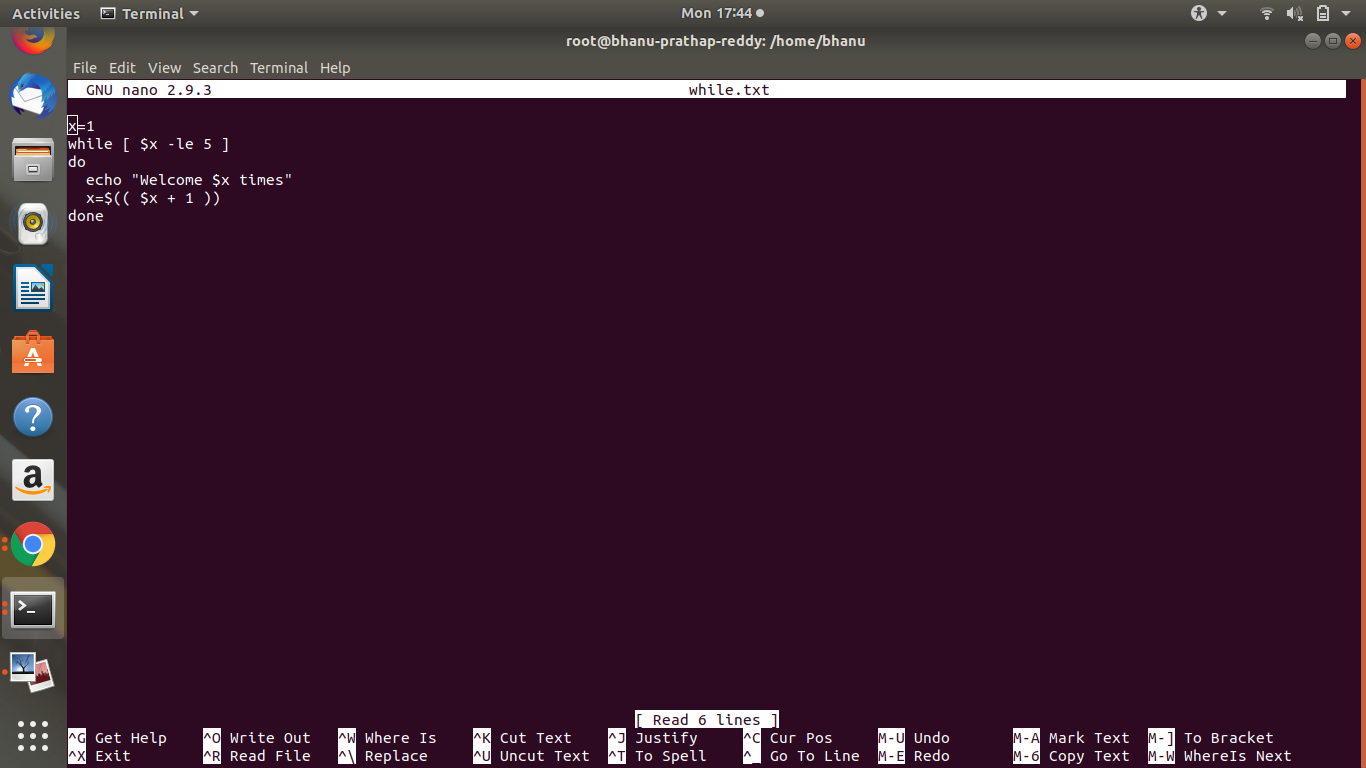
**26.for loop**

for loops iterate through a set of values until the list is exhausted.

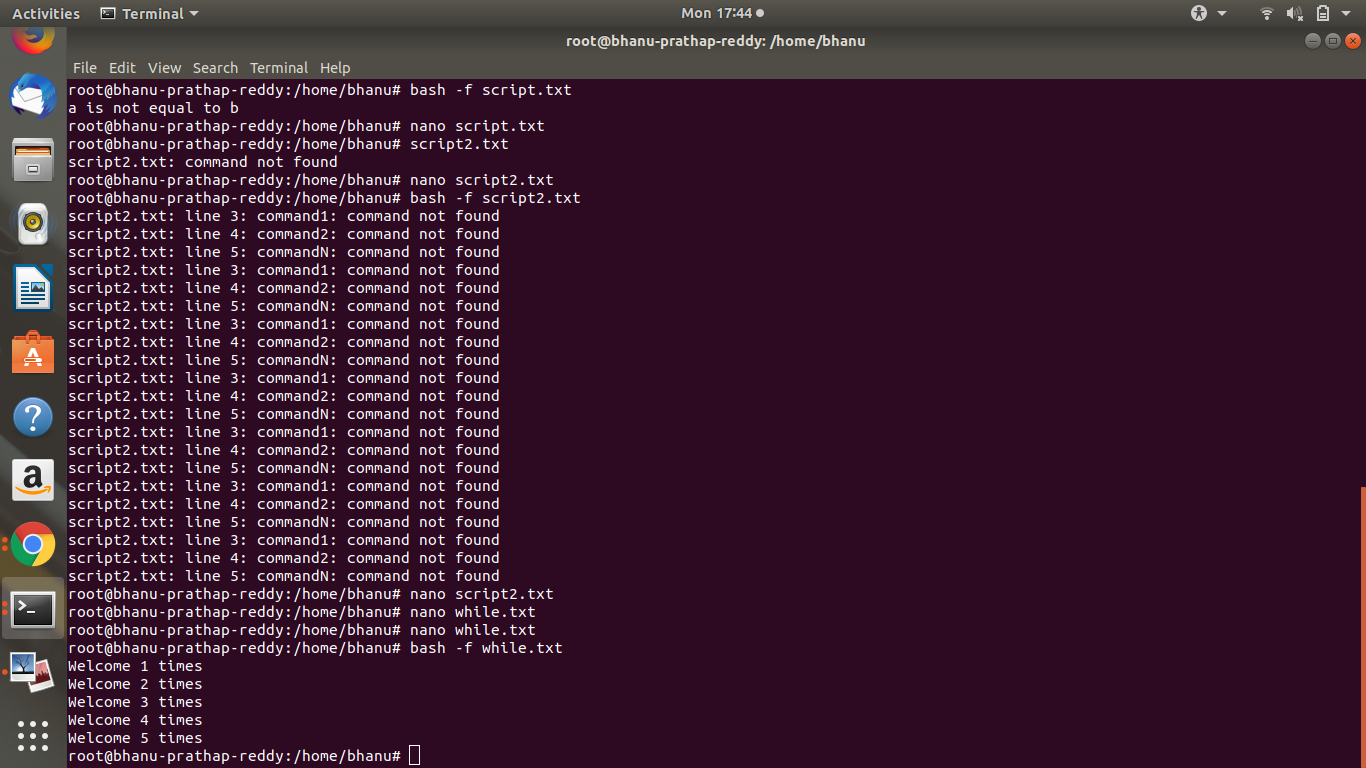




**27.while loop**



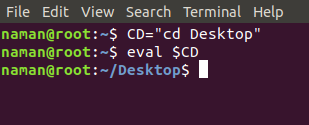
The **while** loop enables you to execute a set of commands repeatedly until some condition occurs. It is usually used when you need to manipulate the value of a variable repeatedly.



**28.eval**

eval reads arguments as input to the shell (the resulting commands are executed). This allows

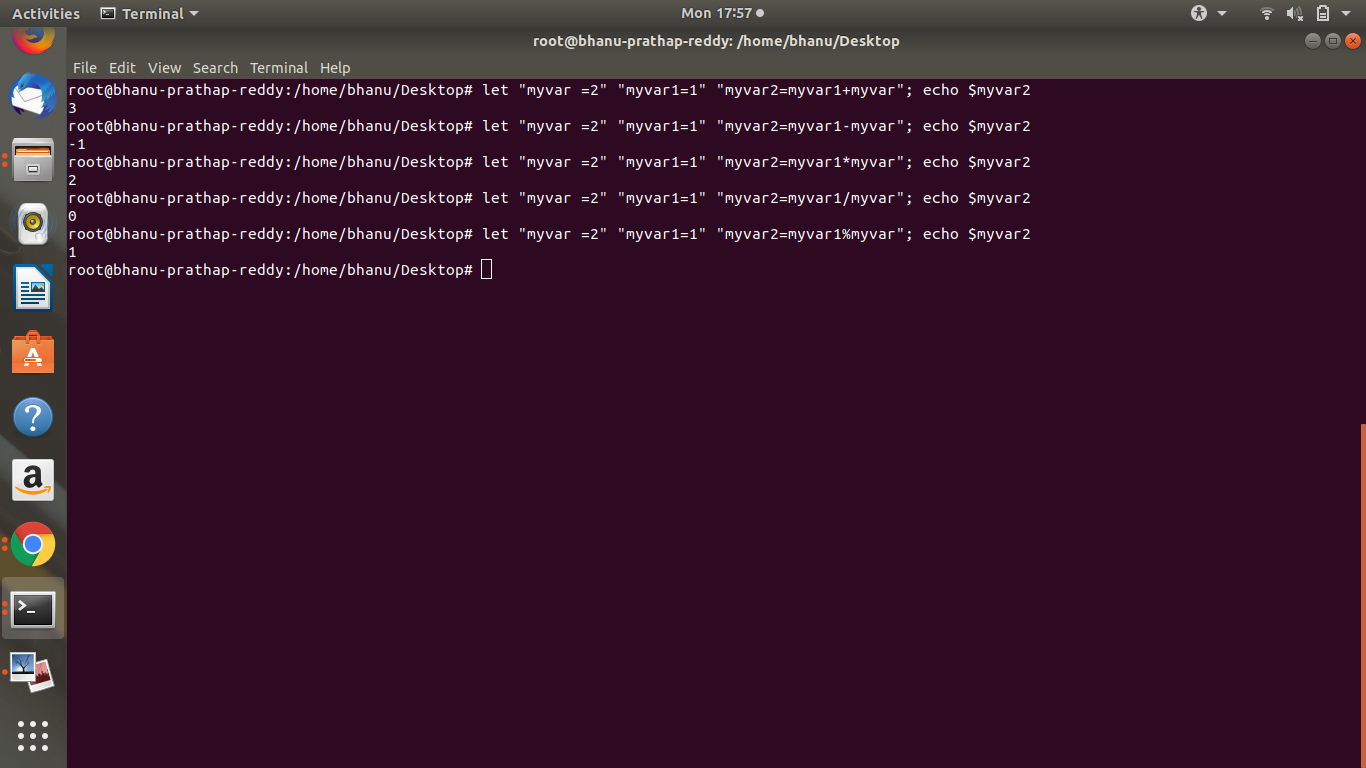
using the value of a variable as a variable.



**29.let**

The let built-in shell function instructs the shell to perform an evaluation of arithmetic

expressions. It will return 0 unless the last arithmetic expression evaluates to 0.



**30.chown**

The user owner of a file can be changed with chown command.

**Assignment : 2-10-2019**

**Question :** Launch an ec2 instance by changing its permissions file and add username and Password in it using bash script

Here is the script that we need to write for above problem

Process:

-->Create one file using with extension .sh

nano Test.sh

Here, .sh means a file in which a script programmed for bash, a type of Unix shell (Bourne-Again SHell). It contains instructions written in the Bash language and can be executed by typing text commands within the shell's command-line interface.

-->write the bash script

#! /bin/bash

cd Downloads

#for creation of ssh key

ssh-keygen

cd /home/delta/.ssh

cp id\_rsa.pub 11.pem

cd

echo "54.193.90.36"

scp -i ~/Downloads/12.pem ~/.ssh/11.pem

ec2-user@54.193.90.36:~/.ssh/

echo "54.193.90.36”

cd Downloads

#connecting to ec2 instance

ssh -i 12.pem ec2-user@54.193.90.36"sudo yum install httpd

cd .ssh/

cat 11.pem >> authorized\_keys

sudo useradd Reddy

echo rkreddy:123 | chpasswd"

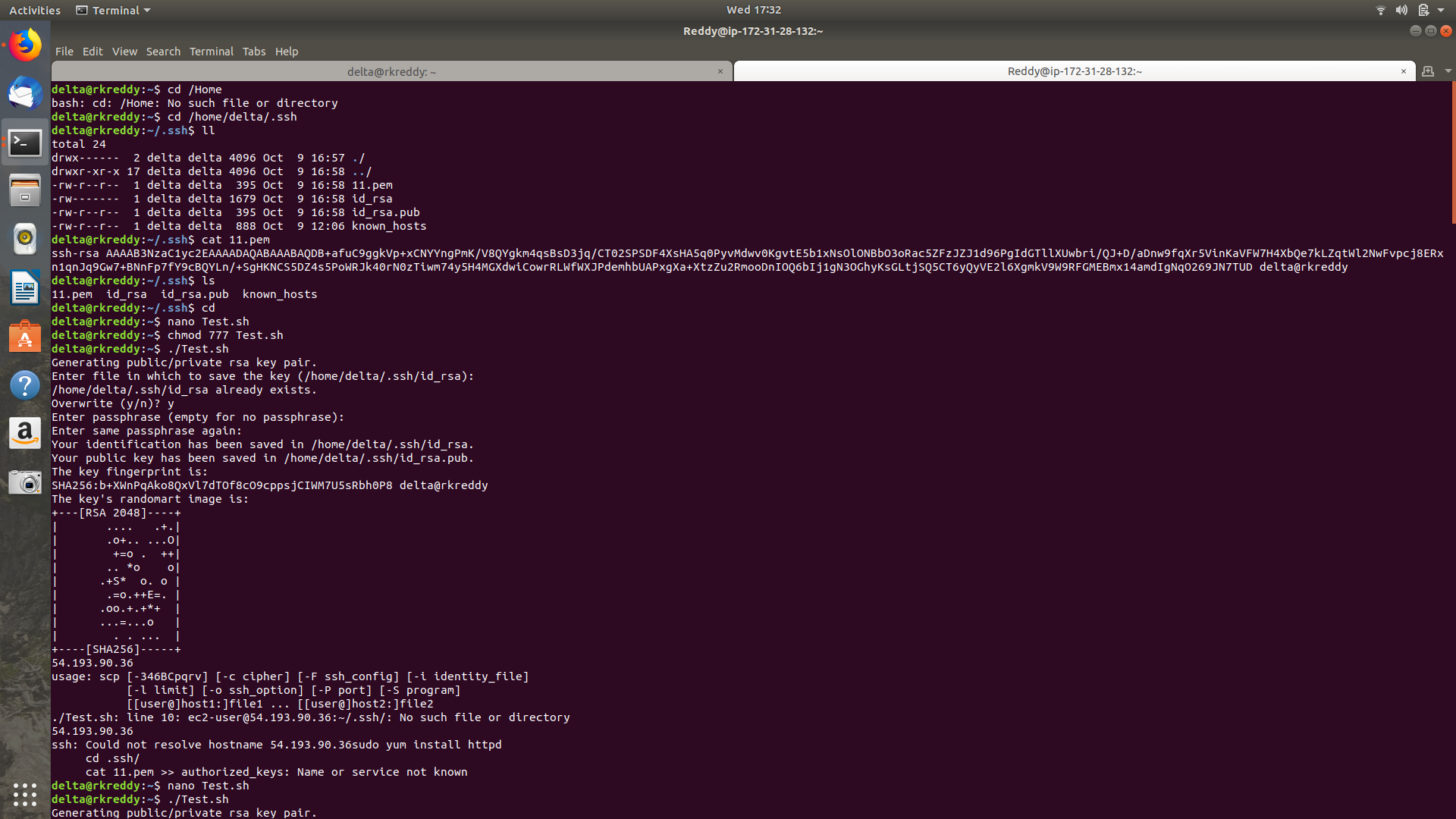
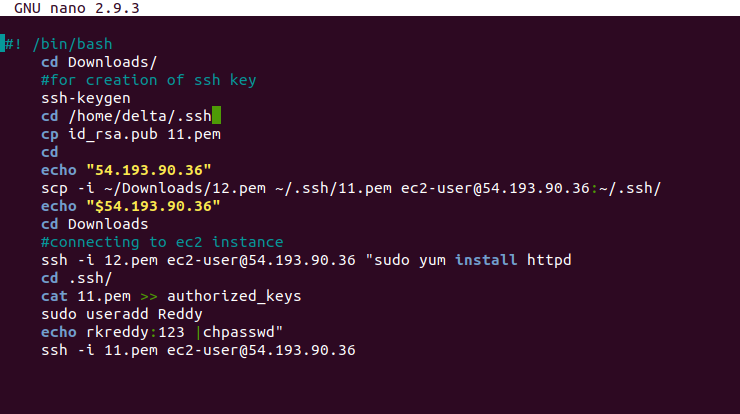
ssh -i 11.pem ec2-user@54.193.90.36

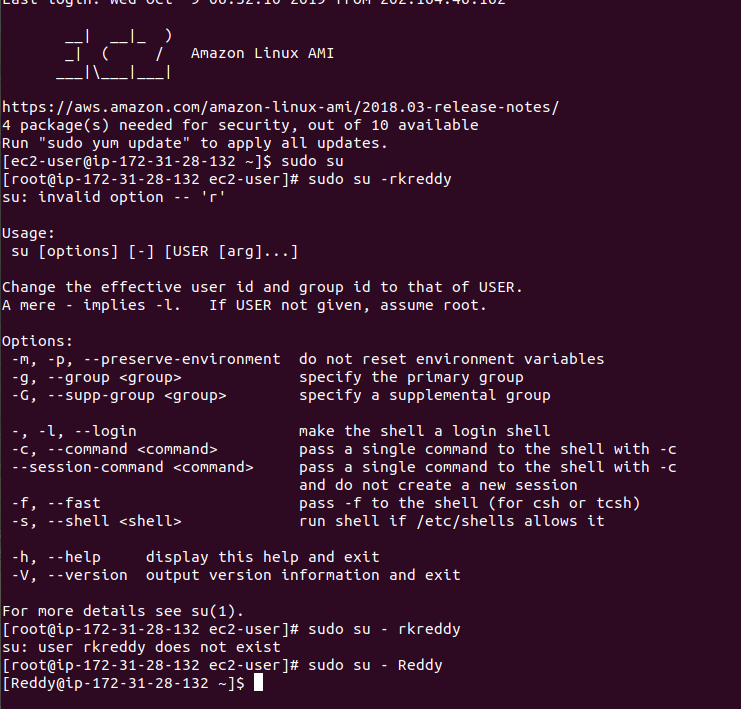
-->save the file

--->chmod 777 Test.sh

--->./Test..sh

**Script**





**Assignment : 3/10/2019**

**Question:**Add a variable in .bashrc and use proxypass in .bkp file and create files from 8001.bkp to 8005.bkp using script and update the variable to 8005 in .bashrc file

**Solution:**

Process:

>Launch an ec2 instance in terminal and goto root/user/

>ls -la #all files will be displayed including hidden files also

>open .bashrc file and add variable var=8001

>cd /etc/httpd/conf.d

>nano 8001.bkp #create file with .bkp extension

<VirtualHost \*:80>

ServerName [www.google.com](http://www.instagram.com)

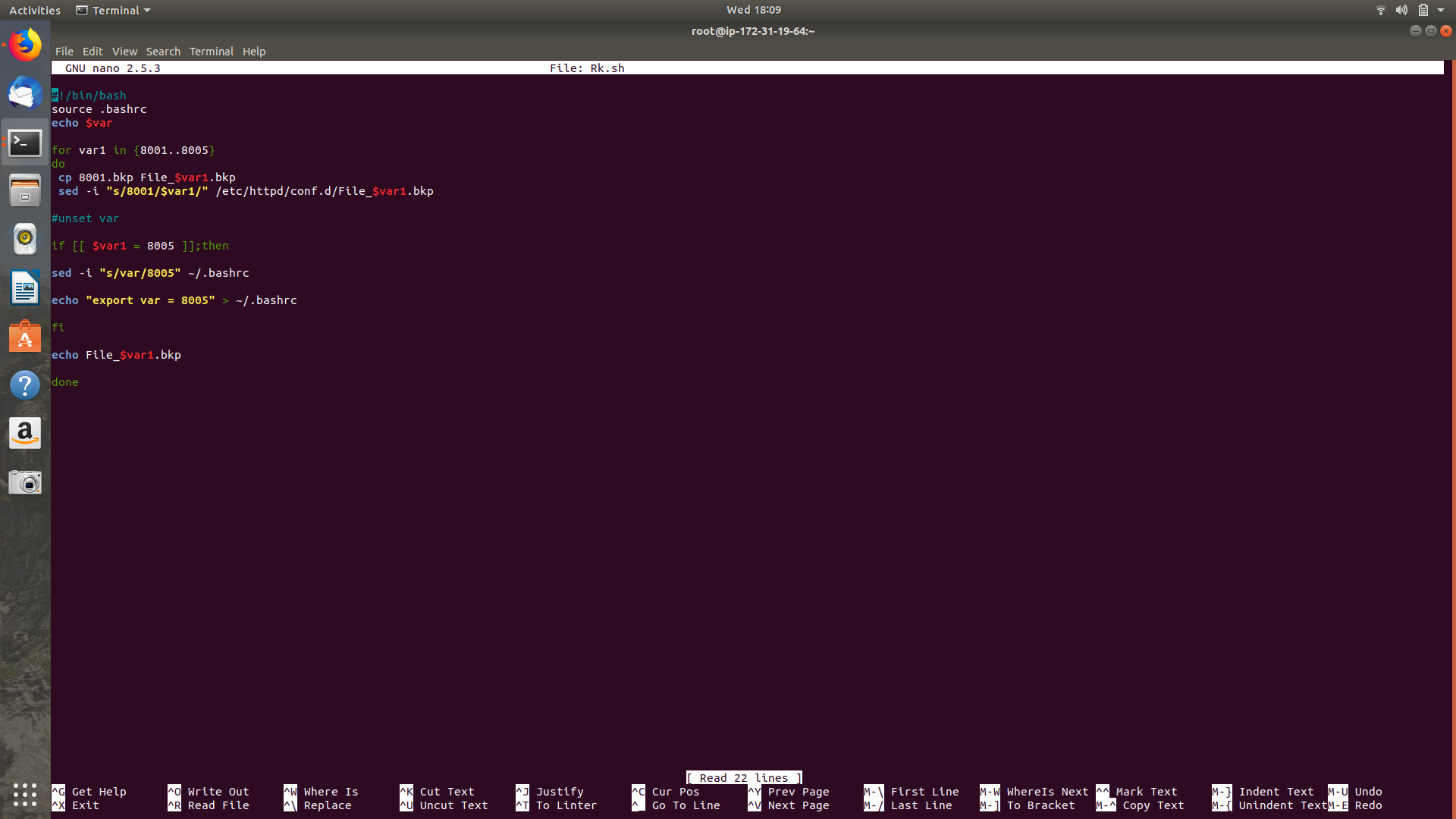
ProxyPass / [http://8.8.8.8:8001/](http://127.0.0.53:8001/)

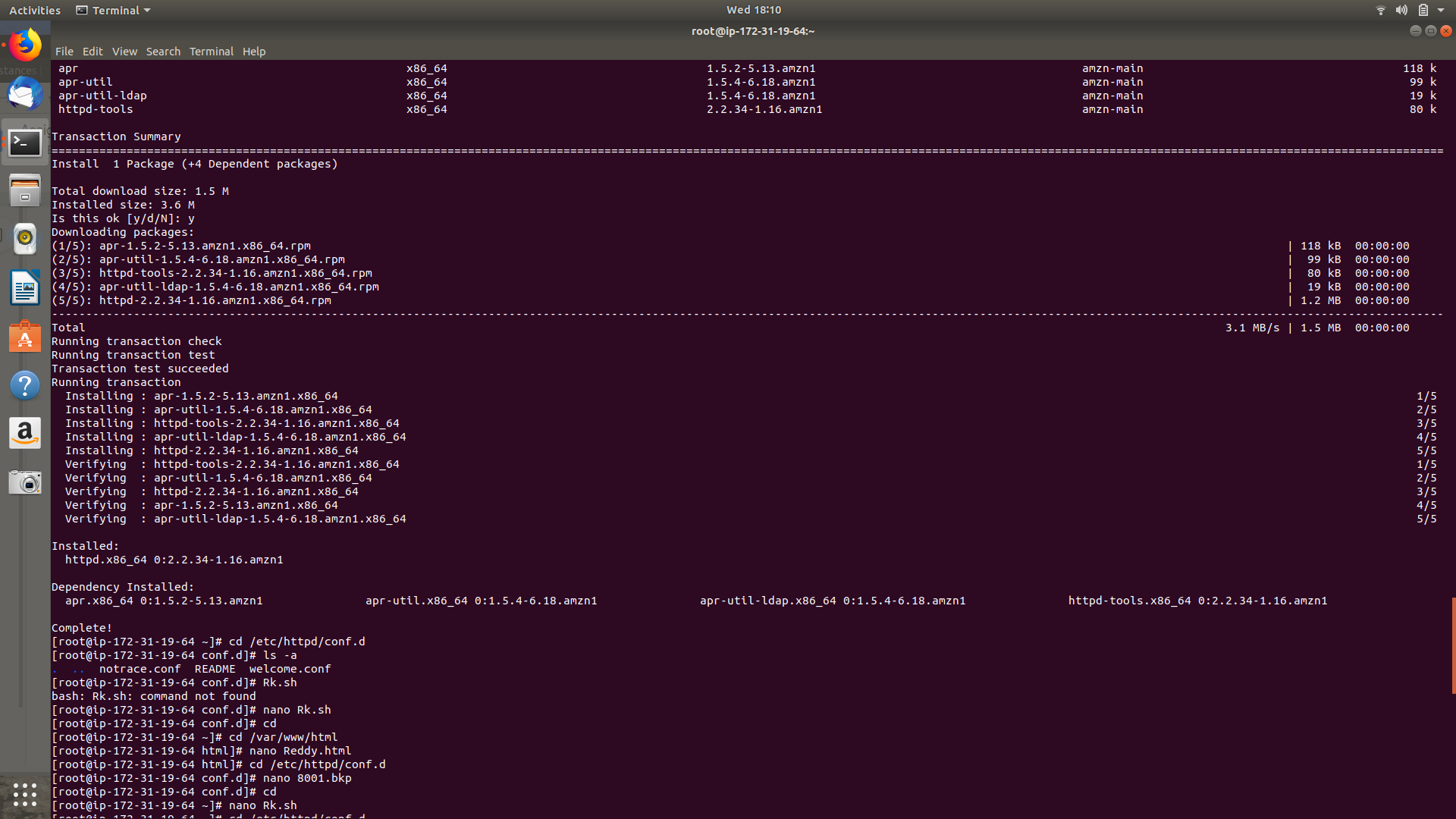
</VirtualHost>

>cd

>create one .sh file

nano Rk.sh







\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_