OPERATING SYSTEMS LAB

LAB # 6

I/O Redirection
and
Shell Programming

Lab # 6

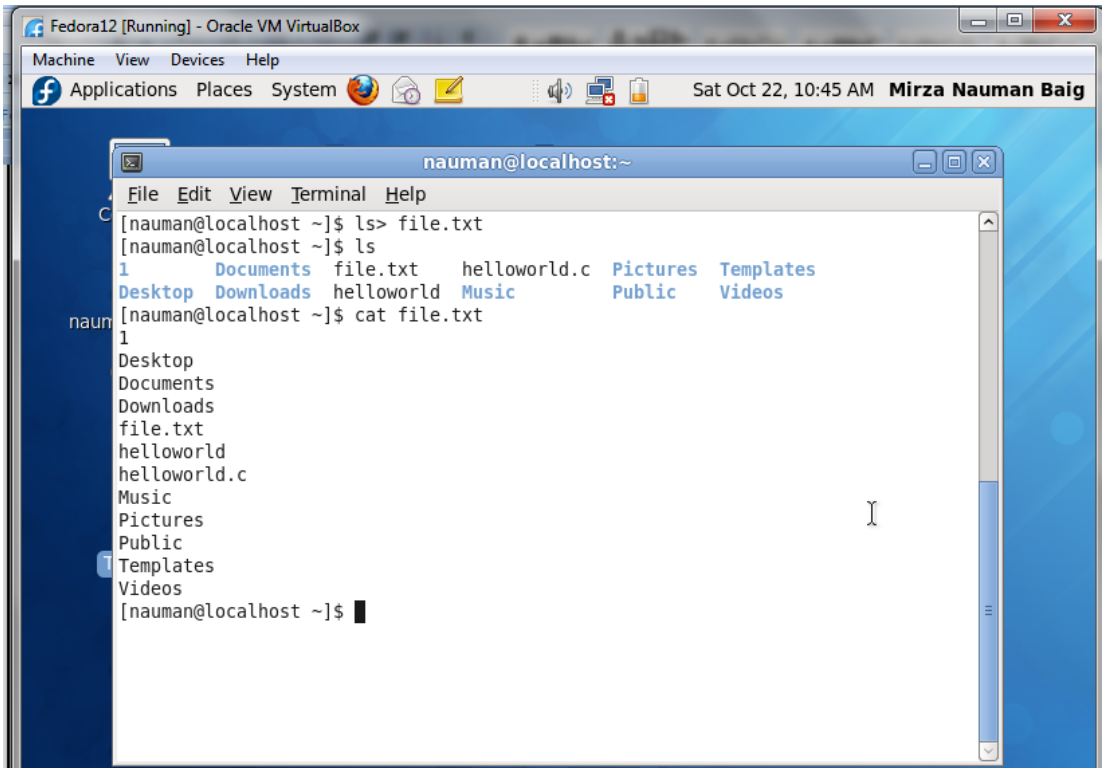
Shell Programming(I/O Redirection and if-else Statement)

Redirection of Standard output/input i.e. Input - Output redirection

Mostly all command gives output on screen or take input from keyboard, but in Linux (and in other OSs also) it's possible to send output to file or to read input from file.

For e.g.
$ ls command gives output to screen; to send output to file of ls command give command

$ ls > filename
It means put output of ls command to filename.



There are three main redirection symbols **>,>>,<**

(1) > Redirector Symbol (Overwrite)

Syntax:
Linux-command > filename
To output Linux-commands result (output of command or shell script) to file. Note that if file already exist, it will be overwritten else new file is created. For e.g. To send output of ls command give

$ ls > myfiles
Now if '**myfiles**' file exist in your current directory it will be overwritten without any type of warning.

(2) >> Redirector Symbol (Append)

Syntax:

Linux-command >> filename

To output Linux-commands result (output of command or shell script) to END of file. Note that if file exist , it will be opened and new information/data will be written to END of file, without losing previous information/data, And if file is not exist, then new file is created.

For e.g. To send output of date command to already exist file give command

$ date >> myfiles

(3) < Redirector Symbol

Syntax:

Linux-command < filename

To take input to Linux-command from file instead of key-board. For e.g. To take input for cat command give

$ cat < myfiles

You can also use above redirectors simultaneously as follows

Create text file sname as follows

$cat > sname

virk
ash
zebra
babu

Press CTRL + D to save.

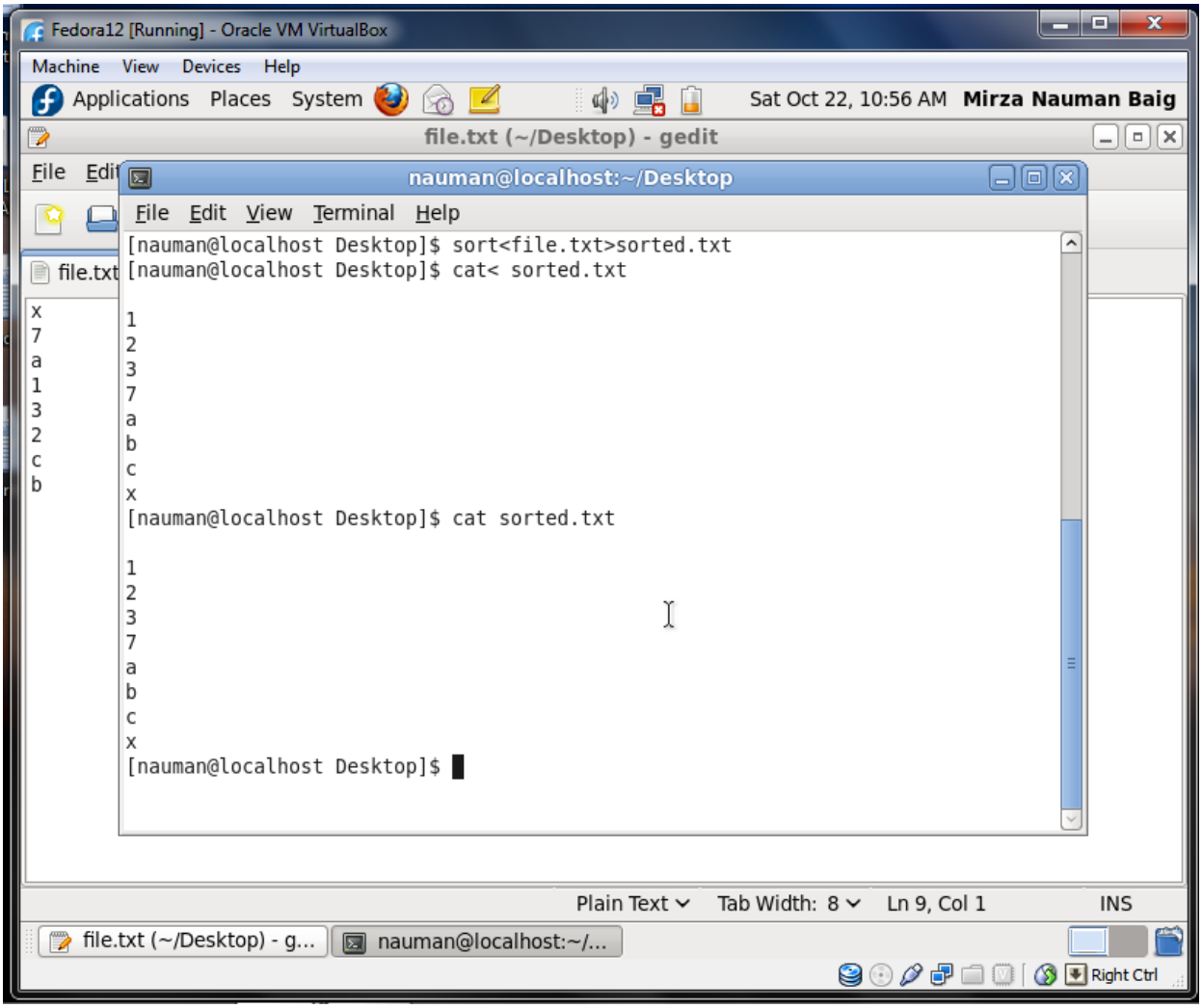
Now issue following command.

$ sort < sname > sorted_names

$ cat sorted_names

ash
babu
virk
zebra

In above example sort (**$ sort < sname > sorted_names**) command takes input from sname file and output of sort command (i.e. sorted names) is redirected to sorted_names file.



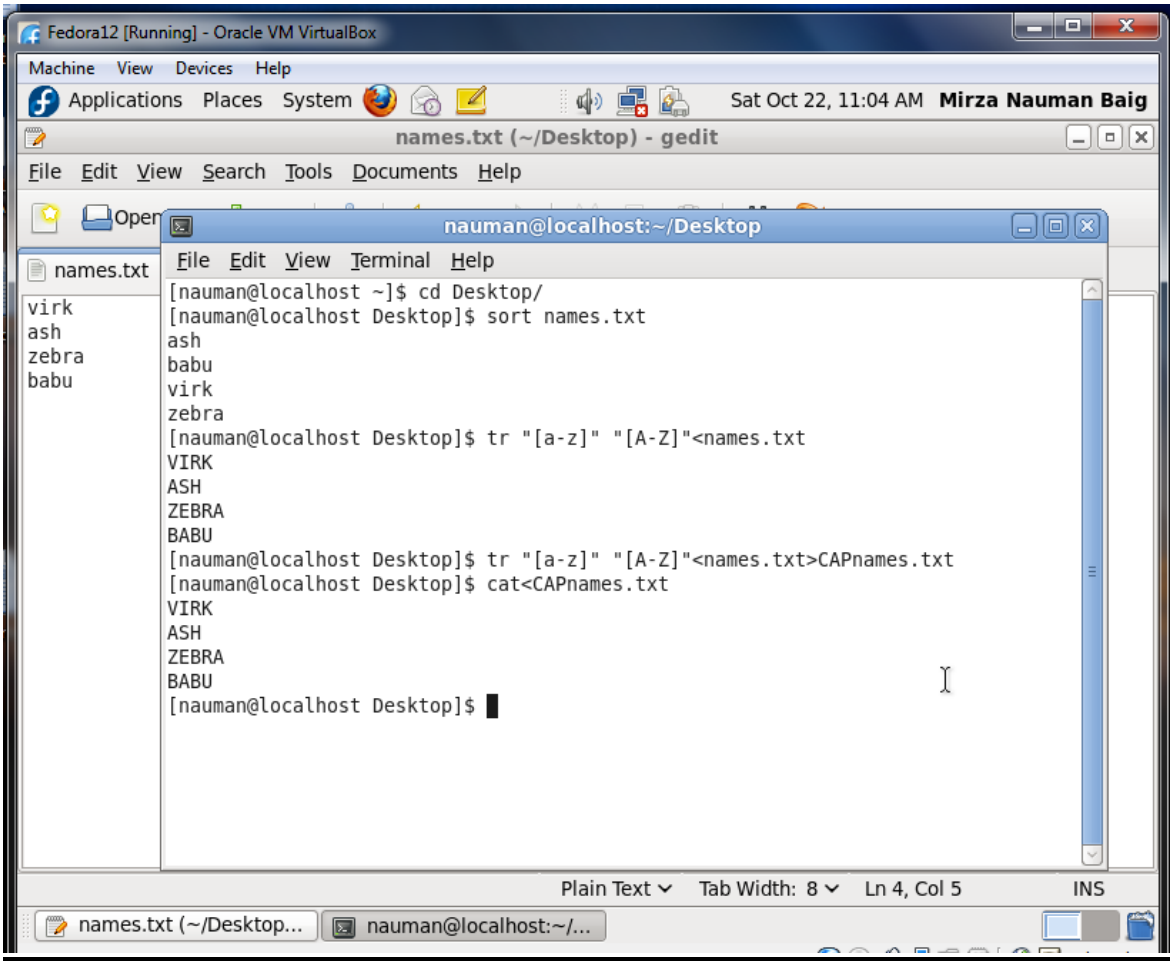
Try one more example to clear your idea:

$ tr "[a-z]" "[A-Z]" < sname > cap_names

$ cat cap_names

VIRK
ASH
ZEBRA
BABU

_command is used to translate all lower case characters to upper-case letters. It take input from sname file, and tr's output is redirected to cap_names file.



Shells (bash) structured Language Constructs

Lets play some game with bc bash calculator - Linux calculator program.
$ bc
After this command bc is started and waiting for your commands, i.e. give it some calculation as follows type 5 + 2 as:
5 + 2
7
7 is response of bc i.e. addition of 5 + 2 you can even try

5 - 2
5 / 2
See what happened if you type 5 > 2 as follows

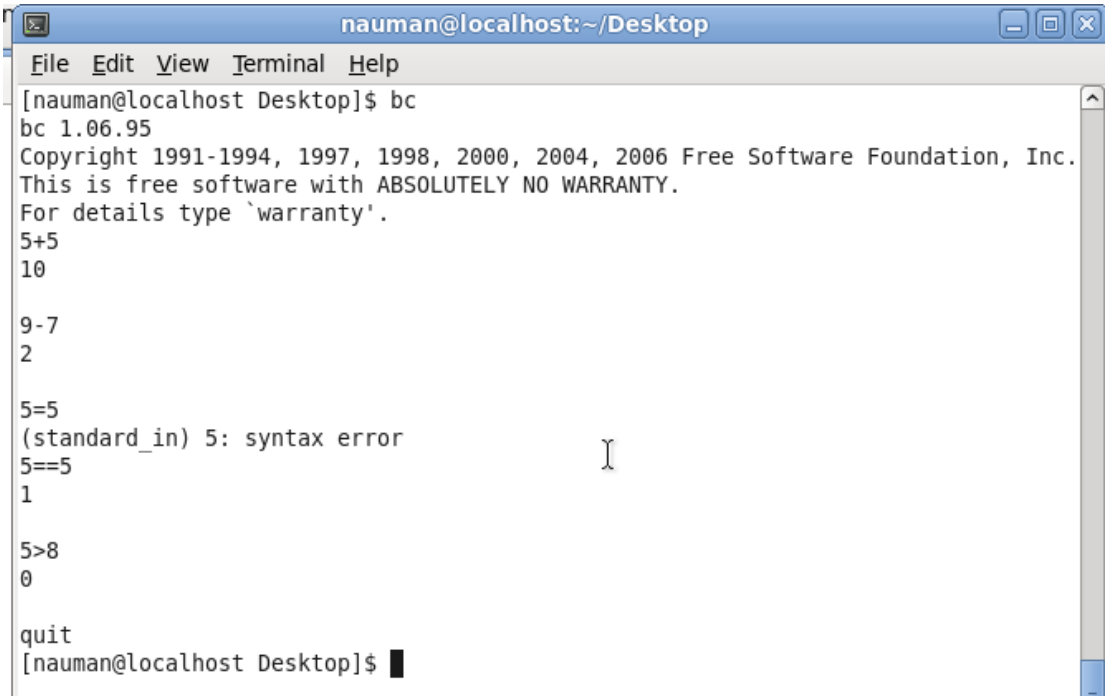
5 > 2
1

1 (One?) is response of bc, How? bc compare 5 with 2 as, Is 5 is greater then 2, (If I ask same question to you, your answer will be YES), bc gives this 'YES' answer by showing 1 value. Now try

5 < 2
0
0 (Zero) indicates the false i.e. Is 5 is less than 2?, Your answer will be no which is indicated by bc by showing 0 (Zero). Try following in bc to clear your Idea and not down bc's response
5 > 12
5 == 10
5 != 2
5 == 5
12 < 2

Expression	Meaning to us	Your Answer	BC's Response
5 > 12	Is 5 greater than 12	NO	0
5 == 10	Is 5 is equal to 10	NO	0
5 != 2	Is 5 is NOT equal to 2	YES	1
5 == 5	Is 5 is equal to 5	YES	1
1 < 2	Is 1 is less than 2	Yes	1

It means when ever there is any type of comparison in Linux Shell It gives only two answers one is YES and NO is other.



In Linux Shell Value	Meaning	Example
Zero Value (0)	Yes/True	0
NON-ZERO Value	No/False	-1, 32, 55 anything but not zero

mkdir FolderName
echo $?
Or
mkdir FolderName; echo $? *What is output*
mkdir FolderName; echo $? *Why?*
mkdr FolderName; echo $?

Remember both bc and Linux Shell uses *different ways to show True/False values*

Value	Shown in bc as	Shown in Linux Shell as
True/Yes	1	0
False/No	0	Non - zero value

How to Run Shell Scripts

There are two ways you can execute your shell scripts. Once you have created a script file:

Method 1
Pass the file as an argument to the shell that you want to interpret your script.

Step 1 : create the script and save it with .sh
For example, the script file show has the following lines
echo Here is the date and time
date

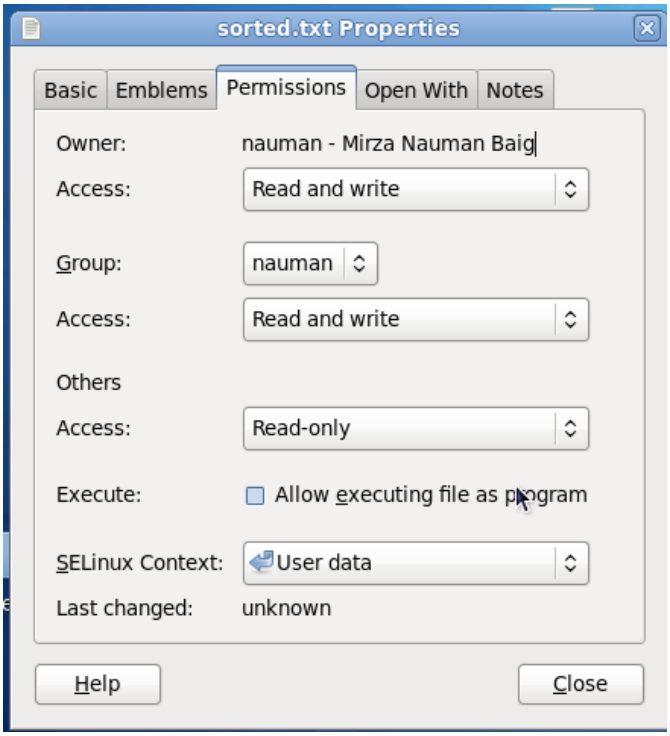
Step 2 : To run the script, pass the filename as an argument to the sh (shell)
$ sh show or sh show.sh
Here is the date and time
Sat jun 03 13:40:15 PST 2011

Method 2:
Make your script executable using the chmod command.
When we create a file, by default it is created with read and write permission turned on and execute permission turned off. A file can be made executable using chmod.

Step 1 : create the script using vi, ex or ed
For example, the script file show has the following lines
echo Here is the date and time
date

Step 2 : Make the file executable
$ chmod u+x script_file
$ chmod u+x show or $chmod u+x /home/selab/show.sh
Step 3 : To run the script, just type the filename
$ show
Here is the date and time
Sat jun 03 13:40:15 PST 2011
Owner u : Group g : other o and Read r : Write w: Execute x

```
$> chmod 755 script.sh
$> chmod u=rwx,g=rx,o=rx script.sh
$> chmod u=rwx,go=rx script.sh
$> chmod u+rwx,g+rx,g-w,o+rx,o-w script.sh
$> chmod u+rwx,go+rx,go-w script.sh
```



if condition

if condition which is used for decision making in shell script, If given condition is true then command1 is executed.
Syntax:

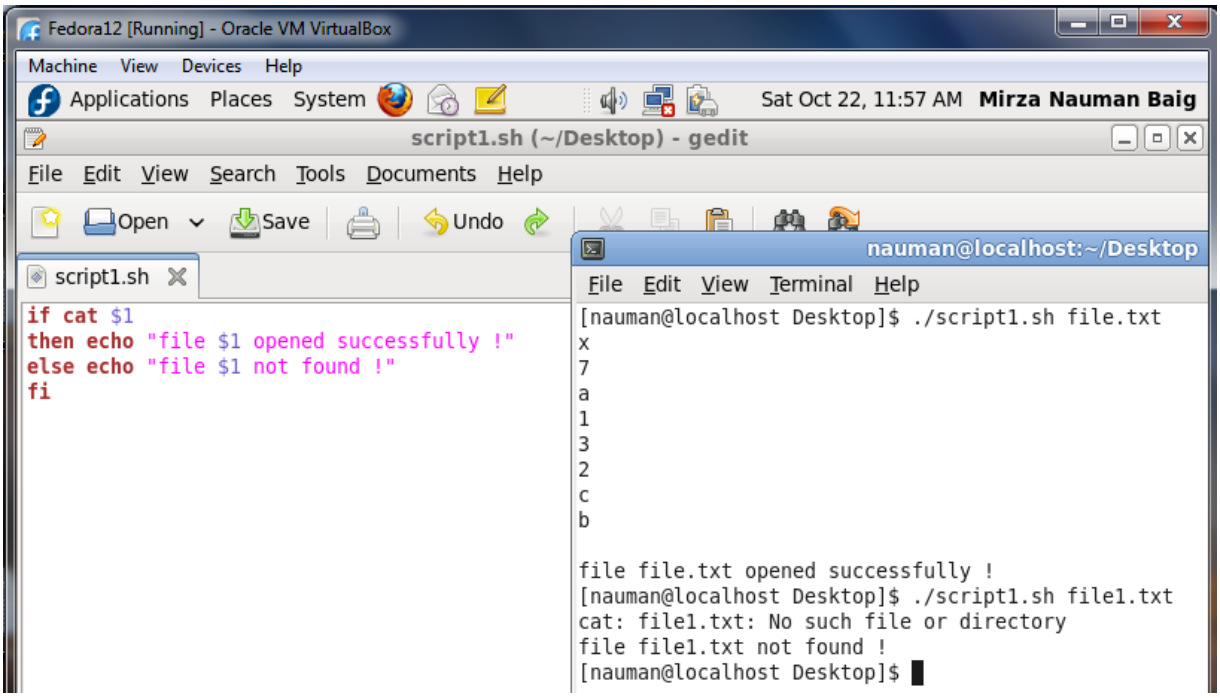
```
if condition
then
        command1 if condition is true or if exit status
        of condition is 0 (zero)
        ...
        ...
fi
```

Condition is defined as:
"*Condition is nothing but comparison between two values.*"

Type following commands (assumes you have file called **foo**)
$ cat foo
A file foo is created
$ echo $?
The cat command return zero(0) i.e. exit status, on successful, this can be used, in if condition as follows, Write shell script as

```
$ cat > showfile
#
#
#Script to print file
#
if cat $1
then
echo  "File $1, found and successfully echoed"
fi
```

Run above script as:
$ chmod 755 showfile
$./showfile foo
Shell script name is showfile ($0) and foo is argument (which is $1).Then shell compare it as follows:
if cat $1 which is expanded to if cat foo.



Detailed explanation
if cat command finds foo file and if its successfully shown on screen, it means our cat command is successful and its exist status is 0 (indicates success), So our if condition is also true and hence statement echo " File $1, found and successfully echoed" is proceed by shell. Now if cat command is not successful then it returns non-zero value (indicates some sort of failure) and this statement echo " File $1, found and successfully echoed" is skipped by our shell.

if...else...fi

If given condition is true then command1 is executed otherwise command2 is executed.
Syntax:

```
if condition
then
            condition is zero (true - 0)
            execute all commands up to else statement

else
            if condition is not true then
            execute all commands up to fi
fi
```

For e.g. Write Script as follows:

```
$ vi isnump_n
#
#
# Script to see whether argument is positive or negative
#
if [ $# -eq 0 ]
then
echo "$0 : You must give/supply one integers"

else
if test $1 -gt 0
then
echo "$1 number is positive"
else
echo "$1 number is negative"
fi

Fi
```

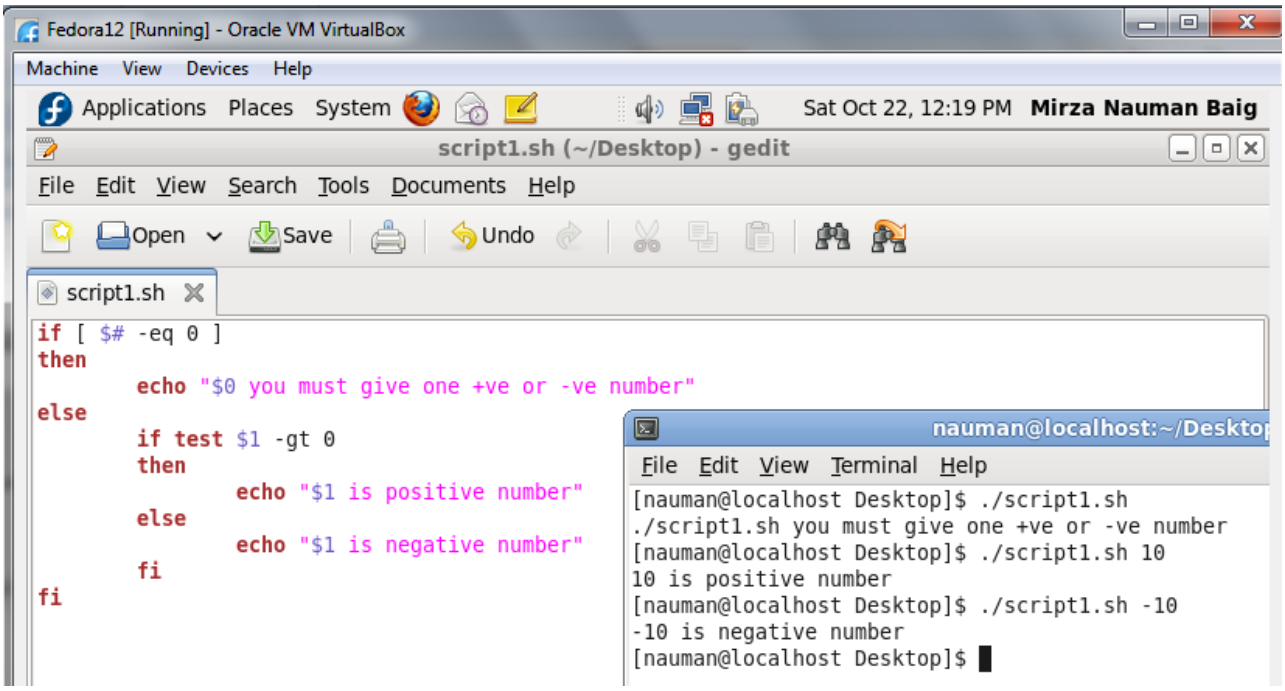
Try it as follows:
$ chmod 755 isnump_n

$ ./isnump_n 5
5 number is positive

$ ./isnump_n -45
-45 number is negative

$ ./isnump_n
./ispos_n : You must give/supply one integers

$ isnump_n 0
0 number is negative



test command or [expr]

test command or [expr] is used to see if an expression is true, and if it is true it return zero(0), otherwise returns nonzero for false.
Syntax:
test expression OR [expression]

Example:
Following script determine whether given argument number is positive.

```
$ cat > ispostive
#
#
# Script to see whether argument is positive
#
if test $1 -gt 0
then
echo "$1 number is positive"
fi
```

Run it as follows
$ chmod 755 ispostive

$./ ispostive 5
5 number is positive

$./ispostive -45
Nothing is printed

$./ispostive
./ispostive: test: -gt: unary operator expected

For Mathematics, use following operator in Shell Script

Mathematical Operator in Shell Script	Meaning	Normal Arithmetical/ Mathematical Statements	But in Shell	
			For test statement with if command	For [expr] statement with if command
-eq	is equal to	5 == 6	if test 5 -eq 6	if [5 -eq 6]
-ne	is not equal to	5 != 6	if test 5 -ne 6	if [5 -ne 6]
-lt	is less than	5 < 6	if test 5 -lt 6	if [5 -lt 6]
-le	is less than or equal to	5 <= 6	if test 5 -le 6	if [5 -le 6]
-gt	is greater than	5 > 6	if test 5 -gt 6	if [5 -gt 6]
-ge	is greater than or equal to	5 >= 6	if test 5 -ge 6	if [5 -ge 6]

NOTE: == is equal, != is not equal.

Multilevel if-then-else

Syntax:

```
            if condition
            then
                        condition is zero (true - 0)
                        execute all commands up to elif statement
            elif condition1
            then
                        condition1 is zero (true - 0)
                        execute all commands up to elif statement
            elif condition2
            then
                        condition2 is zero (true - 0)
                        execute all commands up to elif statement
            else
                        None of the above condtion,condtion1,condtion2 are true
(i.e.
                        all of the above nonzero or false)
                        execute all commands up to fi
            fi
```

For multilevel if-then-else statement try the following script:

```
$ cat > elf
#
# Script to test if..elif...else
#
if [ $1 -gt 0 ]; then
  echo "$1 is positive"
elif [ $1 -lt 0 ]
then
  echo "$1 is negative"
elif [ $1 -eq 0 ]
then
  echo "$1 is zero"
else
  echo "Opps! $1 is not number, give number"
fi
```

Try above script as follows:
$ chmod 755 elf
$ ./elf 1
$ ./elf -2
$ ./elf 0
$ ./elf a

TASK: write a script to ADD two numbers taken from argument
$ ./ADD 5 6

Out put : Sum of 5 and 6 = 5+6 = 11

Note: show error if no argument or more than 2 arguments are passed