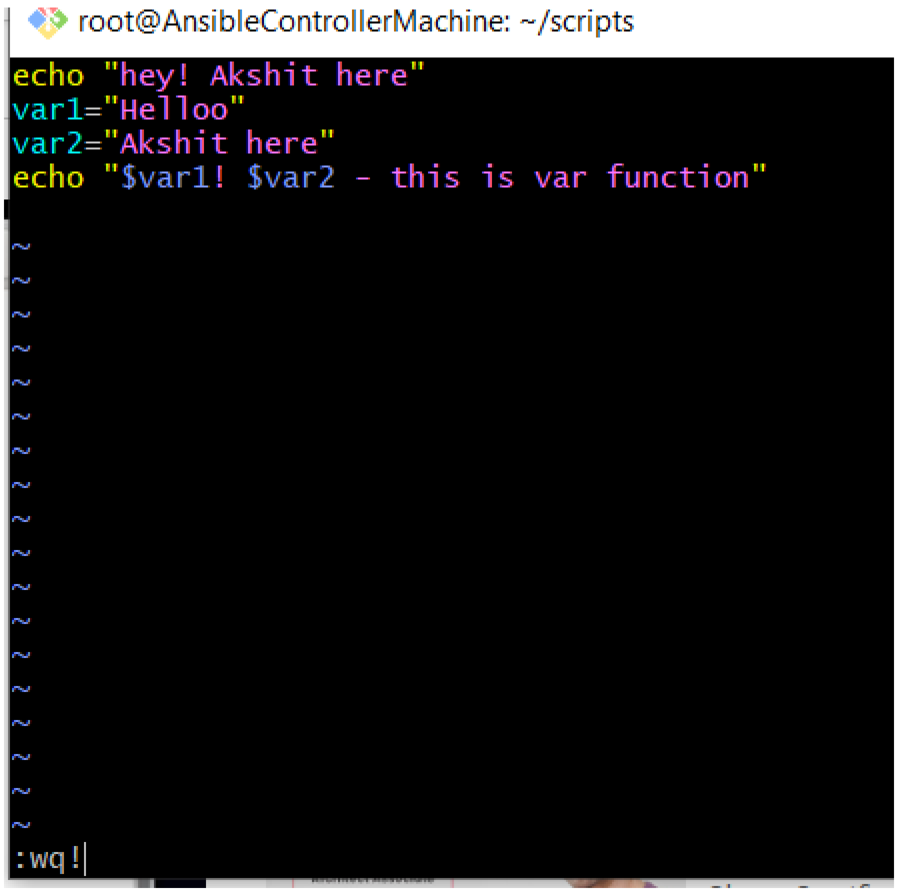
Shell Scripting

We are considering the OS as Linux Distro and Bash as our shell type -:

1. #cat /etc/os-release □ prints the os details of your distribution.
2. #whoami □ prints the name of the current user
3. #pwd □ prints the current working directory
4. #echo $0 □ Prints the shell type.
5. #filename.sh □ .sh defines it is a script file
6. #cat “your data” □ concatenate your file and shows the data within
7. #echo “ data “ □ Stores the data in the file
8. #ls –al □ List the contents of the file along with the permissions given to them.
9. #chmod +x Filename.sh □ Adds the executable permission to the file
10. #./MyFile.sh OR /path to file/MyFile.sh □ Execute/run the script file
11. #bash FileName.sh □ Execute the file without the executable permission
12. # “your comment”□ use to write comments in the file

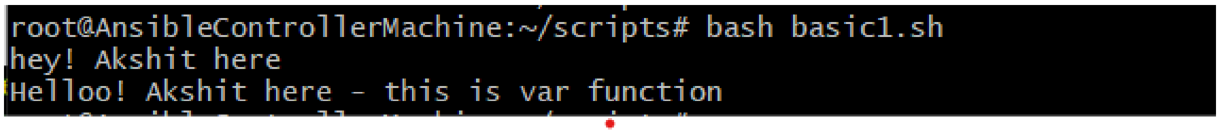
13. <<comment “multi line comment” comment multi line comment add

14. Varname= “some value” , echo “ this is var - $varname” Define the variable

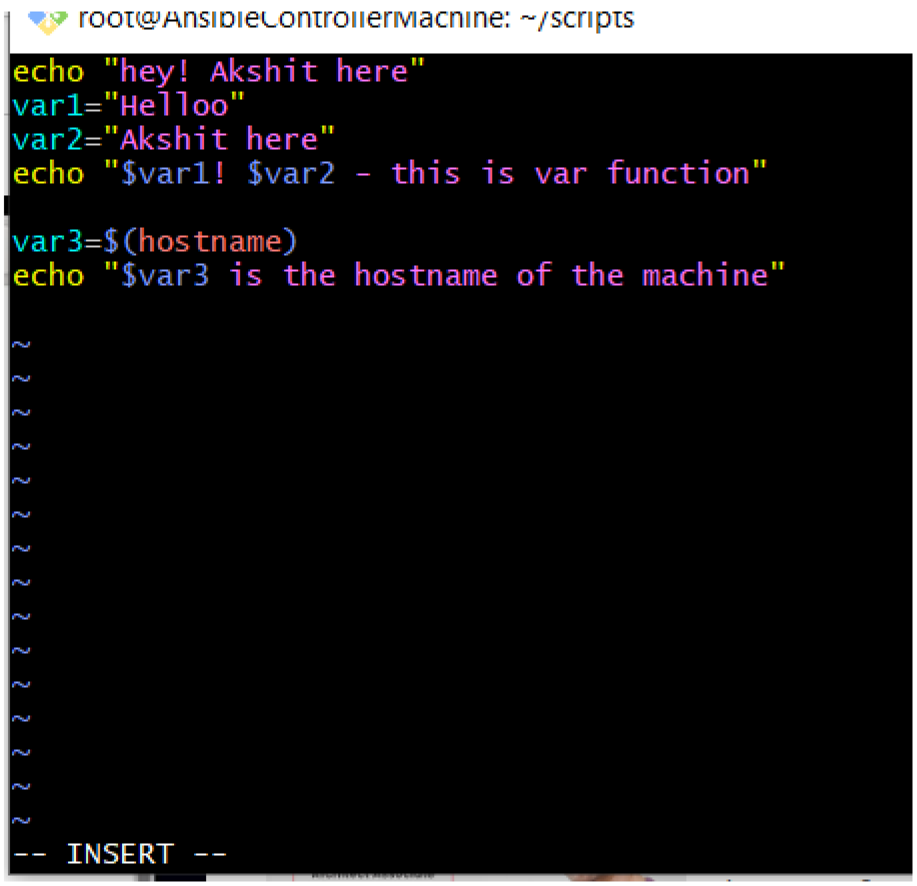


root@AnsibleControllerMachine: ~/scripts

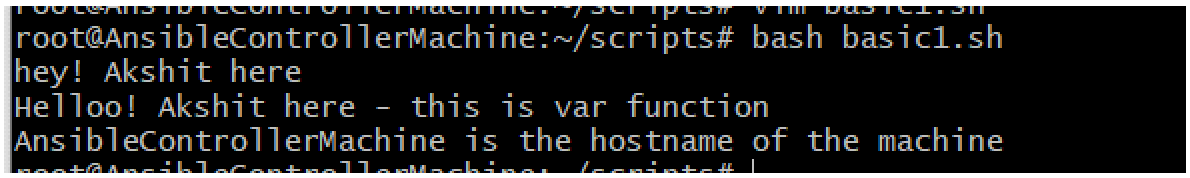
```
echo "hey! Akshit here"
var1="Helloo"
var2="Akshit here"
echo "$var1! $var2 - this is var function"

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:wq!
```



```
root@AnsibleControllerMachine:~/scripts# bash basic1.sh
hey! Akshit here
Helloo! Akshit here - this is var function
```

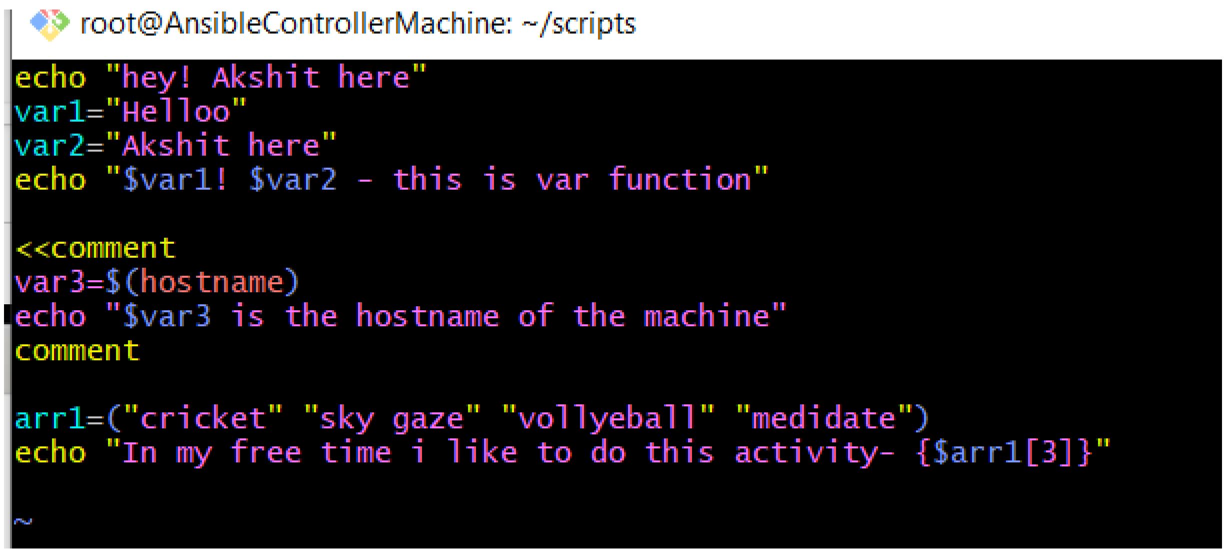


```
root@AnsibleControllerMachine: ~/scripts
echo "hey! Akshit here"
var1="Helloo"
var2="Akshit here"
echo "$var1! $var2 - this is var function"

var3=$(hostname)
echo "$var3 is the hostname of the machine"

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --
```



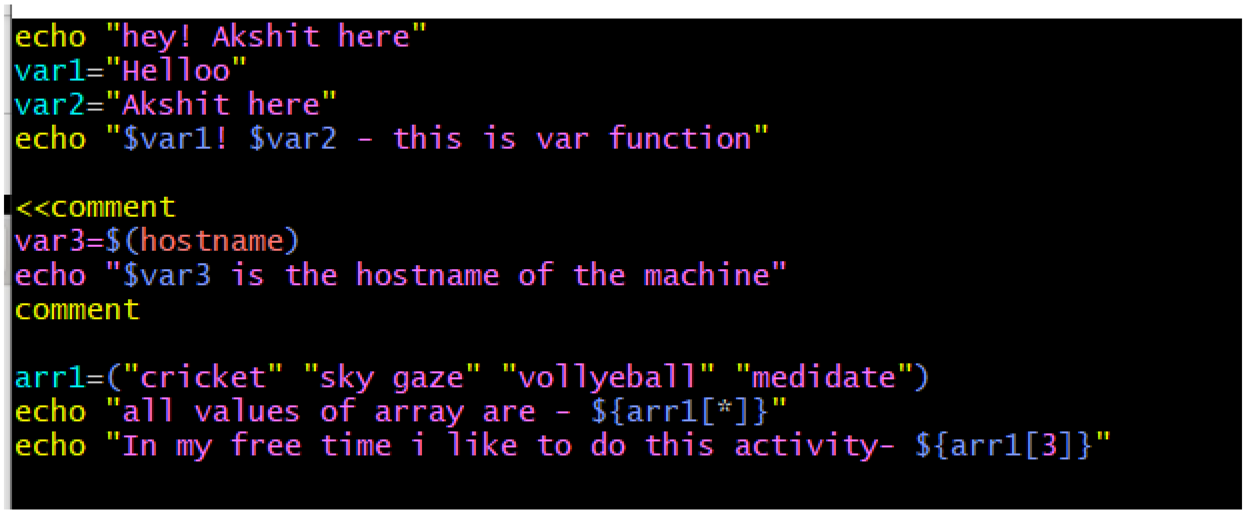
```
root@AnsibleControllerMachine:~/scripts# vim basic1.sh
root@AnsibleControllerMachine:~/scripts# bash basic1.sh
hey! Akshit here
Helloo! Akshit here - this is var function
AnsibleControllerMachine is the hostname of the machine
root@AnsibleControllerMachine:~/scripts#
```

15. Readonly VarName = “ value” □ Defines the constant variable which cannot be changed.

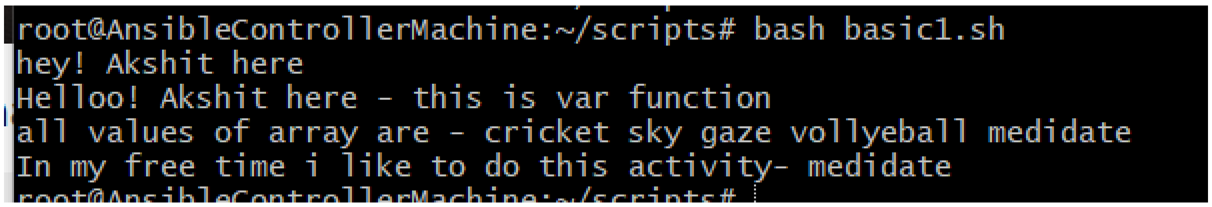

root@AnsibleControllerMachine: ~/scripts

```
echo "hey! Akshit here"
var1="Helloo"
var2="Akshit here"
echo "$var1! $var2 - this is var function"

<<comment
var3=$(hostname)
echo "$var3 is the hostname of the machine"
comment

arr1=("cricket" "sky gaze" "vollyeball" "medidate")
echo "In my free time i like to do this activity- {$arr1[3]}"

~
```

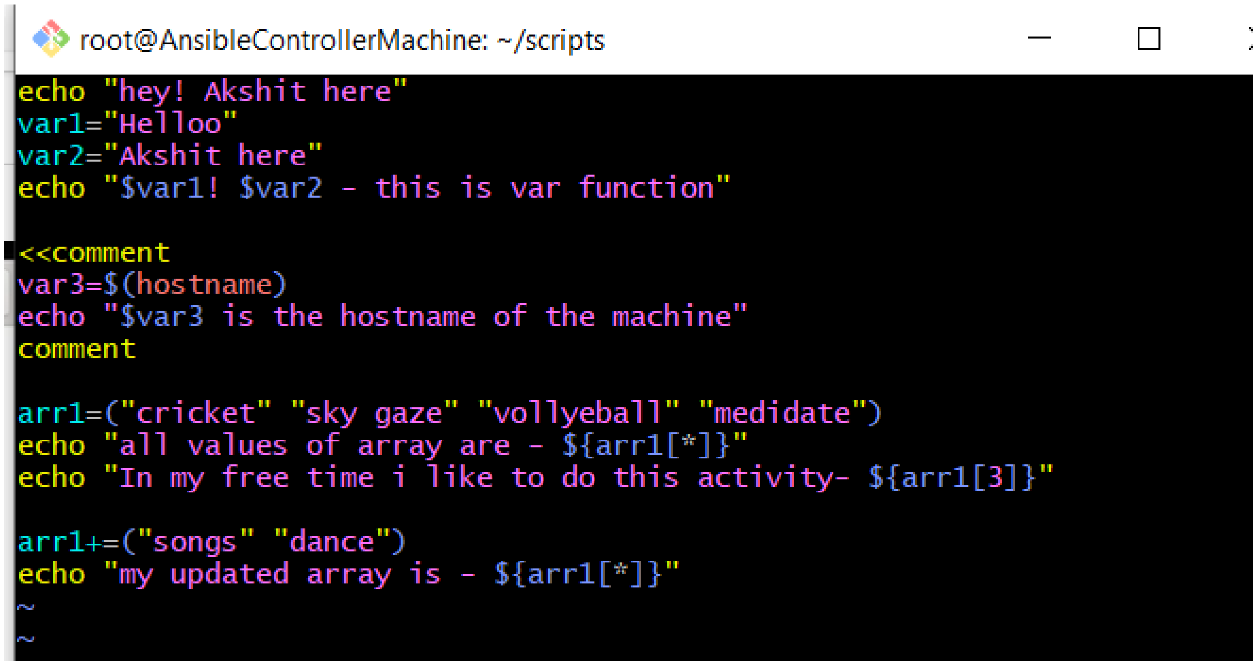
16. Arrays:



```
echo "hey! Akshit here"
var1="Helloo"
var2="Akshit here"
echo "$var1! $var2 - this is var function"

<<comment
var3=$(hostname)
echo "$var3 is the hostname of the machine"
comment

arr1=("cricket" "sky gaze" "vollyeball" "medidate")
echo "all values of array are - ${arr1[*]}"
echo "In my free time i like to do this activity- ${arr1[3]}"
```

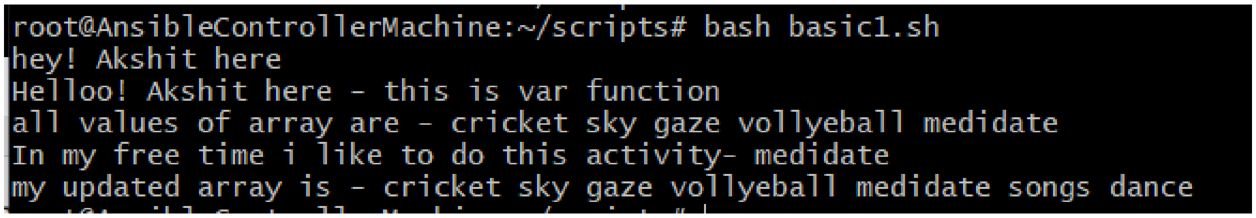


```
root@AnsibleControllerMachine:~/scripts# bash basic1.sh
hey! Akshit here
Helloo! Akshit here - this is var function
all values of array are - cricket sky gaze vollyeball medidate
In my free time i like to do this activity- medidate
root@AnsibleControllerMachine:~/scripts#
```



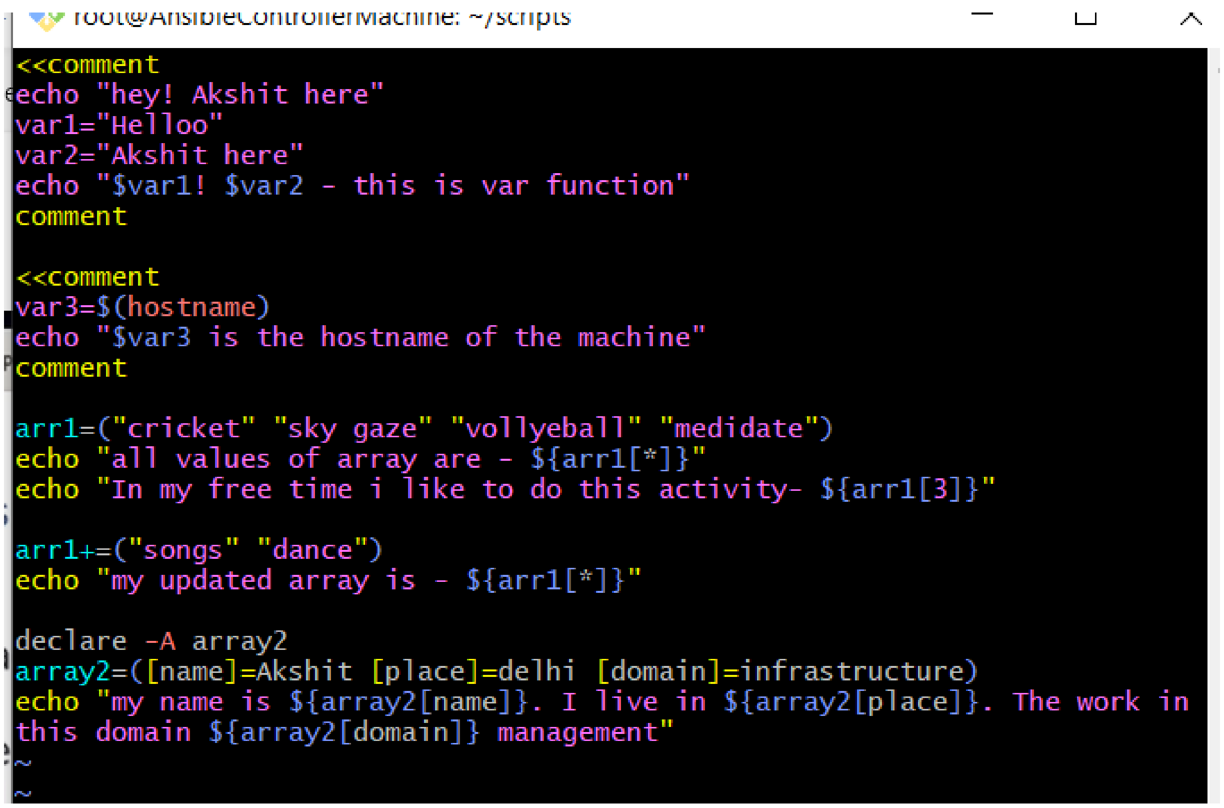
root@AnsibleControllerMachine: ~/scripts

```
echo "hey! Akshit here"
var1="Helloo"
var2="Akshit here"
echo "$var1! $var2 - this is var function"

<<comment
var3=$(hostname)
echo "$var3 is the hostname of the machine"
comment

arr1=("cricket" "sky gaze" "vollyeball" "medidate")
echo "all values of array are - ${arr1[*]}"
echo "In my free time i like to do this activity- ${arr1[3]}"

arr1+=("songs" "dance")
echo "my updated array is - ${arr1[*]}"
~
~
```

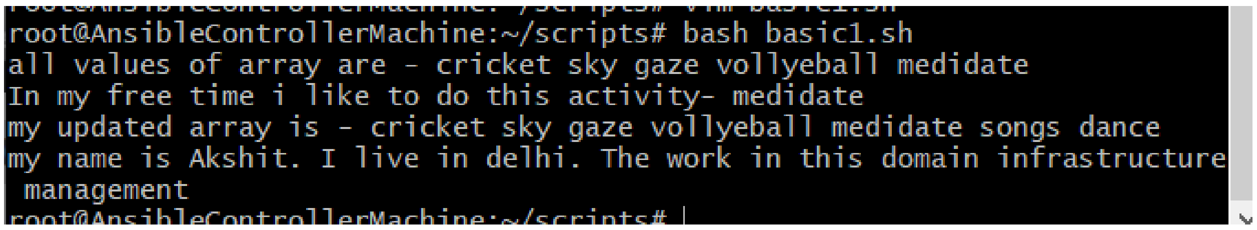


```
root@AnsibleControllerMachine:~/scripts# bash basic1.sh
hey! Akshit here
Helloo! Akshit here - this is var function
all values of array are - cricket sky gaze vollyeball medidate
In my free time i like to do this activity- medidate
my updated array is - cricket sky gaze vollyeball medidate songs dance
```

17. Key value pair in array-

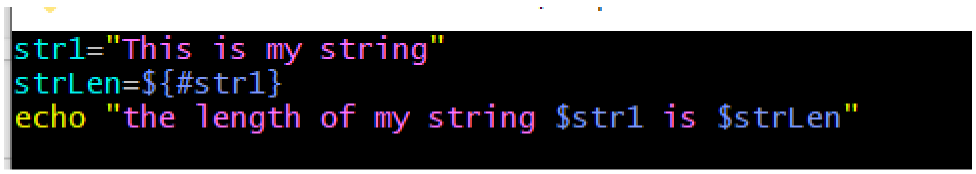


```
<<comment
echo "hey! Akshit here"
var1="Helloo"
var2="Akshit here"
echo "$var1! $var2 - this is var function"
comment

<<comment
var3=$(hostname)
echo "$var3 is the hostname of the machine"
comment

arr1=("cricket" "sky gaze" "vollyeball" "medidate")
echo "all values of array are - ${arr1[*]}"
echo "In my free time i like to do this activity- ${arr1[3]}"

arr1+=("songs" "dance")
echo "my updated array is - ${arr1[*]}"

declare -A array2
array2=([name]=Akshit [place]=delhi [domain]=infrastructure)
echo "my name is ${array2[name]}. I live in ${array2[place]}. The work in
this domain ${array2[domain]} management"
~
~
```

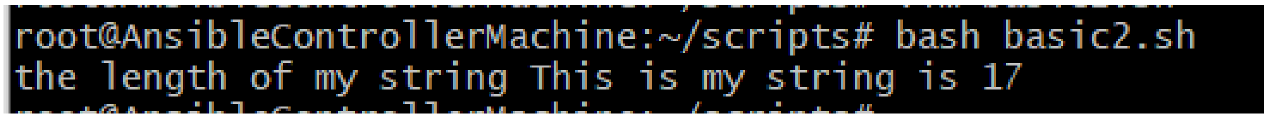


```
root@AnsibleControllerMachine:~/scripts# bash basic1.sh
all values of array are - cricket sky gaze vollyeball medidate
In my free time i like to do this activity- medidate
my updated array is - cricket sky gaze vollyeball medidate songs dance
my name is Akshit. I live in delhi. The work in this domain infrastructure
 management
root@AnsibleControllerMachine:~/scripts#
```

18. Strings

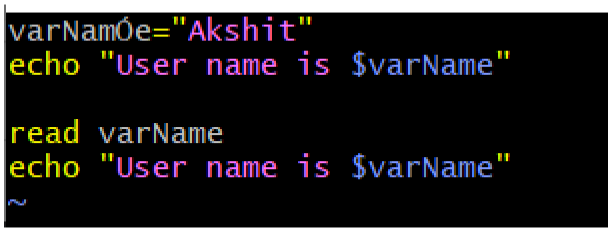


```
str1="This is my string"
strLen=${#str1}
echo "the length of my string $str1 is $strLen"
```

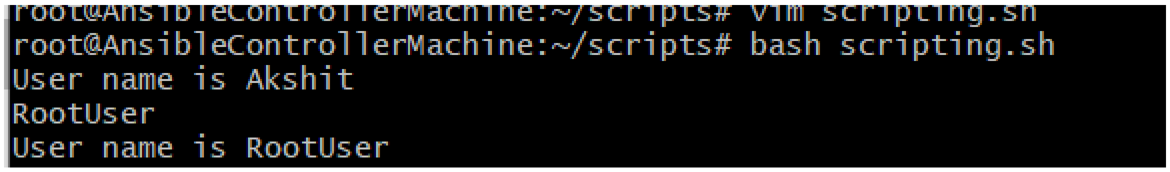


```
root@AnsibleControllerMachine:~/scripts# bash basic2.sh
the length of my string This is my string is 17
```

19. Read Input from user at run time

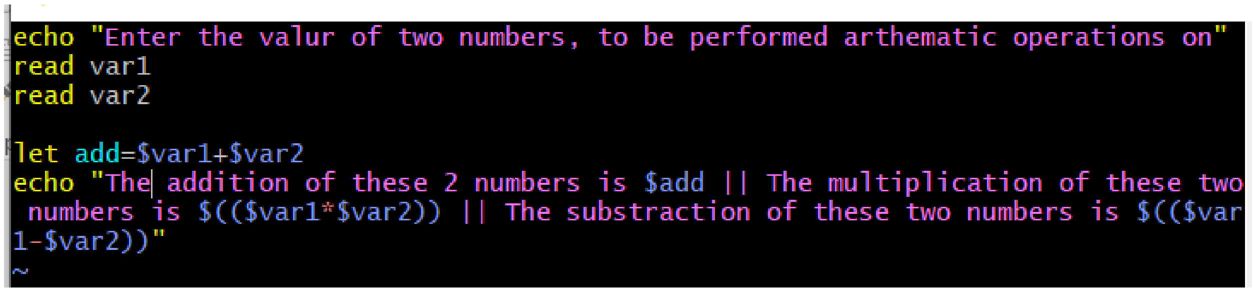


```
varNamÓe="Akshit"
echo "User name is $varName"

read varName
echo "User name is $varName"
~
```

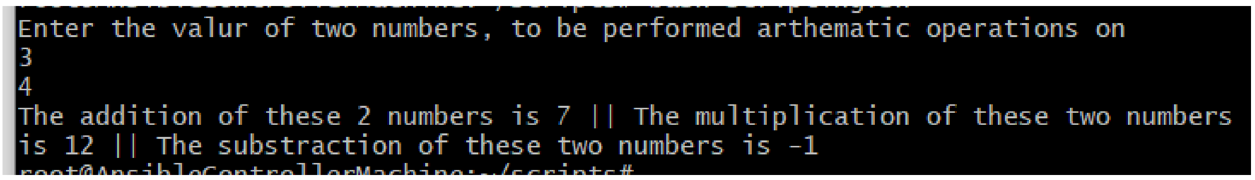


```
root@AnsibleControllerMachine:~/scripts# vim scripting.sh
root@AnsibleControllerMachine:~/scripts# bash scripting.sh
User name is Akshit
RootUser
User name is RootUser
```

20. Arithmetic operations:



```
echo "Enter the valur of two numbers, to be performed arthematic operations on"
read var1
read var2

let add=$var1+$var2
echo "The addition of these 2 numbers is $add || The multiplication of these two
 numbers is $(($var1*$var2)) || The substraction of these two numbers is $(($var
1-$var2))"
~
```

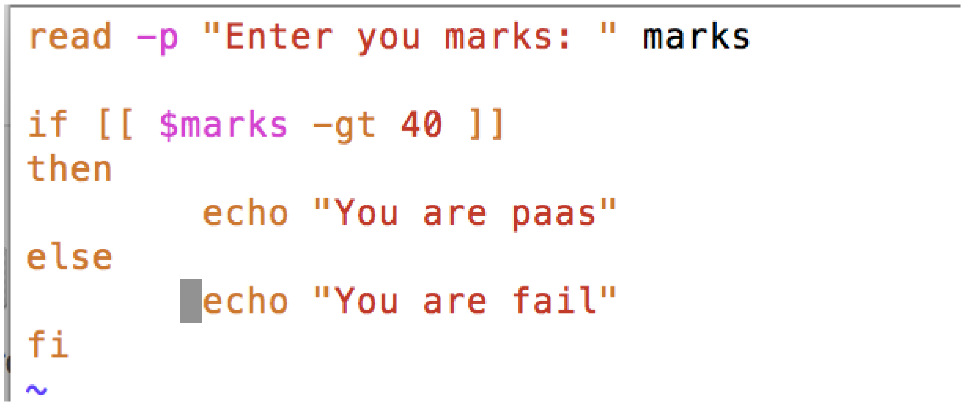


```
Enter the valur of two numbers, to be performed arthematic operations on
3
4
The addition of these 2 numbers is 7 || The multiplication of these two numbers
is 12 || The substraction of these two numbers is -1
root@AnsibleControllerMachine:~/scripts#
```

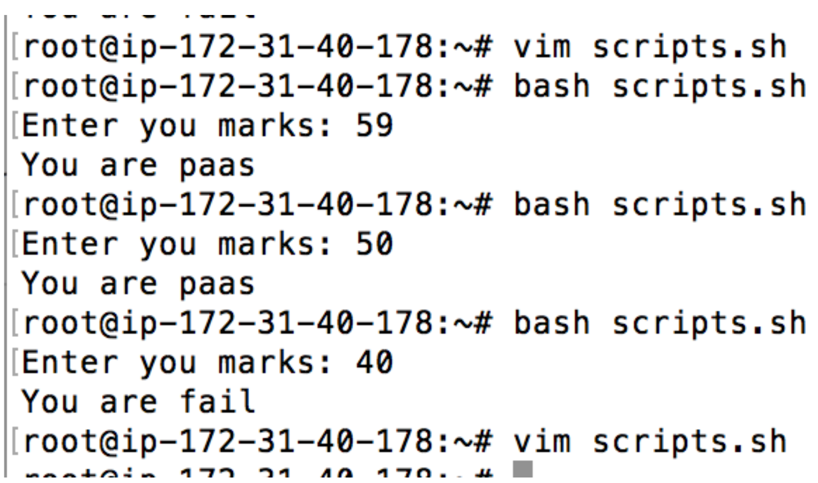
===

1. Sh-Bang command: Command used before every script. It defines the name of shell type we are using. Here, it is bash, so we will write - #!/bin/bash.
   Please note* This is not the mandatory command. This is just ethical way of writing the scripts

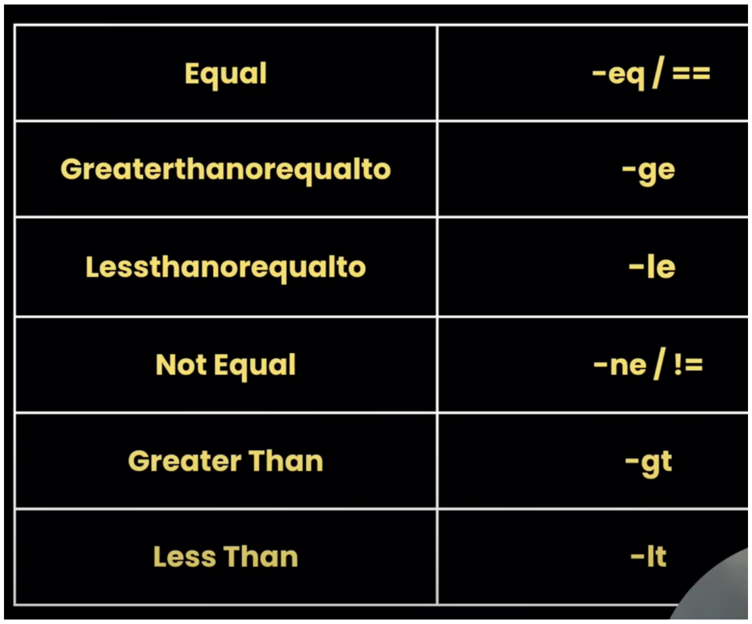
2. If-ElseConditions:



```
read -p "Enter you marks: " marks

if [[ $marks -gt 40 ]]
then
        echo "You are paas"
else
        echo "You are fail"
fi
~
```



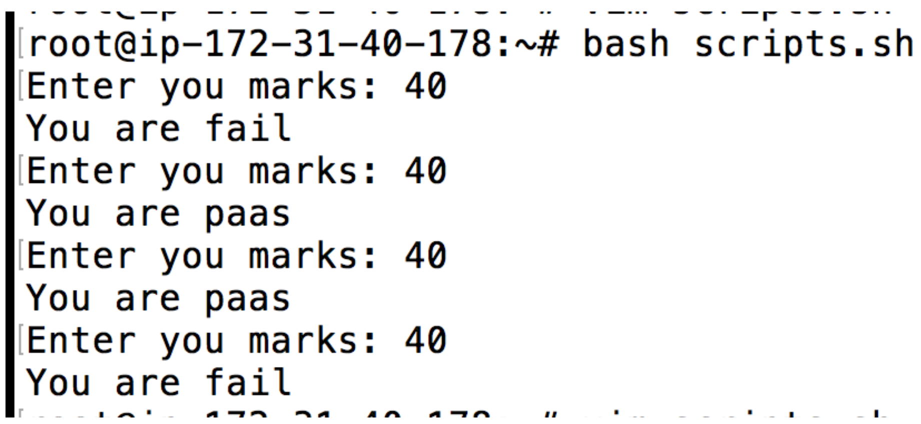
```
root@ip-172-31-40-178:~# vim scripts.sh
root@ip-172-31-40-178:~# bash scripts.sh
Enter you marks: 59
You are paas
root@ip-172-31-40-178:~# bash scripts.sh
Enter you marks: 50
You are paas
root@ip-172-31-40-178:~# bash scripts.sh
Enter you marks: 40
You are fail
root@ip-172-31-40-178:~# vim scripts.sh
```



Equal	-eq / ==
Greaterthanorequalto	-ge
Lessthanorequalto	-le
Not Equal	-ne / !=
Greater Than	-gt
Less Than	-lt

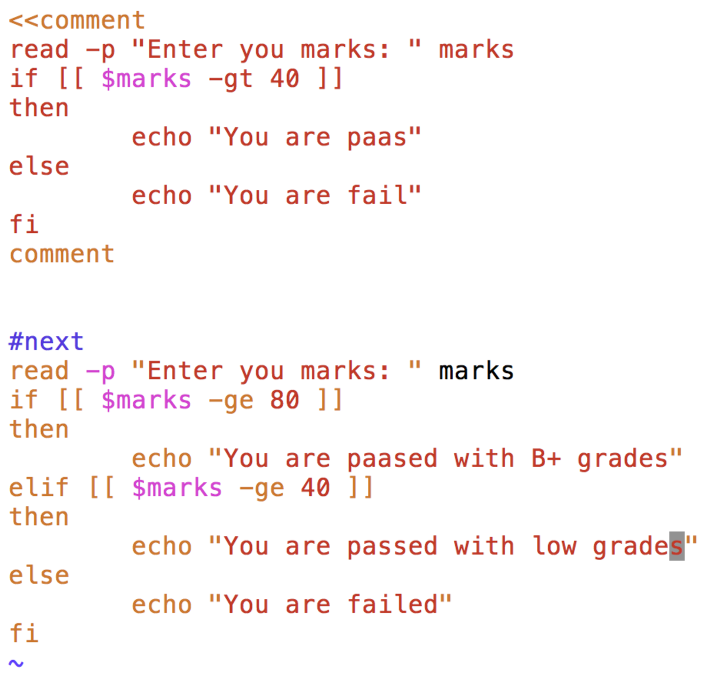
3. Other cases of if-else conditions

```
read -p "Enter you marks: " marks
if [[ $marks -gt 40 ]]
then
        echo "You are paas"
else
        echo "You are fail"
fi

#next
read -p "Enter you marks: " marks
if [[ $marks -ge 40 ]]
then
        echo "You are paas"
else
        echo "You are fail"
fi

#next
read -p "Enter you marks: " marks
if [[ $marks == 40 ]]
then
        echo "You are paas"
else
        echo "You are fail"
fi

#next
read -p "Enter you marks: " marks
if [[ $marks != 40 ]]
then
        echo "You are paas"
else
        echo "You are fail"
fi
```

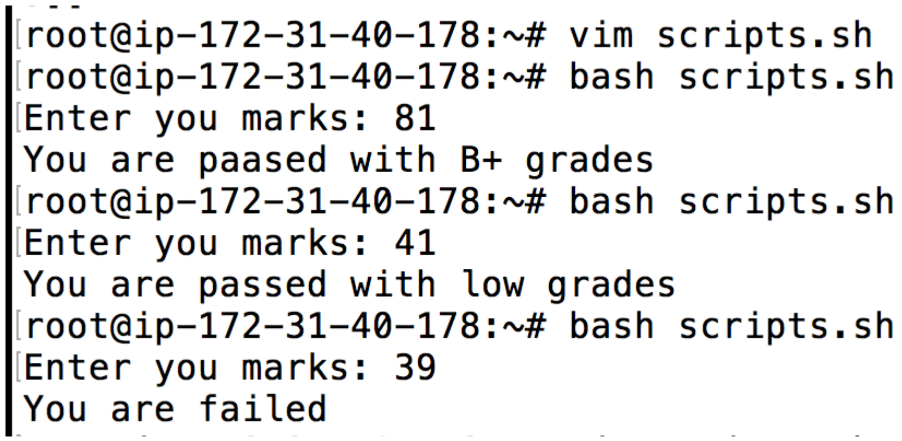


```
root@ip-172-31-40-178:~# bash scripts.sh
Enter you marks: 40
You are fail
Enter you marks: 40
You are paas
Enter you marks: 40
You are paas
Enter you marks: 40
You are fail
```

4. Elseif condition

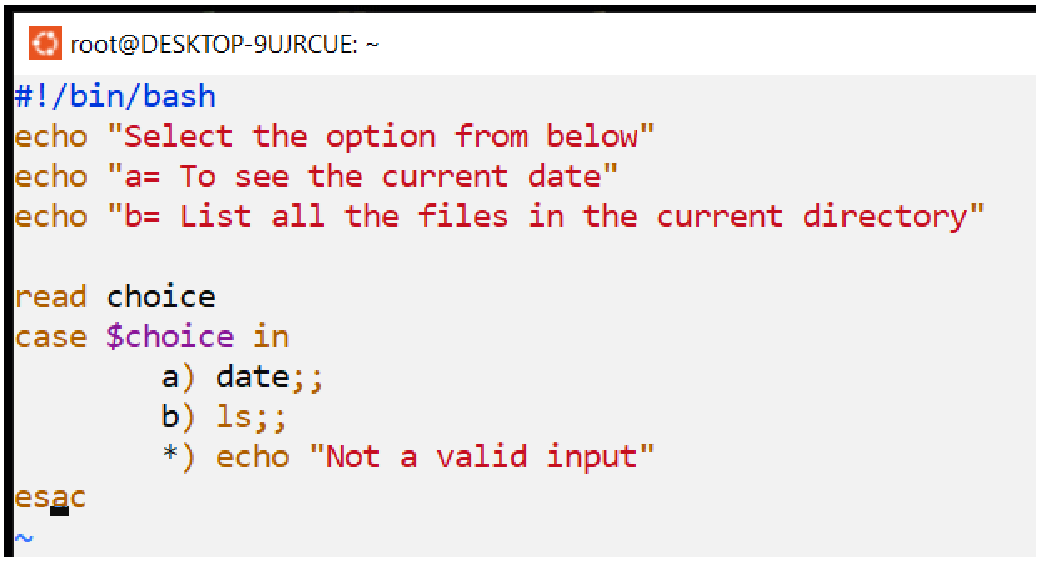


```
<<comment
read -p "Enter you marks: " marks
if [[ $marks -gt 40 ]]
then
        echo "You are paas"
else
        echo "You are fail"
fi
comment


#next
read -p "Enter you marks: " marks
if [[ $marks -ge 80 ]]
then
        echo "You are paased with B+ grades"
elif [[ $marks -ge 40 ]]
then
        echo "You are passed with low grades"
else
        echo "You are failed"
fi
~
```



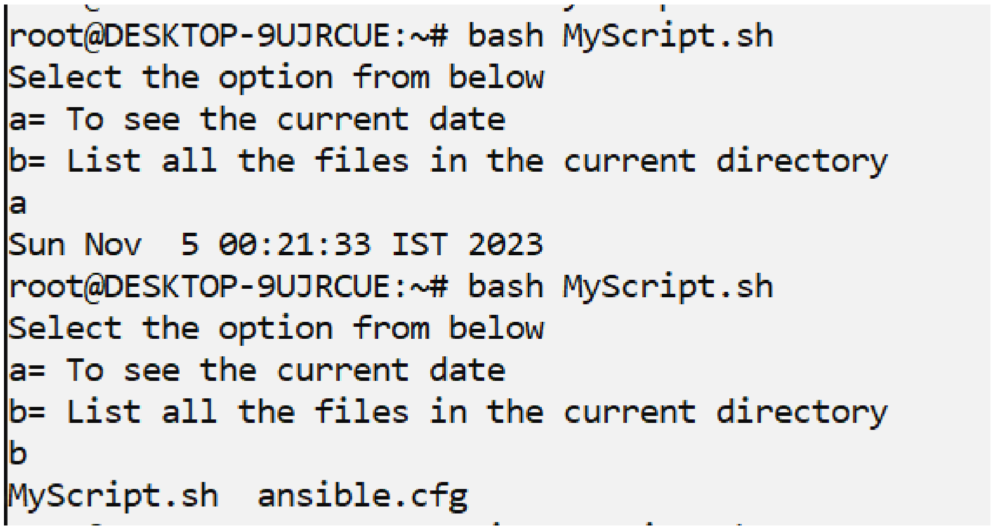
```
root@ip-172-31-40-178:~# vim scripts.sh
root@ip-172-31-40-178:~# bash scripts.sh
Enter you marks: 81
You are paased with B+ grades
root@ip-172-31-40-178:~# bash scripts.sh
Enter you marks: 41
You are passed with low grades
root@ip-172-31-40-178:~# bash scripts.sh
Enter you marks: 39
You are failed
```

5. Case function in shell



root@DESKTOP-9UJRCUE: ~

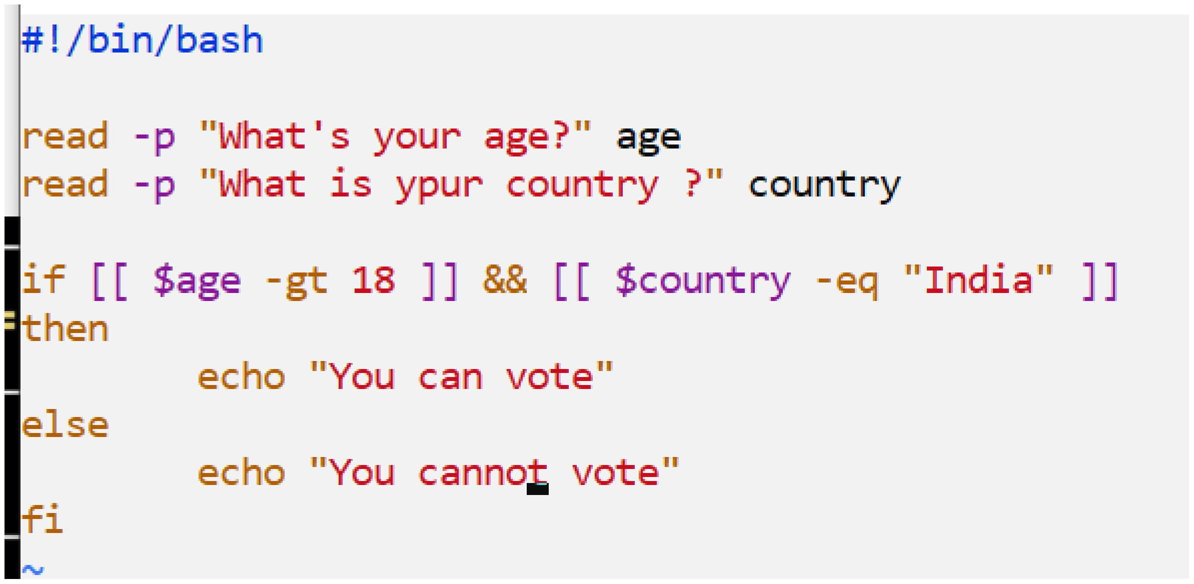
```
#!/bin/bash
echo "Select the option from below"
echo "a= To see the current date"
echo "b= List all the files in the current directory"

read choice
case $choice in
        a) date;;
        b) ls;;
        *) echo "Not a valid input"
esac
~
```



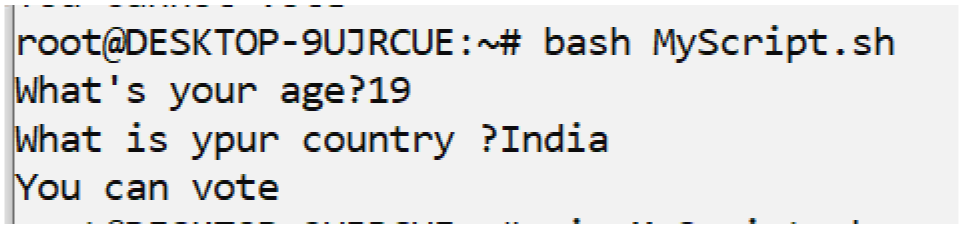
```
root@DESKTOP-9UJRCUE:~# bash MyScript.sh
Select the option from below
a= To see the current date
b= List all the files in the current directory
a
Sun Nov  5 00:21:33 IST 2023
root@DESKTOP-9UJRCUE:~# bash MyScript.sh
Select the option from below
a= To see the current date
b= List all the files in the current directory
b
MyScript.sh  ansible.cfg
```

6. AND Operatory and reading values form the user

AND case

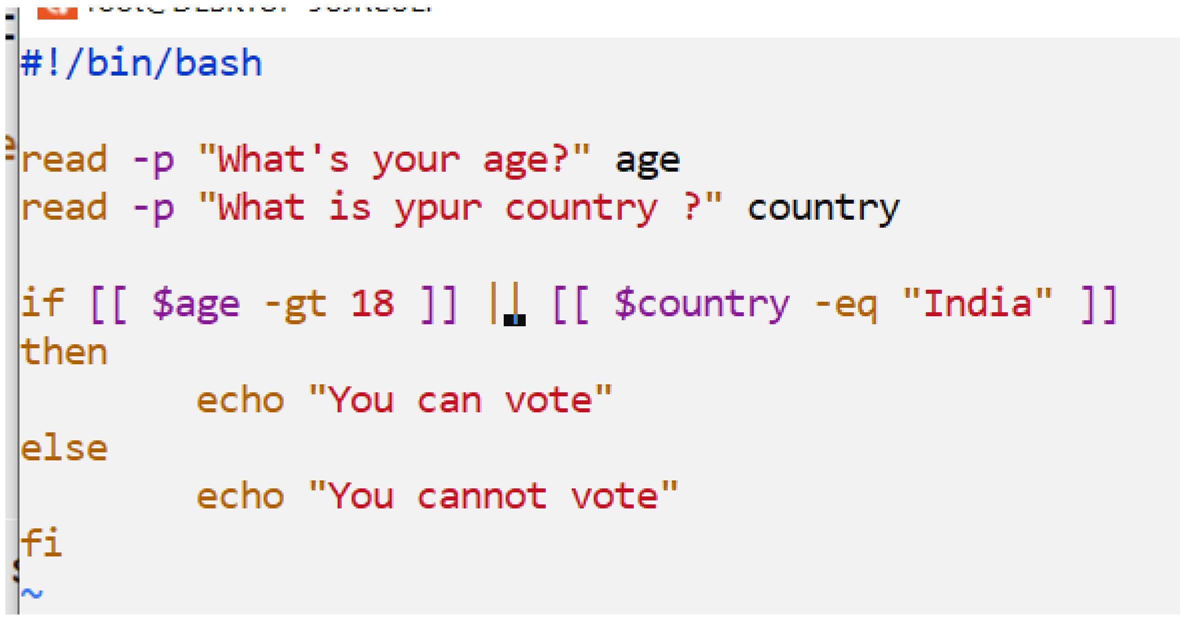


```
#!/bin/bash

read -p "What's your age?" age
read -p "What is ypur country ?" country

if [[ $age -gt 18 ]] && [[ $country -eq "India" ]]
then
        echo "You can vote"
else
        echo "You cannot vote"
fi
~
```

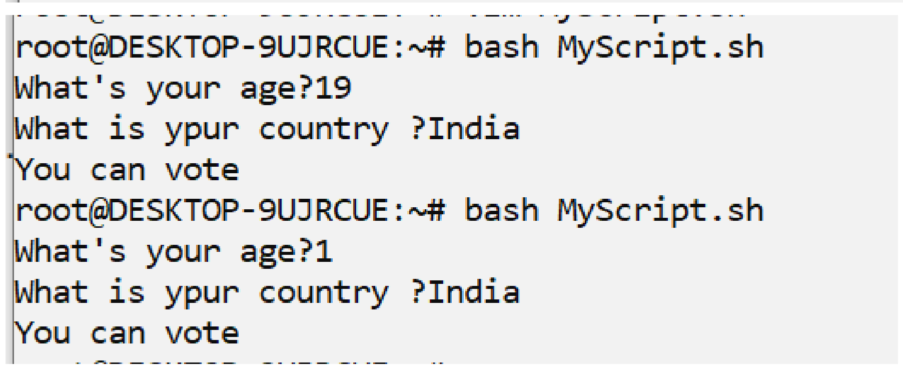


```
root@DESKTOP-9UJRCUE:~# bash MyScript.sh
What's your age?19
What is ypur country ?India
You can vote
```

OR Case

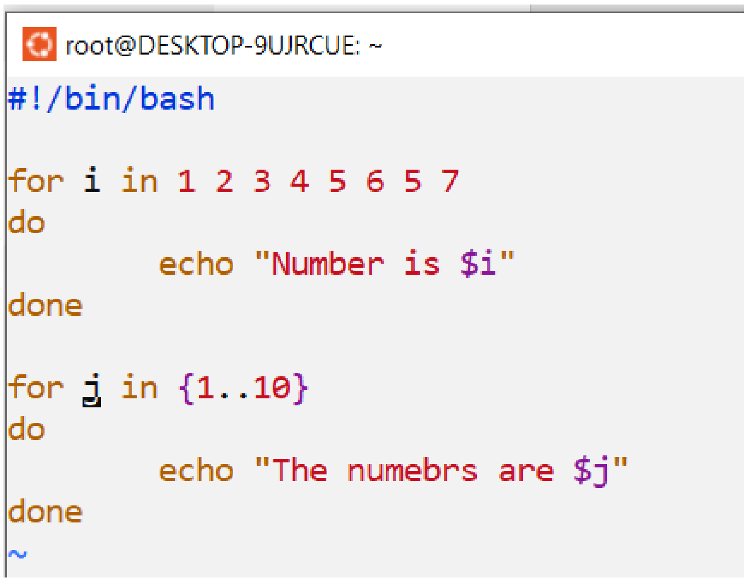


```bash
#!/bin/bash

read -p "What's your age?" age
read -p "What is ypur country ?" country

if [[ $age -gt 18 ]] || [[ $country -eq "India" ]]
then
        echo "You can vote"
else
        echo "You cannot vote"
fi
~
```



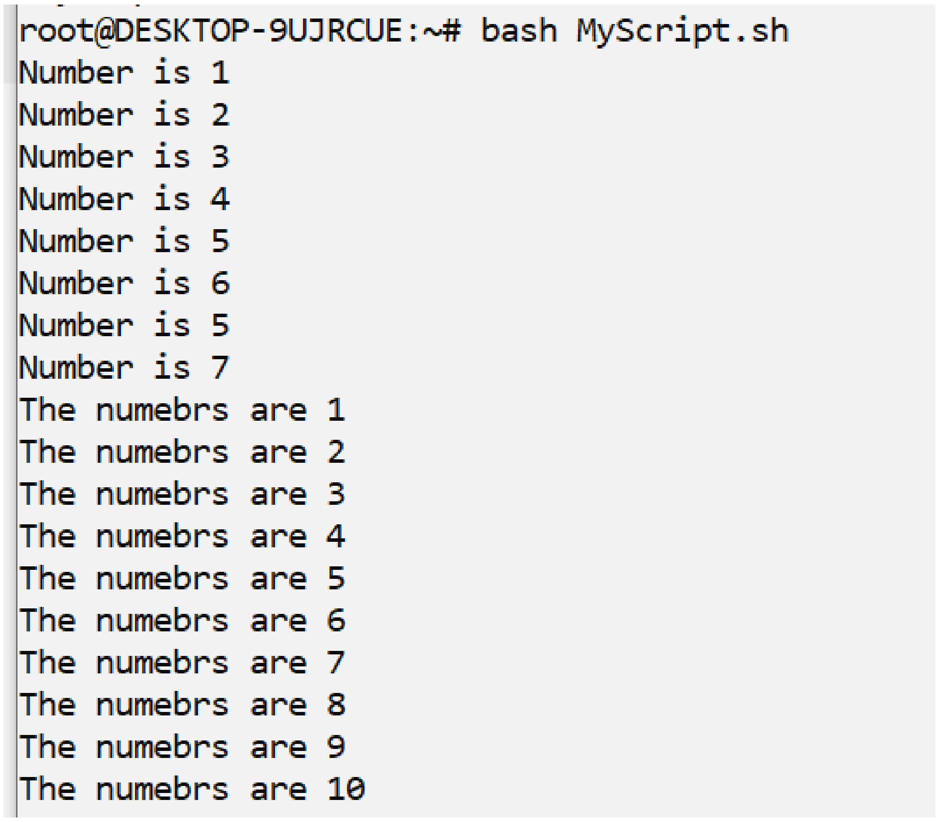
```
root@DESKTOP-9UJRCUE:~# bash MyScript.sh
What's your age?19
What is ypur country ?India
You can vote
root@DESKTOP-9UJRCUE:~# bash MyScript.sh
What's your age?1
What is ypur country ?India
You can vote
```

7. FOR loop

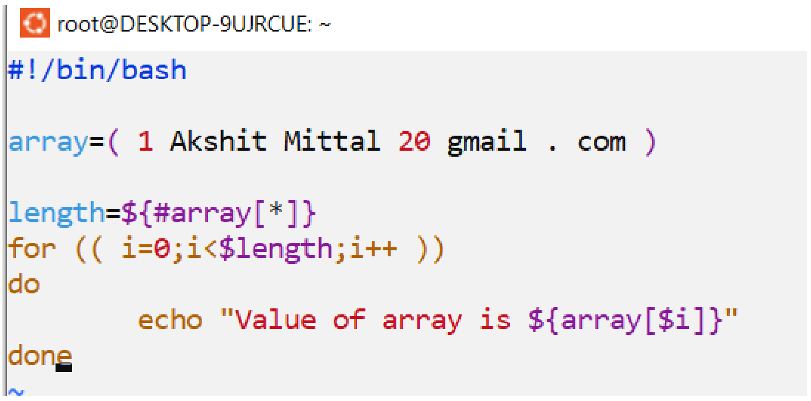


root@DESKTOP-9UJRCUE: ~

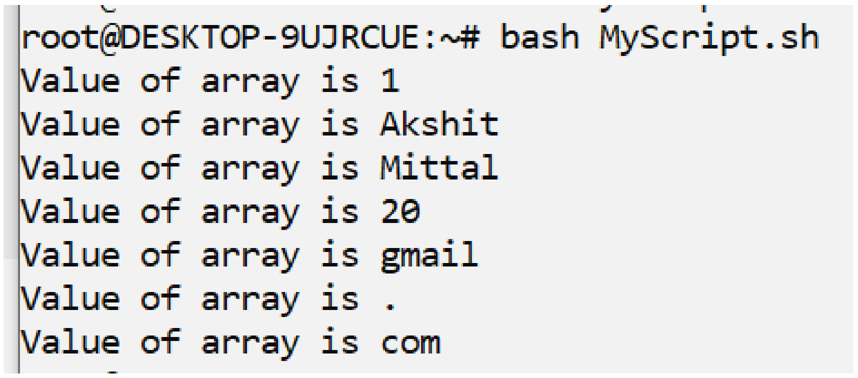
```
#!/bin/bash

for i in 1 2 3 4 5 6 5 7
do
        echo "Number is $i"
done

for j in {1..10}
do
        echo "The numebrs are $j"
done
~
```



```
root@DESKTOP-9UJRCUE:~# bash MyScript.sh
Number is 1
Number is 2
Number is 3
Number is 4
Number is 5
Number is 6
Number is 5
Number is 7
The numebrs are 1
The numebrs are 2
The numebrs are 3
The numebrs are 4
The numebrs are 5
The numebrs are 6
The numebrs are 7
The numebrs are 8
The numebrs are 9
The numebrs are 10
```

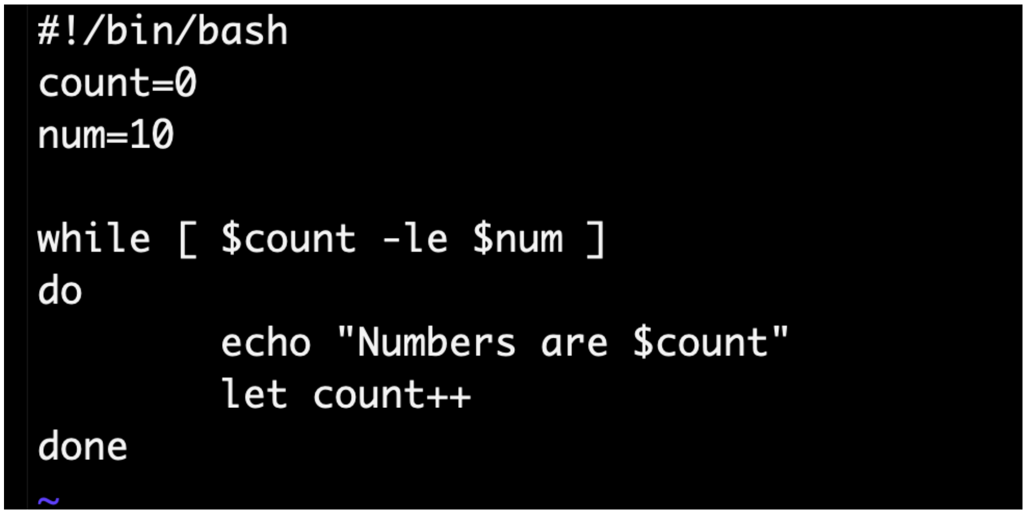


```
root@DESKTOP-9UJRCUE: ~
#!/bin/bash

array=( 1 Akshit Mittal 20 gmail . com )

length=${#array[*]}
for (( i=0;i<$length;i++ ))
do
        echo "Value of array is ${array[$i]}"
done
~
```

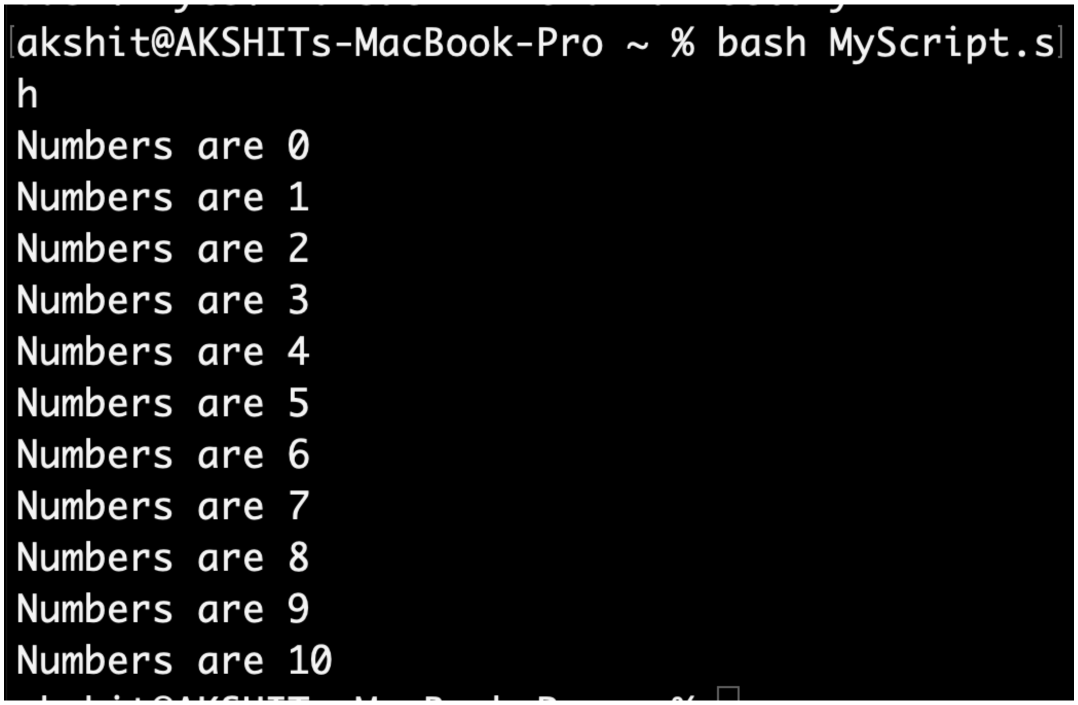


```
root@DESKTOP-9UJRCUE:~# bash MyScript.sh
Value of array is 1
Value of array is Akshit
Value of array is Mittal
Value of array is 20
Value of array is gmail
Value of array is .
Value of array is com
```

8. While Loop

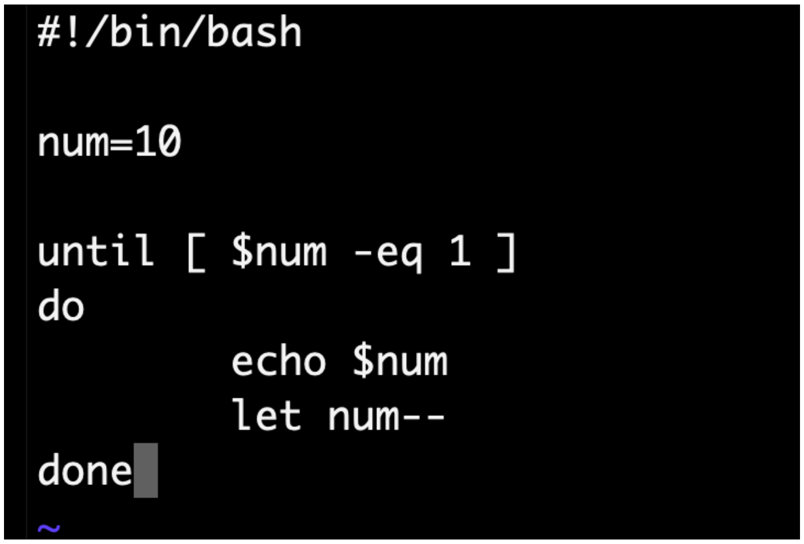


```
#!/bin/bash
count=0
num=10

while [ $count -le $num ]
do
        echo "Numbers are $count"
        let count++
done
~
```

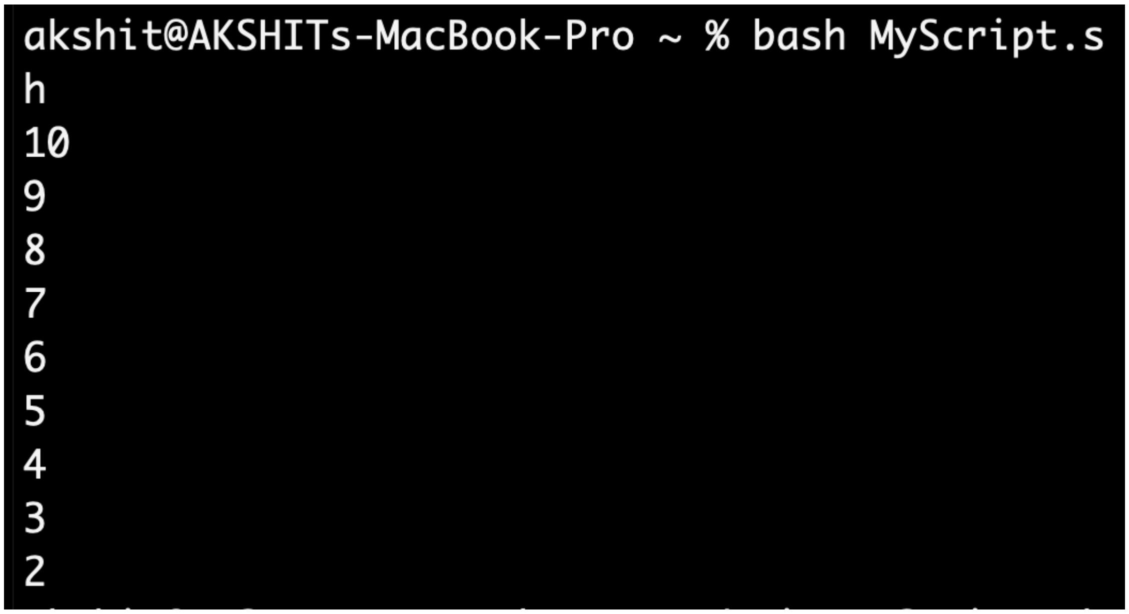


```
akshit@AKSHITs-MacBook-Pro ~ % bash MyScript.s
h
Numbers are 0
Numbers are 1
Numbers are 2
Numbers are 3
Numbers are 4
Numbers are 5
Numbers are 6
Numbers are 7
Numbers are 8
Numbers are 9
Numbers are 10
```

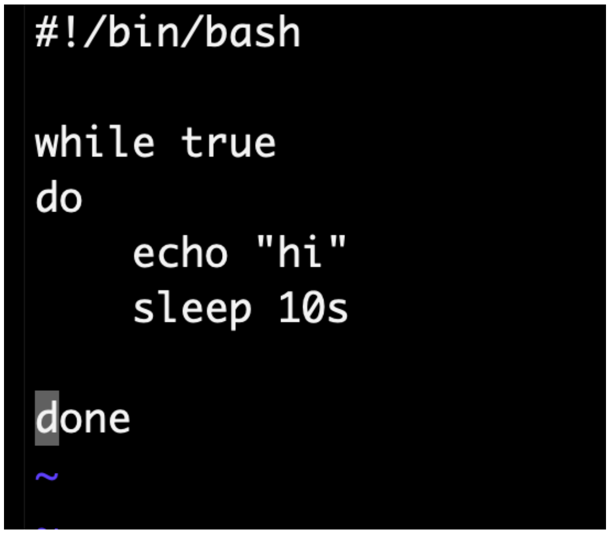
9. Until loop



```
#!/bin/bash

num=10

until [ $num -eq 1 ]
do
        echo $num
        let num--
done
~
```



```
akshit@AKSHITs-MacBook-Pro ~ % bash MyScript.s
h
10
9
8
7
6
5
4
3
2
```



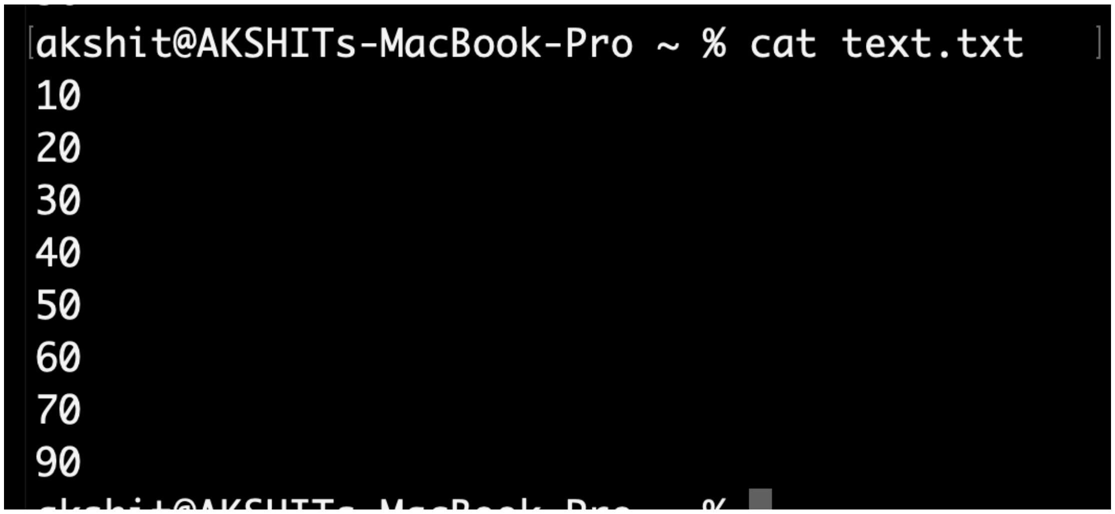
```
#!/bin/bash

while true
do
    echo "hi"
    sleep 10s

done
~
```

You can also create timer in the script to run the loop after every 10 seconds

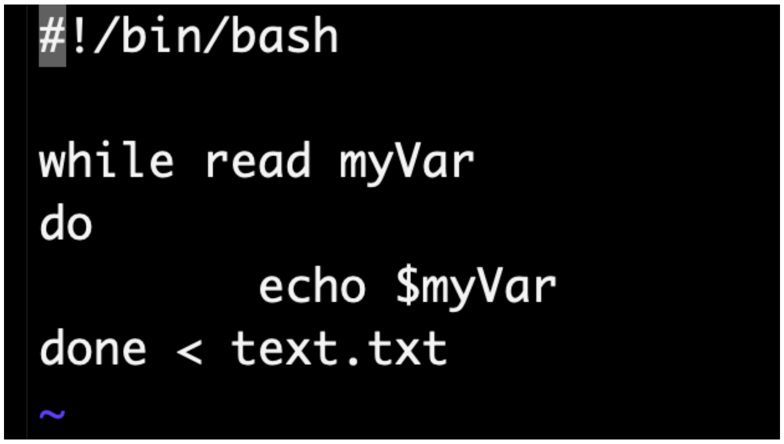
Similarly, we can use all the other loops in the scripts.

10. Read contents from file

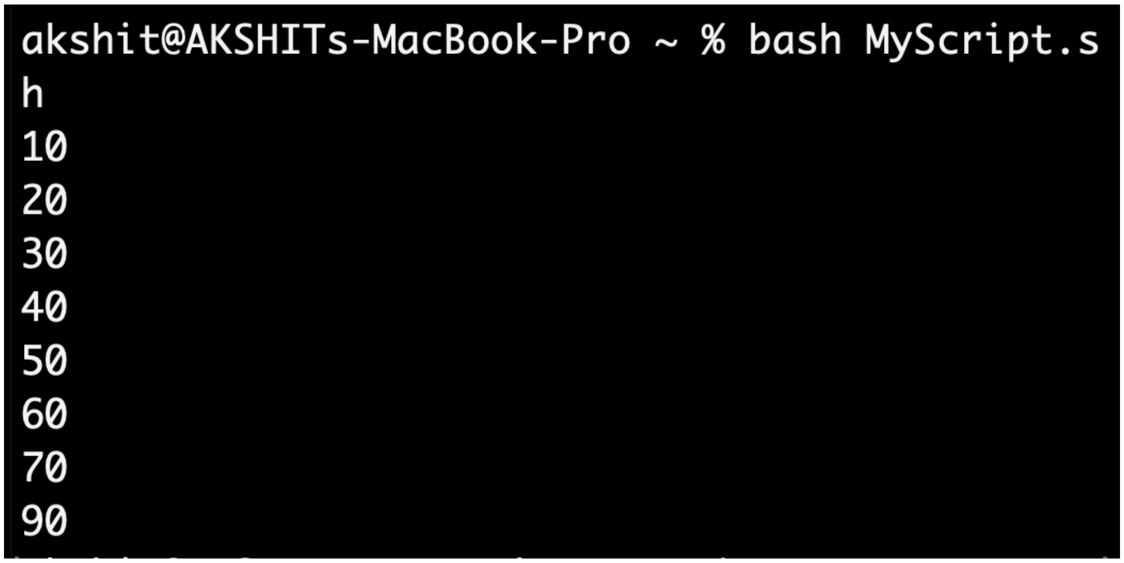
There is a file named text.txt which has some values, which we want to read via script



```
akshit@AKSHITs-MacBook-Pro ~ % cat text.txt
10
20
30
40
50
60
70
90
```



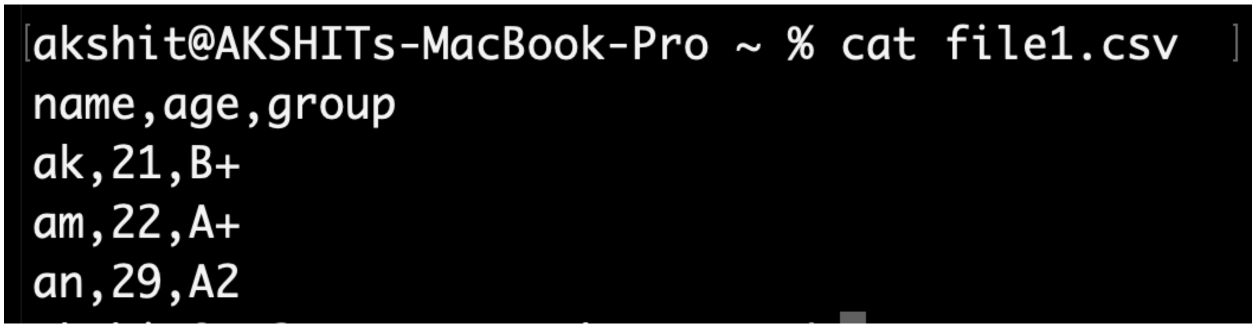
```
#!/bin/bash

while read myVar
do
        echo $myVar
done < text.txt
~
```



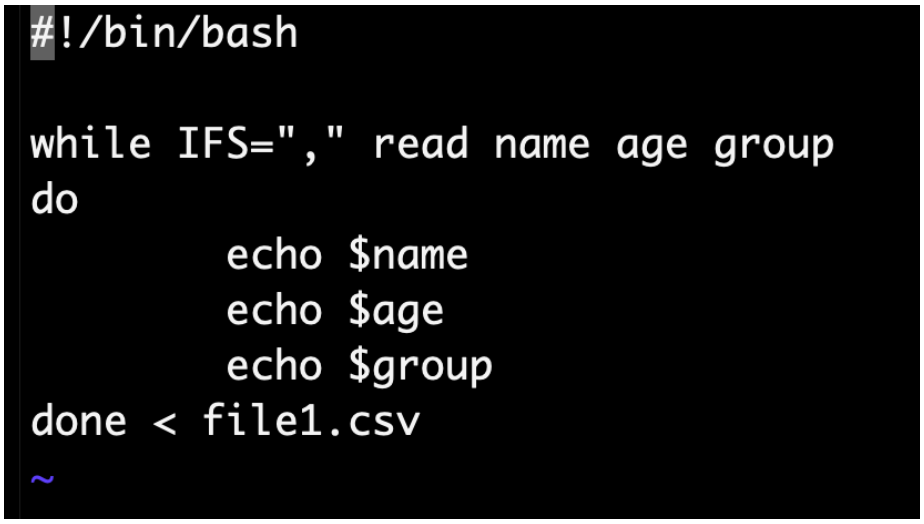
```
akshit@AKSHITs-MacBook-Pro ~ % bash MyScript.s
h
10
20
30
40
50
60
70
90
```

The output shows the content of the file

Now, lets read the contents of .csv file

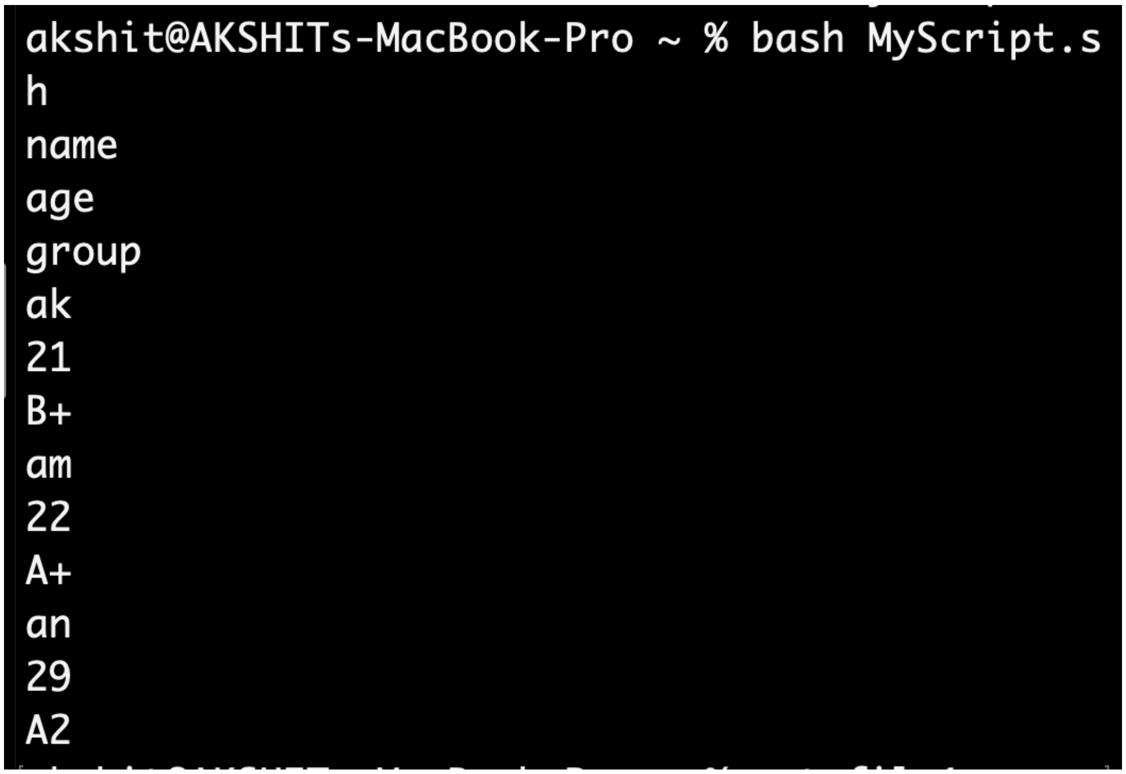


```
akshit@AKSHITs-MacBook-Pro ~ % cat file1.csv
name,age,group
ak,21,B+
am,22,A+
an,29,A2
```



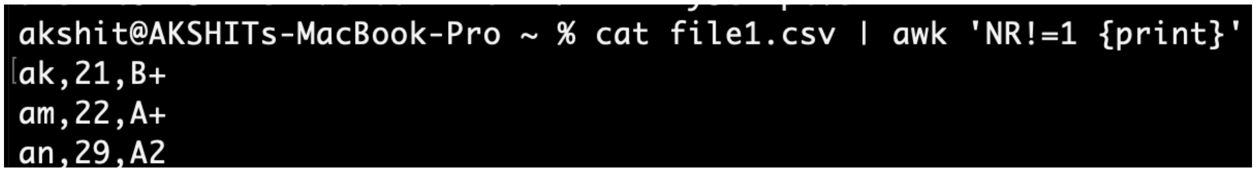
```
#!/bin/bash

while IFS="," read name age group
do
        echo $name
        echo $age
        echo $group
done < file1.csv
~
```

Here, The IFS is the file separator



```
akshit@AKSHITs-MacBook-Pro ~ % bash MyScript.s
h
name
age
group
ak
21
B+
am
22
A+
an
29
A2
```

Now, we will do this using awk tool

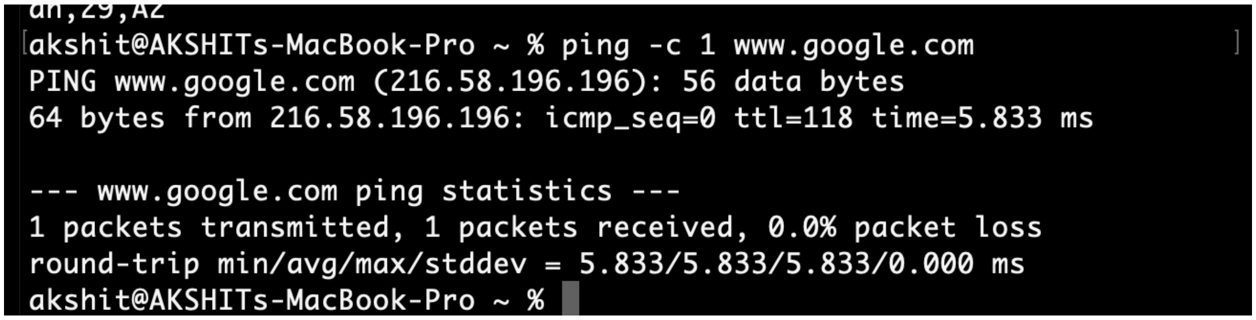


```
akshit@AKSHITs-MacBook-Pro ~ % cat file1.csv | awk 'NR!=1 {print}'
ak,21,B+
am,22,A+
an,29,A2
```

As we can see the 1st header row has been removed

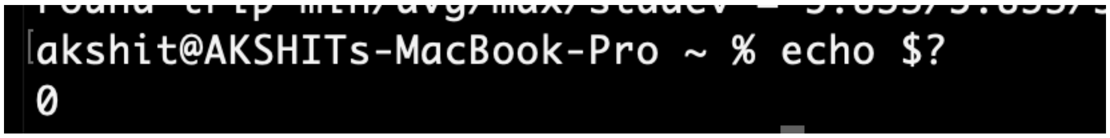
11. Checking connectivity to a server and check the packet loss

    - packet loss in networking happens when the data packets are transmitted from one source to another but doesn't reach there completely.

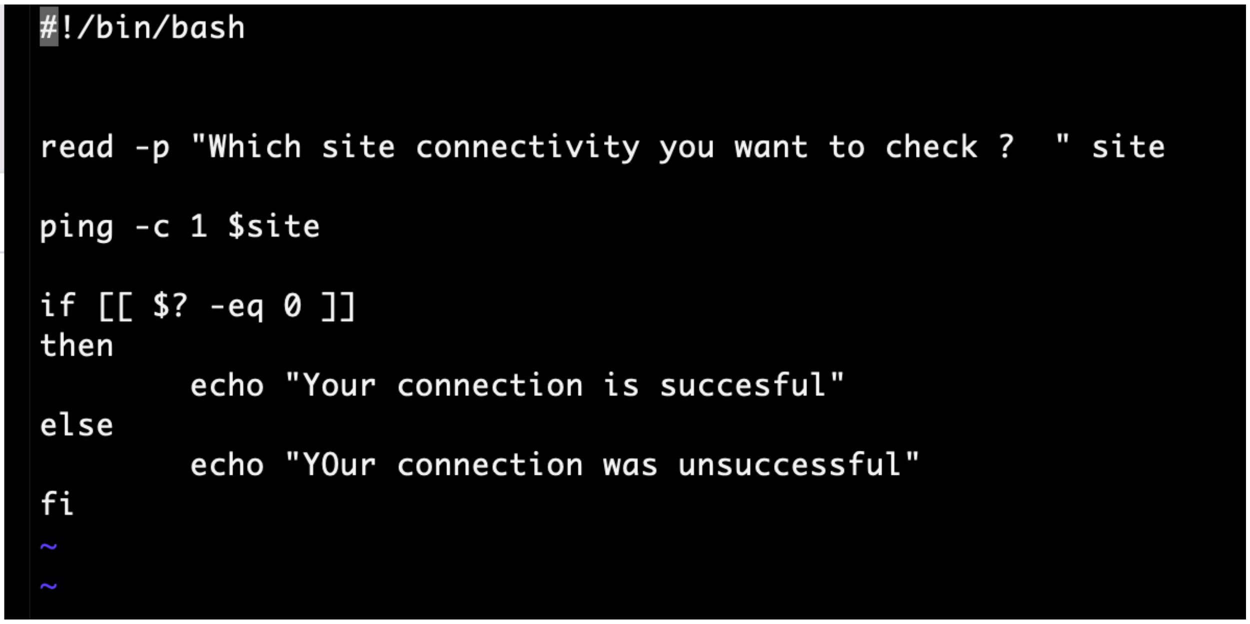


```
an,29,AZ
akshit@AKSHITs-MacBook-Pro ~ % ping -c 1 www.google.com
PING www.google.com (216.58.196.196): 56 data bytes
64 bytes from 216.58.196.196: icmp_seq=0 ttl=118 time=5.833 ms

--- www.google.com ping statistics ---
1 packets transmitted, 1 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 5.833/5.833/5.833/0.000 ms
akshit@AKSHITs-MacBook-Pro ~ %
```

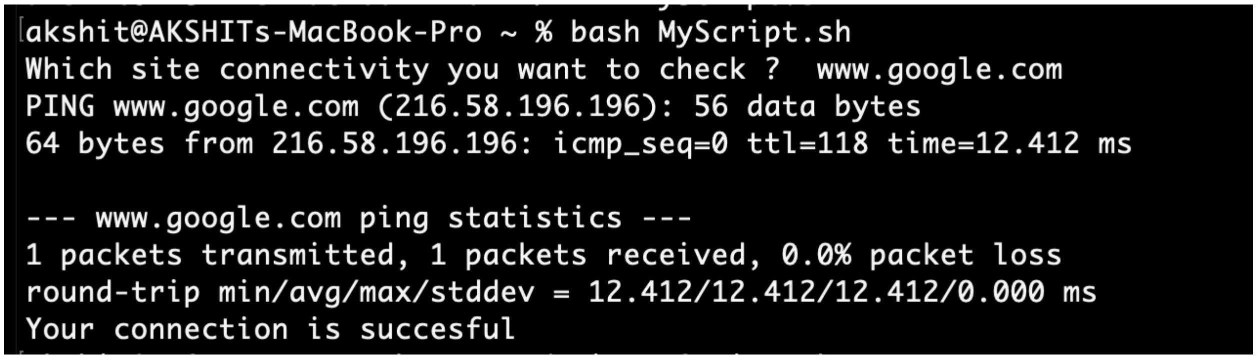
Checking the packet loss with the command



```
akshit@AKSHITs-MacBook-Pro ~ % echo $?
0
```

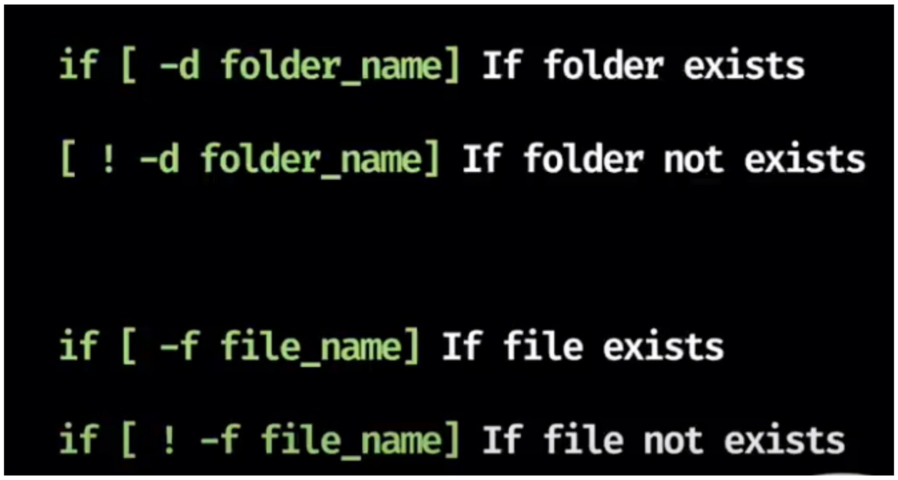


```
#!/bin/bash


read -p "Which site connectivity you want to check ?  " site

ping -c 1 $site

if [[ $? -eq 0 ]]
then
        echo "Your connection is succesful"
else
        echo "YOur connection was unsuccessful"
fi
~
~
```



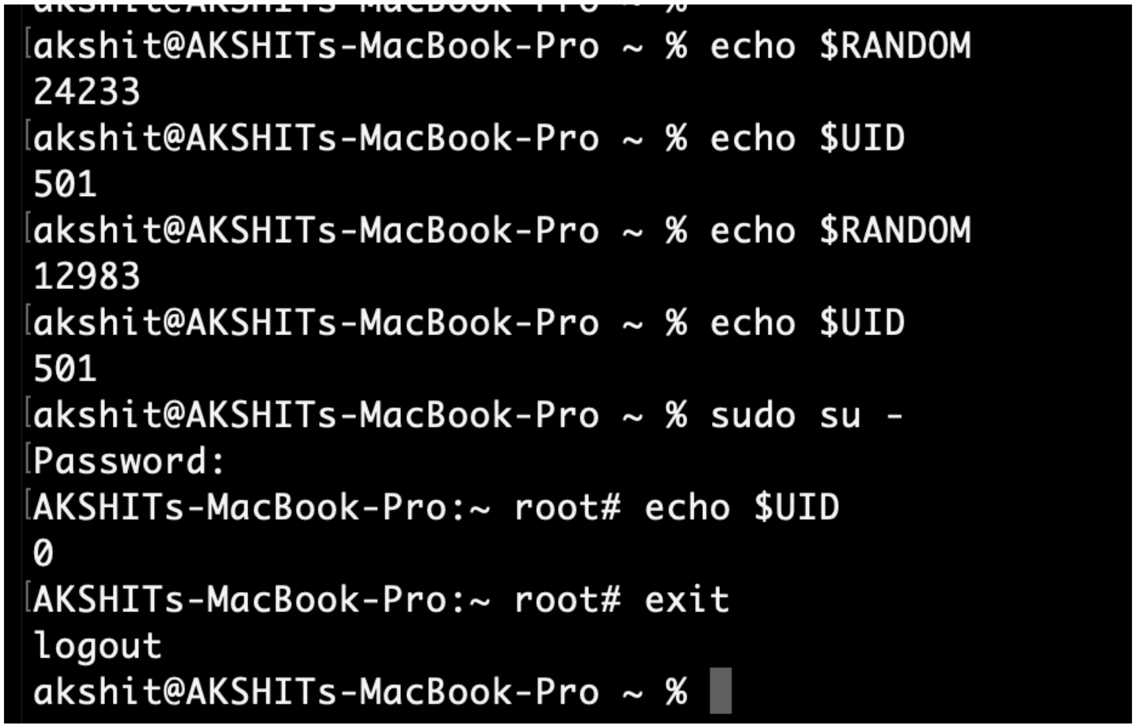
```
akshit@AKSHITs-MacBook-Pro ~ % bash MyScript.sh
Which site connectivity you want to check ?  www.google.com
PING www.google.com (216.58.196.196): 56 data bytes
64 bytes from 216.58.196.196: icmp_seq=0 ttl=118 time=12.412 ms

--- www.google.com ping statistics ---
1 packets transmitted, 1 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 12.412/12.412/12.412/0.000 ms
Your connection is succesful
```

12. Check if file exists or not



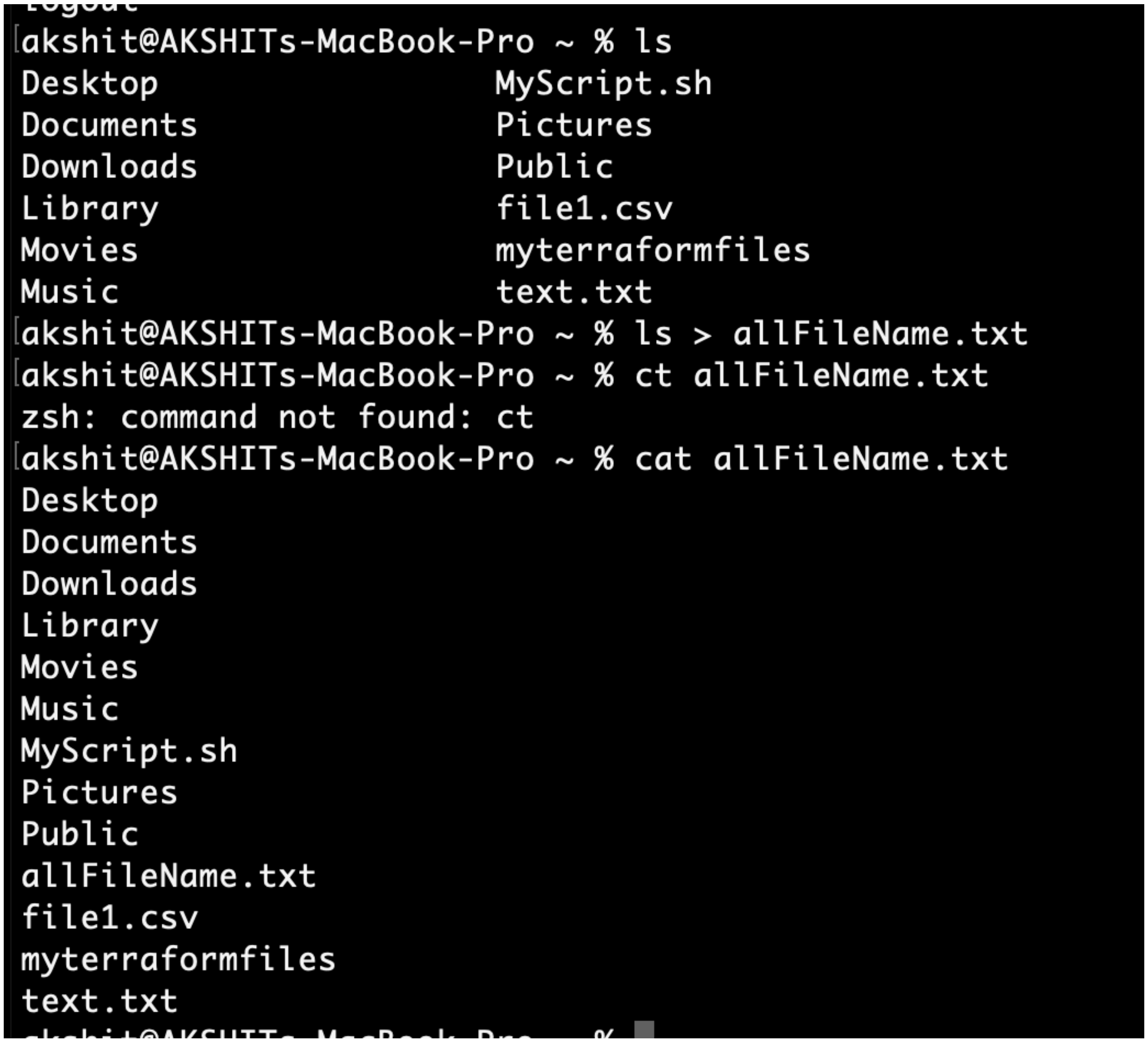
```
if [ -d folder_name] If folder exists

[ ! -d folder_name] If folder not exists


if [ -f file_name] If file exists

if [ ! -f file_name] If file not exists
```

13. Random variable and UID variable
    $RANDOM - defines any random number
    $UID- defines the id of a user which is always unique
    Please note, the root user has always UID as 0



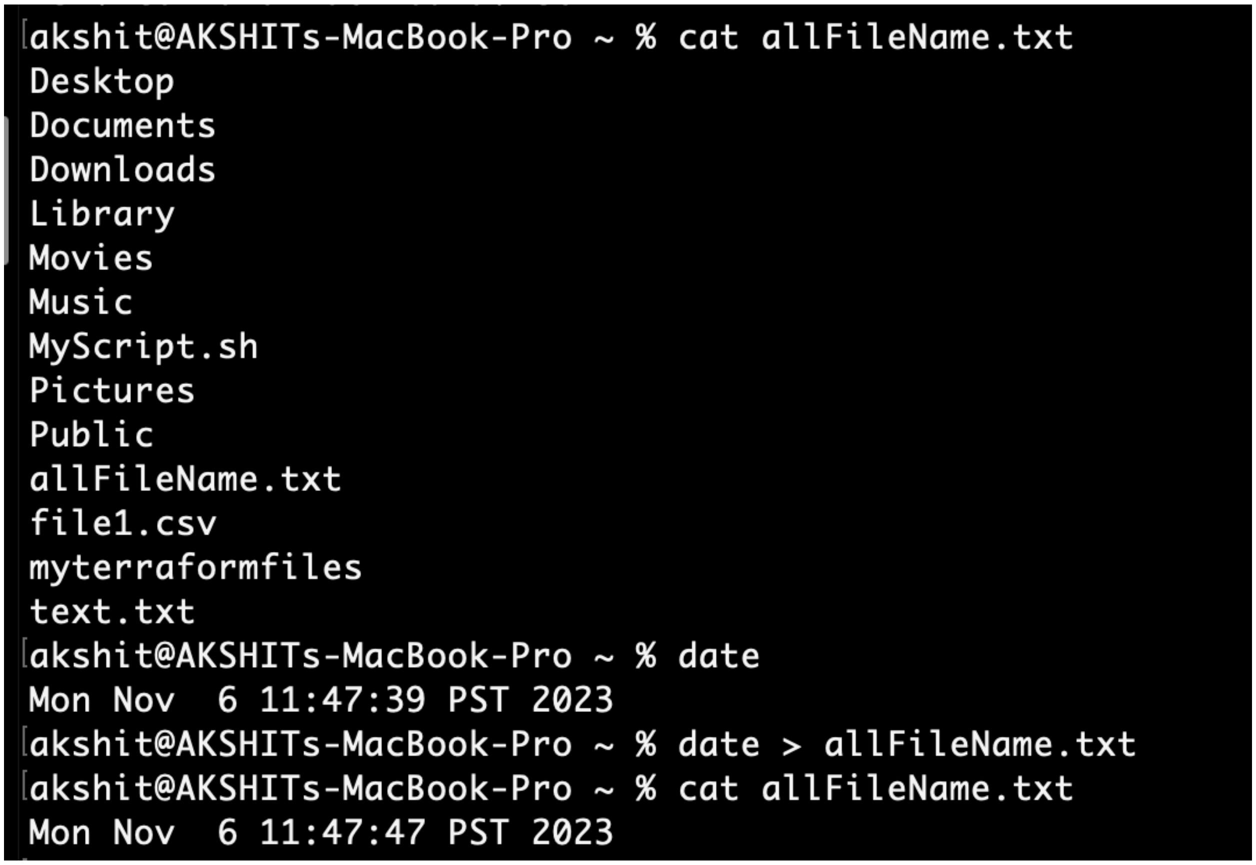
```
akshit@AKSHITs-MacBook-Pro ~ % echo $RANDOM
24233
akshit@AKSHITs-MacBook-Pro ~ % echo $UID
501
akshit@AKSHITs-MacBook-Pro ~ % echo $RANDOM
12983
akshit@AKSHITs-MacBook-Pro ~ % echo $UID
501
akshit@AKSHITs-MacBook-Pro ~ % sudo su -
Password:
AKSHITs-MacBook-Pro:~ root# echo $UID
0
AKSHITs-MacBook-Pro:~ root# exit
logout
akshit@AKSHITs-MacBook-Pro ~ %
```

14. Redirection in Script
    —> pushing some values directly into a file is via redirection function



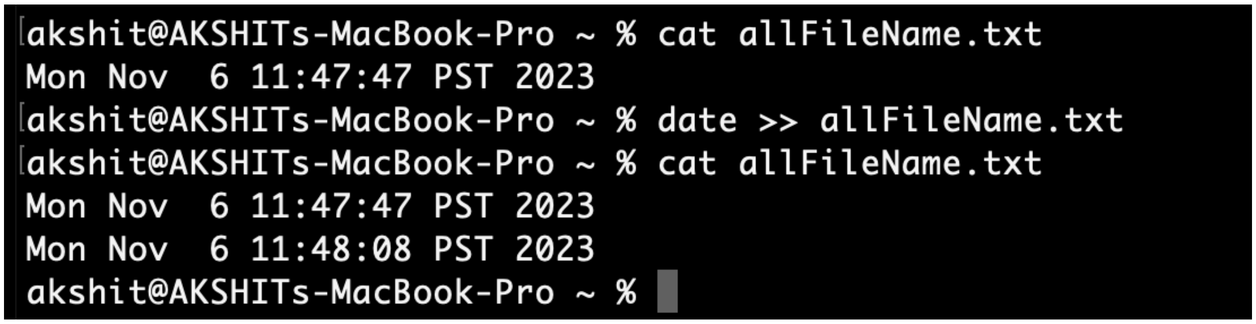
```
akshit@AKSHITs-MacBook-Pro ~ % ls
Desktop                 MyScript.sh
Documents               Pictures
Downloads               Public
Library                 file1.csv
Movies                  myterraformfiles
Music                   text.txt
akshit@AKSHITs-MacBook-Pro ~ % ls > allFileName.txt
akshit@AKSHITs-MacBook-Pro ~ % ct allFileName.txt
zsh: command not found: ct
akshit@AKSHITs-MacBook-Pro ~ % cat allFileName.txt
Desktop
Documents
Downloads
Library
Movies
Music
MyScript.sh
Pictures
Public
allFileName.txt
file1.csv
myterraformfiles
text.txt
```

This command '>' has a feature of over-writing the previous data



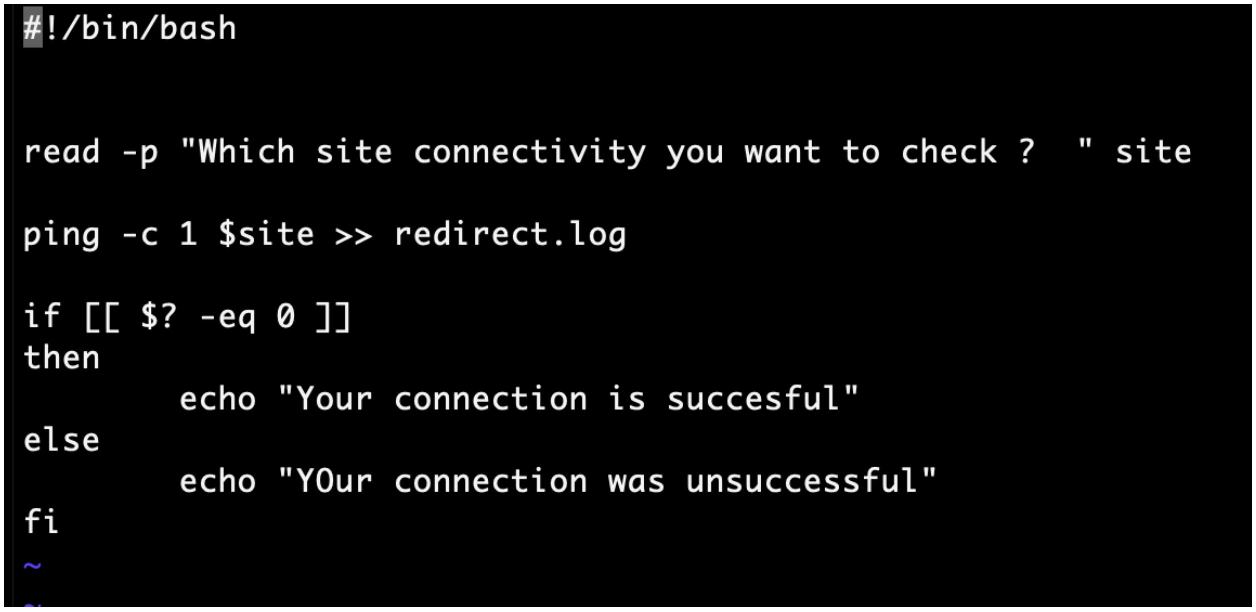
```
akshit@AKSHITs-MacBook-Pro ~ % cat allFileName.txt
Desktop
Documents
Downloads
Library
Movies
Music
MyScript.sh
Pictures
Public
allFileName.txt
file1.csv
myterraformfiles
text.txt
akshit@AKSHITs-MacBook-Pro ~ % date
Mon Nov  6 11:47:39 PST 2023
akshit@AKSHITs-MacBook-Pro ~ % date > allFileName.txt
akshit@AKSHITs-MacBook-Pro ~ % cat allFileName.txt
Mon Nov  6 11:47:47 PST 2023
```

The previous data in file has been over-riden. So here we use '>>' command



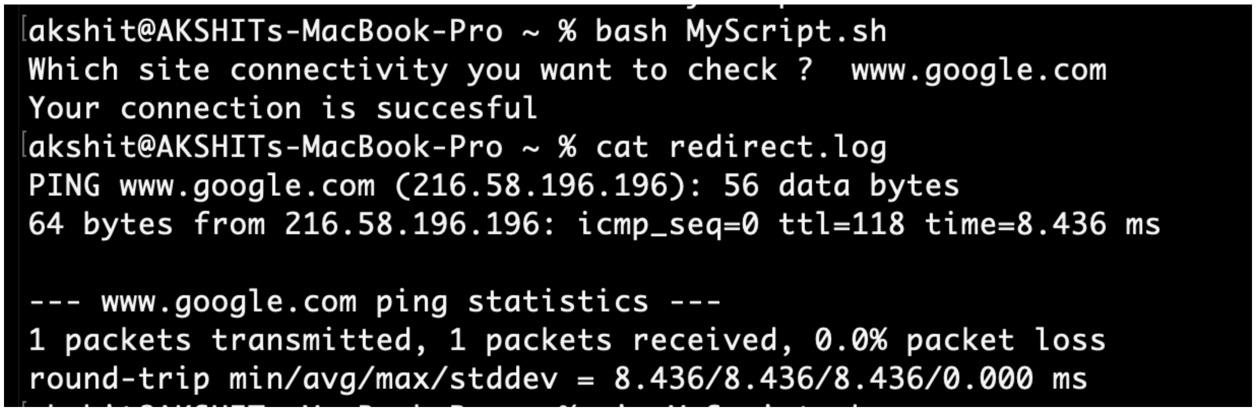
```
akshit@AKSHITs-MacBook-Pro ~ % cat allFileName.txt
Mon Nov  6 11:47:47 PST 2023
akshit@AKSHITs-MacBook-Pro ~ % date >> allFileName.txt
akshit@AKSHITs-MacBook-Pro ~ % cat allFileName.txt
Mon Nov  6 11:47:47 PST 2023
Mon Nov  6 11:48:08 PST 2023
akshit@AKSHITs-MacBook-Pro ~ %
```

Similarly, we can use this command in script to hide or store then un-needed data.



```
#!/bin/bash


read -p "Which site connectivity you want to check ?  " site

ping -c 1 $site >> redirect.log

if [[ $? -eq 0 ]]
then
        echo "Your connection is succesful"
else
        echo "YOur connection was unsuccessful"
fi
~
```

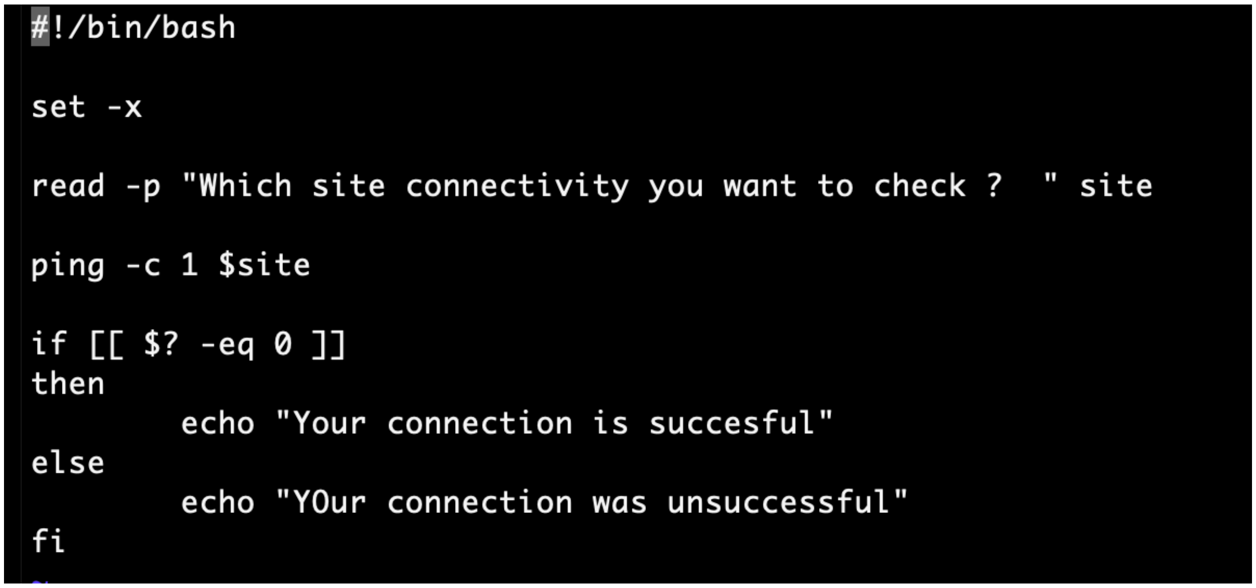
Here we have stored the output of the ping command in a file called as redirect.log. This will allow user to see only necessary data



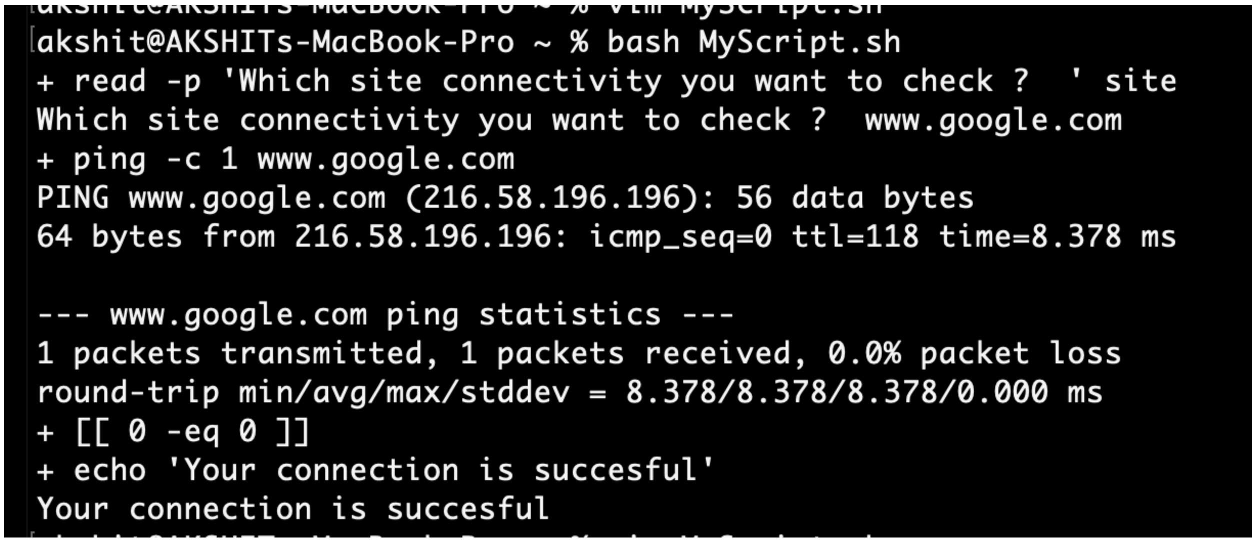
```
akshit@AKSHITs-MacBook-Pro ~ % bash MyScript.sh
Which site connectivity you want to check ?  www.google.com
Your connection is succesful
akshit@AKSHITs-MacBook-Pro ~ % cat redirect.log
PING www.google.com (216.58.196.196): 56 data bytes
64 bytes from 216.58.196.196: icmp_seq=0 ttl=118 time=8.436 ms

--- www.google.com ping statistics ---
1 packets transmitted, 1 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 8.436/8.436/8.436/0.000 ms
```

15. Logger
    This is the variable we use when we have to print something in the logs. It will be saved by default in the directory - /var/log/messages

Debugging the script
via set -x



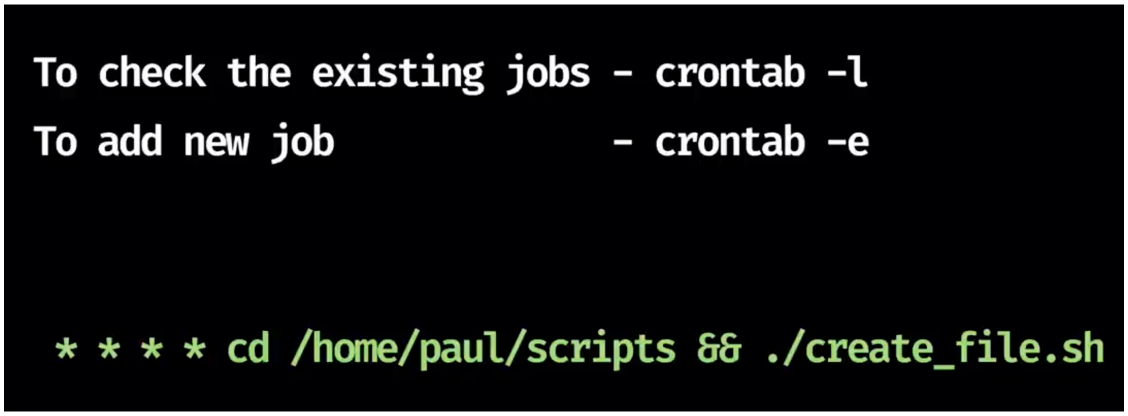
```
#!/bin/bash

set -x

read -p "Which site connectivity you want to check ?  " site

ping -c 1 $site

if [[ $? -eq 0 ]]
then
        echo "Your connection is succesful"
else
        echo "YOur connection was unsuccessful"
fi
```



```
akshit@AKSHITs-MacBook-Pro ~ % bash MyScript.sh
+ read -p 'Which site connectivity you want to check ?  ' site
Which site connectivity you want to check ?  www.google.com
+ ping -c 1 www.google.com
PING www.google.com (216.58.196.196): 56 data bytes
64 bytes from 216.58.196.196: icmp_seq=0 ttl=118 time=8.378 ms

--- www.google.com ping statistics ---
1 packets transmitted, 1 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 8.378/8.378/8.378/0.000 ms
+ [[ 0 -eq 0 ]]
+ echo 'Your connection is succesful'
Your connection is succesful
```

16. Run the script in background
using nohup ./script.sh &
and you may find the output in the /var/log/messages directory

17. Cron Job



```
To check the existing jobs - crontab -l
To add new job             - crontab -e


 * * * * cd /home/paul/scripts && ./create_file.sh
```