Bourne Shell (Bash) Programming

A shell program is a collection of a series of commands for Unix shell.
No separate compiler is required to execute shell script as the shell itself interprets and executes them.
To know the shell in your system type following command:

nagveni@nagveni-H55M-S2V:~$ **echo $SHELL**
/bin/bash

Variables:
we can assgin value to variable as follows:

nagveni@nagveni-H55M-S2V:~$ **x=3**
nagveni@nagveni-H55M-S2V:~$ **y=5**

nagveni@nagveni-H55M-S2V:~$ **expr $x + $y**
8
nagveni@nagveni-H55M-S2V:~$ expr **$x+$y** **//wrong command. Space required between operands and operators**
3+5

nagveni@nagveni-H55M-S2V:~$ z=**`expr $x+ $y`** **//wrong command. Space required between operands and operators**
expr: syntax error
nagveni@nagveni-H55M-S2V:~$ echo $z

nagveni@nagveni-H55M-S2V:~$ z=`expr $x+ $y` **//wrong command. Space required between operands and operators**
expr: syntax error
nagveni@nagveni-H55M-S2V:~$ z=**`expr $x + $y`**
nagveni@nagveni-H55M-S2V:~$ **echo $z**
8
nagveni@nagveni-H55M-S2V:~$ expr 3+5 **//wrong command. Space required between operands and operators**
3+5
nagveni@nagveni-H55M-S2V:~$ expr 3 + 5
8
nagveni@nagveni-H55M-S2V:~$ expr 3 * 5 **//escape character is used for multiplication**
15
nagveni@nagveni-H55M-S2V:~$ let x=15+10 **//let is used for assigning values to the variables and also for evaluating them**
nagveni@nagveni-H55M-S2V:~$ echo $x
25

nagveni@nagveni-H55M-S2V:~$ echo $((15*10)) **//double paranthesis is used instead of let**
150

nagveni@nagveni-H55M-S2V:~$ factor **//factor command is used to factorize the given number and print prime factors**
15
15: 3 5
28
28: 2 2 7

Writing shell scripts:

1. comments should be preceded with #. Comment split over multiple lines must have # at the beginning of each line.
2. More than one assignment can be done in single statement.
3. Multiplication symbol must always be prceded by \.

to type shell script open vi editor with following command:

nagveni@nagveni-H55M-S2V:~$ **vi dispmsg**

type the following code in editor:

```
#!/bin/bash
echo -n "Hello!"       #-n suppresses the new line print
echo "You are wellcome"
echo "we are working in directory `pwd`"
echo  "todays date is `date`"
```

to run the program:

nagveni@nagveni-H55M-S2V:~$ **./dispmsg**
bash: ./dispmsg: Permission denied

Give execution permission to owner:

nagveni@nagveni-H55M-S2V:~$ **chmod 700 dispmsg**

nagveni@nagveni-H55M-S2V:~$ **bash dispmsg**

Hello!You are wellcome
we are working in directory /home/nagveni
todays date is Sat Feb 24 11:50:52 IST 2018

Another way to execute program:

nagveni@nagveni-H55M-S2V:~$ **./dispmsg**
Hello!You are wellcome
we are working in directory /home/nagveni
todays date is Sat Feb 24 11:52:58 IST 2018

Command line parameters:

Shell scripts can read upto **nine command line parameters.** They are named as $1, $2, $3,...$9.

Name of executable script is stored in $0.

$# is the count of the number of arguments.
$* represents all command line arguments.

Type following code in vi editor:

nagveni@nagveni-H55M-S2V:~$ **vi commandparam**

```
#!/bin/bash
echo "The number of parameters are $#"
echo "The parameter are $*"
echo "The parameters are $1 $2 $3"
echo "The shell script command is $0"
```

Run the program:

nagveni@nagveni-H55M-S2V:~$ **bash commandparam a.txt 10 b.tst 25**
The number of parameters are 4
The parameter are a.txt 10 b.tst 25
The parameters are a.txt 10 b.tst
The shell script command is commandparam

Reading input from user:

read command is used to read input typed by user into shell variables.

Open readdemo file in vi editor and type following code

nagveni@nagveni-H55M-S2V:~$ **vi readdemo**

```
#!/bin/bash
echo -n "Enter your first name "
read f
echo -n "Enter your last name "
read l
echo "Your name is $f $l"
```

nagveni@nagveni-H55M-S2V:~$ **bash readdemo**
Enter your first name shachi
Enter your last name natu
Your name is shachi natu

create file samplefile.txt as below using vi editor:

This is sample file.
created forhell programmimg

write a shell script to count number of lines in samplefile.txt

nagveni@nagveni-H55M-S2V:~$ **vi samplefile.txt**
This is sample file.
created forhell programmimg

nagveni@nagveni-H55M-S2V:~$ **vi count**

```
#!/bin/bash
echo "the number of lines in file samplefile.txt are: "
```

echo `wc -l samplefile.txt`
nagveni@nagveni-H55M-S2V:~$ **bash count**
the number of lines in file samplefile.txt are:
2 samplefile.txt

Write a shell script to print current system date.

Instead of using vi editor, u can also write a script file in simple text editor and save it with extension **.sh**

following script file called **date.sh** is created:

```
#!/bin/bash
m=`date +%d/%m/%y`
echo "Current system date is $m"
```

nagveni@nagveni-H55M-S2V:~$ **bash date.sh**
Current system date is 24/02/18

For loop in shell script:

Syntax: **for** *variable* **in** *list-of-variables*

```
do
        command1
        command2
        ..
done
```

write a script in file for.sh to print values in range 1 to 5 using for loop.

```
#!/bin/bash

for x in 1 2 3 4 5
do
        echo "The value of x is $x"
done
```

nagveni@nagveni-H55M-S2V:~$ bash for.sh
The value of x is 1
The value of x is 2
The value of x is 3
The value of x is 4
The value of x is 5

Write a program to print all files/directories in a current working directory.

```
#!/bin/bash
d=`pwd`
echo "current working directory is $d"
l=`ls $d`
for x in $l
do
        echo "The file name is $x"
```

done

save it as filesindir.sh.

nagveni@nagveni-H55M-S2V:~$ **bash filesindir.sh**
current working directory is /home/nagveni
The file name is audch11.m
The file name is audch11.m~~
The file name is audiostegano.m
The file name is auhide.m
The file name is aurecover.m
The file name is b01ae.wav
The file name is b01ah.wav
The file name is b01oa.wav
The file name is b02ae.wav
The file name is b02ei.wav
The file name is b15oa.wav
The file name is b15oo.wav
The file name is BE%20WT%20Def-2017-18.xls_1ods
The file name is bhakti.odt
The file name is Brass_AH31.wav
The file name is commandparam
The file name is count
The file name is date.sh
The file name is date.sh~
The file name is dectobin.m
The file name is deja-dup
The file name is Desktop
The file name is dispmsg
The file name is Documents
The file name is Downloads
The file name is examples.desktop
The file name is fhss.m
The file name is filesindir.sh
The file name is filesindir.sh~
The file name is Flute_A_51.wav
The file name is for.sh
The file name is Guitar_A_52.wav
The file name is IDS.docx
The file name is ls
The file name is lsb-image.odt
The file name is Music
The file name is negation.m
The file name is new2.wav
The file name is New
The file name is Folder
The file name is ns-allinone-2.35
The file name is ns-allinone-2.35(1)
The file name is ns-allinone-2.35.tar.gz
The file name is octave
The file name is octave-workspace
The file name is ofdm.m

The file name is ofdm.odt
The file name is Pictures
The file name is Public
The file name is readdemo
The file name is samplefile.txt
The file name is scenario1.nam
The file name is scenario1.tr
The file name is sensor.tcl
The file name is sensor.tcl~
The file name is Sitar_AH31.wav
The file name is Templates
The file name is testSignal.wav
The file name is Videos
The file name is watermark.wav

the same code can be written without using for loop as follows:
save it in **filesindir1.sh**

```
#!/bin/bash
l=`pwd|ls`
echo "the list of files and directories in current working directory are  $l"
```

nagveni@nagveni-H55M-S2V:~$ **bash filesindir1.sh**
the list of files and directories in current working directory are audch11.m
audch11.m~~
audiostegano.m
auhide.m
aurecover.m
b01ae.wav
b01ah.wav
b01oa.wav
b02ae.wav
b02ei.wav
b15oa.wav
b15oo.wav
BE%20WT%20Def-2017-18.xls_1ods
bhakti.odt
Brass_AH31.wav
commandparam
count
date.sh
date.sh~
dectobin.m
deja-dup
Desktop
dispmsg
Documents
Downloads
examples.desktop
fhss.m
filesindir1.sh
filesindir.sh

filesindir.sh~
Flute_A_51.wav
for.sh
Guitar_A_52.wav
IDS.docx
ls
lsb-image.odt
Music
negation.m
new2.wav
New Folder
ns-allinone-2.35
ns-allinone-2.35(1)
ns-allinone-2.35.tar.gz
octave
octave-workspace
ofdm.m
ofdm.odt
Pictures
Public
readdemo
samplefile.txt
scenario1.nam
scenario1.tr
sensor.tcl
sensor.tcl~
Sitar_AH31.wav
Templates
testSignal.wav
Videos
watermark.wav

Same code can also be written as:

```
#!/bin/bash
for l in `ls $pwd`
do
echo "$l"
done
```

or

```
#!/bin/bash
for l in `ls`
do
echo "$l"
done
```

Write a shell script to display all files and directories starting with letter 'b'

```
#!/bin/bash
for l in b*
do
echo "the list of files and directories in current working directory are  $l"
```

```
done
```

Write a shell script to display names of .sh files starting with f

```
#!/bin/bash
for l in `ls f*.sh`
do
echo "$l"
done
```

```
nagveni@nagveni-H55M-S2V:~$ bash filesindir1.sh
filesindir1.sh
filesindir.sh
for.sh
```

Write a shell script to display contents of .sh files starting with f

```
#!/bin/bash
for l in ls f*.sh
do
cat $l
done
```

while loop in Shell programming:

syntax:

```
while [logical expression]
do
..
..
done
```

IF statement in shell programming

syntax:

```
if [logical expression]
then
        ..
        ..
else
        ..
fi
```

Write a shell script to display numbers from 1 to 10

```
#!/bin/bash
n=1
while [ $n -le 10 ]
do
        echo $n
        (( n++ ))
```

```
done
```

Write a shell script to develope scientific calculator

```
#!/bin/bash
sum=0
i="y"

echo " Enter one no."
read n1
echo "Enter second no."
read n2
while [ $i = "y" ]
do
echo "1.Addition"
echo "2.Subtraction"
echo "3.Multiplication"
echo "4.Division"
echo "Enter your choice"
read ch
case $ch in
    1)sum=`expr $n1 + $n2`
     echo "Sum ="$sum;;
        2)sum=`expr $n1 - $n2`
     echo "Sub = "$sum;;
    3)sum=`expr $n1 \* $n2`
     echo "Mul = "$sum;;
    4)sum=`expr $n1 / $n2`
     echo "Div = "$sum;;
    *)echo "Invalid choice";;
esac
echo "Do u want to continue ?"
read i
if [ $i != "y" ]
then
    exit
fi
done
```

Output:

```
nagveni@nagveni-H55M-S2V:~/Desktop$ bash cal.sh
 Enter one no.
5
Enter second no.
6
1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter your choice
1
```

```
Sum =11
Do u want to continue ?
2
nagveni@nagveni-H55M-S2V:~/Desktop$ bash cal.sh
 Enter one no.
5
Enter second no.
6
1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter your choice
1
Sum =11
Do u want to continue ?
y
1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter your choice
2
Sub = -1
Do u want to continue ?
y
1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter your choice
3
Mul = 30
Do u want to continue ?
y
1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter your choice
4
Div = 0
Do u want to continue ?
Y
```

Until loop in shell programming:

The untill loop is used for repeating the set of instructions **for the time the specified logical expression is false.** The moment the logical expression becomes true, the control will come outof loop.

Syntax:

until logical_expression
do
..
...
done

example:

Write a shell script to print sum of even numbers upto 50

```
#!/bin/bash
s=0
n=2
until [ $n -gt 50 ]
do
        s=$(( $s + $n ))
        ((n+=2))
        echo $n
done
echo $s
```

Output:

```
nagveni@nagveni-H55M-S2V:~/Desktop$ bash evensum.sh
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
52
650
```

test command in shell:

The test command returns true if the expression included is valid otherwise returns false.

Test command can be used to test various file attributes like whether file has read, write or executable permission or whether it is a file or directory etc.

Option	Description
-a filename	Returns true if file has at least one character
-e filename	Returns true if file exists
-f filename	Returns true if file exists and is a regular file
-r filename	Returns true if file has read permission
-w filename	Returns true if file has write permission
-x filename	Returns true if file is executable
-d filename	Returns true if file exists and is a directory
-s filename	Returns true if file exists and has size greater than zero

Write a program to check whether give file name is file or directory

```
#!/bin/bash
echo -n "Enter filename: "
read fname
if test -d $fname
then
   echo "$fname is a directory"
else
   if test -f $fname
then
      echo "$fname is a file"
   else
      echo "$fname is not valid"
      exit 1
   fi
fi
```

```
#!/bin/bash

PASSED=$1

if [ -d "$1" ] ; then
   echo "$1 is a directory";
else
   if [ -f "${PASSED}" ]; then
      echo "${PASSED} is a file";
   else
      echo "${PASSED} is not valid";
      exit 1
   fi
fi
```

Output:
nagveni@nagveni-H55M-S2V:~/Desktop$ bash filedir.sh while.sh
while.sh is a file
nagveni@nagveni-H55M-S2V:~/Desktop$ bash filedir.sh while.sh
while.sh is a file
nagveni@nagveni-H55M-S2V:~/Desktop$ bash filedir.sh WT
WT is a directory
nagveni@nagveni-H55M-S2V:~/Desktop$ bash filedir.sh WT
WT is a directory

Arrays in shell script

the syntax of array initialization −

array_name = (value1 ... valuen)

Other way to initialize array is

array=([index]=<value> [index]=<value> . . .)

We can also read/assign values to array during the execution time using the *read* shell-builtin.

read -a array

Now upon executing the above statement inside a script, it waits for some input. **We need to provide the array elements separated by space (and not carriage return).** After entering the values press enter to terminate.

accessing array element

you access it as follows −

${array_name[index]}

Sample script to access array elements

#!/bin/sh

NAME=(Zara Qadir Mahnaz Ayan Daisy)

echo "First Index: ${NAME[0]}"

echo "Second Index: ${NAME[1]}"

OR

```
#!/bin/sh

NAME[0]="Zara"

NAME[1]="Qadir"

NAME[2]="Mahnaz"

NAME[3]="Ayan"

NAME[4]="Daisy"

echo "First Index: ${NAME[0]}"

echo "Second Index: ${NAME[1]}"
```

nagveni@nagveni-H55M-S2V:~/Desktop$ bash array.sh

First Index: Zara

Second Index: Qadir

Write a shell script to check whether element is present in array

```
#!/bin/sh
echo -n "enter elements of array"
read -a array

echo -n "enter the element to be searched"
read num

for  val in "${array[@]}"
do
        if [ $num == $val ]
        then
                echo "element is present in the array"
                exit
        else
                continue
                #echo "element is not present in the array"
        fi
done
```

nagveni@nagveni-H55M-S2V:~/Desktop$ bash list.sh
enter elements of array1 2 3 4 5
enter the element to be searched3
element is present in the array
nagveni@nagveni-H55M-S2V:~/Desktop$ bash list.sh
enter elements of array1 2 3 4 5
enter the element to be searched6
nagveni@nagveni-H55M-S2V:~/Desktop$

Write a shell script to check whether two strings entered by user are equal or not

```
#!/bin/bash
echo -n "Enter first string: "
read str1
echo -n "enter second string: "
read str2

if [ $str1 = $str2 ]
then
        echo " Two stringa are equal"
else
        echo " Two strings are not equal "
fi
```

Output:

```
nagveni@nagveni-H55M-S2V:~/Desktop$ bash string.sh
Enter first string: s1
enter second string: s2
 Two strings are not equal
```

Functions in shell programming:

Syntax to write a function:

```
function_name ()
{
        statement
        statement
        ..
}
```

Write a function to print sum of sequence of numbers. Limit of sequence will be entered by user.

```
#!/bin/bash

sum()
{
        s=0
        x=1
        while test $x -le $1
        do
                ((s=$s+$x))
                ((x=$x+1))
        done
        return $s
}

sum $1
echo "The sum of sequence is : $?"
```

Output:

```
nagveni@nagveni-H55M-S2V:~/Desktop$ bash seqsum.sh 5
The sum of sequence is : 15
```

same program can be written by taking input from user as:

```
#!/bin/bash
sum()
{
        s=0
        x=1
        while test $x -le $l
        do
                ((s=$s+$x))
                ((x=$x+1))
        done
        return $s
}
echo " Enter limit of sequence"
read l
sum $l
echo "The sum of sequence is : $?"
```

Output:

```
nagveni@nagveni-H55M-S2V:~/Desktop$ bash seqsum.sh
 Enter limit of sequence
5
The sum of sequence is : 15
```

Another way of writing the same program where returned value is stored in variable x.

```
#!/bin/bash

sum()
{
        s=0
        x=1
        while test $x -le $l
        do
                ((s=$s+$x))
                ((x=$x+1))
        done
        return $s
}
echo " Enter limit of sequence"
read l
#sum $1
sum $l
x=$?
echo "The sum of sequence is : $x"
```