Linux Shell Scripts

DevOps
rajeshdeveloper99@gmail.com

Shell Scripting

Program	Shell Script
*) Set of Instructions	*) Set of commands

*) Variable Declaration
int a = 10
char b =‘x’
	*) No Variable Declaration
	a = 10
	b=‘x’

*) Data Types	*) No Data types
*) Compile & Execute	*) Direct Execute
*) Ctrl statements	*) Ctrl Statements
*) Operators	*) Operators
*) Input and Output stmts (printf, scanf…)	*) input and output stmts (echo, print, read)

Schell Scripting

Shell Script : set of commands which can perform a specific task
extension --> “.sh”

Input and Output

Input statements : it will read the data from std Input

Ex :

```
read n
read a b c
```

Output Statements : it will display the data on std output

Ex :

```
echo “Hello World”
printf “\n Good Day”
a = 10
echo $a(echo, print, read)
```

Shell Scripting

Escape Sequences :

\n → New line
\t → Horizontal tab
\b → Backspace
\r → Carriage return

Expr : it is used to perform Arithmetic Operations

Ex:

```
a=10
b=20
c=`expr $a+ $b`
c=`expr $a - $b`
c=`expr $a \* $b`
c=`expr $a / $b`
c=`expr $a % $b`
```

Script : Script to read and display values

```
#vi demo.sh
        printf "\n Enter three values"
        read a b c
        echo value - 1 : $a
        echo value - 2 : $b
        echo value – 3 : $c
```

execution:

```
        #sh demo.sh
        Enter three values 10 20 A
        Value – 1 : 10
        value – 2 : 20
        value – 3 : A
```

Operators

1. Arithmetic Operators : +, -, *, / , %
2. Logical Operators : -a, -0, !
3. Relational Operators : -lt, -gt, -le, -ge, -eq, -ne
4. String Comparison Operators : str1 == str2, str1 != str2

Script : Arithmetic Operations

```
$vi arithmetic.sh
        printf “\n enter two values ”
        read a b
           c=`expr $a + $b`
           echo Add : $c
           c=`expr $a - $b`
           echo Sub : $c
           c=`expr $a \* $b`
           echo Mul : $c
           c=`expr $a / $b`
           echo Div : $c
           c=`expr $a % $b`
           echo Mod : $c
```

Execution:

```
        enter two values 10 20
        Add : 30
        Sub : -10
        Mul : 200
        Div  : 0
        Mod : 10
```

File Copy:

Create two files

```
#vi file1.txt
     File one
#vi file2.txt
    File two
```

```
# vi filecopy.sh
  printf “\n enter two different file names (with extension)”
    read f1 f2
    cp $f1 $f2                    ---- copying file 1 data to file 2
    echo file1 data copied to file2
```

Execution:

```
Enter two different file names (with extension) file1.txt file2.txt
 File1 data copied to file2
```

To verify: cat file1.txt
cat file2.txt

Control Statements

- **Simple if**
- **If – else**
- **Nested if – else**
- **If – elsif**
- **Case**
- **While loop**
- **Until loop**
- **For loop**
- **Break**
- **Continue**
- **Sleep**
- **Exit**

Simple if :

Syntax :

```
if [ condition ]
then
 statements
fi
```

ex :

```
a = 10
b = 20
if [ $a –gt $b ]
then
    echo $a is big
fi
```

If – else

syntax:

```
 if [ condition ]
 then
       statements
 else
       statements
 fi
```

ex :

```
 $vi ifelse.sh
  a = 10
  b = 20
  if [ $a –gt $b ]
  then
      echo $a is big
   else
      echo $b is big
   fi
```

Even or Odd Number

```
$vi evenodd.sh
    printf “\n enter a number\n”
    read a
    if [ `expr $a % 2` -eq 0 ]
    then
              echo $a is even number
     else
              echo $a is odd number
      fi
```

Execution :

```
    $sh evenodd.sh
              enter a number
               11
                11 is odd number
```

Script to verify give IP is valid or not

```
#vi checkipaddress.sh
   printf “\n enter ipaddress / host name : ”
   read ip
   ping $ip -c1 > /dev/nul
   If [ $? -eq 0 ]
   then
      echo $ip is Valid
   else
      echo $ip is in-valid
  fi
```

Execution:

```
enter ipaddress / host name : 10.10.2.3
10.10.2.3 is Valid
```

Script to verify give user is valid or not

```
#vi checkuser.sh
   printf “\n enter user name : ”
   read username
   grep $username /etc/passwd > /dev/nul
   If [ $? -eq 0 ]
   then
      echo $username is Valid
   else
      echo $username is in-Valid
  fi
```

Execution:

```
enter user name : ubuntu
ubuntu is Valid
```

Nested if – else :

syntax :

```
if [ condition – 1 ]
then
    if [ condition – 2 ]
    then
        statements – 1
     else
        statements – 2
     fi
else
     statements – 3
fi
```

ex :

$vi nestedifelse.sh – to find “a is bigger than b & c”

```
printf “\n enter three numbers : \n”
read a b c
 if [ $a –gt $b ]
 then
     if [ $a –gt $b ]
     then
          echo $a is bigger
     else
          echo $a is not big than $b
     fi
 else
     echo $a is not big than $b
 fi
```

if – elif

```
 if [ condition – 1 ]
 then
     statements – 1
 elif [ condition – 2 ]
 then
     statements – 3
  else
    default statements
  fi
```

ex:

$vi ifelif.sh --- to find given number <10 / > 10 / =10

```
  printf "\n enter a number\n"
  read a
  if [ $a –gt 10 ]
  then
    echo a is greater than 10
  elif [ $a –lt 10 ]
   then
    echo a is less than 10
  else
     echo a is equal to 10
 fi
```

Script : to find biggest value

```
$vi biggestvalue.sh
        printf "\n enter three values\n"
         read a b c
         if [ $a –gt $b –a $a –gt $c ]
          then
               echo Biggest value : $a
           elif [ $b –gt $c ]
            then
                     echo Biggest value : $b
             else
                    echo Biggest value : $c
            fi
```

Execution:

```
enter  three values
10 20 21
Biggest value : 21
```

File Operations

-e <File Name> ------> a FIle is Exist
-f <File Name>   -------> a File is a Regular File (txt, img, doc…)
-d <File Name> ------> a FIle is a Directory
-r <File Name>   -------> A File Contains Read Permi.
-w <File Name> --------> a file contains Write Permi
-x <File Name> --------> a File Contains Execute permi.
-l <File Name>   --------> a FIle is a Link File
-s <File Name> --------> a File Contains more than one byte
-O <FIle Name> -------> a File is Owned by User
-G <File Name> -------> a File is Owned by Group
<FIle1> -ef <File2> -------> a file-1 is link with File -2
<File1> -nt <File2> -------> a File-1 is Newer than File-2
<File1> -ot <File2> -------> a File-1 is Older than File-2

Script : to verify the file is exist or not

```
#vi fileexit.sh
  printf "\n enter a file : "
  read file
  If [ -e $file ]
  then
     echo File Exist
   else
     echo File does not exit
  fi
```

Execution:

```
   Enter a file : file1.txt
    FIle Exist
```

Script : To verify the file is a regular file or directory

```
#vi filedir.sh
 printf "\n Enter a file : "
 read file
 if [ -e $file ]
 then
     if [ -f $file ]
     then
         echo Is a Regular file
     elif [ -d $file ]
     then
         echo is a Directory
  fi
  else
       echo File does not exit
 fi
```

Execution:

```
 Enter a file : file1.txt
   Is a Regular File
```

Case Statement

syntax:

```
case $<var> in
1) statements – 1
2) statements – 2
    …….
 n) statements – n
 *) default statements
 esac
```

ex:

```
echo enter a value 1 – 3
read n
case $n in
 1) echo one ;;
 2) echo two ;;
 3) echo three ;;
 *) echo values from 1 – 3 only !! ;;
 esac
```

Script : write a script for case statement

```
$vi caseoption.sh
  printf “\n\n1. server name\n2.ip address\n”
  printf “3. date\n4. user name \n 5.cal”
   read op
   case $op in
   1) hostname –f ;;
   2) hostname –I ;;
   3) date ;;
   4) username
   5) printf “enter month and year : ”
                  read m y
                  cal $m $y ;;
 *) echo invalid input
  esac
```

While loop

Syntax:

```
  while [ condition ]
  do
    statements
  done
```

Ex :

```
  n = 1
  while [ $n –le 10 ]
  do
     echo $n
     n=`expr $n + 1`
  done
```

Script : write a script to find a reverse number

```
$vi reversenumber.sh
printf “\nenter a number : ”
read n
m=0
while [$n –gt 0 ]
do
    r=`expr $n % 10`
    m=`expr $m \* 10 + $r`
    n=`expr $n / 10`
done
Echo Reverse number : $m
```

Execution:

```
Enter a number : 123
Reverse number : 321
```

Until Loop :

Syntax :

```
 until [ condition ]
 do
   statements
 done
```

break – this ctrl statement will terminate a loop

Ex :

```
until false
do
  echo “Hello World”
  break
Done
```

Output:

```
Hello World
```

Continue : this ctrl statement will skip the statements from its execution in that loop.

Ex :

```
i=1
while [ $i –lt 5 ]
do
  i=`expr $i + 1`
  echo hello
  if [ $i –eq 2 ]
  then
     echo if started
     continue
     echo if stopped
   fi
   echo world
done
```

Script : write a script to display stop watch with nested while

```
 $vi stopwatch.sh
   h=0
   while [ $h –lt 24 ]
   do
       m=0
       while [ $m –lt 60 ]
       do
           s=0
           while [ $s –lt 60 ]
           do
              clear
              printf “\n\nSTOP WATCH\n”
              printf “\t $h : $m : $s”
              sleep 1
              s=`expr $s + 1`
          done
        m=`expr $m + 1`
        done
      h=`expr $h + 1`
    done
```

Execution :

```
STOP Watch
  0 : 0 : 1
```

For Loop Syntax :

```
for <var> in <Values>
do
  Statements
done
```

Ex:

```
#vi forloop.sh
echo Enter three values
read a b c
for n in $a $b $c
do
   echo $n
done
```

Execution:

```
Enter three values 10 20 30
10
20
30
```

For Loop : Ex -1

```
for ip in $(cat myips)
do
  ping -c1 $ip > /dev/null
  if [ $? -eq 0 ]
  then
    echo $ip is valid ip /host name
  else
    echo $ip is invalid ip / host name
  fi
done
```

For loop : ex-2

```
for usr in $(cat allusers)
do
  grep $usr /etc/passwd > /dev/null
  if [ $? -eq 0 ]
  then
    echo $usr is valid user
   else
    echo $usr is invalid user
  fi
done
```

Script : write a script to verify set of IP’s valid or not

```
#vi myips                          #cat myips
  10.10.10.2                           10.10.10.2
  10.20.30.40                         10.20.30.40
   2.3.65.45                          2.3.65.45
```

```
#vi checkips.sh
  for ip in $(cat myips)
  do
    echo -------------
    ping $ip -c1 > /dev/null
    if [ $? -eq 0 ]
    then
        echo valid ip : $ip
    else
        echo Invalid ip : $ip
    fi
 done
```

Positional Parameters

"Command line arguments"

$0 → File Name
$# → No. of Arguments
$1, $2, $3, … $9 → other Arguments
$* → All Arguments

Ex:

```
#vi positional.sh
  echo File name : $0
  echo No. of Arguments : $#
  echo All Arguments : $*

   for a in $*
   do
      echo Arg : $a
   done
```

Execution:

```
#sh positional.sh 10 20 30
```

Write a script file copy

```
#vi filecopy.sh
 if [ $# -wq 2 ]
 then
    if [ -e $1 ]
    then
       cp $1 $2
       echo File copied
     else
        echo error : source file does not exist
     fi
   else
      Echo error : Invalid No. of Arguments
 fi
```

Execution:

```
#sh filecopy.sh file1.txt file2.txt
File Copied
```