Shell Scripting

Prabhjeet Singh

Shell Scripting

Bash shell script is a series of bash commands that re stored in a file and can be executed by running the script.

This is first script

$ cat script1.sh

#!/bin/bash

this is a comment

echo "Hello WOrld"

echo " these are files available int he following folder:"

ls -l

How to run a script file

$ bash script1.sh → $ < shell> < script name.sh>

Hello WOrld

these are files available int he following folder:

total 4

-rw-r--r-- 1 kali kali 117 Sep 2 11:12 script1.sh

We can run this script with bash keyword. But we need to make that script as executable

$ chmod 777 script1.sh

Or

$ chmod +x script1.sh

┌──(kali㉿kali)-[~/linux_learning/scripts]

└─**$ ls -l**

total 4

-rwxrwxrwx 1 kali kali 117 Sep 2 11:12 script1.sh

$ ./script1.sh → if a script is executable then we can directly run script like this.

$ ./script1.sh

Hello WOrld

these are files available int he following folder:

total 4

-rwxrwxrwx 1 kali kali 117 Sep 2 11:12 script1.sh

We can $<variable name> to store the values in a variable in scripts

Script 2.sh

$ cat script2.sh

#!/bin/bash

#
****

Executing commands example

Execute whoami command

user=$(whoami)

Execute hostname command

hostname=$(hostname)

Execute print working directory (pwd) command

directory=$(pwd)

Display information

echo "User=[$**user**] Host=[$**hostname**] Working dir=[$**directory**]"

```
# Display contents of directory
echo "Contents:"
ls
```

```
┌──(kali㉿kali)-[~/linux_learning/scripts]
└─$ chmod +x script2.sh
```

```
┌──(kali㉿kali)-[~/linux_learning/scripts]
└─$ ./script2.sh
User=[kali] Host=[kali] Working dir=[/home/kali/linux_learning/scripts]
Contents:
script1.sh  script2.sh
```

Control statements

$ cat script4.sh

```
#!/bin/bash
# demo of if else
if [[ -d /etc/ ]]; then
    echo /etc/ is a directory
fi

if [[ -e example.txt ]]; then
    echo example.txt file is present
else
    echo example.txt file is not present
fi
test="kali"
if [[ $test == "sam" ]]; then
    echo test variable is sam
```

```
elif [[ $test == 'kali' ]]; then
      echo test variable is kali
else
      echo test is something different
fi
```

```
┌──(kali㉿kali)-[~/linux_learning/scripts]
└─$ ./script4.sh
/etc/ is a directory
example.txt file is not present
test variable is kali
```

$ cat control_state1.sh

```
#!/bin/bash

#Numerical comparision
x=5
y=9
z=9
echo x=[$x]
echo y=[$y]
echo z=[$z]

if [[ "$x" -ne "$y" ]]; then
       echo x is not equal to y
fi
```

```
if [[ "$y" -eq "$z" ]]; then
        echo y is equal to z
fi

if [[ "$y" -gt "$z" ]]; then
        echo $y is greater than $x
fi
if [[ $y -ge $z ]]; then
        echo $y ge $z
fi
if [[ $x -lt $y ]]; then
        echo $x lt $y
fi
if [[ $y -le $z ]]; then
        echo $y le $z
fi
```

–**$ ./control_state1.sh**

```
x=[5]
y=[9]
z=[9]
x is not equal to y
y is equal to z
9 ge 9
5 lt 9
9 le 9
```

$ cat control_state1.sh

```
#!/bin/bas
h

                #
                ********************************************************************************
                *
                # Numerical comparison examples

                # Create some variables
                x=1
                echo x=["$x"]
                y=2
                echo y=["$y"]
                z=2
                echo z=["$z"]


                # Perform some comparisons
                # Numeric: Not equals
                if [[ "$x" -ne "$y" ]]; then
                       echo ["$x"] ne ["$y"]
                fi

                # Numeric: Equals
                if [[ "$y" -eq "$z" ]]; then
                       echo ["$y"] eq ["$z"]
                fi

                # Numeric: Greater than
                if [[ "$y" -gt "$x" ]]; then
                       echo ["$y"] gt ["$x"]
                fi

                # Numeric: Greater than or equal to
                if [[ "$y" -ge "$z" ]]; then
                       echo ["$y"] ge ["$z"]
                fi

                # Numeric: Less than
                if [[ "$x" -lt "$y" ]]; then
                       echo ["$x"] lt ["$y"]
```

```
fi

# Numeric: Less than or equal to
if [[ "$y" -le "$z" ]]; then
        echo ["$y"] le ["$z"]
fi



#
*********************************************************************************
*
# String comparison examples

# Create some variables
a="A"
echo a=["$a"]
b="B"
echo b=["$b"]
anotherA="A"
echo anotherA=["$anotherA"]

# Perform some comparisons
# String: Equals
if [[ "$a" == "$anotherA" ]]; then
        echo ["$a"] "==" ["$anotherA"]
fi

# String: Not equals
if [[ "$a" != "$b" ]]; then
        echo ["$a"] "!=" ["$b"]
fi

# String: Less than
if [[ "$a" < "$b" ]]; then
        echo ["$a"] "<" ["$b"]
fi

# String: Greater than
if [[ "$b" > "$a" ]]; then
        echo ["$b"] ">" ["$a"]
fi
```

$ cat case_example.sh

```
#!/bin/bas
h

#
*************************************************************************
*
# Case example

# Switch off of the first command line argument
case $1 in
[1-3])
        message="Argument is between 1 and 3 inclusive"
        ;;
[4-6])
        message="Argument is between 4 and 6 inclusive"
        ;;
[7-9])
        message="Argument is between 7 and 9 inclusive"
        ;;
1[0-9])
        message="Argument is between 10 and 19 inclusive"
        ;;
*)
        message="I don't understand the argument or it is missing"
        ;;
esac

# Print out a message describing the result
echo $message
```

$ cat example_loop.sh

```
#!/bin/bas
h

#
#******************************************************************************
*
# For loop examples

echo ----------------------------------
echo For loops

# Iterate through the numbers 1 through 5 and print them out
echo Print out a hard-coded sequence
for i in 1 2 3 4 5; do
    echo Index=[$i]
done

# Same as above, but generate the sequence
echo Print out a generated sequence
for i in {1..5}; do
    echo Index=[$i]
done

# Same as above, but use a more conventional format
#   NOTE: Double parenthesis are used since we are doing arithmetic
echo Print out a generated sequence using the 3-expression format
for(( i=1; i<=5; i++ ))
do
    echo Index=[$i]
done

# Print out the last line of each shell script in the current directory
echo Print out the last line of each shell script
for FILE in *.sh
do
    echo ====================================
    echo File=[$FILE]
    tail -n 1 $FILE
done

echo ''
```

```
#
*********************************************************************************
*
# While loop example

echo ----------------------------------
echo While loop

# Countdown to blastoff
echo Executing a while loop to countdown to blastoff
counter=5
while [[ $counter -gt 0 ]]; do
    echo Countdown [$counter]
    counter=$(($counter - 1))
done
echo Blastoff
```

$ cat commandline_arg.sh

```
#!/bin/bash
#
*********************************************************************************
# Processing command line arguments
# What is the name of the executed script?
echo Name of script [$0]
# How many were provided?
echo Command line argument count [$#]
# Iterate through each argument
for arg in $@; do
        echo Argument [$arg]
done
# Display all the arguments as a string
echo All arguments [$*]
```

```
# Use parenthesis for arguments with numbers 10 or larger
if [ "${12}" != "" ]; then
        echo Argument 12 is [${12}]
        echo Argument 12 is NOT [$12]
fi
```

$ bash commandline_arg.sh first second third fourth fifth 6 7 8 9 10 11 12 13

```
Name of script [commandline_arg.sh]
Command line argument count [13]
Argument [first]
Argument [second]
Argument [third]
Argument [fourth]
Argument [fifth]
Argument [6]
Argument [7]
Argument [8]
Argument [9]
Argument [10]
Argument [11]
Argument [12]
Argument [13]
All arguments [first second third fourth fifth 6 7 8 9 10 11 12 13]
Argument 12 is [12]
Argument 12 is NOT [first2]
```

$ cat password_generate.sh

```
#!/bin/bash
# Grab the command line arguments
passwd_word_count=$1
separator=$2
# Start with a blank password
password=''
# Get the total number of words in the word list
total_word_count=`wc -l ../wordlist.txt | awk '{print $1;}'`
# Build the password using the specified number of words
for (( i=1; i<=$passwd_word_count; i++ ))
do
   # Generate a random number using OpenSSL to be cryptographically secure
   rand_num_hex=`openssl rand -hex 4`
   rand_num_dec=$((16#$rand_num_hex))
   # Use the random number as an index into the word list
   word_index=$(($rand_num_dec % $total_word_count))
   random_word=`awk -v idx="$word_index" '{if (NR==idx) print $1}' ../wordlist.txt`
   # Capitalize the word
   random_word_upper=`echo ${random_word^}`
   # Insert a separator if this isn't the first word in the password
   if [[ ${#password} -gt 0 ]]; then
      password=$password$separator$random_word_upper
   else
      password=$random_word_upper
   fi
done
echo $password
```

$ bash password_generate.sh 4 '-'