Lab 04 – Shell (BASH) Scripting in Linux Part III

Objectives

- Learn about Loops
- Learn about Functions

FAQ: Read Array Input Using Read Command

Script

```
#!/bin/bash

read line
list=(${line})

for i in ${list[@]};do
  echo $i
done
```

Loops

While Loop

The bash while loop can be defined as a control flow statement which allows executing the given set of commands repeatedly as long as the applied condition evaluates to true.

Syntax

➢ *while [expression];*

do

commands;

multiple commands;

done

Example

```
#! /bin/bash

n=1

while [ $n -le 10 ]
do
        echo "$n"
        n=$(( n+1 ))
done
```

For Loop

- *for variable in list*

  do

  commands

  done

OR

- *for ((expression1; expression2; expression3))*

  do

  commands

  Done

Example 1 (The C Language Style Loop)

```
#! /bin/bash


for((i=0; i<15; i+=1))
do
 echo "Line Number $i"
done
```

Example 2

```
#! /bin/bash

for i in 1 2 3 4 5
do
        echo $i
done
```

Example 3

```
#!/bin/bash
# basic for command

for test in Alabama Alaska Arizona Arkansas California Colorado
do
   echo The next state is $test
done
```

Example 4 (Reading from a file)

```
#!/bin/bash
# reading values from a file

file="states"

for state in $(cat $file)
do
   echo "Visit beautiful $state"
done
```

Problems in For Loop that you may encounter

Example

Things aren't always as easy as they seem with the for loop. There are times when you run into data that causes problems. Here's a classic example of what can cause problems for shell script programmers:

```
# another example of how not to use the for command

for test in I don't know if this'll work
do
   echo "word:$test"
done
```

Fix it like this: for test in I don\'t know if "this'll" work

Example

Another problem you may run into is multi-word values. Remember that the for loop assumes that each value is separated with a space. If you have data values that contain spaces, you run into yet another problem:

```
#!/bin/bash
# another example of how not to use the for command

for test in Nevada New Hampshire New Mexico New York North Carolina
do
   echo "Now going to $test"
done
```

Fix it like this: for test in ..... "New Hampshire" "New Mexico"....

Functions

A function in Shell can be defined as follows:

```
function_name()

{
```

}

E.g.

```
#Defining the function
Greet()
{
    echo Hello
}
#Invoking the function
Greet
```

Point to remember: You should only invoke the functions after it has been defined.

Arguments in a Shell Function

You can define a function that will accept parameters while calling the function. These parameters would be represented by $1, $2 and so on.

Example

```
#! /bin/bash

add()
{
        echo `expr $1 + $2`
}
#Invoking Function
add 10 20
```

Returning a value from a function

$? Depicts the value returned by the last command. Assigning that value to a variable means you are catching the value returned by the function that you invoked in the previous command.

Example 1

```
#! /bin/bash

#Defining Function
add()
{
        temp=`expr $1 + $2`
        return $temp
}

#Invoking Function
add 10 20
sum=$?
        echo "Answer = $sum"
#Invoking Function

add 30 40
sum=$?
        echo "Answer = $sum"
```

Example 2

```
#!/bin/bash
# using the return command in a function

function dbl {
   read -p "Enter a value: " value
   echo "doubling the value"
   return $[ $value * 2 ]
}

dbl
echo "The new value is $?"
```

Example 3 (Capturing the output of function in a shell variable)

Just as you can capture the output of a command to a shell variable, you can also capture the output of a function to a shell variable. You can use this technique to retrieve any type of output from a function to assign to a variable.

```
#!/bin/bash
# using the echo to return a value

function dbl {
   read -p "Enter a value: " value
   echo $[ $value * 2 ]
}

result=$(dbl)
echo "The new value is $result"
```

Example 4 (Passing arguments to the function and checking the count of argument)

```
#!/bin/bash
# passing parameters to a function

function addem {
   if [ $# -eq 0 ] || [ $# -gt 2 ]
   then
      echo -1
   elif [ $# -eq 1 ]
   then
      echo $[ $1 + $1 ]
   else
      echo $[ $1 + $2 ]
   fi
}

echo -n "Adding 10 and 15: "
value=$(addem 10 15)
echo $value
echo -n "Let's try adding just one number: "
value=$(addem 10)
echo $value
echo -n "Now trying adding no numbers: "
value=$(addem)
echo $value
echo -n "Finally, try adding three numbers: "
value=$(addem 10 15 20)
echo $value
```

Lab Assignment

1. Write a shell script to read a numeric array as an input from the user, pass that array to the function and print it.
2. Write a shell script to read a list of parameters from the command-line and print those arguments to the terminal using the while loop. The output should look something like this:

```
Parameter #1 = rich
Parameter #2 = barbara
Parameter #3 = katie
Parameter #4 = jessica
```

Submission Guidelines

1. Open this Google Docs and rename as RollNumber_LabAssignment02
2. Copy/paste the question.
3. Under each question, write the shell script and copy/paste the screenshot of the output.
4. Copy/Paste the URL in the assignment module. (Make sure to change the settings to “Anyone with the link” and “Viewer only”)