Lab *04*– Scripting in Bash

<*Jahmaal Russell: wew 103*>
IS 1003 Spring 2020
<*4/20/20*>

INTRODUCTION

The purpose of this lab is to get familiar with scripting in Bash and understand the different commands being used in the interactive and non-interactive modes and to get a brief understanding of Variables expressions, Control flow statements, Functions and Positional parameters to be used when executing scripts.

PROCESS

```
#!/bin/bash
echo "Good afternoon!"

Current_Date=$(date +"%d-%m-%Y")
echo "the current date is" $Current_Date
Current_Time=$(date +"%H:%M:%S %p")
echo "the current time is" $Current_Time
```

1. In this screenshot I modified the greeting to be executed in my script when writing the code echo “good afternoon!” By creating a variable and using the format option date + %d-%m-%y I will be able to echo the current date in the output. In order to echo the current date I have assigned the current_date variable to be executed in the echo line by typing echo “the current date is $Current_Date” I will then use this same method to write the next line of code for the variable Current_Time

```
jahmaal@jahmaal-VirtualBox:~$ nano date-parse.sh
jahmaal@jahmaal-VirtualBox:~$ chmod +x date-parse.sh
jahmaal@jahmaal-VirtualBox:~$ ./date-parse.sh
Good afternoon!
the current date is 20-04-2021
the current time is 20:47:09 PM
```

2. First I will use the command nano followed with a file name given and the shell command sh to open a text editor to code the shell that I wrote in the 1st screenshot to execute the shell by typing the command nano date.parse.sh I am opening a new file under the name “date-parse” in the text editor in order to execute a shell script I have to give permission for the script to be executed so by typing the command chmod +x date-parse.sh grants permission for the shell to be executed. By using the command ./ followed by the file date-parse.sh will execute the script that I wrote in the 1st screenshot

```
jahmaal@jahmaal-VirtualBox:~$ file date-parse.sh
date-parse.sh: Bourne-Again shell script, ASCII text executable
jahmaal@jahmaal-VirtualBox:~$ cp date-parse.sh date-parse-v2
jahmaal@jahmaal-VirtualBox:~$ file date-parse-v2
date-parse-v2: Bourne-Again shell script, ASCII text executable
jahmaal@jahmaal-VirtualBox:~$
```

3. By writing the command file date-parse.sh I am able to make sure that this file can be executed in shell script. If not the output would have read as "ASCII text" when writing files in nano I have to type #!/bin/bash in order for the script to be executed in bash by using the command cp (copy) I can copy the code that was written in the 1st screenshot into a new file. By using the command cp date-parse.sh date-parse-v2 followed by file date-parse-v2 I can copy the contents of the first file to the new v2 file. This new file can execute in bash as well by typing the command file date-parse-v2 I get the bash output. This tells me that the new file was copied successfully

```
  GNU nano 4.8                    date-parse-v2
echo "Rise and Shine! Carpe Dlem!!"

Current_Date=$(date +"%d-%m-%Y")
echo "the current date is" $Current_Date
Current_Time=$(date +"%H:%M:%S %p")
echo "the current time is" $Current_Time
```

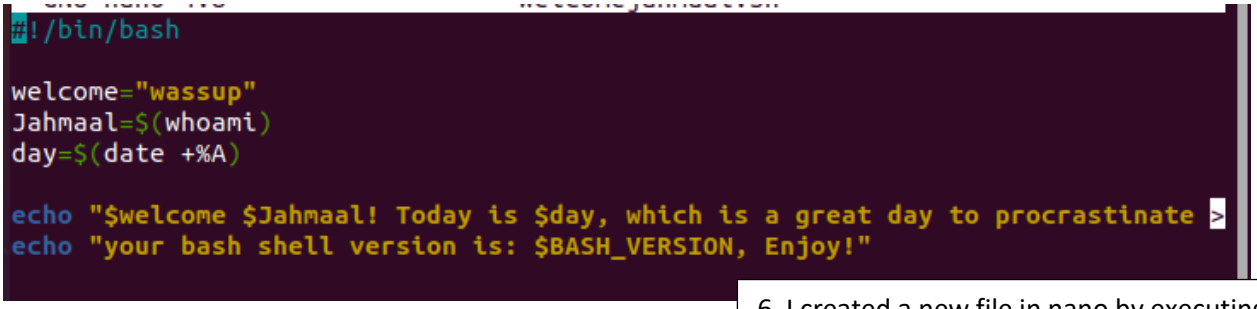
```
jahmaal@jahmaal-VirtualBox:~$ file date-parse-v2
date-parse-v2: ASCII text
jahmaal@jahmaal-VirtualBox:~$
```

4. By using the command nano date-parse-v2. I see that the code is exactly the same to demonstrate that the new file cannot be ran in script I will remove the "shebang" (specifies which program should be run in the script) the shebang in the date-parse-v2 file is #!/bin/bash once removed I can now use the command file date-parse-v2 to get the output ASCII text. The file is no longer identified as a script

```
jahmaal@jahmaal-VirtualBox:~$ echo toilet "onward"! >> date-parse.sh
jahmaal@jahmaal-VirtualBox:~$ cat date-parse.sh
#!/bin/bash
echo "Rise and Shine! Carpe Dlem!!"
```

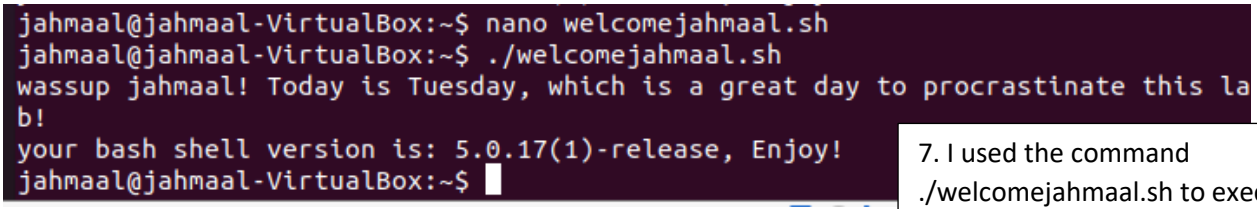
```
Current_Date=$(date +"%d-%m-%Y")
echo "the current date is" $Current_Date
Current_Time=$(date +"%H:%M:%S %p")
echo "the current time is" $Current_Time

toilet LETS GOO!
toilet onward!
jahmaal@jahmaal-VirtualBox:~$ ./date-parse.sh
Rise and Shine! Carpe Dlem!!
the current date is 20-04-2021
the current time is 03:15:33 AM

 m      mmmmmmmmmmmmm  mmmm          mmm   mmmm   mmm
 #      #        #    #"   "        m"   " m"  "m m"
 #      #mmmmm   #    "#mmm         #   mm #    # #
 #      #        #        "#        #    # #    # #    #   "
 #mmmmm #mmmmm   #    "mmm#"         "mmm"  #mm#   #mm#    #



                                        #    m
  mmm   m mm  m     m  mmm    m mm   mmm#    #
 #" "#  #"  # "m m m" "   #   #"  " #" "#    #
 #   #  #   #  #m#m#  m"""#   #     #   #    "
 "#m#"  #   #   # #   "mm"#   #     "#m##    #


jahmaal@jahmaal-VirtualBox:~$
```

5. By using the command >> this will allow me to add anything I want into a script without having to use the nano command. In this example I used the command echo toilet “Lets Goo!” >> date-parse.sh followed by the cat command I am able to verify the contents of the output. When typing the command cat date-parse.sh using the command ./date-parse.sh will execute the modified script



```
#!/bin/bash

welcome="wassup"
Jahmaal=$(whoami)
day=$(date +%A)

echo "$welcome $Jahmaal! Today is $day, which is a great day to procrastinate >
echo "your bash shell version is: $BASH_VERSION, Enjoy!"
```

6. I created a new file in nano by executing nano welcomejahmaal.sh I then modified this script by assigning the welcome variable with “wassup” Jahmaal with =$(whoami) which is a command that displays the administrators name and I assigned the variable day=$(date +%A). In the next line I used the echo command followed with the printed text by using the $followed by the variables I have assigned the output that the variables are assigned will be printed.



```
jahmaal@jahmaal-VirtualBox:~$ nano welcomejahmaal.sh
jahmaal@jahmaal-VirtualBox:~$ ./welcomejahmaal.sh
wassup jahmaal! Today is Tuesday, which is a great day to procrastinate this la
b!
your bash shell version is: 5.0.17(1)-release, Enjoy!
jahmaal@jahmaal-VirtualBox:~$
```

7. I used the command ./welcomejahmaal.sh to execute this script that I created in screenshot 6

```
#1/bin/bash

function newfunction {
  echo "my name is: $(whoami)"
  echo "home directory: $HOME"
  echo "ID: $(id)"
  lslogins  -u
}

newfunction
```

8.By using the function command I don't have to set variables to then echo the command that I tied to the variable I created a new file by using nano newFunction.sh. To execute this script I can simply use the echo command with internal variables to be executed

```
jahmaal@jahmaal-VirtualBox:~$ nano newfunction.sh
jahmaal@jahmaal-VirtualBox:~$ ./newfunction.sh
my name is: jahmaal
home directory: /home/jahmaal
ID: uid=1000(jahmaal) gid=1000(jahmaal) groups=1000(jahmaal),4(adm),24(cdrom
7(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)
 UID USER     PROC PWD-LOCK PWD-DENY LAST-LOGIN GECOS
   0 root       87                              root
1000 jahmaal    73                        03:12 Jahmaal,,,
jahmaal@jahmaal-VirtualBox:~$
```

9. I executed the script by using the command ./newfunction.sh the Id command is used to find out user and group names in the server the lslogins command displays information about the known users in the system by scanning the btmp logs which keep track of failed login attempts

```
jahmaal@jahmaal-VirtualBox:~$ a=20
jahmaal@jahmaal-VirtualBox:~$ b=35
jahmaal@jahmaal-VirtualBox:~$ [ $a -lt $b ]
jahmaal@jahmaal-VirtualBox:~$ echo $?
0
jahmaal@jahmaal-VirtualBox:~$ [ $a -gt $b ]
jahmaal@jahmaal-VirtualBox:~$ echo $?
1
jahmaal@jahmaal-VirtualBox:~$ [ $a -eq $b ]
jahmaal@jahmaal-VirtualBox:~$ echo $?
1
jahmaal@jahmaal-VirtualBox:~$ [ $a -ne $b ]
jahmaal@jahmaal-VirtualBox:~$ echo $?
0
```

10. When comparing numeric values I assigned the variable a=20 and b=35 I then used a numeric comparison operator- lt (less than) to compare these valuables to each other by typing [$a -lt $b followed by echo $? Will return the value 0 to indicate a true statement or 1 to indicate a false statement

```
jahmaal@jahmaal-VirtualBox:~$ [ "axe" = "bin" ]
jahmaal@jahmaal-VirtualBox:~$ echo $?
1
jahmaal@jahmaal-VirtualBox:~$ str1="axe"
jahmaal@jahmaal-VirtualBox:~$ str2="bin"
jahmaal@jahmaal-VirtualBox:~$ [ $str1 = $str2 ]
jahmaal@jahmaal-VirtualBox:~$ echo $?
1
```

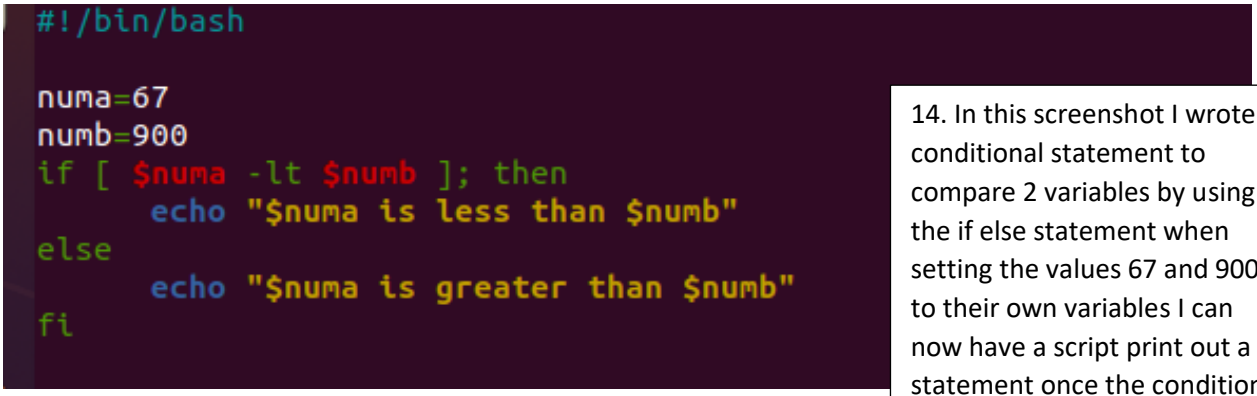
11. When comparing strings I set the word axe to equal the word bin which prompted a false output in the next line I have set a variable to each string and set the variables equal to each other which also promted a false output as these stirngs are not equal to eachother

```
#!/bin/bash

let_a="unix"
let_b="GNU"

echo "is $let_a strings equal to $let_b"
[ $let_a = $let_b ]
echo $?

num_a=50
num_b=50

echo "is $num_a equal to $num_b?"
[ $num_a -eq $num_b ]
echo $?
```

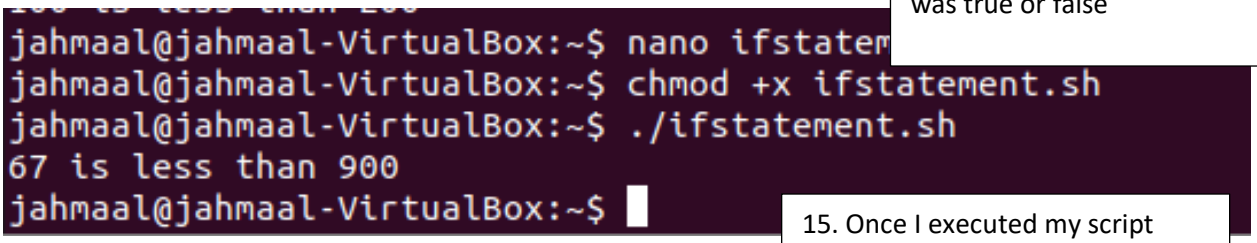
12. Here I created a new script to compare the strings by using the command nano followed by comparison.sh I have created a new file to run a script to compare “unix” to the string “GNU“ by assigning these strings into variables and setting the variables equal to each other. In the next line of code, I set the numerical values 50 to a variable and then I have set those variables to each other as well.

```
jahmaal@jahmaal-VirtualBox:~$ nano comparison.sh
jahmaal@jahmaal-VirtualBox:~$ chmod +x comparison.sh
jahmaal@jahmaal-VirtualBox:~$ ./comparison.sh
is unix strings equal to GNU
1
is 50 equal to 50?
0
```

13. I have to give permission for the script to be executed by using the chmod command when typing chmod +x comparison.sh followed by ./comparison.sh the scripts output shows that the string “unix” is not equal to the string “GNU” since the output 1 that was printed shows that it was a false comparison but the numerical value of 50 is equal to 50 this output is true because 0 was printed showing a true comparison.



14. In this screenshot I wrote a conditional statement to compare 2 variables by using the if else statement when setting the values 67 and 900 to their own variables I can now have a script print out a statement once the condition is met instead of a 0 or a 1 that showed whether the statement was true or false



15. Once I executed my script with the command ./ followed by the file that I created ifstatement.sh the output prints that “67 is less than 900” in the previous screenshot if condition was met since the else condition was not met the output that was written doesn’t show

```
#!/usr/bin/bash

echo $7 $8 $9
echo $#
echo $*


jahmaal@jahmaal-VirtualBox:~$ nano param.sh
jahmaal@jahmaal-VirtualBox:~$ ./param.sh 7 8 9 10

4
7 8 9 10
```

```
4
hello bash scripting world
jahmaal@jahmaal-VirtualBox:~$ nano param.sh
jahmaal@jahmaal-VirtualBox:~$ ./param.sh hello bash scripting world
hello bash world
4
hello bash scripting world
jahmaal@jahmaal-VirtualBox:~$ nano param.sh
jahmaal@jahmaal-VirtualBox:~$ nano param.sh
jahmaal@jahmaal-VirtualBox:~$ ./param.sh hello bash scripting world I am happy
to be learning more about bash
happy to be
13
hello bash scripting world I am happy to be learning more about bash
jahmaal@jahmaal-VirtualBox:~$
```

```
jahmaal@jahmaal-VirtualBox:~$ nano param.sh
jahmaal@jahmaal-VirtualBox:~$ chmod +x param.sh
jahmaal@jahmaal-VirtualBox:~$ ./param.sh 1 2 3 4
1 2 4
4
1 2 3 4
```

16. With positional paremeters you can set a value to produce the output of a character for example by using echo $7 $8 $9 will only display the 7th 8th and 9th character as the output and if that parameter is not meant than nothing will be displayed in the output the echo $# command will display the number of words or values in the output so when I executed my script./param.sh 7 8 9 10 will show the number 4 as that represents that 4 values are being displayed the echo $* will display all of the words or values typed in the output in my case the output shown is 7 8 9 10

```
jahmaal@jahmaal-VirtualBox:~$ nano new loop.txt
jahmaal@jahmaal-VirtualBox:~$ nano forloop.txt
jahmaal@jahmaal-VirtualBox:~$ cat forloop.txt
Lab4
Jahmaal
Newscript
jahmaal@jahmaal-VirtualBox:~$ for i in $( cat forloop.txt ); do echo -n $i | wc
 -c ; done
4
7
9
jahmaal@jahmaal-VirtualBox:~$
```

17. In this screenshot I created a new file to display the for loop to count characters. In my example I used the words “lab4” “Jahmaal” “Newscript” by using for I in $(cat forloop.txt); do echo -n $i |sc -c ; done displays the word count of each word I have typed the output shows lab4=4 characters Jahmaal=7 characters and newscript=9 characters

```
jahmaal@jahmaal-VirtualBox:~$ nano whileloop.sh
jahmaal@jahmaal-VirtualBox:~$ cat whileloop.sh
#!/bin/bash
counter=0
while [ $counter -lt 7 ]; do
 let counter+=1
 echo $counter
done
jahmaal@jahmaal-VirtualBox:~$ chmod +x whileloop.sh
jahmaal@jahmaal-VirtualBox:~$ ./whileloop.sh
1
2
3
4
5
6
7
jahmaal@jahmaal-VirtualBox:~$
```

18. Here I used the while loop I created a new file with the command nano and named this file whileloop.sh then I set a number to the variable that I called counter counter =0 in the next line of code I typed while [$counter -lt 7]; do then I let counter+=1 this will have the output print numbers in multiples of 1 until the condition is false which is the number 7. I then granted permissions for the script to be executed by using the command ./ followed by the file whileloop.sh and the output shows the numbers printed are from 1-7. The While loop will only stop printing numbers in multiples of 1 until the condition 7 being less than 7 is false which is why the last number of the output printed is 7.

```
jahmaal@jahmaal-VirtualBox:~$ nano untilloop.sh
jahmaal@jahmaal-VirtualBox:~$ cat untilloop.sh
#!/bin/bash

counter=7
until [ $counter -lt 4 ]; do
  let counter-=2
  echo $counter
done
jahmaal@jahmaal-VirtualBox:~$ chmod +x untilloop.sh
jahmaal@jahmaal-VirtualBox:~$ ./untilloop.sh
5
3
jahmaal@jahmaal-VirtualBox:~$
```

19. Here I used the until loop I created a new file with the command nano and named the loop untilloop.sh I then set the variable counter to the number 7 and wrote in the next line until [$counter -lt 4]; do this shows that the variable condition will be met if a number printed in the output is less than 4 in the next line I let counter-=2 which will subtract 2 from the number 7 until there is a number that Is less than 4 once I executed the script the output 5 which is false then followed by the number 3 being true thus stopping the execution because the condition is met

CONCLUSION

Please provide a conclusion based on these prompts:

- Goals of this lab were to understand what each command is used for when scripting in bash. Scripting is used in cybersecurity today because you can run a script to a file that can be executed over and over again. Scripting is also used to hack computers as well spreading malware can be caused by using the for loop. With scripting time is saved as you can replicate scripts to serve their purpose in computer related tasks.
- Scripting is the most useful skill to have in the cybersecurity industry as it saves a lot of time when executing code.
- I think this lab was very easy once you understand how each command is used when scripting and why the code is executed in the format it is in.

REFERENCES

Internet Resources

. *What is a positional parameter?* (2019, October 7). https://www.computerhope.com/jargon/p/positional-parameter.htm.

Admin. (2011, April 26). *How to Read /var/log/btmp, Rotate the btmp Log With Logrotate*. studiosacchetti.com. https://www.studiosacchetti.com/2011/how-to-read-varlogbtmp-rotate-the-btmp-log-with-logrotate/.

Standford Journalism. *Basic Conditional Branching with If-Then Statements*, www.compciv.org/topics/bash/conditional-branching/.

Collaboration

none