Vivian Do
Sep 29 2021
CRN 88089
Lab 5 Part 1 Commands Report

CSC3320 System Level Programming
Lab Assignment 5 - In-Lab

Purpose: Learn how to write basic shell script.

In Chapter 2 and 3, you have learned a list of utilities. However, each time we could only type a single command on command line in terminal. It is inconvenient sometimes when a task has to been accomplished by multiple commands. For example, if the task needs to be repeated, you may have to restart the execution of the list of commands by typing the command one by one. For this reason, the shell script file is used to store the commands interpreted by shell. It is more than a regular file containing only the command. You can even write for loop, if else and switch case statement in the shell script. The shell script file can be executed directly by providing the name of it on command line.

Write a report by answering the questions and upload the report (named as Lab5_P1_FirstNameLastName.pdf or Lab5_P1_FirstNameLastName.doc) to google classroom.

This lab assignment is related to the slides #12 to #14 in chapter 4

Part 1:

Now it is your turn to create your first shell script file by following the steps below.

Step 1: Go to your home directory and create a new file named as **simple.sh (vi simple.sh or nano simple.sh),** then include following lines in your **simple.sh.**

```
#!/bin/bash
#
#Simple Script
#
echo Congratulations! Now you know shell script!
echo -n "The current time and date are: "
date
```

Step 2: Save your file and exit editor.

Step 3: Execute this file by invoking its name.

```
$ simple.sh
```

Question 1) : What did you see in the output of step 3?

-bash: simple.sh: command not found

```
[vdo10@gsuad.gsu.edu@snowball ~]$ simple.sh
-bash: simple.sh: command not found
[vdo10@gsuad.gsu.edu@snowball ~]$
```

Step 4: Execute this file by adding ./ before the its name.

```
$ ./simple.sh
```

Question 2) : What did you see in the output of step 4?

-bash: ./simple.sh: Permission denied

```
[vdo10@gsuad.gsu.edu@snowball ~]$ ./simple.sh
-bash: ./simple.sh: Permission denied
[vdo10@gsuad.gsu.edu@snowball ~]$
```

Step 5: Try following command to make **simple.sh** executable.

```
$chmod a+x simple.sh
```

Step 6: Execute this file again.

```
$ ./simple.sh
```

Note: you must type ./ before the name. This is because current working directory is not in PATH. However, you can modify the value of PATH variable and add current working directory into it by referring to next part.

Question 3): Attach a screenshot of the output in step 6.

```
[vdo10@gsuad.gsu.edu@snowball ~]$ chmod a+x simple.sh
[vdo10@gsuad.gsu.edu@snowball ~]$ ./simple.sh
Congratulations! Now you know shell script!
The current time and date are: Thu Sep 23 16:12:11 EDT 2021
[vdo10@gsuad.gsu.edu@snowball ~]$
```

Question 4): Describe the meaning of **-n** option in **echo** command.

-n can be used to remove the newline character ('\n') from the output, and *echo* includes '\n' at the end of every string by default

Read the slides #13 in Chapter 4 and then answer the following two questions.

Question 5): Is "Simple Script" a comment? If not, what is the meaning of it or why we use it?

Yes, "Simple Script" is a comment; you can use # to comment at the beginning of a line

Question 6): Is "#!/bin/bash" a comment? If not, what is the meaning of it or why we use it in first line?

No, "#!/bin/bash" is not a comment; this is a "she-bang". Since the lines are in bash, this executes as a bash script

Part 2:

To discard the ./ before the script file name when executing it, we need to change the PATH variable's value and add current working directory into it.

Step 7: Print out the value stored in PATH variable.

Question 7) : How many directories you can find in the output?
Note: the directories are separated by colon.

There are six (6) directories!

```
[vdo10@gsuad.gsu.edu@snowball ~]$ echo $PATH
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/vdo10/.local/bin:/home/vdo10/bin
[vdo10@gsuad.gsu.edu@snowball ~]$
```

Step 8: Try command below to insert current working directory at the beginning of the string value stored in PATH variable.

```
$PATH=.:$PATH
```

Step 9: Execute simple.sh again by trying following command.

$simple.sh

```
$ simple.sh
```

Question 8) : Can you find errors prompted in step 9 ? If not, please briefly describe why there is no need to put ./ before the file name.

No errors were found in Step 9 since we added our directory to the $PATH variable. Now we can execute the command without './' since it'is our path variable

```
[vdo10@gsuad.gsu.edu@snowball ~]$ PATH=.:$PATH
[vdo10@gsuad.gsu.edu@snowball ~]$ simple.sh
Congratulations! Now you know shell script!
The current time and date are: Thu Sep 23 16:30:10 EDT 2021
[vdo10@gsuad.gsu.edu@snowball ~]$
```

Step 10: Log out the connection to the snowball server and reconnect to it. Or simply close your terminal and then reopen your terminal.

Step 11: Print out the value stored in PATH variable again.

Question 9: Can you find the current working directory . in the PATH variable?

No. We added a temporary value. After closing and reopening the terminal, we have our default path back since the path we added was lost.

```
[vdo10@gsuad.gsu.edu@snowball ~]$ echo $PATH
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/vdo10/.local/bin:/home/vdo10/bin
[vdo10@gsuad.gsu.edu@snowball ~]$
```

Step 11: Execute simple.sh again by trying following command.

$simple.sh

```
$ simple.sh
```

Question 10) : Can you find errors prompted in step 11 ? If yes, please explain why?

Yes. The terminal cannot find the command since the PATH variable is not in this directory. We have to redo the whole process using *./simple.sh* again to add it to the PATH directory in order for it to be executable.

Part 3 - Optional:

This part is optional, but you will find more questions about this part in your next lab.

Step 1:

Create a new file named as **checkError.sh (vi checkError.sh or nano checkError.sh),** then include following lines in your **checkError.sh.**

```
$#/bin/bash

/*  Check Error Script */

echo "Try to find out some errors!!!"

# Seach for the words which can be matched by regex [^a]*ce
# And save the output to file "Result"
echo "The regex [^a]*ce can match the string(s):" > Result
grep '^[^a]*ce$' << END >> Result
lance
ace
brace
decide
piece
-ENDHERE

# Check the existence of file "Result"
# Send the content in "Result" to your emailbox
# $1 is replaced by your campusID
ls    mail $1 < Result

# $1 is replaced by your campusID
echo  "The result has been sent to ${1}@student.gsu.edu"
echo  "Congratulations! You have corrected all the errors!"
```

Or you can directly copy this file from my public directory to your current working directory by following command and then skip step 2.

```
$cp   /home/yye10/public/checkError.sh      checkError.sh
```

Step 2: Save your file and exit editor.

Step 3: Try following command to make checkError.sh executable.

```
$chmod a+x checkError.sh
```

Step 4: Execute this file by following command.
`$./checkError.sh  campusID`
Note: Replace campusID with your own campus ID.
E.g. $./checkError.sh ylong4

Questions:
Can you find some errors when executing the command in step 4? If yes, please point out which lines contain errors. Think about the correction in your next lab. **Before the correction, you could pre-view the slides #15 - #24 in Chapter 4.**

Hints:

- *Following is a sample of the output once all the errors are corrected*

```
$ ./checkError.sh  ylong4
Try to find out some errors!!!
checkError.sh Result
The result has been sent to ylong4@student.gsu.edu
Congratulations! You have corrected all the errors!
```

- *You can use **cat -n checkError.sh** to check line numbers.*
- *You may need to use **CTRL-C t**o terminate the execution of the command, especially for the script file with errors.*