

CSC3320 System Level Programming

Lab Assignment 5 - In-Lab

Purpose: Learn how to write basic shell scripts.

In Chapter 2 and 3, you have learned a list of utilities. However, each time we could only type a single command on the command line in the terminal. It is inconvenient sometimes when a task has to be 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 statements in the shell script. The shell script file can be executed directly by providing the name of it on the 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
```

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
```

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

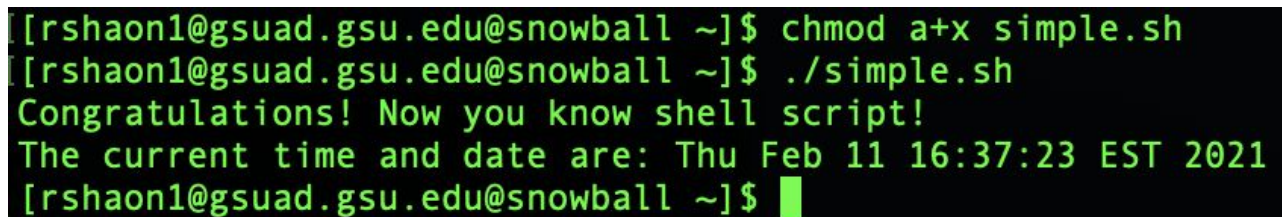
```
$chmod a+x simple.sh
```

Step 6: Execute this file again.

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

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

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



```
[rshaon1@gsuad.gsu.edu@snowball ~]$ chmod a+x simple.sh
[rshaon1@gsuad.gsu.edu@snowball ~]$ ./simple.sh
Congratulations! Now you know shell script!
The current time and date are: Thu Feb 11 16:37:23 EST 2021
[rshaon1@gsuad.gsu.edu@snowball ~]$
```

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

-n option can be used to remove the `\n` (newline character) from the output. By default, **echo** includes `\n` at the end of every string it outputs.

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, It is a comment. In UNIX/Linux, you 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 do we use it in the first line?

#!/bin/bash It is known as 'she-bang'(shabang). It's not a comment. It tells the interpreter that the following lines are written for bash , so execute this file as 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 the current working directory into it.

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

Question 7) : How many directories can you find in the output?

Note: the directories are separated by colon.

```
[rshaon1@gsuad.gsu.edu@snowball ~]$ echo $PATH
|/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/rshaon1/.local/bin:/home/rshaon1/bin
```

I can see 6 directories.

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

\$PATH=.:\$PATH

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

\$ 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 found because we added our directory to the \$PATH variable and now we can execute this without './' as this is in our path variable.

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 the PATH variable again.

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

The value which we added was temporary, so after closing the terminal and reopening it, we have lost the path which was added and now we have the default path shown.

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

\$ simple.sh

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

It does not execute since the PATH variable does not have this directory now so the terminal cannot find the command. To make it executable , we have to use './simple.sh' or it won't execute unless we again add this path to the PATH directory.

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 the following lines in your **checkError.sh**.

```
#!/bin/bash

/* Check Error Script */

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

# Search 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 mailbox
# $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/yje10/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** to terminate the execution of the command, especially for the script file with errors.*