

# CSC3320 System Level Programming

## Lab Assignment 5 - In-Lab

**Purpose:** Learn how to write basic shell script.

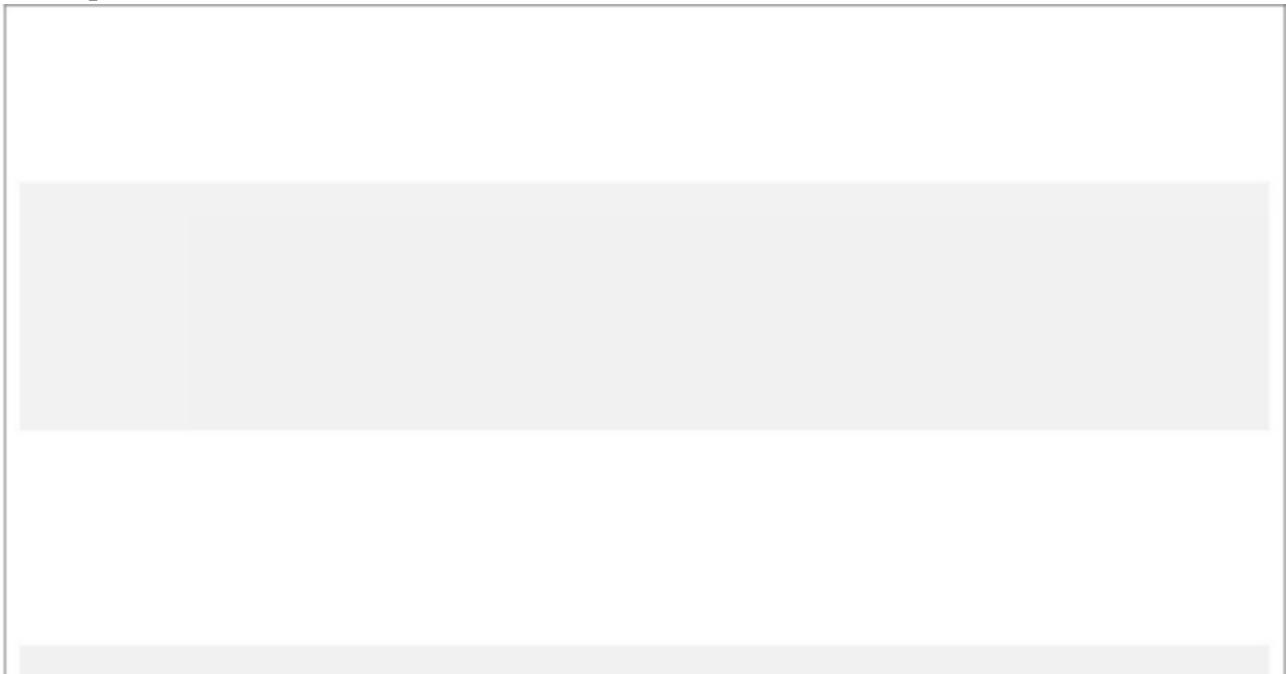
In Chapter 2 and 3, you have learned a list of utilities. However, each time we could only type a single command on command line in terminal. It is inconvenient sometimes when a task has to be accomplished by multiple commands. For example, if the task needs to be repeated, you may have to restart the 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 statement in the shell script. The shell script file can be executed directly by providing the name of it on command line.

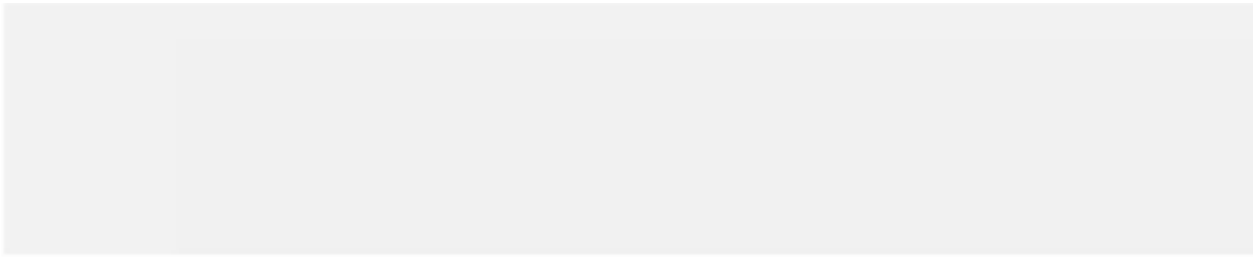
**Write a report by answering the questions** and upload the report (named as **Lab5\_P1\_FirstNameLastName.pdf** or **Lab5\_P1\_FirstNameLastName.doc**) to google classroom.

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.





```
#!/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
```

1

**The output for the step 3 is “-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?

The output for the step 4 is that “-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 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.

```
[~bash-4.2$ ./simple.sh
Congratulations! Now you know shell script!
The current time and date are: Thu Sep 23 16:08:43 EDT 2021
~bash-4.2$
```

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

The **-n** option in the **echo** command ensures that the resultant date and time are printed on the same line as the string than printing it onto the next line.

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, SimpleScript is a comment .

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

The line or command “#!/bin/bash” is not a comment because its job is to instruct the script to use the bash shell. It’s usually mentioned on the first line. It’s usually written as the first line of the script because it basically commands the OS to invoke the specified shell to execute the commands that follow in the 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.

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

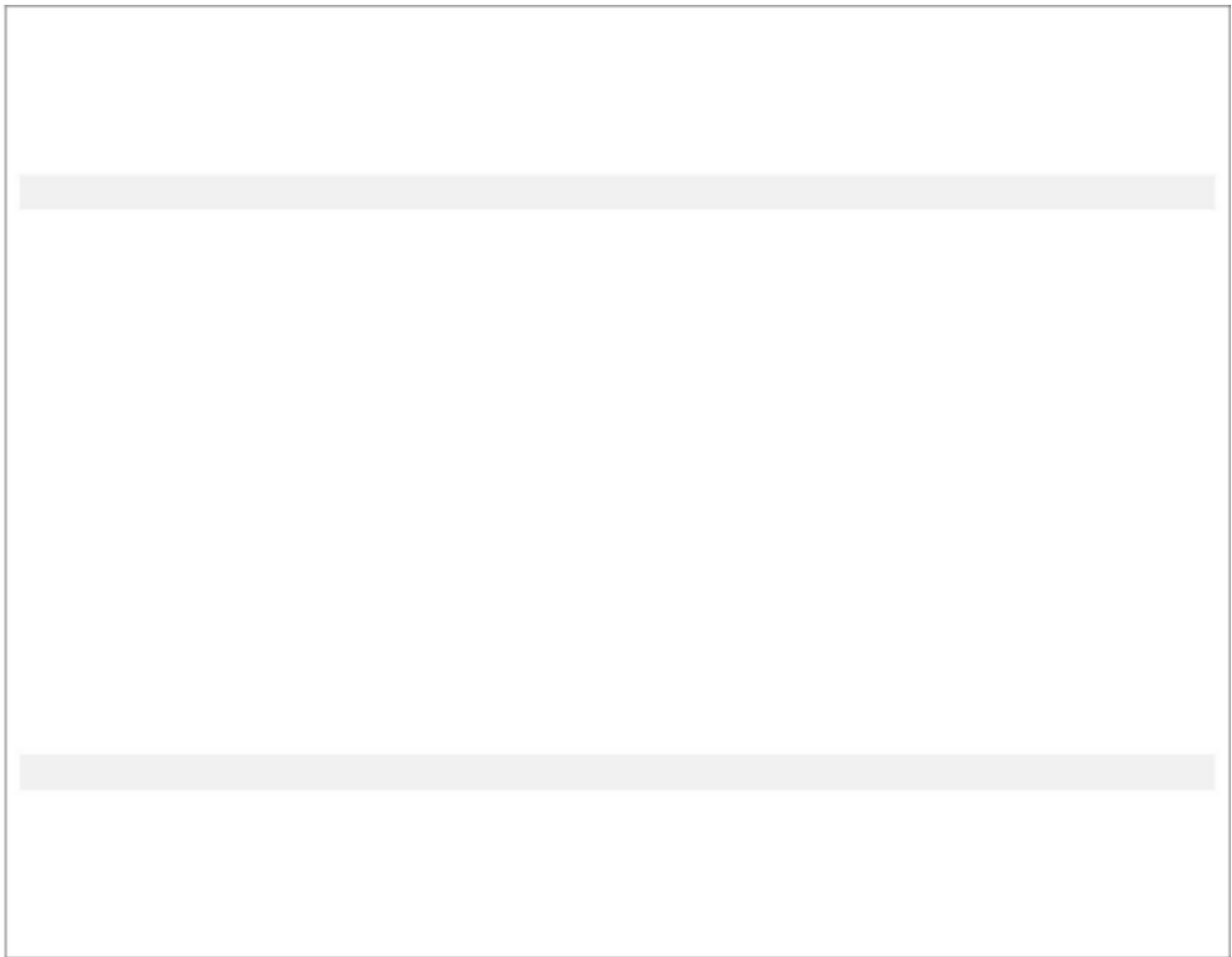
**There are 4 directories in my output**

```
Lab3 Lab4 simple.sh temp_course.txt
-bash-4.2$ echo $PATH
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin
-bash-4.2$
```

**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.

**There were no errors in this step because the current working directory has been added to the path. The file that is ready to be executed is in the current working directory and does not need to be specified.**

**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?**

**I was not able to find the current working directory in the PATH variable.**

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

```
simple.sh
```

```
$
```

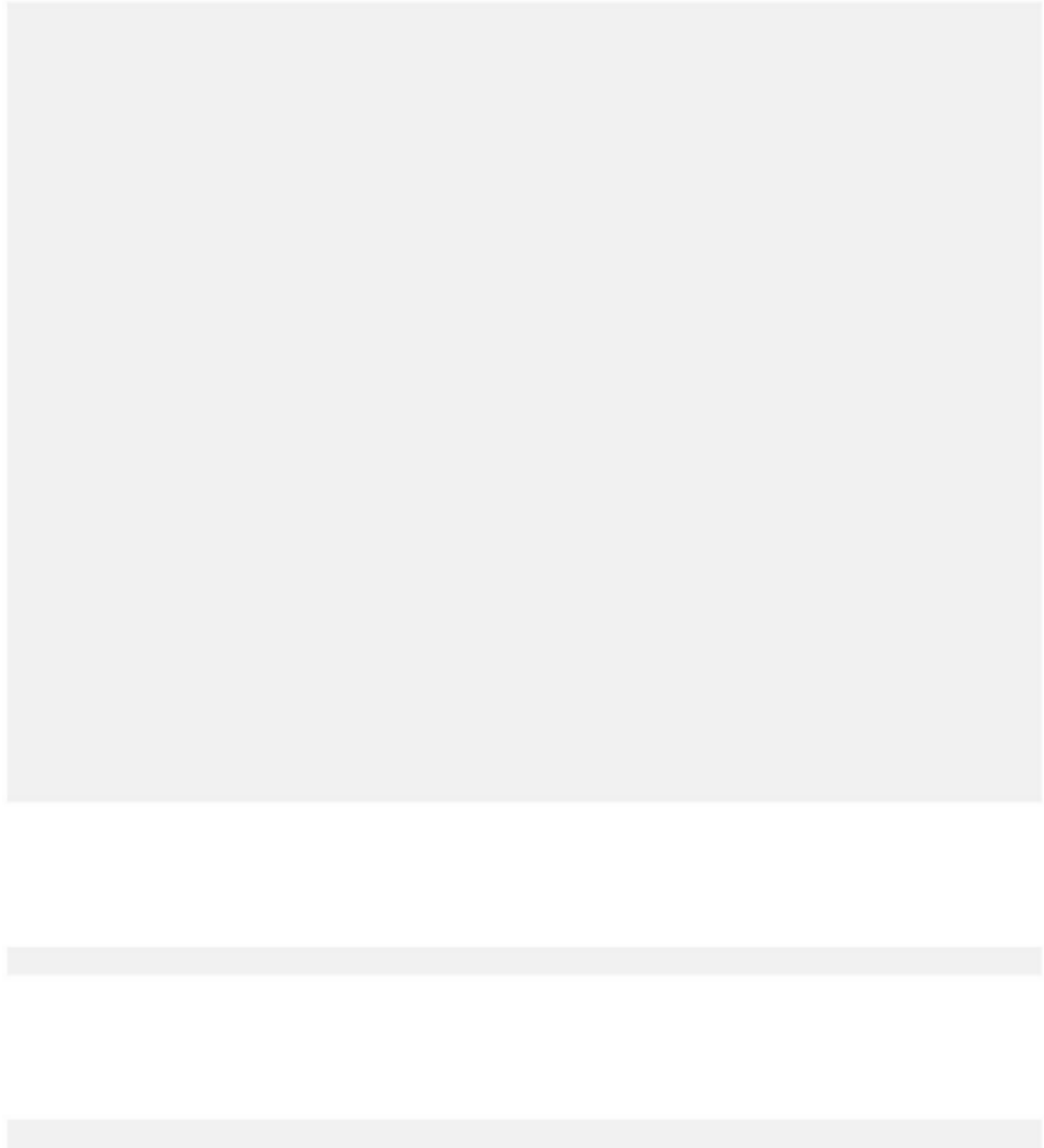
**Question 10) : Can you find errors prompted in step 11 ? If yes, please explain why?**

**As I enter the command simple.sh I had a error saying that “command not found” because I think the path was not specified and the path variable was not assigned to the current working directory**

**Part 3 - Optional:**

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





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!!!"
```

```
# 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/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
```

4

```
./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.**

**As I was executing the Step 4 command, I did run into errors. The lines that contained the errors are line 1,3,15,15, 15.**

```
-bash-4.2$ vi checkError.sh
-bash-4.2$ chmod a+x checkError.sh
-bash: chmod: command not found
-bash-4.2$ chmod a+x checkErrors.sh
chmod: cannot access 'checkErrors.sh': No such file or directory
-bash-4.2$ ls
checkError.sh  homeworks  Lab3  Lab4  Result  simple.sh  temp_course.txt
-bash-4.2$ chmod a+x checkError.sh
-bash-4.2$ ./checkError.sh 002553239
./checkError.sh: line 1: /bin/bash: No such file or directory
./checkError.sh: line 3: /bin: Is a directory
./checkError.sh: line 15: warning: here-document at line 7 delimited by end-of-file (wanted 'END')
./checkError.sh: line 15: syntax error near unexpected token `newline'
./checkError.sh: line 15: `echo "The regex [^a]*ce can match the string(s):" > Result grep '^[a]*ce$' <<END>>'
-bash-4.2$ 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.*

