3.1bash-scripts-lab

CST 334 (Operating Systems)
Dr. Glenn Bruns

Lab: Bash -- scripting

The purpose of this lab is to get you started on creating your own bash scripts. Use the hints at the end of the lab only when you get stuck.

You can use any Linux machine to work on this lab.

1. Enter a command at the bash prompt to display “Hello, World”. (Try it both with quotes and without quotes.)

2. Write a bash script named `hello.sh` that will display “Hello, World”.

3. Set the permissions of your script so that it can be executed, and then run it.

4. What happens if you try to run it like this: `$ hello.sh` ? Why? (The $ symbol indicates the command prompt.)

5. Create a subdirectory of your home directory named `bin`. (You may have previously created it.)

6. Move your bash script `hello.sh` to your bin directory.

7. Now, from your home directory, try running `hello.sh` like this: `$ hello.sh` What happens?

8. Update your `PATH` variable as follows, so that your own bin directory is in the path:

`$ PATH=$PATH:$HOME/bin` (the first `$` is the bash prompt)

9. Display the value of your `PATH` variable now.

10. In the previous problem, why does `PATH` appear without a `$` at the front on the left, but not on the right?

11. From your home directory, enter `hello.sh` at the bash prompt. What happens this time? Why?

12. What directory is indicated by the symbol “.”?

13. Would it be a good idea to add “.” to your `PATH` variable? Think about it, then google to find an answer.

14. Go to a directory where `hello.c` appears. (I expect you to have such a directory. If not, create one and put hello.c there. Get hello.c /home/CLASSES/brunsglenn or write it yourself.

15. Create a makefile in that directory to compile `hello.c`. The file must be named '`Makefile`'. Test that your makefile works.

16. Modify your hello.sh program so that it will print "Hello, ***input***" where ***input*** is the parameter provided on the command line when `hello.sh` is run. For example:

```
$ hello.sh Whirled
```

Should produce "Hello, Whirled" as output.

17. If you have time, create another bash script. This script should take a pathname as a command-line argument, and output a single number that gives the total number of files in that directory. Please don't look online for the solution.

18. If you still have time, figure out how to access the CSUMB library's Safari account, and look for books there on Linux and bash.

19. If you still have time, read the bash man page some more, and experiment with things you see there. Or, if you prefer, create more shell scripts.

Hints:

1. You may need the 'echo' command.

2. Take your command from the previous question, put it in file hello.sh, and don't forget to add the "shebang" line.

3. Use command 'chmod'. For example, 'chmod +x hello.sh'. To execute, don't forget the './' before the command, which tells bash the command is in the current directory.

4. It won't work because the current directory isn't in your PATH, so the command won't be found. For more info, see the good discussion here: stackoverflow.com/questions/6331075/why-do-you-need-dot-slash-before-script-name-to-run-it-in-bash

5. Don't forget command 'mkdir'.

6. Command 'mv'.

7. It shouldn't run, because bash still can't find the command.

8. -

9. Use ‘echo $PATH’.

10. Bash is strange. When you want to get the value of a variable, you put $ in front of it. You don’t need the $ when you are assigning a value to a variable.

11. Your command should run now. The directory containing the command is in your PATH.

12. ‘dot’ indicates your current working directory

13. No, it is avoided because of security problems. See the stackoverflow link above.

14. -

15. The Makefile you create might look like this: (it must be a tab before ‘gcc’)

```
hello: hello.c

    gcc -o hello hello.c
```

Test the makefile by typing command ‘make’.

16. See lecture notes about how to use the parameters of a main program in C.

17. Try combining the ‘ls’ and ‘wc’ commands with a pipe. For example, at the bash prompt, try ‘ls | wc -l’. What does this give you? This will be the core part of your script.

18. -

19. -