Lab - Odds and Ends

==========

Follow the instructions line-by-line.

\* Type in the commands as is, but ignore the beginning prompt.

\* Enter, tab, up and down are represented by <ENTER><TAB>,<UP> and <DOWN>.

\* "No output" or "nothing happens" are valid answers to any of the questions.

==========

==========

1. Open a new terminal window.

[NO OUTPUT]

----------

==========

2. In your home directory, start editing a text file called temp.txt using nano.

Write the command you used to do this below.

----------

nano temp.txt

==========

3. Open another terminal

[NO OUTPUT]

----------

==========

3. In this terminal, show (list) all running processes / programs.

Write the command that you used to do this, and the last two lines of output.

----------

ps

19144 ttys000 0:00.05 nano temp.txt

19146 ttys001 0:00.01 -bash

==========

4. Run the same command, but look for a specific process. (It's the version of the command that has | grep ...). Look for the program that you started to edit a file, nano.

Write the command that you used to do this, and all of the output.

----------

==========

5. Stop (kill) the process that's called nano "temp.txt" by using the process id shown in the output of your previous command (first number after user name).

----------

==========

6. Go to your other terminal window. What happened to nano? What was the message on the screen?

----------

==========

7. Close the terminal window that nano was in, and go back to the terminal where you ran ps.

[NO OUTPUT]

----------

==========

8. Now... using nano, create a shell script in your home directory called hello.sh. It should contain the following text exactly:

#!/bin/bash

echo "hi there!"

Quit and save when you're done.

What command did you use to do this?

----------

touch hello.sh

nano hello.sh

==========

9. Change the permissions (modify) on hello.sh so that the \*user\* (u) can \*execute\* (x) it:

Write the commands that you used to do this below.

----------

chmod u+x hello/sh

==========

10. Run your script (hello.sh).

How did you do this? What was the output?

----------

ent-v314-011:~ student$ ./hello.sh

hi there!

==========

11. Change to the root directory. Try running your script again (hello.sh). What was the output (if there's an error, write it out)?

----------

ent-v314-011:/ student$ ./hello.sh

-bash: ./hello.sh: No such file or directory

==========

12. Now trying using the full, absolute path to your script (that is, starting with /...). What did you write in? What did it do?

----------

cd /users/student

==========

13. Go back to the directory that your hello.sh script was in. What command did you use to change to this directory?

----------

==========

14. Type in the following command:

echo $PATH

Write down the output of this command

----------

==========

15. Type in the following command to show all environment variables:

env

Write down the last two lines of output for this command

----------

==========

16. Set your PATH to include your home directory. Do the following (substituting student or username for professor)

PATH=$PATH:/Users/professor

Now check your path again.

echo $PATH

Write down the output of the last command. It should include your home folder.

----------

==========

17. Go back to root (/)

Try running your script simply by typing

hello.sh

It should work now! What is the output?

----------

==========

18. Save this file in the repository that you created from parts 1 and 2.

Add and commit it to your local repository and push to the remote repository. Check github to see that your work was submitted.

----------

==========

19. Optional - Try writing this shell script!

In your repository, create an executable shell script called make\_5\_files that creates 10 files in the directory that it's called in. The file names should be:

myfile1.txt

myfile2.txt

.

myfile10.txt

Use a for loop to do this. Add and save in your repository, push to the remote.

----------

==========

20. Optional - Try writing this shell script!

In your repository, create an executable shell script called say\_twice. It should take one argument - a filename. It will cat out the contents of that file twice, with a row of dashes between each (use cat, echo... then cat again). Create a test file calle foo.txt ... that contains foo, bar and baz... each on separate lines.

Add and save in your repository, push to the remote.

----------