Homework 8

Due Friday, April 2nd, 2021 at 11:59 pm ET

This homework is an adventure through the fascinating world of bash script. A fun fact: bash stands for Bourne-Again-SHell. Named after Stephen Bourne, the author of the direct ancestor to the current Unix shell sh.

Another fun fact: Bash script is used to make the auto graders for most programming courses in ECE.

We understand that bash might be new to you. The very end of this README has a guide to help you get started.

Goals

This homework has several objectives:

1. Write some simple bash scripts that use environment variables and input/output redirection.
2. Have some fun with bash commands.

Starter Code

Clone your HW8 repository by clicking on the GitHub classroom link distributed through Piazza. This will create a repository on GitHub with your user name. Use `git clone` to clone that repository locally.

There should be thirteen files in your repository when you begin:

1. This README file
2. `test.in` , a sample input file you can use for testing purposes
3. `cmd1` , a python file that is one of the programs we ask you to run
4. `cmd2` , a python file that is one of the programs we ask you to run
5. `cmd3` , a python file that is one of the programs we ask you to run
6. `problem1.sh` , a starter file you can use to help get started with problem 1
7. `problem2.sh` , a starter file you can use to help get started with problem 2
8. `problem3.sh` , a starter file you can use to help get started with problem 3
9. `TC_1.py` , a test case for problem 3
10. `TC_2.py` , a test case for problem 3
11. `TC1` , the expected output for problem 3 using `TC_1.py` .
12. `TC2` , the expected output for problem 3 using `TC_2.py` . (You will not get the expected output when you run `TC_2.py` . We are interested to see re-direction in the error here.)
13. `student_solution_problem1.py` , a python helper file to provide functions to the bash script

In this assignment, you should only modify `problem1.sh`, `problem2.sh` and `problem3.sh`. All other files should NOT be modified.

Instructions

In this homework, you will write three bash scripts.

We will use environment variables called `INFILE` (which will contain a filename that we will read input from) and `OUTFILE` (which will contain a filename that we will redirect output to). When grading the homework, we will specify these variables ourselves (so you won't know what they are ahead of time!) but we suggest you set them to `test.in` and `test` for testing purposes (in your terminal). **(Note: If you hardcode the file names in your starter files, then they are no longer environment variables. This will lead to 0 points beings awarded for this task)** (Remember you can set these environmental variables using the `export` bash command as described in the notes.) We will run python code from bash scripts and hone our skills in redirection. We will also learn the difference between creating and appending to a file.

Please consult the class notes for this assignment. Optionally, we have also provided a short guide to help with bash at the bottom of this README.

Problem 1

Modify `problem1.sh` to run `cmd2`, getting the input from the file specified in the environment variable `INFILE` (i.e. input redirection) and piping the output to `cmd1`. When running with `INFILE` set to `test.in`, the output you see should be:

```
> ./problem1.sh
1 enil si
2 enil si
!gniog spe
enil tsal eht si siht ,gniddik
```

Note: To run the local commands `cmd1`, `cmd2`, and `cmd3`, you will need to use preface by `./`, e.g., `./cmd1` and `./cmd2`.

Problem 2

Modify the script, `problem2.sh` to do the following:

1. Creates a new variable (it does not have to be an environment variable) that is `OUTFILE` *concatenated* with ".out"
2. Prints out the value of this new variable (Hint: Use the `echo` command to print in bash.)
3. Creates a new variable that is `OUTFILE` *concatenated* with ".err"
4. Prints out the value of this new variable (Hint: Use the `echo` command to print in bash.)
5. Write a one line bash command that:
    i. Runs `cmd1` getting the input from `INFILE`
    ii. Pipes the output to `cmd3` redirecting stdout (but not stderr!) to the filename you created in step 1, and stderr (but not stdout!) to the filename you created in step 3.

When running with `INFILE` set to `test.in` and `OUTFILE` set to `test`, you should see the following:

```
> ./problem2.sh
test.out
test.err
> cat test.out
This is line 1
This is line 2
Just kidding, this is the last line
> cat test.err
I do not like exclamation points
```

Problem 3

We can use bash scripts to run python code. When we write `python3 python_script.py` on the terminal we execute code, the same can be done in the bash script as:

```
#!/bin/bash
python3 python_script.py
```

The `diff` command is used in bash to display the differences between two files. Each set of differences is called a "diff" or "patch". For files that are identical, diff normally produces no output. You may want to consult the diff manual for more information or see some diff examples before you begin.

You need to do the following for this problem:

1. Use `python3` to run the script TC_1.py and redirect stdout (but not stderr!) to the filename `test1.txt` and stderr (but not stdout!) to the filename `error1.txt`
2. Use `python3` to run the script TC_2.py and redirect stdout (but not stderr!) to the filename `test2.txt` and stderr (but not stdout!) to the filename `error2.txt`
3. Produce the difference between TC1 and test1.txt and redirect the output to `student.txt`.
4. Produce the difference between TC2 and test2.txt and **append** the output to `student.txt`. Please remember there is a difference between `>` and `>>`. In this step you are *appending* to a pre-existing file that already has data you do not want to over-write.

Run `./problem3.sh`.

Expected contents in:

1. `test1.txt`

   ```
   [16, 64, 4, 36]
   ```

2. `test2.txt` will be an empty file.

3. `error1.txt` will be an empty file.
4. `error2.txt`

```
Traceback (most recent call last):
File "TC_2.py", line 3, in <module>
  data = [a , b, c, d]
NameError: name 'a' is not defined
```

5. `student.txt`

```
1d0
< [None, None, None, None]
```

What to Submit

Please submit your solutions for `problem1.sh` , `problem2.sh` and `problem3.sh` . You can optionally submit all the files (including output files) in your diretory by using the commands `git add --all` , `git commit -m 'I love ECE20875'` and `git push` . Remember to verify the files on GitHub. Do not make any modifications to your code once you submit. This will help avoid the late penalty.

(Optional) Guide to help with bash

Step 0: Read the assignment.

Step 1: Figure out what an environment variable is. Where is it defined? How is it used? Why is it used?

Step 2: Using files as inputs and outputs. Redirection comes super handy in this.

You may want to try a few minor examples referred to henceforth as points:

```
1. Read: [Making an executable script](http://www.compciv.org/topics/bash/scripting/).
2. Execute a command (eg. cmd1) with an input file (eg: test.in) ( Hint: `<` will accept
input from a specified file.)
3. Define a variable and set it to your input file.
4. Try point 2 replacing the input file name with the variable declared.
5. On your terminal and observe the results:
    a. Define a variable VAR
    b. Set VAR to "anything"
    c. Try: echo VAR
    d. Try: echo $VAR (Is it different from the previous step? Why?)
6. Write the commands in Point 4 as a shell script. Try executing the shell script. Refer to
Point 1 in case you have forgotten how to execute.
```

Step 3: Pipes `|` send the output of one command as input of another command.

Try completing Problem 1 with the pipe command.

Step 4: Try a few examples which are henceforth referred to as points.

1. Make a file call it something.txt (.txt extension may be substituted or omitted: Try it!)
2. Declare a variable VAR and set it to a value. (eg: VAR may be assigned as "hi")
3. Display VAR and redirect the output to something.txt (Or whichever extension you have used). (Hint: `>` will redirect the output to a specified file)
4. Observe the contents of the file -> something.txt