CS 279 - Homework 5

Deadline

11:59 pm on **Friday, October 21**

Purpose

To think about some recent Bash odds-and-ends, file globbing options, `grep` command options, and BRE options, and to write a few Bash shell scripts.

How to submit

You will complete **Problems 1-4** on the course Canvas site.

For the rest of the problems, you will create several files and then submit those to the course Canvas site.

NOTE: While I list the separate files you need to submit for each problem below, I am going to set up Canvas to *also* accept `.zip` files.

That is,

- you can submit each file to Canvas,
- OR, if you prefer, you may compress your files to be submitted into a single `.zip` file and submit that `.zip` file to Canvas.

Important notes

Assume, for all `bash` scripts in this course, that the following are required:

- Start each script with the line that is considered good style (and is a CS 279 course requirement), that specifies that this script should be executed using the `bash` shell
- After a blank line, put in one or more **comments** including at least the name of the shell script, your name, and its last modified date
- And follow these comments with a blank line.

Problem 1 - 7 points

Problem 1 is correctly answering the "HW 5 - Problem 1 - Short-answer questions on some recent Bash shell odds-and-ends" on the course Canvas site.

Problem 2 - 5 points

Problem 2 is correctly answering the "HW 5 - Problem 2 - Short-answer questions on more file globbing/filename expansion options" on the course Canvas site.

Problem 3 - 5 points

Problem 3 is correctly answering the "HW 5 - Problem 3 - Short-answer questions on some grep options" on the course Canvas site.

Problem 4 - 5 points

Problem 4 is correctly answering the "HW 5 - Problem 4 - Short-answer questions on some Basic Regular Expression (BRE) options" on the course Canvas site.

Problem 5

In a file `hw5-5.txt`, include:

- your name
- the part you are giving an answer for
- the command that does each of the following (except, for 5 part h, just the specified BRE):

5 part a

Add to/extend your executable path within your current shell to include `~st10/279stuff` and `~/bin` and the current directory.

5 part b

Within your current shell, create a custom version of the `rm` command so that it runs with the `-i` option whenever you type just `rm`

5 part c

Output the names of all files in the current directory whose names start with an uppercase `X` or `Y` or `Z` and end with a lowercase `a` or `b` or `c`.

5 part d

Output the names of all files in the current directory whose names:

- start with `test`
- end with either 0, 2, 4, 6, or 8 directly before an ending suffix `.txt`

(For example, `test2.txt` should match; `test3.txt` should not; `test12.txt` should match; `test2.txty` should not; and `oldtest2.txt` should not. Please ask me if you need more clarification on what should be matched here...)

5 part e

Recall: When you give the `grep` command a regular expression and then a single file name, it outputs the lines in that file that contain the given pattern.

Display all lines in file `fred` that include a string starting with an uppercase `X` and ending with a lowercase `a`

5 part f

Display all lines in file `fred` that include a string of 5 or more lowercase `k`s in a row immediately followed by a lowercase `j`

5 part g

Display all lines in file `fred` that include a string of 5 lowercase `k`s in a row followed by any characters and then followed by a lowercase `j`

5 part h

Consider: a C++ identifier must start with a letter, that can be followed by zero-or-more letters, digits, or underscores.

Write just a BRE that would match a C++ identifier.

Submit your resulting `hw5-5.txt` .

Problem 6

(with thanks to Nathan Peralta for the idea for this Bash shell script!)

Mightn't it be convenient to have a script that properly starts a Bash shell script with the CS-279-required parts? (And if you put this in your `~/bin` directory on nrs-projects, you then can call it from anywhere in your nrs-projects account!)

FUN FACT: the `date` command outputs the current date and time. (See `man date` for its many options for specifying the desired format of its output!)

Write a script `nanobash` or `nanobash.sh` that meets at least the following specifications (although you can add more as well, as long as you include at least the following):

- It expects one command-line argument, the name of a Bash shell script file to be edited.
  - But if it is called with no command-line arguments, it should prompt the user to enter the name of a Bash shell script file to be edited.
- If this file does not currently exist, create it and write to it at least the following lines:
  - `#!/bin/bash`
  - a blank line
  - one or more comments including at least the name of the shell script, your name, and the current date
  - another blank line
- Finally -- whether it already existed or whether you just created it -- call `nano` to open this script file.

OPTIONAL VARIATION: open it using `emacs` or `vi` if you prefer! And in that case you can appropriately change the shell script's name as well.

Problem 7

If you have used `touch` to create a large number of empty files for testing file globbing options, you might find it convenient to have a script that helps you clean these up!

Write a script `rm-empties` or `rm-empties.sh` that tries to remove all regular files that are empty (that have a size of 0) in the current working directory.

TIPS:

- Use `rm -i` in this until you are SURE it is working properly!
- this was a useful reference when I made my version of this:

  https://tldp.org/LDP/Bash-Beginners-Guide/html/sect_07_01.html

(This is another you might want to put this in your `~/bin` directory on nrs-projects.)