School of Computer Science, McGill University

COMP-206 Introduction to Software Systems, Fall 2020

Mini Assignment 2: Bash scripting

Due Date Oct 7th, 23:55

This is an individual assignment. You need to solve these questions on your own. If you have questions, post them on Piazza, but do not post major parts of the assignment code. Though small parts of code are acceptable, we do not want you sharing your solutions (or large parts of them) on Piazza. If your question cannot be answered without sharing significant amounts of code, please make a private question on Piazza or utilize TA/Instructors office hours. Late penalty is -5% per day. Even if you are late only by a few minutes it will be rounded up to a day. Maximum of 2 late days are allowed.

You MUST use `mimi.cs.mcgill.ca` to create the solution to this assignment. You must not use your Mac command-line, Windows command-line, nor a Linux distro installed locally on your laptop. You can access `mimi.cs.mcgill.ca` from your personal computer using **ssh** or **putty** as seen in class and in Lab A.

All of your solutions must be composed of commands from the list provided at the end of this assignment description.

For this assignment, you will have to turn in two shell scripts. Instructors/TAs upon their discretion may ask you to demonstrate/explain your solution. No points are awarded for commands that do not execute at all. (Commands that execute, but provide incorrect behavior/output will be given partial marks.) All questions are graded proportionally. This means that if 40% of the question is correct, you will receive 40% of the grade.

Please read through the entire assignment before you start working on it. You can lose up to 3 points for not following the instructions.

Lab C provides some background help for this mini assignment.
Total Points: 20

Ex. 1 — Creating a backup script (9 Points)

1. **(5 points)** Your first task is to create a script which we will call `backup.sh`. This script will take an individual file or directory, and back it up into a tar file. Specifically, this script will take two inputs:
   1. the directory where the tar file should be saved;
   2. an individual file or directory to backup.

   Furthermore, the name of the tar file created will need to contain the name of the directory or file (without the extension) and the date the backup was created in `YYYYMMDD` format. Finally, the script will need to return with **error code 0** upon success and the appropriate error code otherwise (see below).
   For example, let's imagine that the following is executed on September 28, 2020:

   ```
   $ ./backup.sh $HOME/backups asgn1
   ```

   where `backups` and `asgn1` are directories. This would produce a file called `asgn1.20200928.tar` in ∼`/backups` containing directory `asgn1` and all files therein.
   You can assume that directory names will not contain a period ('`.`').
2. **(1 point)** Your script should be able to deal with both absolute and relative paths.

3. **(1 point)** If the script does not receive 2 input parameters, it should print an error message, display the usage, and exit with **error code 1**.

```
$ ./backup.sh
Error: Expected two input parameters.
Usage: ./backup.sh <backupdirectory> <fileordirtobackup>
```

4. **(1 point)** If the directory to store the tar file does not exist, the file or directory to back up do not exist, or if both arguments are the same directory, your script should print an error message indicating so, and exit with **error code 2**.

```
$ ./backup.sh dir_does_not_exist file.txt
Error: The directory 'dir_does_not_exist' does not exist.
```

5. **(1 point)** If a tar file with the same name already exists, your script should ask the user whether they want to overwrite the file. If the user enters 'y' (lowercase letter Y), the tar file should be created, and the script should exit with **error code 0** (no error). Otherwise, the tar file should not be overwritten, and the script should exit with **error code 3**.

```
$ ./backup.sh $HOME/backups/ asgn1
Backup file 'asgn1.200922.tar' already exists. Overwrite? (y/n) y
```

Ex. 2 — Source code checker (11 Points)

We will now work on a script that compares the text files between two directories, for example the source files in a directory to files with the same name in a backup directory.

1. **(6 points)** Create a script called `srcdiff.sh`. This script will take two directories as input parameters, iterate over the lists of files **(2 points)**, and report files which are either present in one directory but missing in the other **(2 points)**, or present in both directories but differ in content **(2 points)**.

```
$ ./srcdiff.sh $HOME/comp206/asgn2 $HOME/comp206/asgn2-bak
/home/20xx/csuser/comp206/asgn2/backup.sh differs
/home/20xx/csuser/comp206/asgn2-bak/file1.txt is missing
/home/20xx/csuser/comp206/asgn2/file2.txt is missing
```

   Your script does not need to consider subdirectories or files therein. The script should work with both relative and absolute paths.

2. **(1 point)** Your script should not display any output beside the requested output. That is to say that you need to handle the output of any commands you use so that it is not displayed.
3. **(1 point)** If the script does not receive 2 input parameters, it should print an error message, display the usage, and exit with **error code 1**.

```
$ ./srcdiff.sh
Error: Expected two input parameters.
Usage: ./srcdiff.sh <originaldirectory> <comparisondirectory>
```

4. **(1 point)** The script should verify that both input parameters are directories, and that they are different directories. Otherwise, it should print an error message, display the usage, and exit with **error code 2**.

```
$ ./srcdiff.sh some_dir some_file.sh
Error: Input parameter #2 'some_file.sh' is not a directory.
Usage: ./srcdiff.sh <originaldirectory> <comparisondirectory>
```

5. **(1 point)** The script should function even if one or both of the input directories are empty.
6. **(1 point)** If no differences are found between the directories, the script should exit with **error code 0**. Otherwise, it should exit with **error code 3**.

WHAT TO HAND IN

Upload both of your scripts, `backup.sh` and `srcdiff.sh`, to MyCourses.

COMMANDS ALLOWED

You may use any option provided by these commands, even ones that have not been discussed in class.

`[[ ]]`	`awk`	`basename`	`bash`	`break`
`cd`	`continue`	`date`	`diff`	`echo`
`exit`	`export`	`for`	`grep`	`if`
`ls`	`printf`	`pwd`	`read`	`sed`
`shift`	`tar`	`while`		

You may also use commands discussed in class but not listed here.

MINITESTER

A tester script, `mini2tester.sh`, is provided with the assignment so that you can test how your scripts are behaving.

It is recommended that you first run your scripts yourself, test each of the options and arguments using the examples above. Once you are fairly confident that your script is working, you can test it using the tester script.

When you are ready, in order to run the tester, put the tester script in the same folder as your scripts for this assignment and run

```
$ ./mini2tester.sh
```

The idea is that the tester's output should be very similar or even identical to that of the example output provided, except for directory names.

FOOD FOR THOUGHT!

The following discussion is meant to encourage you to search independently for creative and optimal ways to perform rudimentary tasks with less effort and does not impact the points that you can achieve in the above questions.

- Can you include a third optional argument in your Ex2 script so that it compares only files with a specific extension?

  For example,

  ```
  $ ./srcdiff.sh $HOME/comp206/asgn2 $HOME/comp206/asgn2-bak sh
  ```

  would only compare the files in `asgn2` and asgn2-bak which end with the `.sh` extension.

  This option does not need to be included in the usage message, and will not be tested by TAs.