School of Computer Science, McGill University

COMP-206 Introduction to Software Systems, Fall 2023

Assignment 2: Bash Scripting

Due Date Oct 5, 17:59 EST
TA Office Hours will start on Sept 29: See MyCourses announcement for list

This is an individual assignment. You need to solve these questions on your own. Use `mimi.cs.mcgill.ca` to create the solution to this assignment, and write all the code using Vim.

An important objective of the course is to make students practice working completely on a remote system. Therefore, 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. **If we find evidence that you have been instead using your laptop without using ssh, etc., to do some parts of your assignment work, you might lose all of the assignment points.** All of your solutions should be composed of commands that are executable in `mimi.cs.mcgill.ca`.

For this assignment, you will have to turn in three 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.) **TAs WILL NOT modify your scripts in any ways to make them work.**

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

Total Points: 20

Ex. 1 — Simple backup and archive script (5 points)

In this exercise, we will see how Bash scripts can be used to run sequences of shell commands.

1. Using vim, create a script called `txtbackup.sh` Make sure that your script starts with a shebang to execute it using `bash` and is followed by a small comment section that includes your name and student ID (format of this comment section is up to you). **-1 point** if not followed. Remember that you might need to change permissions to be able to execute the script. The script should do the following in order (and nothing more):
2. **(1 point)** Print the absolute pathname of the current directory (where the script is ran from) and display the list of all `.txt` files inside of it using `ls` (don't use any flags for the command). This question should take two commands to implement.
3. **(1 point)** Create a `backup` subdirectory. Move inside of it. Display "Moved to backup directory", then display the absolute pathname of the backup directory. This question should take four commands to implement.
4. **(1 point)** Copy all `.txt` files from the parent directory to the `backup` subdirectory. Display "Copied all text files to backup directory". This question should take two commands to implement.
5. **(1 point)** Using redirection, create a file `date.txt` in the backup directory with the content "Current backup:". Then, using redirection again, append the current date and time using `date` (with no additional flags or arguments) to the end of `date.txt`. Display `date.txt` using `cat`. This question should take three commands to implement.
6. **(1 point)** Create a compressed archive `txtarchive.tgz` of all the txt files that are in the backup directory. This archive should be located in the backup directory. When creating this archive, use the verbose option so we can see all the files that are archived. Display "Created archive txtarchive.tgz", and call `ls -l`
   This question should take three commands to implement.

Note that this is a simple script that makes the following assumptions (which we will also make when grading):

1. There are some `.txt` files in the directory you are running the script from. This means that when testing, you should create some `.txt` files in the directory where the script is ran from.

2. There is no file named `date.txt` in the directory you are running the script from
3. The `backup` subdirectory does not already exist. When testing your script, I'd recommend deleting the backup directory everytime by doing `rm -r backup` (be sure you do not delete a real backup directory that you previously had before this assignment though!)

Here are two additional tips to follow (also applies to following exercises):

1. The TAs might use testing scripts when grading your assignment. Make sure to display the exact text that the assignment asks for as the scripts might automatically compare your output to the expected output. For the same reason, use the same filenames.
2. Remember that the TAs will not run the script in the same directory than you do! Be sure that you do not hardcode any pathnames.

Ex. 2 — Analyzing user input (5 points)

We will now write a short script that reads user input and returns the number of words and characters.

1. Using vim, create a script called `inputinfo.sh` Make sure that your script starts with a shebang to execute it using `bash` and is followed by a small comment section that includes your name and student ID (format of this comment section is up to you). **-1 point** if not followed.
2. **(2 points)** Display "Please enter your sentence:" and get input from the user
3. **(3 points) In a single line (-2 points otherwise)**, display the following: "Your input has # words and # characters", where the number symbols are replaced by the word count and the character count respectively. You are allowed to repeat the same command twice with different options to achieve this. Do not use a temporary file to achieve this question, even if you delete the file after. Note that the newline at the end of the user input counts as a character.
   Here is an example run:

```
$ ./inputinfo.sh
Please enter your sentence:
hello world, how are you!
Your input has 5 words and 26 characters
```

Ex. 3 — Sorted directory copy (10 points)

We will now write a script that will copy all the files of a directory to a new directory, appending a number in front of each files so that they are sorted in reverse alphabetical order.

1. Using vim, create a script called `sortedcopy.sh` Make sure that your script starts with a shebang to execute it using `bash` and is followed by a small comment section that includes your name and student ID (format of this comment section is up to you). **As this is a longer script, you should also have additional comments for the important parts of the code. -1 point** if not followed.
2. **(7 points)** Your script will take two inputs:
   - The name of an existing directory whose files will be copied. This will be referred to as the "source directory".
   - The name for the new directory where files will be copied (Will be overwritten if it exists, see part 5 below). This will be referred to as the "target directory".

   Note that the script should work with both relative and absolute paths, **-1 point** otherwise. Your script will create the target directory and copy all files from the source directory into it, appending "#." in front of each filenames, where # is the position of the file in the original directory in reverse alphabetical order.
   For example, if

```
$ ./sortedcopy.sh documents sorted_docs
```

   is called, and the source directory `documents` has the following files

```
bravo.txt
echo.jpg
tango.gif
```

   Your script will create the target directory `sorted_docs` and copy the files such that the content of `sorted_docs` is:

```
1.tango.gif
2.echo.jpg
3.bravo.txt
```

Finally, the script will terminate with **exit code 0** upon success, and the appropriate exit code otherwise (see the other parts below).
Note that the script should function even the source directory is empty, **-1 point** otherwise. You should also only copy files from the source directory, you should not copy subdirectories and the files they contain. **-2 points** otherwise. For example, if you had a directory "my_dir" in the source directory from earlier:

```
bravo.txt
echo.jpg
my_dir
tango.gif
```

The target directory's content would still be:

```
1.tango.gif
2.echo.jpg
3.bravo.txt
```

, regardless of the presence of my_dir in the source directory and whatever files that subdirectory might contain.

3. **(1 point)** If the script is not invoked with 2 arguments, it should print an error message, display the usage, and exit with **exit code 1**.

```
$ ./sortedcopy.sh
Error: Expected two input parameters.
Usage: ./sortedcopy.sh <sourcedirectory> <targetdirectory>
```

4. **(1 point)** The script should verify that the first input parameter is a directory. Otherwise, it should print an error message, display the usage, and exit with **exit code 2**.

```
$ ./sortedcopy.sh some_file.sh some_dir
Error: Input parameter #1 'some_file.sh' is not a directory.
Usage: ./sortedcopy.sh <sourcedirectory> <targetdirectory>
```

5. **(1 point)** If a directory with the same name as the target directory already exists, your script should ask the user whether they want to overwrite the directory. If the user enters 'y' (lowercase letter Y), the conflicting directory should be deleted and replaced by the new empty directory of the same name. Otherwise (for any other response), the directory should not be overwritten, and the script should exit with **exit code 3**.

```
$ ./sortedcopy.sh some_dir sorted_dir
Directory 'sorted_dir' already exists. Overwrite? (y/n)
```

WHAT TO HAND IN

Upload your three scripts, `txtbackup.sh`, `inputinfo.sh`, and `sortedcopy.sh`, to MyCourses under the **Assignment 2** folder. You do not have to zip all of the files together, but you might need to if MyCourses does not accept the file format. Re-submissions are allowed, but please try to upload all of the files again (and not just the modified ones) so that TAs do not have to go over multiple submissions to find correct files. You are responsible to ensure that you have uploaded the correct files to the correct assignment/course. **No emailing of submissions**. If it is not in MyCourses in the correct submission folder, it does not get graded.

Late penalty is -10% 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. Even if you only (re)submit a part of the assignment (e.g., one script) late, late penalty is applicable to the entire assignment.

COMMANDS ALLOWED

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

```
[[ ]]  $( )      $(( ))  $[ ]   basename
bc     break     case    cat    cd
cp     continue  date    diff   dirname
echo   exit      export  expr   for
grep   if        ls      mkdir  pwd
read   rm        set     shift  sort
tar    wc        while
```

You can also use redirection, logical and/or/negation/comparison/math operators and any check operators (such as checking if something is a file) as required with the `$[[ ]]` operator. You can use the keywords that go with conditionals and loops (do/done/fi/etc.). You can also use shell variables and use/manipulate them as needed. You may use the concept of shell functions, but it might be an overkill for this assignment.

You may not use any commands that are not listed here or explicitly mentioned in the assignment description. **Using commands not allowed will result in 2 points deduction per such command.**

ADDITIONAL RESTRICTIONS

- Your scripts should not produce any messages/errors in the output unless you explicity produce it using an echo statement (following the message examples given in the question). Violating this would result in 2 points deduction per such message occurence.
- Your scripts should not "hang", i.e. all scripts you wrote here should run in less than a second unless you are trying to copy hundreds of files. Hanging is usually a result of some logical error in your code.
- Your scripts must not create any files internally (other than what it is asked to produce as the final output tar file), even as temporary output storage that the script deletes by itself. Doing this will result in 3 points deduction on the assignment score.
- DO NOT Edit/Save files in your local laptop, not even to edit comments in the scripts. This can interfere with the fileformat and it might not run on `mimi` when TAs try to execute them. This will result in a 0. TAs will not modify your script to make it work.

ASSUMPTIONS

- You may assume that you will have the necessary permissions as needed by the script for a correct invocation (e.g., it will not be executed on a file that you cannot read to archive, etc.)
- You can assume that that no file or directory names will contain any white space characters and will contain only alpha-numeric characters, underscore (_) and period (.) characters.

QUESTIONS?

If you have questions, post them on the MyCourses discussion board, under the Assignment 2 forum, 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 the discussion board. If your question cannot be answered without sharing significant amounts of code, please use TA/Instructors office hours. Also check the MyCourses FAQ under "Assignment 2" before you post a new question. If it is already discussed there, it will not get a response.

Emailing TAs and Instructors for assignment clarifications, etc., is not allowed. You can email your TA only if you need clarification on the assignment grade feedback that you received.