School of Computer Science, McGill University

COMP-206 Introduction to Software Systems, Winter 2021

Mini Assignment 2: Intro to Shell Scripting

Due Date Feb 10th, 18:00 EST

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 -15% 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** and also transfer files to your computer using **filezilla** or **scp** as seen in class and in Lab A and mini assignment 1.

All of your solutions must be composed of commands from the list provided at the end of this assignment description and your scripts must be executable as is in mimi.

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 loose several points for not following the instructions. There are also some helpful hints given at the end of this document that can be useful to you.

Lab C provides some background help for this mini assignment.

Total Points: 20

Ex. 1 — Creating a backup script (10 Points)

1. Your first task is to create a shell script, `backup.sh`. This script will take a backup directory name and a filename as its input and create a backup of the file in the said backup directory.
   An example invocation is given below.

```
$ ls -l ~/backups206
total 9
-rwx------ 1 jdsilv2 root 702 Jan 22 10:05 tester.sh.20210122
$ ls -l ~/COMP206/finalproj/tester.sh
-rwx------ 1 jdsilv2 root 702 Jan 29 12:49 /home/2013/jdsilv2/COMP206/finalproj/tester.sh
$ ./backup.sh ~/backups206 ~/COMP206/finalproj/tester.sh
$ ls -l ~/backups206
total 17
-rwx------ 1 jdsilv2 root 702 Jan 22 10:05 tester.sh.20210122
-rwx------ 1 jdsilv2 root 702 Jan 29 12:50 tester.sh.20210129
$
```

   Here the directory to which we want to create backup files is `~/backups206` . The file we are interested in backing up is `~/COMP206/finalproj/tester.sh`

The source file **MUST NOT** be removed once the file is backed up.

2. **(1 Point)**
   If the script is not passed its two arguments, the script should throw a usage message and terminate its execution immeidately with an error code of 1

```
$ ./backup.sh
Usage: ./backup.sh backupdirname filetobackup
$ echo $?
1
$
```

3. **(1 Point)** If the directory name passed to the script is not valid (does not exist or is not actually a directory), the script should throw the following error and terminate with an error code of 1

```
$ ./backup.sh /nosuchdir ~/COMP206/finalproj/tester.sh
Error!! /nosuchdir is not a valid directory
```

4. **(1 Point)** If the file name passed to the script is not valid (does not exist or is not actually a file), the script should throw the following error and terminate with an error code of 1

```
$ ./backup.sh ~/backups206 nosuchfile
Error!! nosuchfile is not a valid file
```

5. **(1 Point)** Your script should work correctly irrescpective of whether relative path or absolute path are used for the backup directory name and file name arguments.
6. **(4 Points)** The correct execution of the script will result in a copy of the file being made to the backup directory, but with the date of execution appended to its name. In the sample output given in 1, `tester.sh` is stored in the backup directory with the date on which it was backed up (2021, Jan 29 formatted as 20210129). You should follow this formatting convention. (see the Hints section). The script should then terminate with code 0.
7. **(2 Points)** If the script detects that a backup of the said file was already taken for the day to that backup directory, it must not make another backup or overwrite the backup file, but report this to the user and terminate with error code 1.
   So continuing from the example in 1, if we run the backup script again on the same file (to the same backup directory), we will receive the following message.

```
$ ./backup.sh ~/backups206 ~/COMP206/finalproj/tester.sh
Backup file already exists for 20210129
```

8. **(0 Points)** Any error messages thrown from your script should be made by specific echo statements (which results in the messages shown above). You must suppress any other error/output messages that might result from the commands that you are using and not let it show up in the output. Violating this would result in **-1 point** penalty.
9. **(0 Points)** At no point in your backup script should you create any other files (other than your backup file), even for temporary purposes. Violating this would result in **-1 point** penalty.
10. **(0 Points)** Any instance of an incorrect termination code (from what is specified for each circumstance) would result in **-1 point** penalty.

Ex. 2 — Script to check for missing backups (10 Points),
In this task, you will write a script, `chkbackups.sh` that will help you check if there are any files in a directory that you forgot to backup today.

1. **(1 Point)** The backup script expects the first argument to be the directory to which files are getting backed up. The second argument is meant to be the directory which contains the files that you usually backup. If the second argument is not provided to the script, it will assume that you want to check if all the files in the current directory is backed up. In other words, if the script is not invoked in one of these forms, it must throw an error and terminate with error code 1.

```
$ ./chkbackups.sh
Usage: ./chkbackups.sh backupdirname [sourcedir]
$ echo $?
1
```

2. **(1 Point)** Similar to the previous exercise, if any of the directory arguments passed to the script is not a valid directory name, it should throw an error message and terminate with error code 1.

```
$ ./chkbackups.sh ~/backups206 /nosuchdir
Error!! /nosuchdir is not a valid directory
```

3. **(2 Points)** If the source directory is empty (has no files in it) then the script should display this message and terminate with error code 1.

```
$ ./chkbackups.sh ~/backups206 ~/emptydir
Error!! /home/2013/jdsilv2/emptydir has no files
```

   For simplicity, you can assume that source directories do not have any directories inside them.

4. **(2 Points)** If all the files in the source directory are already backed up for today in the backup directory, the script should display this information and terminate with code 0.

```
$ ./chkbackups.sh ~/backups206 ~/COMP206/finalproj
All files in /home/2013/jdsilv2/COMP206/finalproj have backups for today in /home/2013/jdsi
lv2/backups206
```

5. **(4 Points)** If there are files in the source directory that is missing backup for today then the script should list all such files and terminate with code 0.

```
$ ./chkbackups.sh ~/backups206 ~/COMP206/finalproj
tester.sh does not have a backup for today
```

   You may list the file names (if multiple files are involved) in any order.

6. **(0 Points)** Any error messages thrown from your script should be made by specific echo statements (which results in the messages shown above). You must suppress any other error/output messages that might result from the commands that you are using and not let it show up in the output. Violating this would result in **-1 point** penalty.
7. **(0 Points)** At no point in your backup script should you create any other files (other than your backup file), even for temporary purposes. Violating this would result in **-1 point** penalty.
8. **(0 Points)** Any instance of an incorrect termination code (from what is specified for each circumstance) would result in **-1 point** penalty.

You must write a reasonable amount of comments (to understand the logic) in both your scripts. You can lose up to **-2 points** for not writing comments.

WHAT TO HAND IN

Upload both of your scripts, `backup.sh` and `chkbackups.sh`, to MyCourses. If MyCourses is not allowing files with `.sh` extension, you may use the `tar` command as discussed in class to make a `tar` file or a `tar.gz` file that contains your scripts and submit them. Please download your submitted tar files to double check if they are good. Errorneous invocation of the tar command can sometimes result in a bad file. Such submissions cannot be unfortunately graded.

ASSUMPTIONS

You may assume that any files and directories that your script needs to access will have the necessary permissions for it to execute the tasks outlined in the assignment. You can also assume that the backup directory and source directory do not have any subdirectories.

HINTS

- Explore the `date` command using its man page. Check what options you can pass to it so that the output produced by the `date` command is formatted in the exact way you want it to be so that you can easily append it to the name of the backup file.
- Explore the `basename` command to figure out how to extract only filename from the absolute path.
- How do you get rid of unwanted messages from the commands and utilities that you use in a shell script? Linux provides a special file `/dev/null` to which you can discard any unwanted outputs. Explore how it is commonly used in writing shell scripts.
- Learn the sytnax for performing logical OR, AND, NOT operations with your condition check/test operator.

COMMANDS ALLOWED

You may use any option provided by these commands, even ones that have not been discussed in class. Not all of them may be required for you to build your solution.

`[[ ]]`	`!`	`basename`	`dirname`	`break`
`cd`	`continue`	`date`	`diff`	`echo`
`exit`	`export`	`for`	`grep`	`if`
`ls`	`printf`	`pwd`	`$( )`	`cp`
`shift`	`while`	`mv`		

You may also use commands discussed in class but not listed here. You must not use the commands `sed`, `awk`, and `find`.

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 <u>first</u> 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 in format or even identical to that of the example output provided, `minitester.out.txt` except for directory names and timestamps.

QUESTIONS?

Please use piazza. However, before posting your question, use the search functionality to check it has been already discussed. You should also look under "Mini 2 general clarifications" pinned post to check if a popular question has been already included there. They will not get individual responses again.

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.

When you make the backup copy of the file, what option can you use to the utitlity used to copy the files, so that it retains the modified timestamp (shown by `ls -l`) of the original file in the backupfile as well?