Workplace Development and Skills (CS3306) Assignment 1

Basic Unix Skills (Version 1)

(Due: February 8. 2022. Marks: 50)

Contents

1 Introduction

This assignment is about implementing some *Bourne* shell scripts: not *Bash* but *Bourne* shell scripts. There are at least two good reasons why you may not use Bash.

- Bourne shell scripts are more portable.
- I want you to learn more about command substitution. With the Bourne shell you will *have* to use these techniques; with the Bash shell you can get away with not using these techniques (at the expense of ending up with a less portable script).

There are 5 tasks in progressive levels of difficulty/work. By the end of this assignment you should know how to:

- Write a portable shell script;
- Test the command line parameters for validity;
- Do proper error handling;
- Use file name and command substitution;
- Implement pipelines and redirection;
- Use some frequently needed shell tools;
- …

2 Assignment Details

The remainder of this section explains the assignments. Please note/remember that:

- Scripts should *Bourne* shell scripts, so *Bash* commands and techniques are not allowed: no `((…))`, no `[[…]]`, no `$(…)`, no `${VAR[…]}`, no `${!VAR[…]}`, no `${#VAR}`, no `${VAR:-…}`, no `${VAR%…}`, no `${VAR/…}`, no `${VAR^…}`, no `${VAR@…}`, no `${VAR:=…}`, no `${VAR:?…}`, no `${VAR:+…}`, no `${VAR:…}`, …Please note that `$((…))` is not a standard Bourne shell feature.
- Scripts which rely on command line parameters should check their parameters for validity and conformance.
- Scripts should proper output warnings and error messages which should inform the user about mistakes and errors.
- Scripts should send prompts, warnings. and error messages `stderr`.
- Scripts should exit with a proper exit status.
- Scripts should contain proper, meaningful comments.
- Scripts should **only** output things which are mentioned in the requirement specifications.[1]
- Scripts should (properly) abort if (1) they detect errors in the command line parameters; or (2) when a parameter does not conform to the specifications—e.g. when parameters are supposed to be directories but one of them is a regular file. Other valid reasons for aborting the script's operation may also be possible but the scripts should *definitely* abort if the previous conditions arise.
- Scripts should be saved/submitted as **plain text files**. Scripts which are submitted in other file formats can't be accepted.

Task 1 (10%)**.** *Implement a shell script which outputs a single line consisting of three integers:*

1. *The number of files and directories for which you have write permission;*
2. *The number of files and directories for which you have read permission; and*
3. *The number of files and directories for which you have execute permission.*

For example.

```
$ ls -l
-rwx------  1 dongen dongen  190 Jan 22 06:40 a
--wx------  1 dongen dongen  238 Jan 22 06:40 b
d--x------ 13 dongen dongen 4096 Jan 13 15:56 c
$ ./ex1.sh
1 2 3
```

Task 2 (20%)**.** *Implement a script which outputs the names of the* regular *files in the current directory and its (recursive) subdirectories, which contain the words* Lee, Dublin, *or* Galway. *The output should not contain duplicates.*

When you test the implementation, you may use the test data which are provided in the file `test-input.tgz`*, which which you may find on* Canvas. *When you decompress the data (use:* `tar xvfz test-input.tgz`*) this will create a directory called* `test-input` *which you can use for testing.*

Task 3 (20%)**.** *Count the total number of* regular *files (not directories) in the directories which are supplied as arguments to the script. If no arguments are provided, then the script should use the current directory.*

[1] I always get submission, which output what the script *is doing,* e.g. I am now doing this, now doing that, …Clearly you don't want that; if every script did that, the user would not be able to see the wood from the trees.

Task 4 (25%)**.** *Implement a script which outputs a summary of the assignment submissions for each of the students in a class. The input consists of lines with the following format:*

```
ID:GIVEN NAMES:SURNAME:SPACE-DELIMITED SEQUENCE OF MARKS
```

For example:

```
3:Judy:Garland:8 8 9 10 10
2:Dean:Martin:10
5:Frank:Sinatra:1 2 3 4 5 6 7 8 9 10
```

The summary of a student should consist of four lines:

1. *The `ID` of the student;*
2. *The name of the student: given names and surname;*
3. *A list consisting of the student's marks; and*
4. *The average of the marks. Averages should be rounded up to the smallest integer which is greater than or equal to the average.*

Task 5 (25%)**.** *Output the name of the largest* regular *file in the current directory or any (recursive) sub-directory.*

3 Tools and Tricks

The following are some useful commands.

`find`: Search for files in a directory hierarchy.

`expr`: Carry out a simple computation.

`wc`: Compute the number of lines, words, and characters in the input.

`test`: Check file types and compare values. You usually carry out the test inside square brackets. For example, `[ -f file ]`.

`read`: Read text and assign to variable.

`sed`: Stream editor for filtering and transforming text.

`awk`: Pattern scanning and processing language.

Please remember the following:[2]

`${VAR}` Expands to the value of `VAR`.

`$@` Expands to the positional parameters, starting from one. When the expansion occurs within double quotes, each parameter expands to a separate word: `"$@"` expands to `"$1" "$2" ...`.

`$#` Expands to the number of positional parameters in decimal.

`$0` Expands to the name of the shell or shell script.

[2] See `man sh` for more information.

4 Assignment Submission

You should submit your work in the Assignment 1 section on the CS3306 area on Canvas. Please implement your assignment in a directory named `Lab01` and implement the 5 tasks as separate scripts, which should be named `ex1.sh`, `ex2.sh`, ... The assignment should be submitted as a tarred, gzipped archive of the your assignment directory. To create the submission, go to the parent directory of `Lab01` and execute

- `tar cvfz Lab01.tgz Lab01`

When you tar up your project, please make sure the directory `Lab01` only contains the files `ex1.sh`–`ex5.sh`, and does not have any subdirectories or hidden files/directories.

5 Submission Deadline

The submission deadline of February 8 is a hard deadline. Request for project submissions after the submission deadline won't be granted (unless there is a valid reason). **I suggest you submit regular incremental updates of your project before February 8.** That way you always have *some* submission and don't risk having no submission.

6 Questions?

If you have any questions about the assignment, please ask the lecturer. The demonstrators and the lecturer will help with trouble shooting during the lab sessions but please. You may ask for help if you don't understand something but we can't do the tasks for you.