CIS241 Project

Bash Script Programs

(Due: Dec 6)

Write a Bash script to perform each task described below. Your scripts should follow the conventions of Linux commands so they can be used just like Linux commands. In addition, they should be able to handle command line errors as stated in the exceptions section.

1. Write a Bash script called *newname* to change the name of a file, but if the destination file exists, it creates an index number to append to the name of the destination file, sort of version number. Suppose that the user types a command line as shown below (where the $ sing is the shell prompt):

   $newname a.txt b.txt

   Its intention is to change the filename from a.txt to b.txt. However, if file b.txt exists, your script will try to rename it as b.txt.1. If b.txt.1 is already there, it will try b.txt.2, and so on, until it can successfully rename the file to a name that has not been used by any file in the current working directory.

   Exceptions: (1) incorrect number of command-line arguments; (2) a non-existing file is given as the first argument.

2. Blackboard allows a student to submit several files for a programming project (usually including source and data files) and also allows a professor to download all of them from students in a class as a single zip file. However, the professor has to unzip a downloaded file, rename and reorganize files from each student, as well as build and run each program. Such a procedure is quite tedious and time consuming. Here, you are asked to develop a script to automate this process so the processor can focus on grading.

   Given below is a short list of files from two students in format as they appear in a downloaded zip file. Blackboard has added a prefix to the original name of each file that a student submitted, including project title, user name, and so on. And the fourth file for each student (the text file that does not have its own name except the added prefix) was created by Blackboard to provide information about each submission as well as a message the student wrote along with the submitted files.

   ```
   Project 1_edwardsj_attempt_2018-03-08-10-16-08_cipher.c
   Project 1_edwardsj_attempt_2018-03-08-10-16-08_LetFreq.txt
   Project 1_edwardsj_attempt_2018-03-08-10-16-08_makefile
   Project 1_edwardsj_attempt_2018-03-08-10-16-08.txt
   Project 1_harrisr_attempt_2018-03-02-15-56-34_cipher.c
   Project 1_harrisr_attempt_2018-03-02-15-56-34_LetFreq.txt
   Project 1_harrisr_attempt_2018-03-02-15-56-34_makefile
   Project 1_harrisr_attempt_2018-03-02-15-56-34.txt
   ```

   Before writing a script for this part of the project, you need to create a directory, named project1, and run the following command line to copy a downloaded file into the project1 directory. Note that you have to include the period (.) in the command line:

   cp /home/tao/Labs/cis241.zip .

Specifically, your script needs to take the following steps to achieve the goal:

1) Prompt the user for the name of the zip file and unzip it into a data directory under the project1 directory. For simplicity, cis241.zip contains files from three students; but you must make sure that your script would work for such a zip file with files from any number of students.
2) Create a directory, named grading, under the project1 directory and create a sub-directory under the grading directory for each student and name it by the student's user name (that can be extracted from the added prefix of a filename); for example, creating sub-directories edwardsj and harrisr according to the sample data given above.
3) Remove the added prefix from each filename in the data directory and place them into each student's sub-directory respectively. For example, the sub-directory edwardsj would have four files that he submitted, namely cipher.c, LetFreq.txt, makefile, and memo.txt. Note that the name of the file created by Blackboard is renamed as memo.txt.

In this script, you can assume that the downloaded zip file and user input are valid; so no error checking is necessary.

Naturally a further step to facilitate the professor's grading effort is to test and run each student's program automatically; that is, building the program from a makefile, executing it against an input data, and displaying the test results. It is possible for the professor to use one's own makefile and run all the programs against a data file that one provides. But such a step is not required in this project.

You may work on this project as a group of two people. When your programs work correctly, use the script command to capture your test session for each script. In case of the second part of this project, you may use the ls command with option -R to show all sub-directories and files that your script have created under your project1 directory. Make a Word or PDF file that includes your bash scripts and test sessions for submission.