CIS 241 System-Level Programming and Utilities
Project 3: Bash Shell Script -- File Backup

Due: Thursday, April 12, 2018 by the end of the day

Educational Objectives

This project is designed to help students gain hands-on experience with:

1) Writing and executing a bash shell script
2) Taking and processing command inputs from a shell script
3) Control structures in bash shell programming
4) Bash shell commands

Programming Environment

The project should be implemented with Bash shell programming language in a Linux environment. If you do not demonstrate your code with your instructor, your code will be tested on the EOS computer with the installed Bash shell (GNU bash, version 4.2.46(2) – release). A different Bash version can be used if you will demo your code working with your instructor by the due time.

Project Description

For this project, you are going to write a Bash shell script to perform file backup. Your code should follow the conventions of Linux commands so that it can be used just like a Linux command. It should be able to handle command line input errors and exceptions that are described in this assignment.

Synopsis

Let **backup.sh** be the name of your script. Your script can be executed as follows:

```
backup.sh [options] targetFileList
```

where options include **-l**, **-n**, and **--help**

Description

Its argument **targetFileList** is a list of filenames and directories in current directory. **backup.sh** copies files and directories (including the files in the directories) in the **targetFileList** into the subdirectory named as **.backup** that is located in the home directory of current user (e.g., if the user name is **wangx**, the home directory is **/home/wangx** in the EOS system). If the subdirectory **.backup** does not exist, it is automatically created by your script. If a file in **targetFileList** already exists in the **.backup** subdirectory, keep the newer version of the file in the subdirectory.

The **-l** option lists the current contents of the **.backup** subdirectory. The **-n** option shows the number of files in the **.backup** subdirectory and the amount of memory space they consume. The **--help**

option displays a brief description about this script as well as each of the options. Note that all the activities related to the options should be performed after the intended actions (as specified in the previous paragraph) are done if present. In addition, like a typical Linux command, any number of options (zero, one, two, or three) is allowed and ordering of these options is irrelevant. Combinations of the options, such as **`-ln`** or **`-nl`**, should be allowed. **`-ln`** and **`-nl`** are equivalent to **`-l -n`** or **`-n -l`**. In any case, the **`--help`** option if performed lastly no matter where it is located in the command input.

Example

```
backup.sh file1 file2 file2 -l --help
```

Copy **file1**, **file2** and **file3** to the subdirecgtory **.backup** first, and then list the contents in the subdirectory, finally, display the **--help** contents.

```
backup.sh --help file_a directory_1
```

Copy **file_a** and **directory_1** first, then display the **--help** contents.

Error Handling and Exceptions

1) No command-line argument is provided (i.e., only "**`backup.sh`**" is typed). In this case, a message such as "**The usage of this command is : backup.sh [options] targetFileList**" should be displayed and then exit.
2) If user input an option other than **-l**, **-n**, **-ln**, **-nl**, **--help**, your code should display a message like "backup.sh: invalid option b", if **-b** is one of the user input. However, files in **targetFileList** should be copied correctly.
3) Files in the **targetFileList** do not exist. Your script should be able to display a message such as "**File1** does not exist!". Meanwhile, other existing files in the **targetFileList** should be copied to the **.backup** subdirectory in the home directory of current user, as well as operations in [**options**] should work well.
4) If **targetFileList** is empty, your program should be able to perform corresponding operations according to [**options**].
5) Directories can be included in **targetFileList**. In this case, the entire directory including all the files and subdirectories should be copied to the **.backup** subdirectory in the home directory of current user.

Grading

1) File copy: 20%
2) Three options: 30%
3) Error and exception handling: 50%

Submission

If you can demonstrate that your program works correctly as specified, you will receive 100 on this project and you don't need to submit your code.

If you cannot demo it by the due time, please name your script as **backup.sh** and turn it in through Black Board by due time. Your project will be graded accordingly.