EE 463 | **Operating Systems** | **Date: 26/05/1444**
Winter 2023 | **Homework #2** | 20/12/2022

Dr. A. M. Al-Qasimi | **Due: 01/01/2023**

Operating Systems
Homework #2

EE 463
Winter 2023
Dr. A. M. Al-Qasimi

Date: 26/05/1444
20/12/2022
Due: 01/01/2023

Objectives:

The objective of this assignment is to apply what you learned in the previous labs about Linux commands and shell programming by writing some bash shell scripts.

What to Do:

Write *generalized* bash-shell scripts that are able to work correctly on any Unix/Linux computer, with any user account, and starting from any working directory without any modification, to carry out each of the following tasks:
Hint: Use environment variables, string patterns, and wild-card characters to generalize your scripts.

a. **Safe Delete (srm)**:
When you use the standard "`rm`" command in Linux, it will delete the specified files, with no chance for recovering them back later. Write a script (called **srm**) that will safely delete the file(s) passed to it as command-line argument(s). For example, typing: "`srm file1 file2 file3 …etc.`", the script shall not actually delete these files, but instead it shall move them to a *trash* directory and record their original paths and names in a *hidden* recovery record file called *trash/.srm*.

At its start, it shall always check for the existence of the directory: */home/user-name/trash*, and the file: */home/user-name/trash/.srm*, if they do not exist, the script must create them. At any change to the contents of the trash, the script shall update the recovery record file accordingly. The script shall also accept switches that modify its behavior to do the following:

1. If the script is called with a **`-d`** switch, first it shall check the *trash* directory for all files older than *40 days* and permanently delete them, and then perform its normal task.
2. If the script is called with only **`-l`** switch, then it shall list all the recoverable files in the *trash*.
3. If the script is called with a **`-b`** switch followed by a list of files (e.g. `srm -b file1 file2 file3 …etc.`), or a wildcard file pattern (e.g. `srm -b file*`) then it shall not remove the files, but rather it shall recover the matching files in the trash, by moving them back to the same directories they were deleted from.
4. If the script is called with a **`-m`** switch followed by a list of files (e.g. `srm -m file1 file2 file3 …etc`), or a wildcard file pattern (e.g. `srm -m file*`) then it shall do the same as the **`-b`** switch, but all matching files shall be moved to the *current* work directory.

b. **Phone (phone)**:
Write a shell script (called **phone**) that creates a simple telephone list (create an empty file called "phonelist" in your home directory: */home/user-name/phonelist*). Each line in this file represents one record consisting of two fields: a name and a phone number. The script shall do the following:

1. When the user types the command: "**`phone -a`** *`name number`*", this will add a new record (*name*, *number*) to the list. If the *name* or the *number* or both are missing, nothing happens. Also, if a record already exists with the same *name* and *number*, the new record is not added.
2. When the user types the command: "**`phone -s`** *`name`*" then the script should search in the file "*/home/user-name/phonelist*" and get and display all (*name*, *number*) entries matching the specified *name*. It should work the same way even if the **`-s`** switch is omitted.
3. When the user types the command: "**`phone -r`** *`number1 number2`*" then the script should replace the existing matches of *number1* in the file "*/home/user-name/phonelist*" to *number2*.

4. When the user types the command: "`phone -d name number`" then the script should delete from the file "*/home/user-name/phonelist*" *all* existing records that match (*name*, *number*).
5. When the user types the command: "`phone -d name`" then the script should delete from the file "*/home/user-name/phonelist*" all existing records that have a matching *name*.
6. When the user types the command: "`phone -d number`" then the script should delete from the file "*/home/user-name/phonelist*" all existing records that have a matching *number*.
7. When the user types the command: "`phone`" alone or with `-l` switch, then the script should show a list of all the (*name*, *number*) entries it finds in "*/home/user-name/phonelist*" file.

c. **Mailmerge (mmerge):**
Suppose that you want to write the same letter to many people, except that you want each one addressed to each recipient personally by his/her name:

1. Create a simple text file (called *template*) containing your *letter*, which has the word NAME wherever you want the person's name to appear in the letter.
2. Create a file (called *names*) containing a name-list of the letter recipients (one per line).
3. Write a shell script (called **mmerge**) that will accept three command-line parameters: the name of the letter *template* file, the word NAME to be replaced and the name of the recipient *names* file. The script shall produce a copy of the *letter* file addressed to each recipient by his/her own name.

<u>*Notes*</u>:

- The output of your script shall be as many files as the number of recipients in the *names* file, i.e. one file for each recipient, where the filenames are appended by the names of the recipients to distinguish them from one another, e.g. *letter-to-Ali*, *letter-to-Amal* …etc.
- All the letter files must include an automatically generated header line showing on the left *to* whom the letter is sent and on the right the current date, e.g.

```
| to:  Ali                    Date: 20/12/2022 |
|                                              |
```

Test your scripts **thoroughly** before you submit the assignment, to make sure they will do their specified tasks exactly as described above, even if invoked from any directory or by any user, and to make sure that all possible errors are **handled** correctly or appropriately.

Scripts that need modification to work or do not work as submitted will receive zero credit

<u>What to Turn-in:</u>

1. Each script must begin with comment lines showing its function, its author and date of last update.
2. Collect all your text files, script files, and test files (all in one flat directory) into one .zip file and submit at Blackboard no later than the due date and time.