Glasgow Caledonian University
Department of Computing

Systems Programming (M3I324183)
Diet 1 Coursework 1 18/19 Tri B ALC
`safeDel` – A bash Script Utility for better File Deletion

This coursework constitutes 50% of the total module mark. You must follow the submission instructions at the end of this specification or marks will be zeroed. This is an individual piece of coursework (i.e. not a group assignment/project) and is the only piece of coursework that will be issued for Coursework 1 in session 2018/2019 Tri B.

Introduction

Linux does not provide a tool to restore files if they are accidentally removed from the file system. Your task is to implement a new utility, **`safeDel`**, using **bash** script programming, which simulates the removal of one or more files by moving them to the user's trash can directory. Typical usage to do this is as follows:

```
safeDel.sh file [file ...]
```

However, **`safeDel`** requires additional functionality as described in the full requirements below. **`safeDel`** must automatically create the user's trash can directory (**`~/.trashCan`**) if it does not already exist. All development will be as a **bash** script (bash version 4.3). The development environment must be **Linux Mint 17.3 64 bit.**

Coursework Requirements

`safeDel` will be **both** command line option driven and menu driven i.e. **`safeDel`** will display a menu to the user **only** if the user provides no command line options or arguments. The functionality made available to a user using command line options or the menu will be the same. In terms of the command line, the options available will be:-

- **-l** output a **list** on screen of the contents of the trashCan directory; output should be properly formatted as "file name" (without path), "size" (in bytes) and "type" for each file ("**`safeDel.sh -l`**")
- **-r** **recover** i.e. get a specified file from the trashCan directory and place it in the current directory ("**`safeDel.sh -r file`**")
- **-d** **delete** interactively the contents of the trashCan directory ("**`safeDel.sh -d`**").
- **-t** display **total** usage in bytes of the trashCan directory for the user of the trashcan ("**`safeDel.sh -t`**")
- **-m** start **monitor** script process (*see requirements below* i.e. "**`safeDel.sh -w`**")
- **-k** **kill** current user's monitor script processes (i.e. "**`safeDel.sh -k`**")

Your name and student number should be hard-coded into the safeDel script. These must be displayed once when the script starts.

A **trap** should be set up to execute before **`safeDel`** terminates on receipt of `SIGINT`. The trap should indicate the current total number of regular files in the user's trashCan directory in an appropriate message and then terminate the script.

The script should print a warning message if the disk usage in the trashCan directory exceeds 1Kbytes. This value is chosen for testing purposes.

You may assume the use of the options is mutually exclusive.

** You **must** use the **`getopts`** statement to implement the command line argument parsing; and the **`select`** statement to implement the menu. **You will fail this coursework if you just copy a solution based on another method from a book. You must also use bash**. The code should be structured using bash functions.

The program should be **fully robust**: so for each option implement a check that the arguments make sense (e.g. check that the files exist, the files are readable, that the files are not directory files as appropriate).

A Monitor script optional enhancement: this should be a separate bash script and started from the safeDel script as a separate process with the **`-m`** option. The **`-k`** option will terminate the monitor script (if it is running). The monitor script is more challenging to implement. Its purpose is, as a separately executing process ideally in a separate window, to monitor the creation, change and deletion of ordinary files in the user's trashCan directory and to update this information to the user every 15s. Your name and student number should be hard-coded into the monitor script. These must be displayed once when the script starts.

The submission instructions are in an Appendix below. It is an electronic only submission via GCULearn. Please follow the instructions on what to submit exactly or you may fail this coursework.

Testing

Testing should be documented as a table identifying test, expected result and actual result for each of the implemented options. This should be submitted as a word file. Better testing will include examples of relevant test runs i.e. screen captures etc.

Annotation

The script (and monitor enhancement) should contain comments to explain what your script does and how it works. Note that two versions should be submitted of the script(s) - one version with the comments and one without.

Linux manual page

A Linux manual page should be created for the script(s) using `groff`. This should be in the standard manual page format. There is no need to install the manual page. However the `groff` script and the resultant manual page (which should be in a form viewable in a text editor) should be submitted.

Marking Scheme:- safeDel bash script: 50%
monitor bash script (enhancement): 20%
Testing: 10%
Annotation: 10%
Linux manual page: 10%

Submission deadline: 12noon Friday 22nd February 2019 to GCULearn (end of Week 6). All students must attend their timetabled class contact session hours that week.

You are warned that any plagiarism or any failure to submit the coursework by the submission date will be dealt with according to your Student Handbook which details the relevant University Assessment Regulations (and it would be wise to re-read these sections now).

Coursework Demonstrations:-

These will take place in the timetabled class contact sessions in Week 6 (w/b 18th February 2019) You must demonstrate your code working to the demonstrator on the Linux Mint 17.3 64 bit VMWare image or on your own laptop. Failure to attend or to undertake the demonstration means a zero mark will be entered for this coursework.

Appendix: Submission Instructions:-

Follow these instructions carefully. Failure to do so will mean sections or all of the coursework marks being zeroed.

Your coursework must be submitted electronically to your drop box on GCULearn as a single 'zip' file. The *zip* file must have the specified contents below only. Do not use *rar* or any other compression format. The *zip* file must not be password protected. Do not use encryption. The zip file must be named in the form: **`<student_id>_SP_1.zip`** e.g. **`S1512345_SP_1.zip`**

1. Create a directory called
   `<Family Name>_<Given Name>_<student_id>`
   e.g. **`Hainey_Brian_S1512345`**
   Put everything in this directory and zip up the directory.
2. The zip file should ***not*** contain any subdirectories.
3. The zip file should contain the following only ***with these filenames***:-
   - safeDel script with no comments (safeDel.sh)
   - safeDel script with same code and additionally annotation as comments (safeDel-comments.sh)
   - monitor script (monitor.sh) – optional enhancement
   - monitor script with same code and additionally annotation as comments (monitor-comments.sh) – optional enhancement
   - groff manual file (monitor-manual.1)
   - manual page as a generated text file viewable in gedit (monitor-manual.txt)
   - testing document as a word file (testing.docx)

Your 'zip' file should be submitted using the link in the "Assignments" section on GCULearn. You are advised to keep more than one copy of all your files in case of loss or corruption of the originals; it is your sole responsibility to do this. Any loss of material/PC failures/not being able to access the module on GCULearn etc. will not be accepted as a valid reason for not submitting the coursework by the deadline.