CS 392 Spring 2024 Systems Programming — Homework 1

1 Objective

Your task is to write a simple bash script to provide the basic functionality of a **recycle bin**. In addition to moving files into the recycle bin directory, the script must be able to list and empty the files that have been placed into the recycle bin. This script acts as a substitute for the `rm` command, giving the user a chance to recover files deleted accidentally. Note that restoring the files is not part of this exercise.

You should consult with the following websites for details about programming in bash: http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html and http://tldp.org/LDP/abs/html/.

2 Recycle Bin

You will create a bash script called `rbin.sh`, and finish the following tasks.

2.1 Task 1: Parse Arguments

When the script is executed, it first parses the command line arguments. The usage message is shown below to help you determine what features you need to implement.

```
$ ./rbin.sh
Usage: rbin.sh [-hlp] [list of files]
   -h: Display this help;
   -l: List files in the recycle bin;
   -p: Empty all files in the recycle bin;
   [list of files] with no other flags,
        these files will be moved to the
        recycle bin.
```

Notice there are three flags `-h`, `-l`, and `-p` that the script must handle. You must use `getopts` in your solution to receive credit for this part of the assignment. When no argument is supplied or when the user passes `-h`, the script should produce the output seen in the box above.

If an unexpected flag is found, the script should display an error message and repeat the correct usage. This is true, even if the user inputs too many flags. If one is unknown, the script terminates with the message seen below.

```
$ ./rbin.sh -z
Error: Unknown option '-z'.
Usage: rbin.sh [-hlp] [list of files]
   -h: Display this help;
   -l: List files in the recycle bin;
   -p: Empty all files in the recycle bin;
   [list of files] with no other flags,
        these files will be moved to the
        recycle bin.
```

If more than one (valid) flag is specified, the script should display an error message and repeat the correct usage. See below.

```
$ ./rbin.sh -l -p
Error: Too many options enabled.
Usage: rbin.sh [-hlp] [list of files]
   -h: Display this help;
   -l: List files in the recycle bin;
   -p: Empty all files in the recycle bin;
   [list of files] with no other flags,
        these files will be moved to the
        recycle bin.
```

If one or more flags are specified and files are supplied, the script also tell the user that too many options have been supplied.

```
$ ./rbin.sh -l note.txt
Error: Too many options enabled.
Usage: rbin.sh [-hlp] [list of files]
   -h: Display this help;
   -l: List files in the recycle bin;
   -p: Empty all files in the recycle bin;
   [list of files] with no other flags,
        these files will be moved to the
        recycle bin.
```

Note, you must `exit 1` when there's an error (*e.g.,* illegal options). However, only printing help message does not count as error.

2.2 Task 2: Creating a Directory for Recycle Bin

The second task is relatively straightforward: your script must create an empty directory called `.recycle`. When you invoke the script with `-p`, the script should remove all the files and directories in `.recycle`. When you invoke the script with a list of files, the script will move all the files listed to `.recycle`.

There are some details you need to pay attention:

- If there's no `.recycle` in yoru home directory, your script must create a new one;
- The path to `.recycle` must be declared as a read-only variable, and must be used throughout the script whenever the path to `.recycle` is referred;
- It is a very bad idea to hardcode the absolute path to your home directory, and you will fail many test cases when CAs are grading. Remember, your home directory is always at `/home/<uname>/` where `<uname>` is your user name, but you must not assume it's the same `<uname>` everywhere.

2.3 Task 3: Implementing Recycle Bin Functionalities

At this point, the script is ready to do its main task. So, depending on what flag (if any) the user supplied, the script needs to either display the usage message, list the files in the recycle bin, empty the files in the recycle bin, or move the files or folders specified on the command line into the recycle bin.

If a file or folder is not found, the junk script should warn the user. If the user is attempting to junk multiple files or folders and some of them are not found, the script should warn about the missing ones and move forward

with copying the ones that exist to the recycle bin. See below for the expected output:

```
$ ./rbin.sh notfound1.txt found.txt notfound2
Warning: 'notfound1.txt' not found.
Warning: 'notfound2' not found.
```

When listing files in the recycle bin, use `ls -lAF`.

3 Tester

We provide a tester program for you to test your script. Type the following and you'll see help menu:

```
$ qemu-aarch64 -L /usr/aarch64-linux-gnu/ rbintester -h
```

By default, the tester will assume the script is in the same directory, so it will invoke `./rbin`. However, if you want to specify a path to the script, you can use `-f` flag:

```
$ qemu-aarch64 -L /usr/aarch64-linux-gnu/ rbintester -f <PATH_TO_SCRIPT> -t <TASK>
```

Note that task 2 does not have testing cases — it's integrated into task 3.

4 Grading

The homework will be graded based on a total of 100 points:

- Task 1 (20 pts): 10 test cases in total, **2** points each;
- Task 2 & 3 (80 pts): 32 test cases in total, **2.5** points each.

After accumulating points from the testing above, we will inspect your code and apply deductibles listed below. The lowest score is 0, so no negative scores.

- Task 1 (20 pts):
  - **-20:** did not use `getopts` to parse the arguments;
  - **-20:** did not use heredoc to display usage;
  - **-10:** the path to the recycle bin is not declared as a readonly variable, or is not a variable;
- Task 2 & 3 (80 pts):
  - none.
- General (only deduct once):
  - **-100:** the script has run-time error, or syntax error;
  - **-100:** the script is generated by AI, and/or through reverse engineering on the tester;
  - **-10:** no pledge and/or name in the script file.

Earlybird Extra Credit: 2% of extra credit will be given if the homework is finished two days before the deadline. For specific policy, see syllabus.

Deliverable

Submit a single bash script `rbin.sh`.