CSCD 340
Lab 7

We will develop a shell and then add specific features. The goal of this assignment is to write a shell (called *posh* which will stand for "pretty ordinary shell"). Specifically, you are to write the framework for a Linux shell using the C programming language. Once the shell is executing it will prompt the user with the "command?: " prompt. The user now may issue any command line Linux command. The basics are provided in the main from cscd340Lab6.c

Here is the general battle plan

1) In lab you wrote a forkIt function that executed a valid Linux command or did nothing if the command was invalid. Enhance the forkIt function (if you still want to call it that) to handle the &. Meaning I should be able to execute a command that will run in the background. If a command is invalid print the appropriate error message to the screen.

2) In lab you wrote a pipeIt function. You will need to enhance the pipeIt function to handle one or two pipes. You should be able to handle ls -l | sort | wc -w

3) Implement a path extension feature, e.g., `PATH=$PATH:/data/myprogs`, at the command line.

4) The `posh` shell is to read input from a file called *.poshrc* and execute each line in this file as a command. When the program is run *.poshrc* will be read. This makes the *.poshrc* file a batch file. (BASH reads from .bashrc)

5) Implement a path extension feature where the user can have predefined paths in the *.poshrc* file. For example in *.poshrc* the user has already defined the path to be `PATH=$PATH:/data/myprogs`. This will be preloaded.

6) Implement a "history" mechanism which will allow to scroll though the last *N* shell commands. Your .poshrc will contain two environment variables HISTCOUNT and HISTFILECOUNT. (BASH has HISTSIZE and HISTFILESIZE). It is up to you to determine how HISTSIZE and HISTFILESIZE are used by BASH. Your shell will history mechanism will mimic BASH including each environment variable. At a minimum you should be able to recall commands by number, for example **!513**, or **!!** to re-execute the last command. Similar to BASH the history is stored in a file called *.posh_history*.

7) Implement aliasing, i.e. define an alias and undefine (unalias) it. Alias and unalias may come from the .poshrc or from the command line.
    a. For example you should be able to make the alias LA which is ls -a
    b. Then you should be able to delete that alias
    c. The *.poshrc* should be able to have predefined aliases. For example, it would contain alias LA="ls -a"

8) Implement redirection (>). You should be able to redirect stdout.

9) Implement redirection (<). You should be able to redirect stdin

10) Implement the cd command.

NOTES:

1. Your shell has to be BASH syntax compatible. If you divert from this, your shell will not pass the input used for testing! If it works in BASH it should work in yours.

2. Design your code so it is **clean** and **modular. Don't wait to start** you won't finish if you don't chip away at this. Don't try to do everything in one sitting. Pick and choose and work on one topic.

3. Draw a design document before you start coding. You need to understand how all the pieces work together before you start coding. If you don't you will quickly start breaking things by trying to add the next item.

4. Your linked list will come in really handy, multiple times.

5. You may have very very few global variables.

6. History will act like the history command of BASH

7. You can work in teams of two if you would like. If you choose this option each team member will turn in their own copy of the shell. If you work in teams of two you will need to complete all 10 items. If you choose to work on your own you only need to complete 1, 2, 4, 6, 7, 8.

GENERAL FLOW

- The program starts up and checks for:
  - . *poshrc* – if it exists reads in the variables – file format is below – doesn't exist then default HISTCOUNT and HISTFILECOUNT are used
  - *.posh_history* is loaded if it exists
  - prompt is displayed
  - each command is handled appropriately
  - upon exit *.posh_history* is written

. poshrc file format – if exists (guaranteed to have HISTCOUNT and HISTFILECOUNT)

```
HISTCOUNT=value
HISTFILECOUNT=value

Alias1
…
AliasN

PATH=$PATH:/data/myprogs
```

TO TURN IN

- The source code of your program. Name your file that contains main cscd340lab7.c
- A Makefile so your code can easily be compiled with the target of posh
- Some output captures to illustrate you shell is working. This does not need to be extensive. We will test your code this is just to illustrate you did your own testing. Name this cscd340lab7out.txt
- A README with your name and your teammate's name. If single just your name. Also include a summary of what is working and what is not working if anything.

You will submit a zip file named your last name first letter of your first name lab7.zip. (Example steinerslab7.zip)