CS 421 – Fall 2017
Project #1

150 Points **Due: Tuesday, 10/10/17 at 11:59PM in D2L**

Create a command-interpreter or shell that will accept and execute various Linux commands.

The shell should operate in this basic way: when you type in a command (in response to its prompt), the shell creates a child process that executes the command you entered and then prompts for more user input when it has finished.

1. When your program is started up, you should display a prompt at which users can type a command. (E.g. `>>`)
    a. You should support the standard processes accessible from `/bin`.
    b. You should also be able to run other executable programs with the complete path provided.

   Note that many of these processes take arguments. You should correctly parse the process name and its arguments, then `fork` the process and use `exec*` to execute it. Your shell should wait for the process to terminate, and display the result of the execution of the process. (E.g. `>>ls –al`).

2. Allow two processes to be run in sequential order with the **;** separating the process names (and arguments, if any) in a command. Your shell should wait for the processes to terminate, and display the results of their execution. (E.g. `>>ls –al ; wc out.txt`)

3. Allow a process to be run in the background with the **&** at the end of the command. In this case, the shell will return right away and not wait for the completion of the process. (E.g. `>>ls -al &`)

4. Allow the output of one process to be piped as the input of another process. (E.g. `>>ls -al | wc`)

5. Allow the output of a process to be redirected to an output file specified by the user. (E.g. `>>ls -al > out.txt`)

6. To exit the shell, the user can type `quit`. (E.g. `>>quit`)

To keep your parsing simple, I will only test your program with exactly one instance of the "operators" **;**, **&**, **|** and **>** in any command to your shell.

As usual, your program should be *robust to user error*, *modular*, *well documented* and *cleanly designed*.

Submission

To complete this assignment, simply submit your C code files, along with a make file, zipped up in a single folder, to the D2L submission dropbox folder **Project 1** by the deadline specified.