CS 421 – Spring 2025
Project 2

150 Points **Due: Thursday, 4/10/25 at 11:59PM in Canvas**

Create a command-interpreter or shell that will accept and execute various Linux commands. *Note that your code must compile and execute without any warnings or errors on our Linux lab computers.*

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 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 &`). It is okay if the output gets "intermingled" between the two processes, and you have to press an `<enter>` to get the prompt back.

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

4. Provide a history feature that allows the last 5 commands entered to be displayed and executed as needed. As an example, assume that the history consists of the commands (from least to most recent):
   `ps, ls-l, cal, who, date`
   The user will be able to list the history by entering `hist`:
```
>>hist
5 ps
4 ls-l
3 cal
2 who
1 date
```

Your program should allow the user to *execute* a specific command by typing `r` followed by the command number. So

```
>>r 3
```

will execute the `cal` command. (Note that this execution will itself change the command history. The most recent command is now cal, not `r 3`!). You may also assume that each command is no longer than 100 characters in length.

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

As usual, your program should be *robust to user error*, *modular*, *well documented* and *cleanly designed*. I will test your code on our Linux systems.

Submission
To complete this assignment, simply submit your C code files, along with a make file, zipped up in a single folder, to the Canvas submission folder **Project 2** by the deadline specified.