Project is compiled using “clang++ CS470P1.cpp -o CS470P1” or equivalent

Discussion:

The program creates a shell that accepts user input commands and executes them. The main program holds the main run function that will run as long as the should\_run flag is set. The run function runs as expected, handles extraneous characters in the command line, calls the parsing function, builds the history container, and ends by starting the startFork function. The startFork function starts the forking process and determines if it the processes will run in the background or not, and waits for background process to finish before proceeding.

For the history feature I chose to use a vector for its versatility, and ease of element access. Recently used commands are pushed back into the vector and older functions are popped off of the back to maintain a history size of 10. !! and !# work as expected. The most recent command is at the zero index. Access to latter commands are implemented with simple if statements and provided indexes.

Sample Test Plan:

ls //shows all files in current directory

vim //opens vim editor

clear //pushes everything up and puts a blank command line at the top of the terminal

history //shows 10 most recent commands

!! //executes the most recent command

!1 //executes the most recent command as well

cal //displays a calendar

date //displays the current date

mkdir name //creates a directory with the name name

exit //ends the running shell

1. What aspect of process manipulation did you find most difficult to understand?

*The idea of forking makes sense in my head, and works how I think it should, but actually implementing it properly is tricky and I often doubt it works the way that I conceive it to.*

2. What aspect of process manipulation did you find least difficult to understand?

*Waiting; we did a lot of this in later projects of Small Computer Software. Threads and Background Workers of different priorities run off to complete tasks, but all wait until the relative functions or parent processes are ready for them.*

3. What, if anything, would you change in your current design?

*I am trying to be more and more modular in my design, but typically more errors appear the more I break things up. I would have liked to include a few more functions to handle misc. tasks.*

4. What, if anything, did you find interesting or surprising about process manipulation that you did not know before doing this project?

*That it actually works. Running my shell actually mimics the command line I am used to. I didn’t purposely seek out unusual commands seeing if it would break, but it seems to work for the commands that I use most often.*