Project 1 Midpoint Report

Ethan Tran, Daniel Eckman
9/14/2025

Project Description

Part 1 of the project is to implement a simplified version of a shell into our code. As such, the program needs to be able to parse a given string and interpret it and its arguments.

Achievements

What we accomplished in part I of the project was implement a “Param” class to handle parsed command details, implement the “parseInput” function to parse command strings and populate the “Param” object, create the “myshell” loop to read the input, parse it, and output the debug information. The libraries we used were <iostream>, <cstring>, <cstdlib>, and <string>. Issues that we encountered were coding syntax errors of sometimes missing a semicolon and misuse of characters and variables when not needed.

Preliminary Testing

A few test cases that we developed for my code were that the shell should be able to run the “ls -1” function parsing it as “-1.” Another test was “cat < input.txt > output.txt &” where it should recognize “< input.txt > output.txt &”, this would set the input to “input.txt” and the output to “output.txt” pushing the argumentCount to “1” and setting the value for backgrounding to 1. We believe the program did pass the tests correctly as there weren’t errors and the shell is correctly holding the strings and parsing them.

Next Steps

The next steps to solving the problem in the project is to update the myshell.cpp file to handle process creation. We will use the fork() command call to create new processes as children. Then, the child runs either the exec(), execl(), execlp(), execvp(), and execv() to execute the command from the user. Create input/output redirection with the “Param” object for before execution of the exec function. Processes ending in “&” should be forced to run in the background. Zombie processes will be handled by implementing a terminating process using waitpid() or wait() to clean up child processes.