Project 1 Midpoint Report

Ethan Tran
9/12/2024

Project Description

Part 1 of the project is to implement a simplified version of a shell into our code. This calls for our program to be able to parse and input string into a structure for copying a shell.

Achievements

What I 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, and create the “myshell” loop to read the input, parse it, and output the debug information. The libraries I used were <iostream>, <cstring>, <string>, <cstdlib>, <cctype>, and the <sstream> libraries. Issue that I encountered we little coding syntax error of sometimes missing a “;” and misuse of characters and variables when not needed.

Preliminary Testing

A few test cases that we develop for my code we the shell should be able to run the “ls -l” function parsing it as “-1.” Another test was “cat < input.txt” where is should recognize “<input.txt” as a redirection for the argumentCount to hold “cat” as a argument vector in the shell. Another test is “echo Hello World < input.txt > output.txt &”, this would set the input to “input.txt” and the output to “output.txt” pushing the argumentCount to “2” and holding, “echo”, “Hello”, “World” as the argumentVector for the background. I believe the program did pass the tests correctly as there weren’t error and correctly holding the strings and parsing them.

Next Steps

Describe in a paragraph the next steps to solve the problem given in the project description. Be sure you break down the description into the different parts that need to be accomplished such as how to create a new process, how to do input/output direction, handle foreground and background.

The next steps to solving the problem in the project is to update the myshell.cpp to handle process creation. We will use the fork() command call to create new processes of creating child and parent. Have the child run 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 fo the exec function. With processes ending in “&” to not let them wait on the child processes to finish and implement. Also handle zombie processes by implementing a terminating process of waitpid() or wait() to clean up child processes.