Project 1 Midpoint Report

Sarah Wallis, Dustin Evans

9/12/2024

Project Description

Part 1 focused on building the foundation of the program to parse user's input and store those into tokens. This way when moving onto part 2 we have most of the necessary functionality needed to further extend the shell to then create processes, handle the input and output, and its execution.

Achievements

We implemented the logic for prompting the user for input and parsing said input into separate arguments delimited by whitespace. This was accomplished by implementing the functionality for the provided param class template as well as a separate parse class for directly managing the input and output streams for the program. In order to ensure that the program could continue to run and take in user input, we also added the clearArguments function to the param class in order to reset instances of it for reuse. Furthermore, a destructor was added to the param class to free up memory.

Preliminary Testing

We have tested multiple cases of typical commands a shell may have such as "ls -l", "cat < input.txt", or "ls -l > output.txt". But also, the parsing of a command where someone might leave out the space before the redirect file, such as "cat <input.txt". The output in debug mode shows that our program correctly parses the input from user like their argument counts, the list of arguments, which redirects it has, and the background processes. With an input like "cat < input.txt" it would show that the input redirect was "input.txt", the output redirect is null, background is currently not applicable, and the argument count is 1 with the argument list showing "cat". This shows our shell program correctly parses the user's input.

Next Steps

The next steps include creating functions in our shell program to create processes with a user's commands. Depending on the user's command we will be able to create a new process using "fork" to then execute an input or output direction like opening a file to read or write from it, then wait for the foreground or background process to end before continuing. Once those functions are created the shell program will be able to successfully take in a user's input and then execute the command.