Ceren Dinç 28220
CS307 PA2

How the Interpreter Works

The program starts by reading each line from 'command.txt', considering each line as a separate command. For each command, it figures out the main action, any additional options, and whether the command needs to send its output to a file or run without disturbing the main program flow. This is done by tokenization, a function named parse_command_line splits each line into these parts. After breaking down the command, the program then starts a new process for each command with fork method. This way, each command runs in its own space without affecting others. The program pays special attention to special symbols like '>' and '<', which tell it to send the command's output to a file, and '&', which means the command should run in the background.

Special Features: Redirection and Background Processes

One key feature of this interpreter is its ability to handle redirection, which means sending the output of a command to a file instead of showing it on the screen. It does this by changing the standard output path to the specified file. Another important feature is its ability to run commands in the background. This is done by keeping track of these background commands in a list and making sure the main program knows about them.

Safe Operation with Threads and Pipes

The interpreter uses mutexes to make sure different parts of the program don't interfere with each other when trying to print information. It also uses pipes for commands that don't need to send their output to a file, allowing these commands to communicate efficiently.

Handling Special Characters

In commands, certain characters like '-' have special meanings and need careful handling. The interpreter identifies these characters as part of the command options and handles them.