Improved Unix/Linux Command Line Interpreter

Nolan Tuttle & Mathew Hobson

CST-315

Grand Canyon University

2.23.2025

**Updated Description of Project 2** The previous project involved creating a basic Unix/Linux command-line interpreter capable of processing user input and executing commands. The interpreter supported batch file execution and single command execution. However, it lacked advanced features such as directory navigation (cd command), piped command execution, and background process handling.

**Methodology/Approach**

1. **Refactored Code Structure:** Improved modularity and readability by refactoring functions for command parsing and execution.
2. **New Features:**
   * **Change Directory (cd Command Support)**: Implemented the ability to change directories.
   * **Piped Commands (|)**: Implemented to allow multiple commands to pass output between each other.
   * **Background Processes (&)**: Implemented to allow execution of commands in the background.
3. **Error Handling Improvements:**
   * Added robust error handling for command execution failures.
   * Improved input validation to handle edge cases.
4. **Concurrency Handling:**
   * Implemented support for running multiple commands simultaneously.

**Improved Algorithm for Parsing and Processing Shell Commands**

1. Read user input.
2. Remove trailing newline characters.
3. Check for special characters (cd for directory changes, | for pipes, & for background execution).
4. Tokenize the input based on space and semicolon (;).
5. Handle cd command by invoking chdir().
6. Handle piped commands using pipe() and fork().
7. Handle background commands by using fork() without waitpid().
8. Execute commands using execvp() instead of system() for better security and efficiency.
9. Repeat until user exits the shell.

**Key Code-Related Explanations**

**Handling Directory Change (cd command):** if (command[0] == 'c' && command[1] == 'd') {

char \*temp = strtok(command, " ");

temp = strtok(NULL, " ");

if (chdir(temp) != 0) {

perror("chdir failed");

}

continue;

}

* + chdir(temp) changes the working directory.
  + perror() prints an error message if the directory change fails.

**Handling Piped Commands:** int pipefd[2];

pid\_t pid;

if (pipe(pipefd) == -1) {

perror("pipe failed");

exit(EXIT\_FAILURE);

}

pid = fork();

if (pid == 0) { // Child process

dup2(pipefd[1], STDOUT\_FILENO);

close(pipefd[0]);

execlp("ls", "ls", NULL);

}

* + pipe() creates a pipe between two processes.
  + dup2() redirects output to the pipe.

**Handling Background Processes:** pid\_t pid = fork();

if (pid == 0) {

setsid();

execlp("sleep", "sleep", "10", NULL);

exit(EXIT\_SUCCESS);

}

* + fork() creates a new process.
  + setsid() allows it to run independently of the terminal.