Name: Darshit Bimal Gandhi

Roll No: 22M0824

CS744: Design and Engineering of Computing System

Project Title: Implementation of Simple Shell using System calls

Overview

This C-written custom shell application offers a simple command-line interface like Unix along with several other sophisticated capabilities. It mirrors the capabilities of the Linux shell, guarantees that all instructions are executed as child processes of the shell and enables users to handle signal interruptions, control background processes, and execute advanced shell commands like Piping and I/O Redirection.

Features

- Shell supports all standard shell commands like `exit`, `pwd`, `clear`, `cd`, `echo`, `top`, `ps`, `grep`, `wc`, `cat`, etc.

- Commands can be executed in the background by appending an & at the end of the command. Shell supports simultaneous execution of background commands.
  - Example: sleep 50 &

- Piping is implemented (`<cmd1> | <cmd2>`) via `pipe` and `dup2` syscalls. Multiple piping is allowed.
  - Example: command 1 | command 2 | command 3

- `I/O Redirection` is also supported. You can overwrite a file, append to a file, or give input to a shell command from a file.
  - Example:
```
command 1 > file1
command 2 >> file2
command 3 < file3
```

- Shell handles the CTRL+C shortcut to kill long-running commands (the shell is not exited).

- Program invocation is done by forking and child processes.

Source Code Structure

- my_shell.c: Main source file containing the shell implementation.
- utils/commands.h: Header file for command execution functions.
- utils/signalHandler.h: Header file for handling interrupts.
- utils/tokenizer.h: Header file for tokenization of the given command.
- utils/vars.h: Header file containing the global variables.

Compilation and Usage

Recommended operating system for running this program is Ubuntu.

1. To compile and run the program, execute the following in the directory that contains the my_shell.c file:

   ```
   make
   ```

   ```
   ./shell
   ```

2. Executing commands in foreground:

   ```
   > ls -al
   ```

3. Executing command in the background:

   ```
   > sleep 50 &
   ```

4. Execute some commands with pipe:

   ```
   > cat my_shell.c | grep "main" | wc -l
   ```

5. Perform I/O Redirection:

   - Overwrite a file with ">" `echo "Hello World" > test.txt`
   - Append to a file with ">>" `echo "Hello" >> test.txt`
   - Input redirection with "<" `cat < test.txt`

6. Terminate a long-running process running in the foreground with Ctrl + C

7. Exit the custom shell and kill all background processes:

```
> exit
```

8. Clean Object files:

```
make clean
```