SimpleShell

SOLO-MEMBER GROUP, ID- 111

Tarandeep Singh

2023553

Overview

SimpleShell is a lightweight command-line shell implementation in C that allows users to execute commands, including those with multiple pipes and background processes. This shell tracks command execution history, including the command, process ID, start time, and duration.

Features

Command Execution: Execute single or piped commands.

History Tracking: Keep track of executed commands with details such as PID, start time, and duration.

Background Processing: Run commands in the background using the `&` symbol.

Graceful Termination: Exit the shell while displaying the command history.

Prerequisites

- A C compiler (e.g., `gcc`)

- A Unix-based environment (Linux, macOS, etc.)

Compilation

To compile the SimpleShell, use the following command:

gcc -o simpleshell simpleshell.c

Usage

1. Run the Shell:

   Start the shell by executing the compiled binary:

./simpleshell

2. Basic Commands:

You can execute commands just like in a standard shell.

3. Piped Commands:

To execute multiple commands in a pipeline, separate them with the `|` character. For example:

ls -l | grep ".c" | wc -l

This command lists files in long format, filters for `.c` files, and counts them.

4. Background Processes:

To run a command in the background, append `&` to the command:

sleep 10 &

The shell will print the PID of the background process.

5. **Command History:**

To view the history of executed commands, simply type:

history

This will display a list of all commands executed in the current session along with their details.

6. Exiting the Shell

To exit the shell gracefully, press `Ctrl+C`. This will display the command history before exiting.

Core Components

- Command History Struct: The `CommandHistory` struct is used to store details about each executed command, including:

  - Command string

  - Process ID (PID)

  - Start time

  - Duration of execution

- Functions:

  - `add_to_history()`: Adds executed command details to the history.

  - `show_history()`: Displays the command history.

  - `tokenize_command()`: Splits a command string into tokens for execution.

  - `exec_single_cmd()`: Executes a single command without waiting for input/output redirection.

  - `execute_piped_commands()`: Manages the execution of multiple piped commands, including setting up pipes and redirecting input/output.

  - `parse_and_execute_command()`: Parses a command string to determine if it should execute a single command or multiple piped commands.

Signal Handling

The shell uses the `signal()` function to handle graceful termination, allowing it to cleanly exit and show the command history when the user presses `Ctrl+C`.

LIMITATIONS

No Input/Output Redirection

Description: Input and output redirection using <, >, and >> is not supported.

Reason: Handling redirection requires additional logic to parse the command and set up file descriptors correctly, which is not implemented in the current version.

No Built-in Command Support(cd, exit, set etc.)

Description: SimpleShell does not natively support built-in commands such as cd, exit, or set.

Reason: The shell is designed primarily to execute external commands through execvp(). Built-in commands require specific handling within the shell process, which is not currently implemented.

Lack of Command Line Editing:

Description: The SimpleShell implementation does not support **cursor movement** using the arrow keys (e.g., **left** and **right** keys). In more advanced shells, users can use these keys to move the cursor within the current command line to edit parts of it.

Reason: This feature requires integration with more sophisticated libraries (such as **GNU Readline**) for handling input.

□