SimpleShell: A Unix Shell in C from Scratch

Contributions

Samyak Mehta

- Creating child processes to execute commands.
- Implemented history command.
- Implemented pipe
- Support “&” for background processes in SimpleShell

Nimit Panwar

- User Input, parsing in a do while loop
- Exit using Ctrl+C
- Created rs to execute the commands from inside a Shell Script (by reading that file)
- Implemented cd command
- Authored Design Document.

Implementation

NOTE -

- To run the history command type 'history'
- To Read from shell-script type 'rs '
- All the commands from the linux introductory session are working.

The list of working commands are - ls, pwd, mkdir, cd, cp, mv, rm, uname, whoami, cat, clear, ps, man, grep, echo, sort, whereis

1. The code includes various C libraries for handling signals, processes, and user input.
2. It defines a structure called `cmnd_Elt` to store information about executed commands, such as the command itself, process ID (PID), start time, and execution time.
3. An array of `cmnd_Elt` structures called `cmnd_Array` is created to store command history, and a counter called `cmnd_count` keeps track of the number of commands executed.
4. There's a flag `ctrl_clicked` to indicate if the Ctrl+C signal (SIGINT) was received.
5. A signal handler function called `my_handler` is defined to handle the Ctrl+C signal. It sets the `ctrl_clicked` flag to 1 when the signal is received.

6. The `read_user_input` function reads user input from the command line, splits it into a command and arguments, and stores them in appropriate variables. If Ctrl+C is detected during input, it returns a special value.
7. The `create_process_and_run` function is responsible for creating child processes to execute commands. It uses `fork` to create a new process and `execvp` to replace the child process with the desired command. It also measures the execution time of the command.
8. The `launch` function is a wrapper that calls `create_process_and_run` to execute a command.
9. There's a `cd_Func` function to change the current directory using the `chdir` system call.
10. The `print_History` function prints the command history stored in `cmnd_Array`.
11. The `print_On_Exit` function prints a summary of executed commands, including PID, start time, and execution time.
12. A function named `executeShellScript` reads and executes commands from a script file.
13. The main function sets up a signal handler for Ctrl+C (SIGINT) and enters a shell loop.
14. Inside the shell loop, it continuously prompts the user for input, handles special commands like "history," "cd," and "rs" (for running scripts), and executes other commands using `launch`.
15. The loop continues until the user decides to exit by pressing Ctrl+C.

Link to Private github repository :

https://github.com/nejim3h/OS-assignments