Simple Shell in C

a UNIX shell in C from scratch

how to run

1. make sure you are in the `SimpleShell` directory
2. compile the shell by running the makefile

```
make
```

3. then simply run the shell by executing

```
./shell
```

4. the prompt will be displayed

```
ishaan&&divyansh@SimpleShell:~$
```

5. execute any commands!
6. to exit, type `exit` or press `[CTRL-C]`

sample files fib, helloworld and a commands.sh containing a list of commands are given for testing purposes.

Implementation

The entire code is written in the C Language, in the file `simpleshell.c`.

1. Shell User Interface The The user is continuously prompted with a custom prompt (`ishaan&&divyansh@SimpleShell:~$`) and `read_command()` method is used by the shell to read input by the user. One can execute commands directly, `cd` to change directories, support pipes (`|`), or use `history` to view the command history. The `history_entries` array, which records the command, process ID (PID), start time, and execution duration, is where input commands are stored.
2. Management of History Up to 100 command history entries can be stored by the shell. The command, PID, start time, and execution duration are all included in the history. The history command outputs every command that has ever been run. The shell prints all of the commands in the history, together with their PIDs and runtime details, when you hit `[CTRL+C]`.
3. Implementation of Piping Pipelining (`|`) is supported by the shell, which enables the execution of several commands by passing the output of one command as an input to the next. The `execute_pipe()` function breaks the command by utilizing pipes, and it uses child processes to run them sequentially.
4. Command Execution (including the application of BONUS `&`) Using `execl()`, commands are carried out by forking a child process. By adding `&` to commands, background processes can be managed without obstructing the shell. The command history contains information about the runtime, PID, and execution state.
5. Managing Signals Signal handling for `SIGINT` (`CTRL+C`) is included in the shell. This signal causes the shell to interrupt, print the history of commands, and then exit.

Limitations

1. we have explicitly handled `&` and pipe commands with `execvp`. attempting to run built-in shell commands in background or in pipes would lead to something like an execvp error and might print unexpected output. e.g. `history | wc` will not work
2. commands containing both `&` as well as `|` won't work because handling such a command will get very complicated and out of our scope.

Link to repo

Contribution

Both members contributed to the assignment equally and with passion. We both worked on coding the assignment together at all times including the brainstorming, writing the code and debugging it.

Contributors

- Divyansh Kumar Gautam (2023208)
- Ishaan Raj (2023248)