COSE322 System Programming Assignment #3

Instructions

- **Deadline: December 19th, 2024, by 11:59 PM.**
- Carefully read the problem description and write the corresponding code.
- Create a Cargo package for the problem, with the following package name:
  - Problem_1
- Write a report explaining how you solved the problem. The report is limited to four A4 pages, and it should include the following contents.
  - **Implementation method**: outline specific implementation methods to meet the requirements. No need to describe source code.
  - **Demo examples**: show examples of commands that work in your shell.
  - **Any known limitations or issues:** if you've implemented everything perfectly, you don't need to write it down.
- Save the report as a PDF file, named in the following format:
  - Report_[Your student ID].pdf (e.g., **Report_2024210046.pdf**)
- Compress your Cargo package and report into a single .zip file and submit it on Blackboard. The .zip file should be named as follows:
  - Assignment3_[Your student ID].zip (e.g., Assignment3_ 2024210046.zip)
- In summary, your .zip file should include the following file tree structure

```
Assignment3_2024210046/
├── Report_2024210046.pdf
└── Problem_1/
```

- **Important:** If the submission format (file name, type, compression, etc.) is not followed, points will be deducted without exception

- **Assumption:** You may assume that your shell program always receives valid input. Valid input refers to commands that operate without errors in Linux environments. In other words, you do not need to handle errors for invalid commands rigorously.

Problem 1. Oh My Shell

A **shell** is a command-line interface (CLI) that provides an environment for users to interact with the operating system. It allows users to execute commands, run scripts, and manage system resources. The shell acts as a command interpreter, translating user input (commands) into instructions for the operating system. Shells also provide advanced features like piping, input/output redirection, and scripting capabilities, making them essential tools for developers, system administrators, and power users.

Objective

In this assignment, you will implement a simplified shell program in Rust. The shell should support basic Linux command execution, input/output redirection, and pipes.

Requirements

Your shell program, named ***oh-my-shell***, must meet the following criteria:

1. **Prompt display**
   - Display a shell prompt (`>>>`) indicating that oh-my-shell is waiting for input.

```
sangeunoh@desktop % cargo run
######### oh-my-shell starts! #########
>>>
```

2. **Command execution**
   - The shell should be able to execute basic Linux commands like `ls`, `grep`, `echo`, etc.
   - The shell should be able to execute any user programs, too.
   - Use Rust's nix crate to handle process creation (`fork`) and execution (`execvp`).
   - When a child process is terminated, display its PID and exit status.
   - After completing a user command, the shell must be ready to accept input again.

```
######### oh-my-shell starts! #########
>>> ls -l
total 32
-rw-r--r--@ 1 ohsang1213  staff  1202 11 28 11:04 Cargo.lock
-rw-r--r--@ 1 ohsang1213  staff   136 11 28 11:05 Cargo.toml
-rw-r--r--@ 1 ohsang1213  staff    14 11 28 22:14 input.txt
-rw-r--r--@ 1 ohsang1213  staff    16 11 28 22:28 output.txt
drwxr-xr-x@ 3 ohsang1213  staff    96 11 28 23:14 src
drwxr-xr-x@ 5 ohsang1213  staff   160 11 28 11:04 target
[oh-my-shell] Child process terminated: pid 35130, status 0

>>> ../summer/target/debug/summer
?# 4
sum=4
?# 1
sum=5
?# 7
sum=12
?# 0
[oh-my-shell] Child process terminated: pid 35133, status 0

>>>
```

3. Input Redirection (<)

- The shell should support reading input for commands from a file.
- Use Rust's nix crate to handle input redirection (`dup2`).
  - The `dup2()` system call duplicates a file descriptor to a specified target file descriptor.
  - For more information, refer to the following links
    `nix` documentation: https://docs.rs/nix/0.29.0/nix/unistd/fn.dup2.html
    Linux manual page: https://man7.org/linux/man-pages/man2/dup.2.html

```
>>> cat < input.txt
Hello, world!
[oh-my-shell] Child process terminated: pid 31983, status 0

>>>
```

4. Output Redirection (>)

- The shell should support writing command output to a file.
- Use Rust's nix crate to handle output redirection (`dup2`).

```
>>> echo "Hello, World!" > output.txt
[oh-my-shell] Child process terminated: pid 32315, status 0

>>>
```

5. Pipe (|)

- The shell should support piping the output of one command as the input to another.
- The shell should wait for the termination of all children processes.
- Use Rust's `nix` crate to handle a pipe (`pipe` and `dup2`).
  - The `pipe()` system call creates a unidirectional data channel for IPC that allows processes to exchange data.
  - For more information, refer to the following links
    `nix` documentation: https://docs.rs/nix/0.29.0/nix/unistd/fn.pipe.html
    Linux manual page: https://man7.org/linux/man-pages/man2/pipe.2.html

```
>>> ls -l | grep txt
-rw-r--r--@ 1 ohsang1213  staff    14 11 28 22:14 input.txt
-rw-r--r--@ 1 ohsang1213  staff    16 11 28 22:28 output.txt
[oh-my-shell] Child process terminated: pid 35257, status 0
[oh-my-shell] Child process terminated: pid 35258, status 0

>>> ../summer/target/debug/summer | ../bingo/target/debug/bingo
4
1
Bingo!
7
0
[oh-my-shell] Child process terminated: pid 35259, status 0
[oh-my-shell] Child process terminated: pid 35260, status 0

>>>
```

6. Pipe chaining

- Support multiple piping redirections in a single command.

```
>>> ls -l | grep txt | wc -l
      2
[oh-my-shell] Child process terminated: pid 35257, status 0
[oh-my-shell] Child process terminated: pid 35258, status 0
[oh-my-shell] Child process terminated: pid 35259, status 0

>>> ../summer/target/debug/summer < input.txt | ../bingo/target/debug/bingo > output.txt
[oh-my-shell] Child process terminated: pid 35262, status 0
[oh-my-shell] Child process terminated: pid 35263, status 0
[oh-my-shell] Child process terminated: pid 35264, status 0

>>> cat output.txt
Bingo!
[oh-my-shell] Child process terminated: pid 35265, status 0

>>>
```

7. Shell exit

- Terminate oh-my-shell whey you type the command “exit”.

```
>>> exit
Exit oh-my-shell. Bye!
```