System Programing Assignment #3

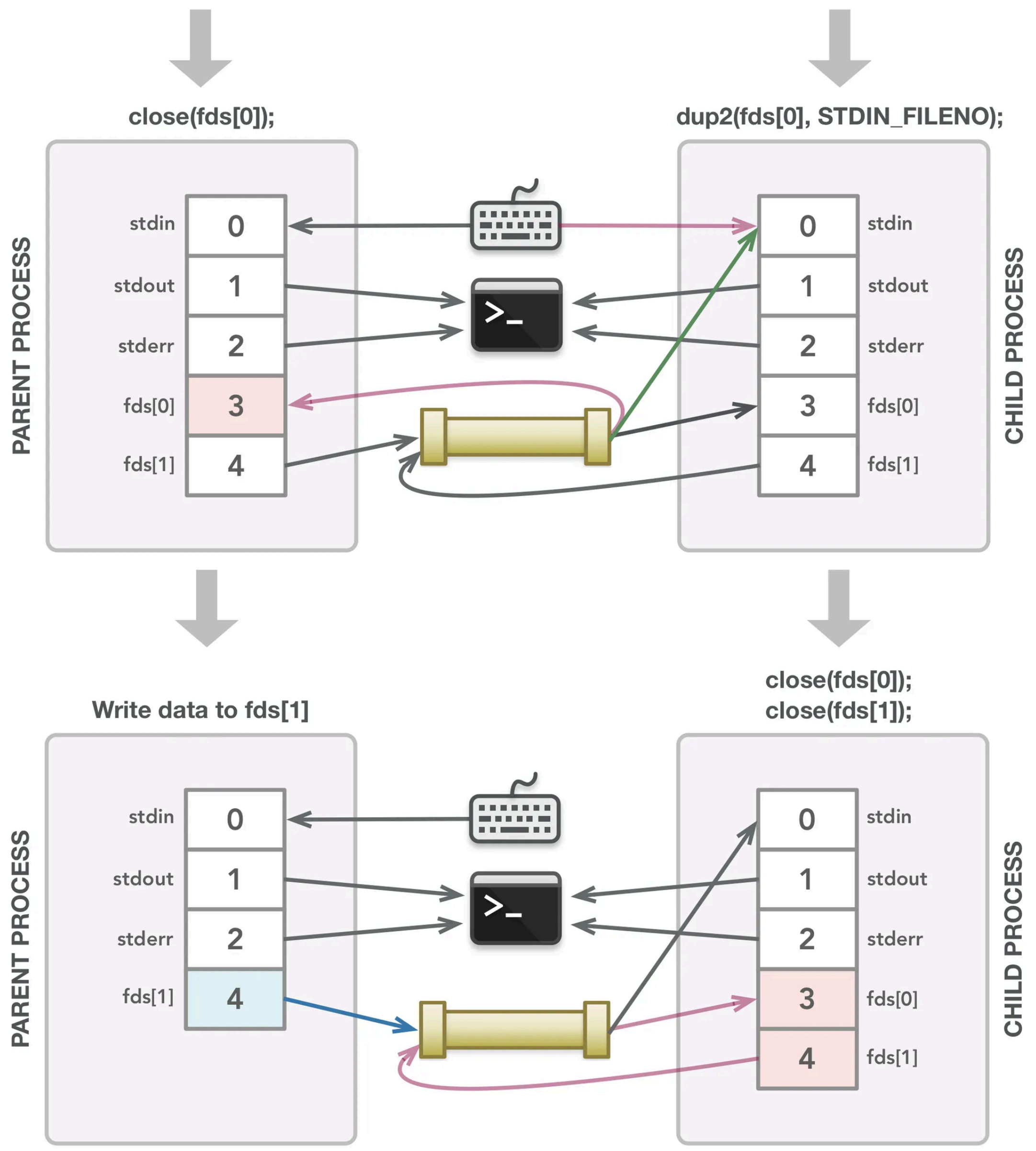
2022320039 컴퓨터학과 정진욱

Problem 1. Oh My Shell

Objectives

The goal was to implement a simple shell capable of executing commands with support for **piping**, **input/output redirection**, and process management. This shell adheres to UNIX principles and interacts directly with system calls for process creation and I/O redirection.

Understanding dup2



The `pipe()` call finds the next two available file descriptors and associates each with the appropriate end of the created pipe. In this case, a process can read 3 and write 4. The `fork()` call creates the child process, which is a copy of the parent's memory and file descriptor table at that point in time. The child calls `dup2()` to make its `stdin` be a copy of fds[0]. Now the child's `stdin` is connected to the read end of the pipe. This allows the child process to receive data as `stdin` when the parent writes to the write end of the pipe.

Implementation Strategy

To implement the shell, the input command is processed in the following sequence. First, the input command is split into segments based on pipe (`|`) operators. Then, pipes required for communication between child processes are created in advance. Each pipe-separated command is executed in a child process created using `fork`. Redirection is handled within each child process, and the Linux command is executed using `execvp()`.

The child process writes the result of the executed command to the pipe. The next child process reads the result from the previous pipe and writes its output to the pipe again. This process is repeated, enabling communication between child processes.

Key Features

`fn handle_piping`:

1. Split command into segments based on pipe (`|`) operators.
2. Create pipes for communication between child processes.
3. loop
    a. Create child process using `fork()`
    b. Child process:
        - Connect previous pipe's read end to STDIN `dup2(pipes[i - 1].0, 0);`
        - Connect current pipe's write end to STDOUT `dup2(pipes[i].1, 1);`
        - Close all previous and current pipe ends.
        - Execute command (Pass command to `fn handle_redirection`)
    c. Parent process:
        - Close all previous pipe ends.
4. When all child process is terminated, display its PID and exit status.

`fn handle_redirection`:

If command has redirection operators (`<`, `>`), redirect STDIN or STDOUT to the file.

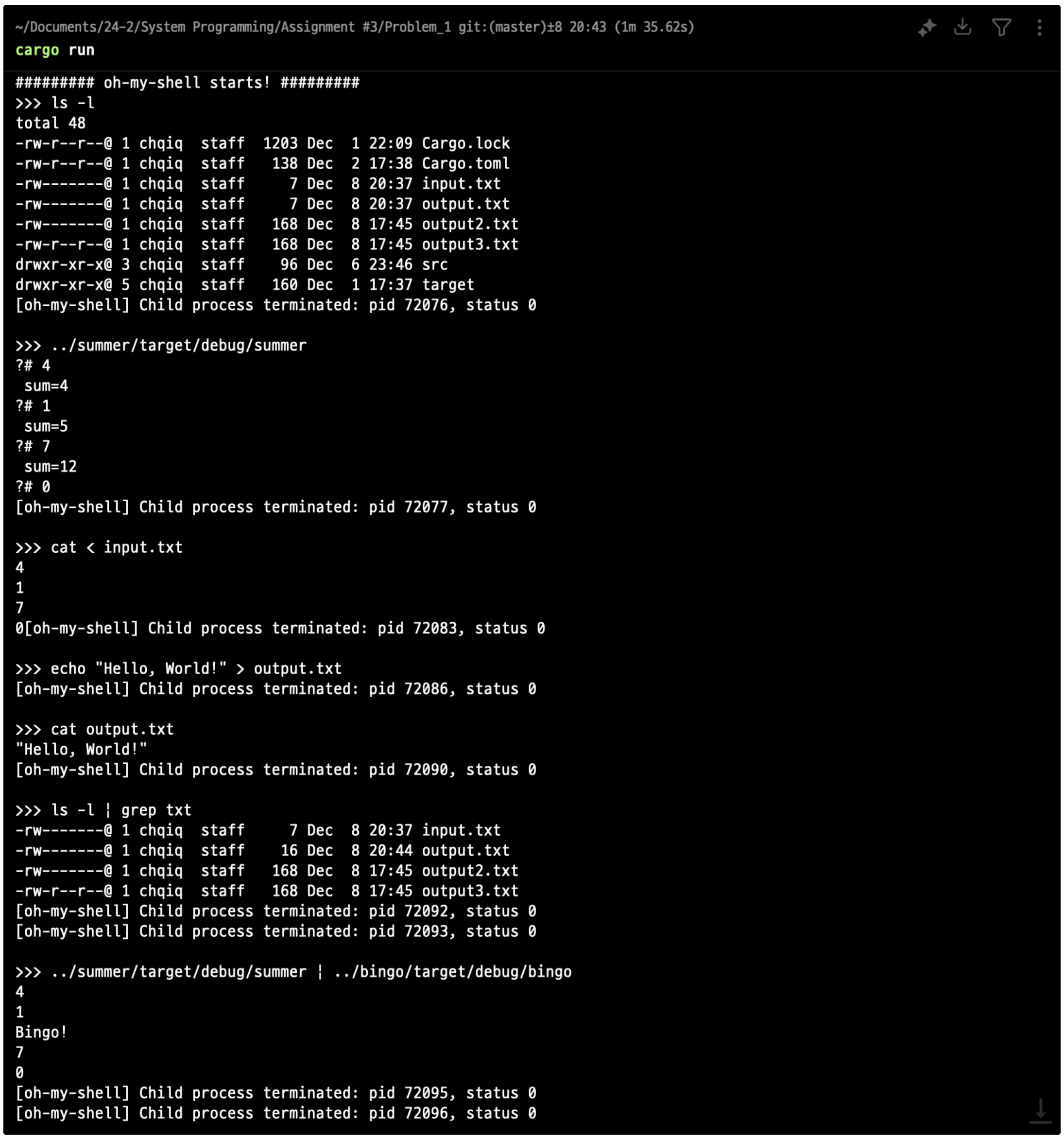
`fn exec_command`:

execute command using `execvp`.

Any known limitations or issues

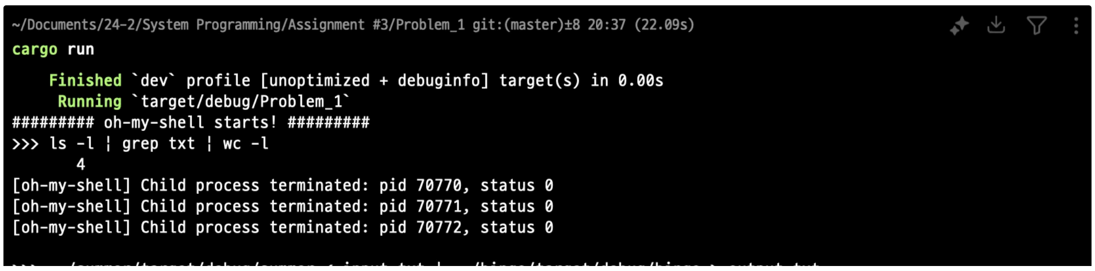
- To ensure that the execution results of commands are displayed in order, the parent process waits specifically for the last command's child process to terminate and stores its output to display it at the end. I am curious if this approach can be further improved.

Demo Examples



```
~/Documents/24-2/System Programming/Assignment #3/Problem_1 git:(master)±8 20:43 (1m 35.62s)
cargo run
```

```
######### oh-my-shell starts! #########
>>> ls -l
total 48
-rw-r--r--@ 1 chqiq  staff  1203 Dec  1 22:09 Cargo.lock
-rw-r--r--@ 1 chqiq  staff   138 Dec  2 17:38 Cargo.toml
-rw-------@ 1 chqiq  staff     7 Dec  8 20:37 input.txt
-rw-------@ 1 chqiq  staff     7 Dec  8 20:37 output.txt
-rw-------@ 1 chqiq  staff   168 Dec  8 17:45 output2.txt
-rw-r--r--@ 1 chqiq  staff   168 Dec  8 17:45 output3.txt
drwxr-xr-x@ 3 chqiq  staff    96 Dec  6 23:46 src
drwxr-xr-x@ 5 chqiq  staff   160 Dec  1 17:37 target
[oh-my-shell] Child process terminated: pid 72076, status 0

>>> ../summer/target/debug/summer
?# 4
 sum=4
?# 1
 sum=5
?# 7
 sum=12
?# 0
[oh-my-shell] Child process terminated: pid 72077, status 0

>>> cat < input.txt
4
1
7
0[oh-my-shell] Child process terminated: pid 72083, status 0

>>> echo "Hello, World!" > output.txt
[oh-my-shell] Child process terminated: pid 72086, status 0

>>> cat output.txt
"Hello, World!"
[oh-my-shell] Child process terminated: pid 72090, status 0

>>> ls -l | grep txt
-rw-------@ 1 chqiq  staff     7 Dec  8 20:37 input.txt
-rw-------@ 1 chqiq  staff    16 Dec  8 20:44 output.txt
-rw-------@ 1 chqiq  staff   168 Dec  8 17:45 output2.txt
-rw-r--r--@ 1 chqiq  staff   168 Dec  8 17:45 output3.txt
[oh-my-shell] Child process terminated: pid 72092, status 0
[oh-my-shell] Child process terminated: pid 72093, status 0

>>> ../summer/target/debug/summer | ../bingo/target/debug/bingo
4
1
Bingo!
7
0
[oh-my-shell] Child process terminated: pid 72095, status 0
[oh-my-shell] Child process terminated: pid 72096, status 0
```



```
~/Documents/24-2/System Programming/Assignment #3/Problem_1 git:(master)±8 20:37 (22.09s)
cargo run
```

```
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.00s
     Running `target/debug/Problem_1`
######### oh-my-shell starts! #########
>>> ls -l | grep txt | wc -l
       4
[oh-my-shell] Child process terminated: pid 70770, status 0
[oh-my-shell] Child process terminated: pid 70771, status 0
[oh-my-shell] Child process terminated: pid 70772, status 0

>>> ../summer/target/debug/summer < input.txt | ../bingo/target/debug/bingo > output.txt
```