

# CSE 344 HW2 REPORT

Bahadir Etkä Kilinc

1901042701

The program is a simple shell that allows the user to execute multiple commands in a pipeline, separated by the "|" character. The shell supports a maximum of 20 commands in a pipeline, and it logs the commands and their respective process IDs (PIDs) to a file with a timestamped filename.

The `log_commands` function takes an array of command strings, an array of PIDs, and the number of commands as input, and logs them to a file. The function first opens the file with the specified filename in write mode. Then, it iterates over the commands and PIDs arrays and writes each entry to the file using the `fprintf` function. Finally, the function closes the file.

The `timestamp` function returns the current time in the format "YYYY-MM-DD\_HH-MM-SS" as a dynamically allocated string. The function uses the `localtime` and `strftime` functions from the standard C library to format the current time and returns a duplicate of the formatted string using the `strdup` function.

The `execute_command` function takes a command string, an input file descriptor, and an output file descriptor as input, and executes the command using the `execl` function. If the input or output file descriptors are not the standard input or output file descriptors, respectively, the function redirects the standard input or output to the specified file descriptors using the `dup2` function before closing them. If the `execl` function fails, the function prints an error message using the `perror` function and exits with an error code.

The `run_commands` function takes a pipeline string as input and executes the pipeline by forking a child process for each command in the pipeline. The function first parses the pipeline string into an array of command strings using the `strtok` function. Then, it iterates over the command strings and forks a child process for each command. For each child process, the function calls the `execute_command` function with the command string, the input file descriptor of the previous command's output (if any), and the output file descriptor of the next command's input (if any). The function also maintains an array of PIDs for each child process and closes any file descriptors that are not the standard input or output file descriptors. Finally, the function waits for all child processes to terminate using the `wait` function and logs the commands and PIDs to a file using the `log_commands` function with a timestamped filename.

The main function takes no command line arguments and reads input from the user using the `fgets` function. It repeatedly prompts the user for input with the ">" prompt until the user enters the ":q" command to exit the shell. Otherwise, it calls the `run_commands` function with the user's input string.

## Test Cases

Single command with no arguments

```
bek_shell> pwd
/Users/bahadiretka/Desktop/hw2
bek_shell> █
```

Single command with arguments

```
bek_shell> ls -l /tmp
lrwxr-xr-x@ 1 root  wheel  11 Apr  1 19:46 /tmp -> private/tmp
bek_shell> █
```

Two commands with pipe, with arguments

```
bek_shell> ps aux | grep root
root      881  0.8  0.4 410767712 31088  ??  Ss   Wed09AM 10:12.85 /Syst
/A/Support/mds_stores
root      518  0.7  0.2 408527840 18576  ??  Ss   Wed09AM  7:50.85 /Syst
mds
root      569  0.6  0.1 408277440 10224  ??  Ss   Wed09AM  3:15.43 /usr/
root       1  0.2  0.2 409220000 15824  ??  Ss   Wed09AM 16:34.71 /sbin
root      514  0.1  0.0 408069232  2128  ??  Ss   Wed09AM  0:04.45 /usr/
root      531  0.1  0.1 408270448 10320  ??  Ss   Wed09AM  2:27.69 /Syst
root      491  0.1  0.2 408343968 16384  ??  Ss   Wed09AM  6:19.64 /usr/
root    35597  0.0  0.1 408227072  4304  ??  Ss    6:16AM  0:00.10 /usr/
root    35560  0.0  0.0 408197744  2960  ??  Ss    6:16AM  0:00.06 /usr/
root    35550  0.0  0.0 408226912  3328  ??  S    6:16AM  0:00.06 /usr/
```

Command with input redirection, with arguments

```
1  PID: 37652, Command: grep 'hello' < input.txt
2

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

bek_shell> grep 'hello' < input.txt
bek_shell> █
```

Command with input redirection

```
1  PID: 37494, Command: cat < input.txt
2

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

bek_shell> cat < input.txt
bek_shell> █
```

Command with output redirection, with arguments

```
≡ output.txt
1 total 162896
2 -rwxr-xr-x 1 root wheel 204592
3 -rwxr-xr-x 1 root wheel 227440
4 -rwxr-xr-x 1 root wheel 205696
5 -rwxr-xr-x 76 root wheel 167136
6 -rwxr-xr-x 76 root wheel 167136
7 -rwxr-xr-x 1 root wheel 256544
8 -rwxr-xr-x 1 root wheel 161008
9 -rwxr-xr-x 1 root wheel 203040
10 -rwxr-xr-x 76 root wheel 167136
11 -rwxr-xr-x 76 root wheel 167136
12 -rwxr-xr-x 1 root wheel 168800
13 -rwxr-xr-x 76 root wheel 167136
14 -rwxr-xr-x 76 root wheel 167136
15 -rwxr-xr-x 1 root wheel 344960
16 -rwxr-xr-x 16 root wheel 167088

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bek_shell> ls -l /usr/bin > output.txt
bek_shell>
```

Command with both input and output redirection, with arguments

```
≡ input.txt
1 12
2 45
3 23
4 59
5 90
6 14
7 85

≡ output.txt
1 12
2 14
3 23
4 45
5 59
6 85
7 90
8

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bek_shell> sort < input.txt > output.txt
bek_shell>
```

## Multiple commands with pipes and input/output redirection, with arguments

```
≡ input.txt
1  apple
2  apple
3  apple
4  apple
5

≡ output.txt
1  apple
2

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

bek_shell> grep -i 'apple' < input.txt | sort | uniq > output.txt
bek_shell>
```

## Extras

```
bek_shell> ls | grep hw2
hw2
hw2.c
hw2.dSYM
bek_shell>

≡ input.txt
1  bahadir etka kilinc

≡ output.txt
1  bahadir etka kilinc

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

bek_shell> cat input.txt > output.txt
bek_shell>

≡ input.txt
1  12
2  45
3  23
4  59
5  90
6  14
7  85

PROBLEMS  OUTPUT  DEBUG CONSOLE

bek_shell> sort < input.txt
12
14
23
45
59
85
90
bek_shell>
```