CSE344 SYSTEM PROGRAMMING

HOMEWORK 2 REPORT

Ayşe Begüm NUR – 1901042613

HOMEWORK OVERVIEW:

For this homework, we were asked to implement a shell that can handle multiple commands separated by the pipe ("|") symbol. The shell can also handle input and output redirection using the "<" and ">" symbols respectively. Every command in the given line is to be executed by a new child process while the parent process sits in a tight loop waiting. The shell also has signal handlers for the SIGINT and SIGTERM signals that kills any child process and frees up the memory.

BRIEF EXPLANATION OF THE FUNCTIONS:

I have implemented functions for getting a line of commands from stdin, parsing the given line of commands with pipes into smaller commands, executing every child process and handling the SIGINT and SIGTERM signals.

1 – void SignalHandler(int signal) - This is a signal handler function that handles the **SIGINT** and **SIGTERM** signals. If any of these signals are received, the **RECEIVED_SIGNAL** flag is set.

2 – void log_child_proccesses(pid_t pids[], char* commands [MAX_ARGS] [MAX_ARGS], int num_commands) - This function logs the child processes to a single file after every given line. It generates a timestamp for the log filename and opens the file for writing. It then writes the child process information (PID and command) to the file for each child process.

3 – void free_memory(char* commands [MAX_ARGS] [MAX_ARGS], int num_commands) - This function frees the allocated memory for the commands.

4 – int split_into_commands(char* line, char* command [MAX_ARGS]) - This function splits the given input into commands. It takes a line of input and splits into commands separated by the "|" symbols. The commands are stored in the command array.

5 – void execute_command(char* command [MAX_ARGS]) - This function executes a single command and handles redirection. It takes a command as input and redirects input and/or output if the "<" or ">" symbols are present in the command.

6 – void execute_pipeline(char* command [MAX_ARGS] [MAX_ARGS], int num_pipes) - This function executes multiple commands separated by the "|" symbol. It takes a 2D array of

commands and the number of pipes as input. It creates pipes for communication between the child processes and forks child processes. Each child process is responsible for executing a single command. The output of each chil process is passed to input of the next child via the pipes.

EXAMPLE RUNS:

Compile and print the instructions.

```
begum@begum:~/Desktop$ make
gcc -Wall -Wextra  -c shell.c
gcc -Wall -Wextra  -o shell shell.o
begum@begum:~/Desktop$ ./shell
----- SHELL EMULATOR -----
This shell can support bin/sh commands with pipes and redirection.
Please type "[command] --help" to learn how to use a specific command.
Please type ":q" to exit the program.
>ls
 Ayse_Begum_Nur_PA1.cpp   pa3.cpp
 b.txt                   'Screenshot from 2023-04-14 07-39-44.png'
 CSE344                  'Screenshot from 2023-04-14 07-47-25.png'
 exam.txt                'Screenshot from 2023-04-14 07-50-33.png'
 file.ppm                'Screenshot from 2023-04-14 07-51-14.png'
 file.txt                'Screenshot from 2023-04-14 07-53-19.png'
 first.png                shell
 image2.ppm               shell.c
 image3.ppm               shell.h
 image.ppm                shell.o
 Makefile                 testme.c
 nxn.cpp                  test.ppm
 out.txt                  test.txt
 pa2.cpp
```

```
>cat test.txt
slkmdlslksf
djsahdjhsdfkf
dskjfhksf
sdjskaaaşd
>cat < test.txt | grep a > mytest.txt | echo hello
hello
>cat mytest.txt
djsahdjhsdfkf
sdjskaaaşd
```

```
>cat test.txt
slkmdlslksf bggdgds nfdb
djsahdjhsdfkf eefuw
dskjfhksf ads wdwe efwr defr
sdjskaaaşd d erwer dn euuw ejwj
>cat test.txt | grep f | sort > my_out.txt
>cat my_out.txt
djsahdjhsdfkf eefuw
dskjfhksf ads wdwe efwr defr
slkmdlslksf bggdgds nfdb
```

```
>ps -ef | grep process | sed -n 2,5p > processes.txt
>cat processes.txt
begum       3326    3266  0 05:25 ?        00:00:00 /opt/brave.com/brave/brave -
-type=renderer --crashpad-handler-pid=3255 --enable-crash-reporter=61ec8c5d-4194
-40ce-b911-5099ea3dd71e, --origin-trial-public-key=bYUKPJoPnCxeNvu72j4EmPuK7tr1P
AC7SHh8ld9Mw3E=,fMS4mpO6buLQ/QMd+zJmxzty/VQ6B1EUZqoCU04zoRU= --change-stack-guar
d-on-fork=enable --brave_session_token=16864338667114979508 --first-renderer-pro
cess --lang=en-US --num-raster-threads=2 --enable-main-frame-before-activation -
-renderer-client-id=7 --time-ticks-at-unix-epoch=-1681439045102605 --launch-time
-ticks=71346728 --shared-files=v8_context_snapshot_data:100 --field-trial-handle
=0,i,2618735924829489777,17569767036175468260,131072
begum       3353    3266  0 05:25 ?        00:00:00 /opt/brave.com/brave/brave -
-type=renderer --crashpad-handler-pid=3255 --enable-crash-reporter=61ec8c5d-4194
-40ce-b911-5099ea3dd71e, --extension-process --origin-trial-public-key=bYUKPJoPn
CxeNvu72j4EmPuK7tr1PAC7SHh8ld9Mw3E=,fMS4mpO6buLQ/QMd+zJmxzty/VQ6B1EUZqoCU04zoRU=
 --change-stack-guard-on-fork=enable --brave_session_token=16864338667114979508
--lang=en-US --num-raster-threads=2 --enable-main-frame-before-activation --rend
erer-client-id=5 --time-ticks-at-unix-epoch=-1681439045102605 --launch-time-tick
s=71600023 --shared-files=v8_context_snapshot_data:100 --field-trial-handle=0,i,
2618735924829489777,17569767036175468260,131072
begum      14427   14388  0 08:09 pts/0    00:00:00 grep process
>
```

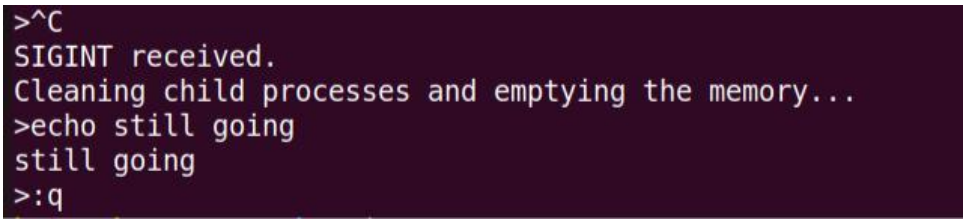
```
>find . | grep png | wc -c > last.txt
>cat last.txt
139
```

```
>cat test.txt
Apple is the best fruit.
>cat test.txt | grep Apple | sed s/Apple/Banana/g > output.txt
>cat output.txt
Banana is the best fruit.
```

Here is the log file created after execution of the command above:

```
Child process information:
PID: 12680      Command:  cat  test.txt
PID: 12681      Command:  grep  Apple
PID: 12682      Command:  sed  's/Apple/Banana/g'  >  output.txt
```

Here is an example case of a signal handling process of this program. Instead of simply exiting the program, all the child processes are killed, and memory is freed after specific signals are received. The shell loop continues to run and wait for new commands.



Only after the input ":q" is the program terminated.