

CSE 344 – System Programming

Homework 2 Report

13.04.2023

- **General Information**

- In general, I made some error checks in all parts. I checked for errors in system calls such as fork, wait and pipe as much as possible.
- I also clean the created .txt and .log files in the makefile.

- **How I solved the problem?**

- My assignment consists of two C files and a header file. The files "terminalEmulator.h" and "terminalEmulator.c" are the two important files that perform the actual task, while "main.c" simply checks for running and calls the terminalEmulator.
- **Firstly , I specify my functions:**
 - Primary function :
 - o terminalEmulator()
 - Helper functions:
 - o parseUserInput()
 - o createPipes()
 - o executeCommand()
 - o childProcess()
 - o parentProcess()
 - o parseCommand()
 - o redirectInputOutput()
 - o handleInputRedirection()
 - o handleOutputRedirection()
 - o execute()
 - Signal function:
 - o signalHandler()

- Logging function:
 - logChildPidAndCommand()
- **This is the steps how I solved it :**
 - Firstly, I define the MAX_COMMANDS and MAX_COMMAND_SIZE that the user can enter.
 - When the program is run, I first use “fgets()” to get the command from the user.
 - Then, I set four signals (SIGINT, SIGQUIT, SIGTSTP, SIGCHLD).
 - After that, I create a log file with the current timestamp.
 - Then, I separate the entered command according to pipes (|) (parseUserInput()). I create pipes for the program based on the resulting number from parseUserInput and connect the commands to each other (createPipes()).
 - Then, I iterate over the number of parsed commands in a loop and execute each command within it (executeCommand()). If multiple commands are entered, I set the input-output using pipes within this loop.
 - Regarding the part where each command is executed, with “fork()” a new child process is created. Depending on the pid output, the child calls the child function (childProcess()) and the parent calls the parent function (parentProcess()).
 - In the child process, I parse the command again because I separated it according to pipes at the beginning, but there may be redirections inside, so I parse it again to handle them. If there are redirections, I redirect them to files according to the file descriptor. After that, I execute the fully parsed command using “execvp()” (execute()).
 - In the parent process, I wait for the children using “wait()” and if the child has died, I print the PID and command to the log file (logChildPidAndCommand).
- In summary, I solved the problem of executing multiple shell commands from a single input string by parsing the user input into separate commands, creating pipes between the commands, forking a child process for each command, and redirecting the input and output streams if needed.

- **Which requirements I met?**

- I create a new log file with the current timestamp in each user input command.
- I create new child process for each command. You can see in the log files after running the program: If input command have multiple commands in it then it means there are more than one child.
- I handle pipes and redirections.
- On the first run, if the program is not called correctly, usage information is printed.
- Program waits for ":q" to finalize its execution.
- Error messages and signals (4 signal) during the execution printed and program returns to the prompt and it keeps getting commands.
- Since the SIGKILL signal is sent by the operating system, it is not possible to define a signal handler to handle this signal. So I don't have signalHandler for SIGKILL signal.
- I used system calls such as fork(), execl(), wait(), and exit() when executing commands.
- I didn't leave zombies from child processes.
- There are no memory leaks.

(You can see its tests below)

- Which commands I tested?

- Usage :

```
esoo@esra:~/Desktop/eryilmaz_esra_171044046$ make
gcc -c main.c -std=gnu99
gcc -c terminalEmulator.c -std=gnu99
gcc -o hw2 main.o terminalEmulator.o
esoo@esra:~/Desktop/eryilmaz_esra_171044046$ ./hw2 file
Usage: 1 command line arguments are required!
EXAMPLES:
$ ./hw2
$ valgrind --leak-check=full --show-leak-kinds=all ./hw2
(You can enter a command after running.)
esoo@esra:~/Desktop/eryilmaz_esra_171044046$ ./hw2
Welcome to the terminal emulator!
>> |
```

- The images on the left are /bin/sh tests, while the images on the right are tests of my program :

(I first try with \$ /bin/sh because my program leaves a log with every command. This way, I can test without increasing the logs.)

```
esoo@esra:~/Desktop/eryilmaz_esra_171044046$ /bin/sh
$ ls -ll
total 80
-rwxrwxr-x 1 esoo esoo 18360 Nis 14 17:17 hw2
-rw-rw-r-- 1 esoo esoo 21851 Nis 14 16:20 hw2_report.odt
-rw-rw-r-- 1 esoo esoo 738 Nis 14 01:19 main.c
-rw-rw-r-- 1 esoo esoo 2016 Nis 14 17:17 main.o
-rw-rw-r-- 1 esoo esoo 239 Nis 14 00:59 makefile
-rw-rw-r-- 1 esoo esoo 7757 Nis 14 16:20 terminalEmulator.c
-rw-rw-r-- 1 esoo esoo 935 Nis 14 01:06 terminalEmulator.h
-rw-rw-r-- 1 esoo esoo 8504 Nis 14 17:17 terminalEmulator.o
$ |
```

```
esoo@esra:~/Desktop/eryilmaz_esra_171044046$ valgrind --leak-check=full --show-leak-kinds=all ./hw2
==9976== Memcheck, a memory error detector
==9976== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==9976== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==9976== Command: ./hw2
==9976==
Welcome to the terminal emulator!
>> ls -ll
total 80
-rwxrwxr-x 1 esoo esoo 18360 Nis 14 17:17 hw2
-rw-rw-r-- 1 esoo esoo 21851 Nis 14 16:20 hw2_report.odt
-rw-rw-r-- 1 esoo esoo 738 Nis 14 01:19 main.c
-rw-rw-r-- 1 esoo esoo 2016 Nis 14 17:17 main.o
-rw-rw-r-- 1 esoo esoo 239 Nis 14 00:59 makefile
-rw-rw-r-- 1 esoo esoo 7757 Nis 14 16:20 terminalEmulator.c
-rw-rw-r-- 1 esoo esoo 935 Nis 14 01:06 terminalEmulator.h
-rw-rw-r-- 1 esoo esoo 8504 Nis 14 17:17 terminalEmulator.o
>> |
```

```
$ echo "hellooo"  
hellooo  
$
```

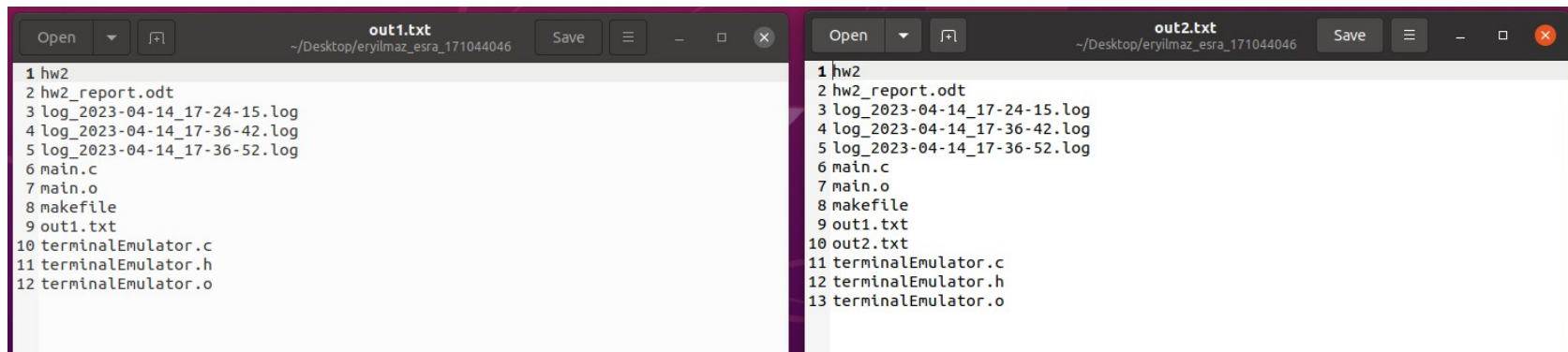
```
>> echo "hellooo"  
"hellooo"  
>>
```

```
$ ls | grep hw2  
hw2  
hw2_report.odt  
$
```

```
>> ls | grep hw2  
hw2  
hw2_report.odt  
>>
```

```
$ ls > out1.txt  
$
```

```
>> ls > out2.txt  
>>
```



```
out1.txt  
~/Desktop/eryilmaz_esra_171044046  
1 hw2  
2 hw2_report.odt  
3 log_2023-04-14_17-24-15.log  
4 log_2023-04-14_17-36-42.log  
5 log_2023-04-14_17-36-52.log  
6 main.c  
7 main.o  
8 makefile  
9 out1.txt  
10 terminalEmulator.c  
11 terminalEmulator.h  
12 terminalEmulator.o  
  
out2.txt  
~/Desktop/eryilmaz_esra_171044046  
1 hw2  
2 hw2_report.odt  
3 log_2023-04-14_17-24-15.log  
4 log_2023-04-14_17-36-42.log  
5 log_2023-04-14_17-36-52.log  
6 main.c  
7 main.o  
8 makefile  
9 out1.txt  
10 out2.txt  
11 terminalEmulator.c  
12 terminalEmulator.h  
13 terminalEmulator.o
```

```
$ sort < out2.txt
hw2
hw2_report.odt
log_2023-04-14_17-24-15.log
log_2023-04-14_17-36-42.log
log_2023-04-14_17-36-52.log
main.c
main.o
makefile
out1.txt
out2.txt
terminalEmulator.c
terminalEmulator.h
terminalEmulator.o
$
```

```
>> sort < out2.txt
hw2
hw2_report.odt
log_2023-04-14_17-24-15.log
log_2023-04-14_17-36-42.log
log_2023-04-14_17-36-52.log
main.c
main.o
makefile
out1.txt
out2.txt
terminalEmulator.c
terminalEmulator.h
terminalEmulator.o
>>
```

```
$ grep log out2.txt | sort | uniq
log_2023-04-14_17-24-15.log
log_2023-04-14_17-36-42.log
log_2023-04-14_17-36-52.log
$
```

```
>> grep log out2.txt | sort | uniq
log_2023-04-14_17-24-15.log
log_2023-04-14_17-36-42.log
log_2023-04-14_17-36-52.log
>>
```

```
$  
$  
$ cat main.c > out3.txt  
$  
$
```

```
>>  
>> cat main.c > out4.txt  
>>
```

```
Open  out3.txt  Save  -  □  ×
~/Desktop/eryilmaz_esra_171044046

1 /*****
2  *      CSE344 - System Programming -
   Homework2
3  *      Author: Esra|
   Eryilmaz
   *
4  *      Date:
   12.04.2023
   *
   *
   *
6  *      This program is a basic implementation of a terminal emulator
7  *****/
8
9 #include <stdio.h>
10 #include "terminalEmulator.h"
11
12 int main(int argc, char const *argv[])
13 {
14     if (argc == 1) {                // S./hw2
15         terminalEmulator();
16     }
17     else {
18         fprintf(stderr, "Usage: 1 command line arguments are required!-
\nEXAMPLES:\n$ ./hw2\n$ valgrind --leak-check=full --show-leak-kinds=all ./hw2\n(You
can enter a command after running.)\n");
19         return -1;
20     }
21
22     return 0;
23 }
```

```
Open Save - □ ×
~/Desktop/eryilmaz_esra_171044046
out4.txt

1 /*****
2 *      CSE344 - System Programming -
   Homework2
3 *      Author: Esra
   Eryilmaz
4 *      Date:
   12.04.2023
5 *
6 *      This program is a basic implementation of a terminal emulator
7 *****/
8
9 #include <stdio.h>
10 #include "terminalEmulator.h"
11
12 int main(int argc, char const *argv[])
13 {
14     if (argc == 1) {                // $.hw2
15         terminalEmulator();
16     }
17     else {
18         fprintf(stderr, "Usage: 1 command line arguments are required!-
19 \nEXAMPLES:\n$. ./hw2\n$. valgrind --leak-check=full --show-leak-kinds=all ./hw2\n(You
20 can enter a command after running.)\n");
21         return -1;
22     }
23     return 0;
24 }
```

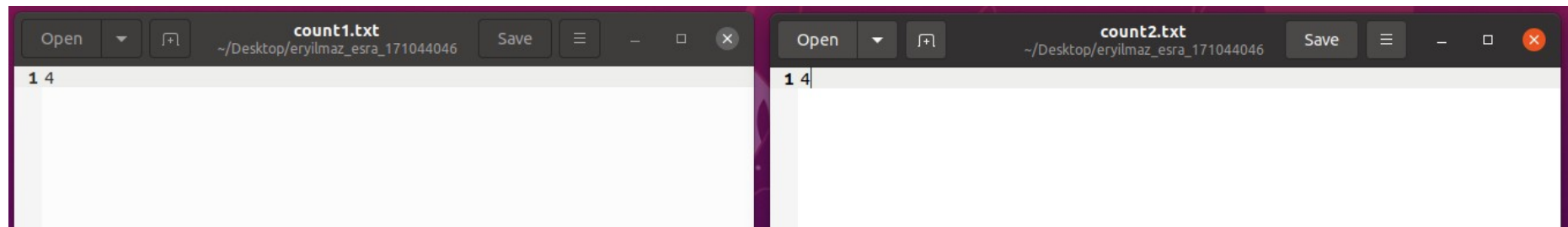
- I didn't leave zombies from child processes : (there are no Z or Z+)

```
$ ps aux | grep hw2
esoo    9976  0.0  0.5 88600 43732 pts/0    S+   17:24   0:00 /usr/bin/valgrind
esoo    16531  0.0  0.0 164364 4756 ?        Sl   18:51   0:00 /usr/lib/libreof
esoo    16565  9.8  3.6 1577520 286556 ?        Sl   18:51   1:21 /usr/lib/libreof
esoo    18132  1.6  1.4 1513328 113896 ?        Sl   19:02   0:02 evince /home/esoo
esoo    18493  0.0  0.0 11776 660 pts/1      S+   19:05   0:00 grep hw2
$
```

```
>> ps aux | grep hw2
esoo    9976  0.0  0.5 88600 43732 pts/0    S+   17:24   0:00 /usr/bin/valgrind
esoo    16531  0.0  0.0 164364 4756 ?        Sl   18:51   0:00 /usr/lib/libreof
esoo    16565  9.3  3.6 1577520 286556 ?        Sl   18:51   1:21 /usr/lib/libreof
esoo    18132  1.2  1.4 1513328 113896 ?        Sl   19:02   0:02 evince /home/esoo
>>
```

```
$ ls | grep out | wc -l > count1.txt
$
```

```
>> ls | grep out | wc -l > count2.txt
>>
```



- Signal tests :

```
>> ^C
Received SIGINT signal
date
Cum 14 Nis 2023 19:34:09 +03
>>
>>
>> ^Z
Received SIGTSTP signal
date
Cum 14 Nis 2023 19:35:03 +03
```

- There are no memory leaks in the homework :

```
>> :q
Exiting...
==9976==
==9976== HEAP SUMMARY:
==9976==      in use at exit: 0 bytes in 0 blocks
==9976==    total heap usage: 80 allocs, 80 frees, 90,630 bytes allocated
==9976==
==9976== All heap blocks were freed -- no leaks are possible
==9976==
==9976== For lists of detected and suppressed errors, rerun with: -s
==9976== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
esoo@esra:~/Desktop/eryilmaz_esra_171044046$
```

- Some generated logs :

