CSE 344 System Programming

HW2 Report

Kahraman Arda Kılıç
1901042648

1- How the code works?

Firstly, the user is prompted to enter a command. Once the user inputs the command, the parse_filename() function is called. Here, the commands, which are an array of char arrays, are split and placed based on the '|' symbol. Then, within the main() function, the number of commands in the commands array is counted. A for loop is executed for each command. During each iteration, the commands are executed. Afterwards, the child processes are waited for in the parent using the waitpid() method, and the parent process terminates.

```
(ARTER)$:
```

(shell is waiting for input from user)

2- Executing all the processes

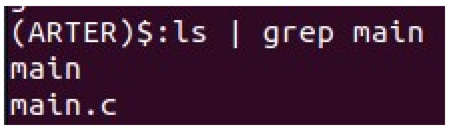
Each command is stored in the commands array. These commands are iterated in order within a for loop. For each iteration, a fork() operation is first performed for the incoming command. Then, a pipe() is opened depending on the position of the command (first, middle, or last) within the for loop. If the command is the last one, no pipe() is opened because the number of pipes is N-1 for N processes. After that, a switch-case structure is executed depending on the pid value returned from fork(). In the case where the value is 0, a child has been created. Then, the values of the pipe that the process will read and write from are set using dup2(). The unused parts of the pipe() are closed and the command is executed with execl(). If there is an error, the error message is printed with perror() for the execl() function. In the absence of an error, the child process completes and exits with exit(1). Afterwards, the unused pipe()s are closed from the switch-case structure and the information of the process is written to the log file. Additionally, for each process within the for loop, SIGINT and SIGQUIT are ignored, while SIGCHLD is blocked.

3- How pipes are connected?

N processes require N-1 pipes to be created. For each command, the i value is incremented in a for loop. If the command is the first one, it has access to the write end of the i-th pipe. Middle commands with i values other than the first or last one have access to the read end of the (i-1)-th pipe and the write end of the i-th pipe. The last command has access to the read end of the (i-1)-th pipe, and its write end is fixed to STDOUT_FILENO.

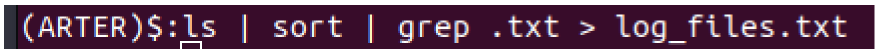
4- Test Cases

a.

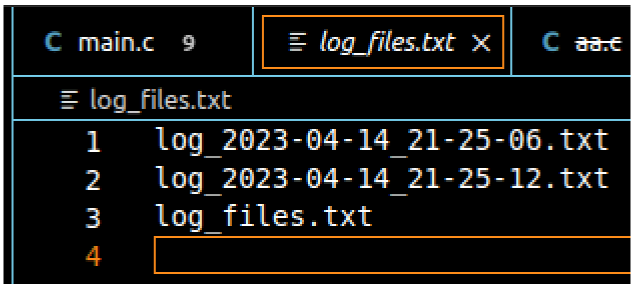


```
(ARTER)$:ls | grep main
main
main.c
```

b.

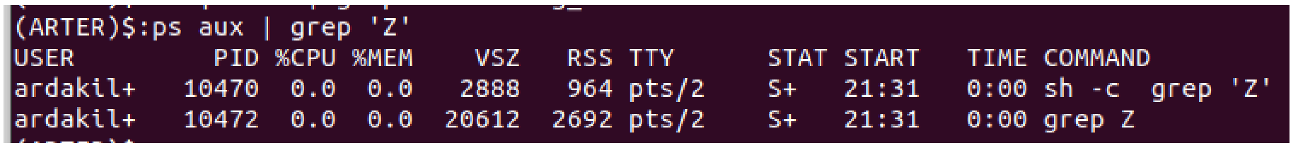


```
(ARTER)$:ls | sort | grep .txt > log_files.txt
```



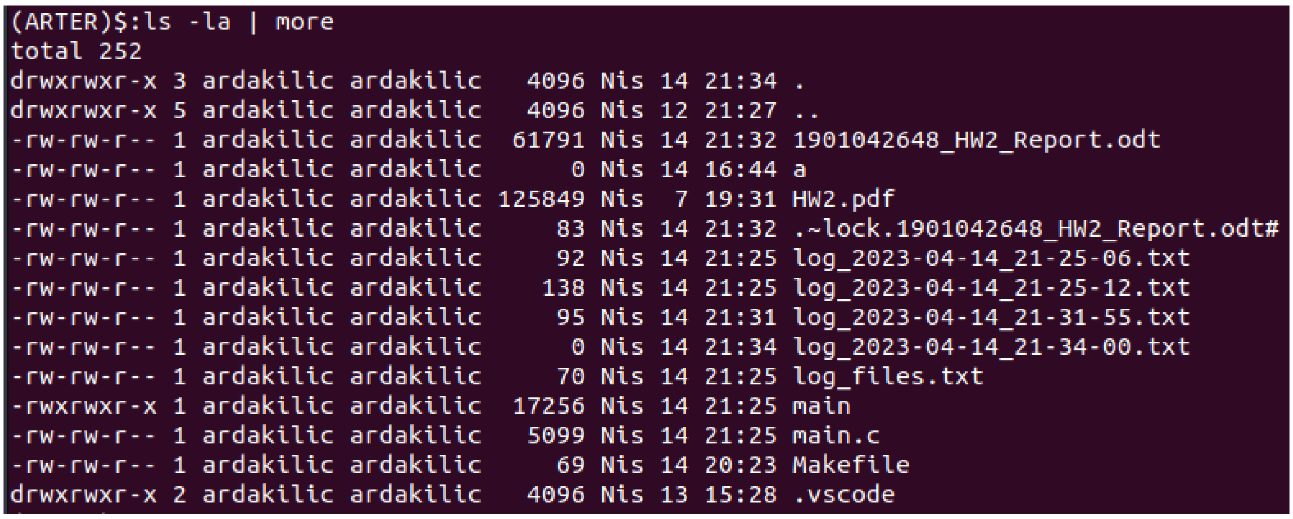
```
log_2023-04-14_21-25-06.txt
log_2023-04-14_21-25-12.txt
log_files.txt
```

c.



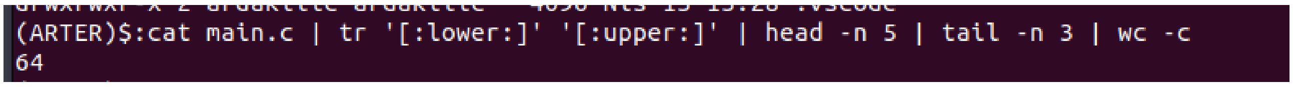
```
(ARTER)$:ps aux | grep 'Z'
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
ardakil+   10470  0.0  0.0   2888   964 pts/2    S+   21:31   0:00 sh -c  grep 'Z'
ardakil+   10472  0.0  0.0  20612  2692 pts/2    S+   21:31   0:00 grep Z
```

d.

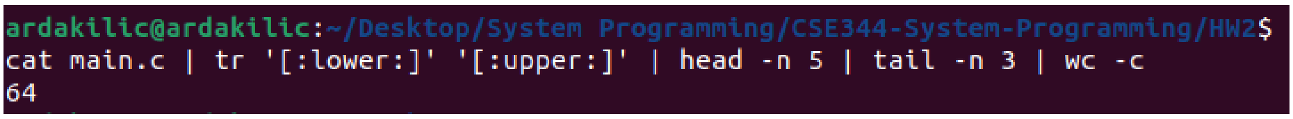


```
(ARTER)$:ls -la | more
total 252
drwxrwxr-x 3 ardakilic ardakilic   4096 Nis 14 21:34 .
drwxrwxr-x 5 ardakilic ardakilic   4096 Nis 12 21:27 ..
-rw-rw-r-- 1 ardakilic ardakilic  61791 Nis 14 21:32 1901042648_HW2_Report.odt
-rw-rw-r-- 1 ardakilic ardakilic      0 Nis 14 16:44 a
-rw-rw-r-- 1 ardakilic ardakilic 125849 Nis  7 19:31 HW2.pdf
-rw-rw-r-- 1 ardakilic ardakilic     83 Nis 14 21:32 .~lock.1901042648_HW2_Report.odt#
-rw-rw-r-- 1 ardakilic ardakilic     92 Nis 14 21:25 log_2023-04-14_21-25-06.txt
-rw-rw-r-- 1 ardakilic ardakilic    138 Nis 14 21:25 log_2023-04-14_21-25-12.txt
-rw-rw-r-- 1 ardakilic ardakilic     95 Nis 14 21:31 log_2023-04-14_21-31-55.txt
-rw-rw-r-- 1 ardakilic ardakilic      0 Nis 14 21:34 log_2023-04-14_21-34-00.txt
-rw-rw-r-- 1 ardakilic ardakilic     70 Nis 14 21:25 log_files.txt
-rwxrwxr-x 1 ardakilic ardakilic  17256 Nis 14 21:25 main
-rw-rw-r-- 1 ardakilic ardakilic   5099 Nis 14 21:25 main.c
-rw-rw-r-- 1 ardakilic ardakilic     69 Nis 14 20:23 Makefile
drwxrwxr-x 2 ardakilic ardakilic   4096 Nis 13 15:28 .vscode
```

e.

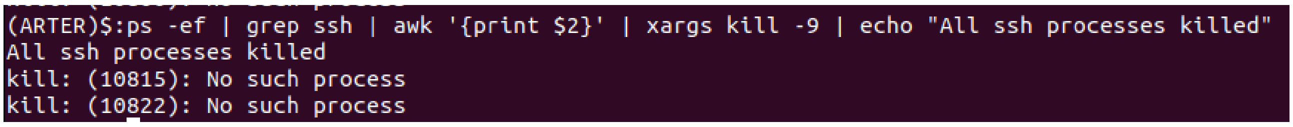


```
(ARTER)$:cat main.c | tr '[:lower:]' '[:upper:]' | head -n 5 | tail -n 3 | wc -c
64
```



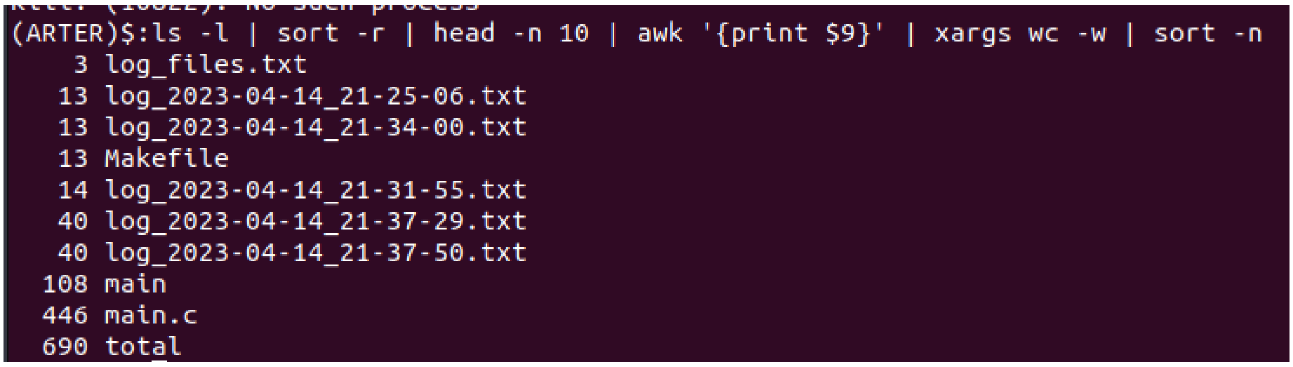
```
ardakilic@ardakilic:~/Desktop/System Programming/CSE344-System-Programming/HW2$
cat main.c | tr '[:lower:]' '[:upper:]' | head -n 5 | tail -n 3 | wc -c
64
```

f.



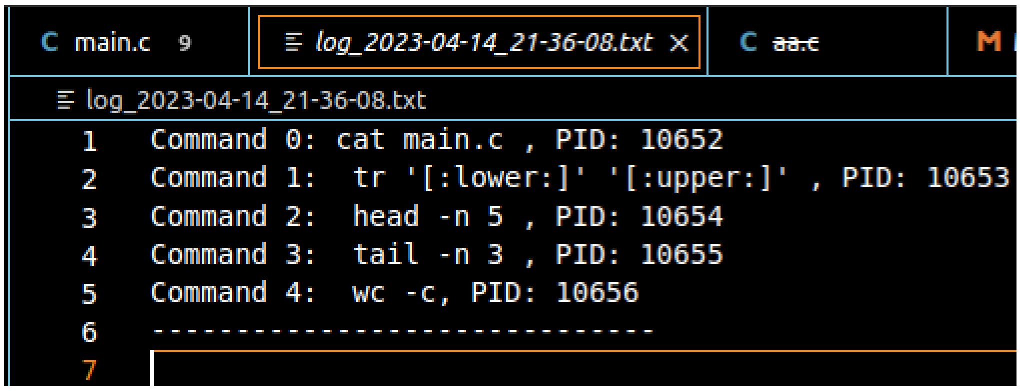
```
(ARTER)$:ps -ef | grep ssh | awk '{print $2}' | xargs kill -9 | echo "All ssh processes killed"
All ssh processes killed
kill: (10815): No such process
kill: (10822): No such process
```

g.

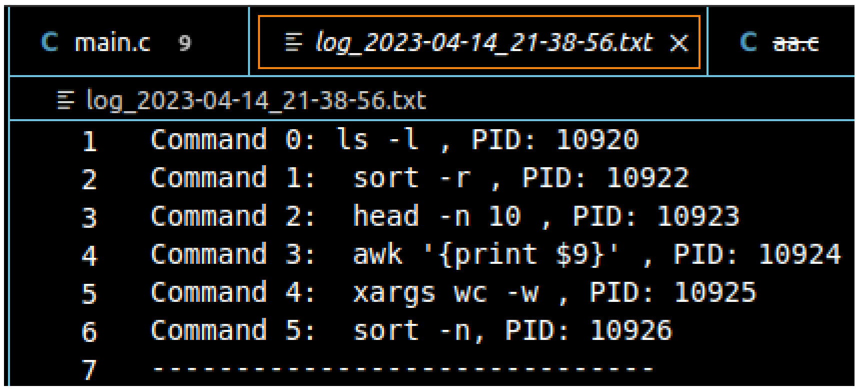


```
(ARTER)$:ls -l | sort -r | head -n 10 | awk '{print $9}' | xargs wc -w | sort -n
    3 log_files.txt
   13 log_2023-04-14_21-25-06.txt
   13 log_2023-04-14_21-34-00.txt
   13 Makefile
   14 log_2023-04-14_21-31-55.txt
   40 log_2023-04-14_21-37-29.txt
   40 log_2023-04-14_21-37-50.txt
  108 main
  446 main.c
  690 total
```

5- Log Files



```
Command 0: cat main.c , PID: 10652
Command 1:  tr '[:lower:]' '[:upper:]' , PID: 10653
Command 2:  head -n 5 , PID: 10654
Command 3:  tail -n 3 , PID: 10655
Command 4:  wc -c, PID: 10656
------------------------------
```



```
Command 0: ls -l , PID: 10920
Command 1:  sort -r , PID: 10922
Command 2:  head -n 10 , PID: 10923
Command 3:  awk '{print $9}' , PID: 10924
Command 4:  xargs wc -w , PID: 10925
Command 5:  sort -n, PID: 10926
------------------------------
```