# CSE 344 – Spring 2023
# HW2 Report

GONCA EZGİ ÇAKIR – 151044054

## 1. Solution Approach

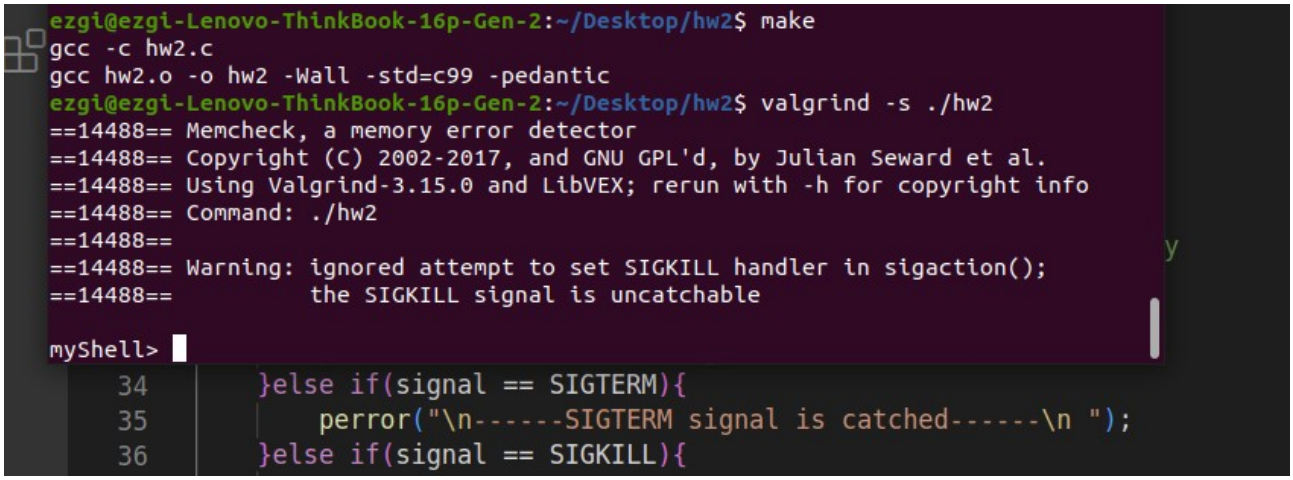
Terminal emulator program implementation steps are listed below.

- Program works in a while(1) loop in order to keep getting commands from user till the exit command (:q) received.
- Command taken from user after the program started with “myShell>”. **fgets** takes the command as a line and checks validity. The command shouldn’t be containing “&&, ||, &, ;” seperators also there can’t be more than 20 commands in a single line. If these cases occurs program simply prints the error with proper usage information and asks for new command. If the command is valid, the line splitted according to **|** token by using **strtok**. Then these commands are stored in a string array, also the number of commands are calculated.
- Children processes are created in a for loop that iterates as number of commands. Child is created by using **fork** system call. Fork’s return value checked if it is equal to 0, child process called. If it is greater than 0, parent process runs. Otherwise there is an error occurred while fork system call, program prints the error and exits.
- For the basic of the program children processes have to communicate between each other. Because each child process executes one command and gives the result to other child process in order to get the proper result at the end.

  So, pipe is created at the beginning of the for loop before child process created by fork, and stored in an array of pipes. Each child process closes the read end of the pipe (pipe[0]) and duplicates the STDOUT_FILENO to pipes write end (pipe[1]). This provide an access to other child process to continue from the STDOUT_FILENO. At the end child process closes the pipes write end (pipe[1]) and executes **execl** system call to run command.

  Parent firstly closes the write end of the pipe (pipe[1]) and duplicates the STDIN_FILENO for pipe to read end (pipe[0]). At the end parent process closes the pipes read end (pipe[0]).
- If the for loop reaches the *last child* (by simply checking loop iterator to be equal number of command amount minus 1) shell command’s result which is taken form pipes read end(pipe[0]) is printed to terminal. If there is nothing to print (might be written to file), program simply continues.
- Signals are handled in a signal handler method which is set with sigaction struct and implemented at the beginning of the program. When a program received SIGINT or SIGTERM signal handler print the signal name and program requests another command prompt from the user.

For SIGKILL it is not possible to catch or handle this signal because it is handle by the kernel. Also valgrind gave a warning, thats why it not added in the code.

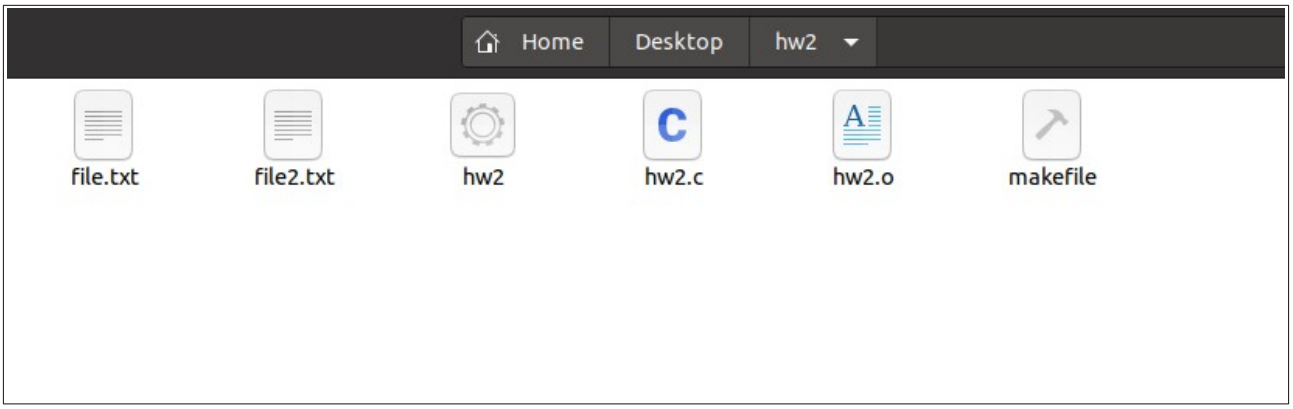



- At the end of executing the command all child process pids and their executed command information are written into log file with a filename as log_timestamp.txt .
- When exit command(:q) received all children process are ended by **wait** system call.

## 2. Test Results

Multiple test case senarios applied and screenshots are added.

Below you will see some program input result. This is the hw2 folder content at the beginning:

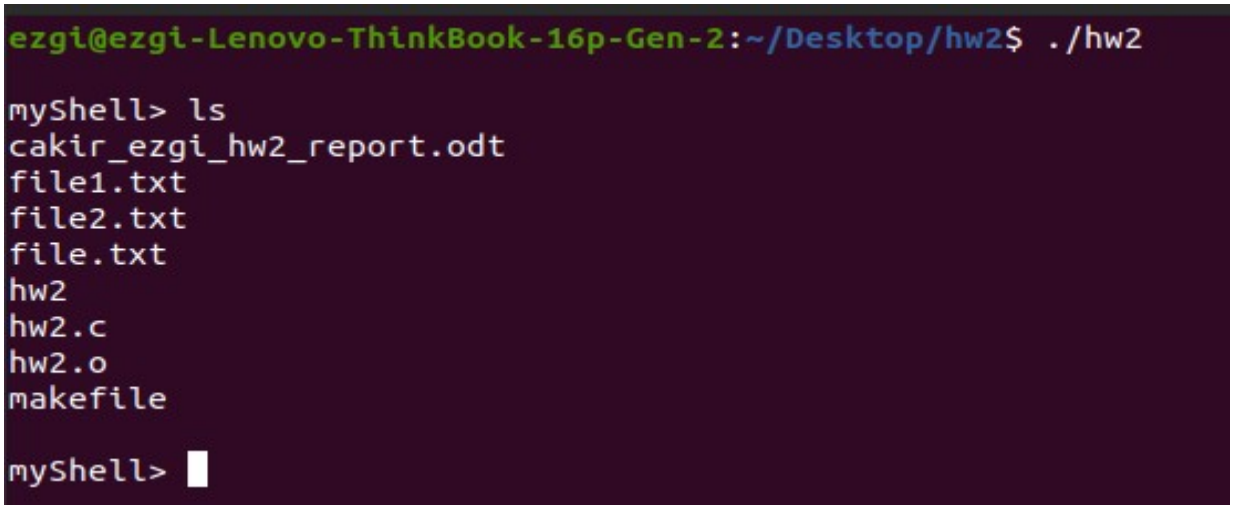


Content of file.txt:

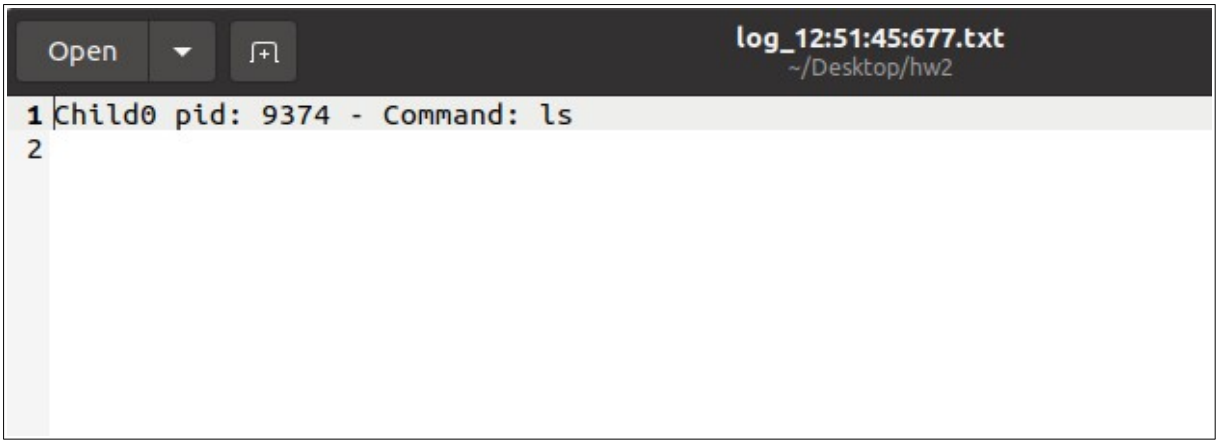

Content of file2.txt:


## 2.1. ls

Program output for command:

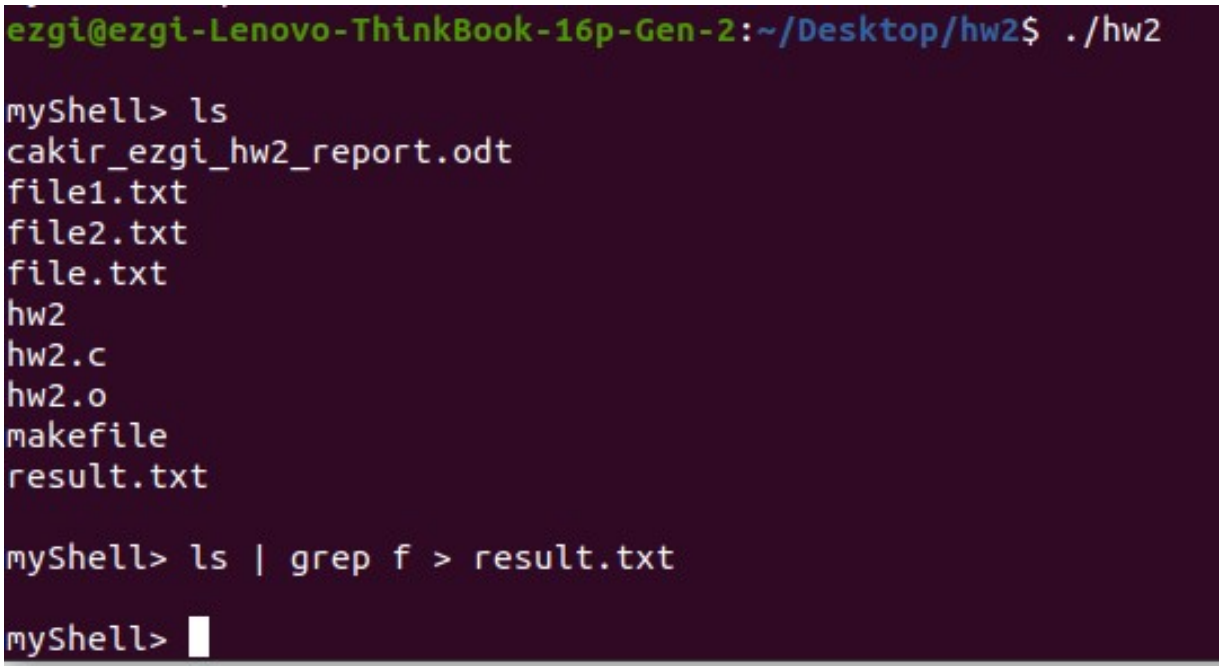


Log file of the command:

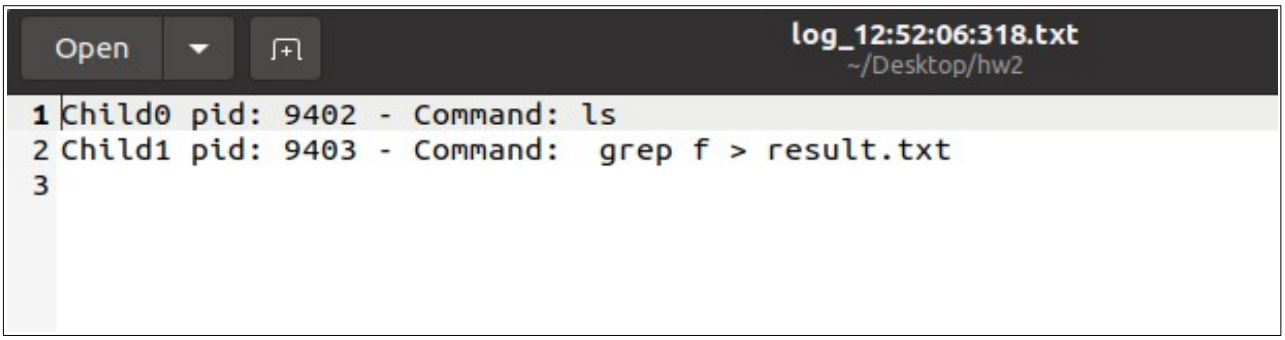



## 2.2. ls | grep f > result.txt

Program output for command:



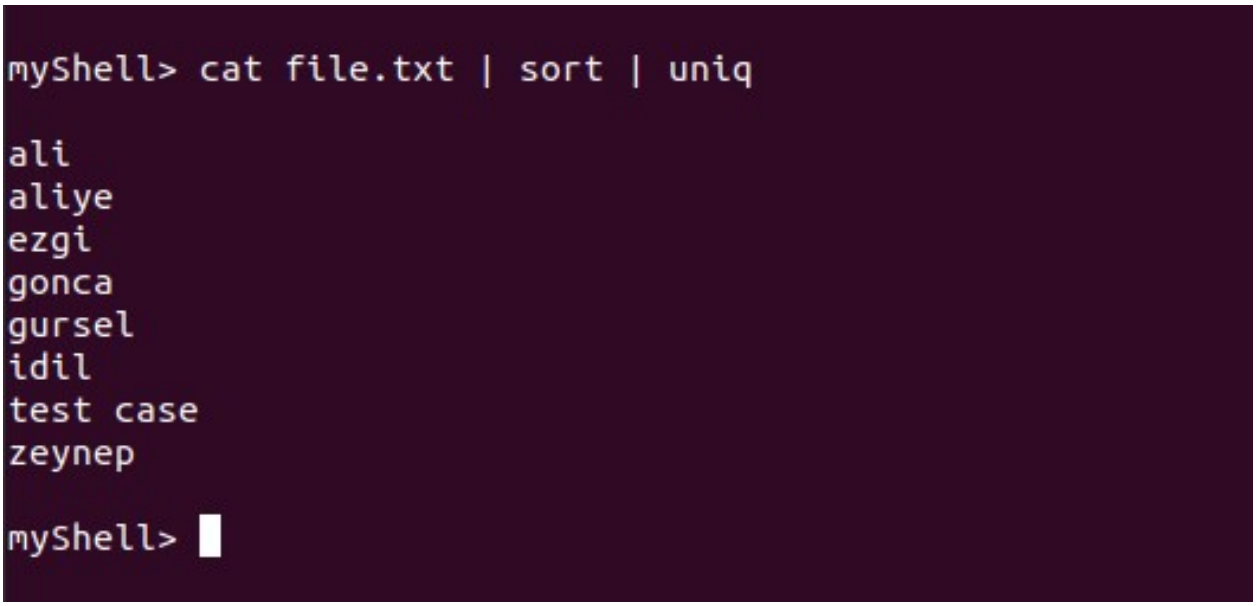
Log file of the command:



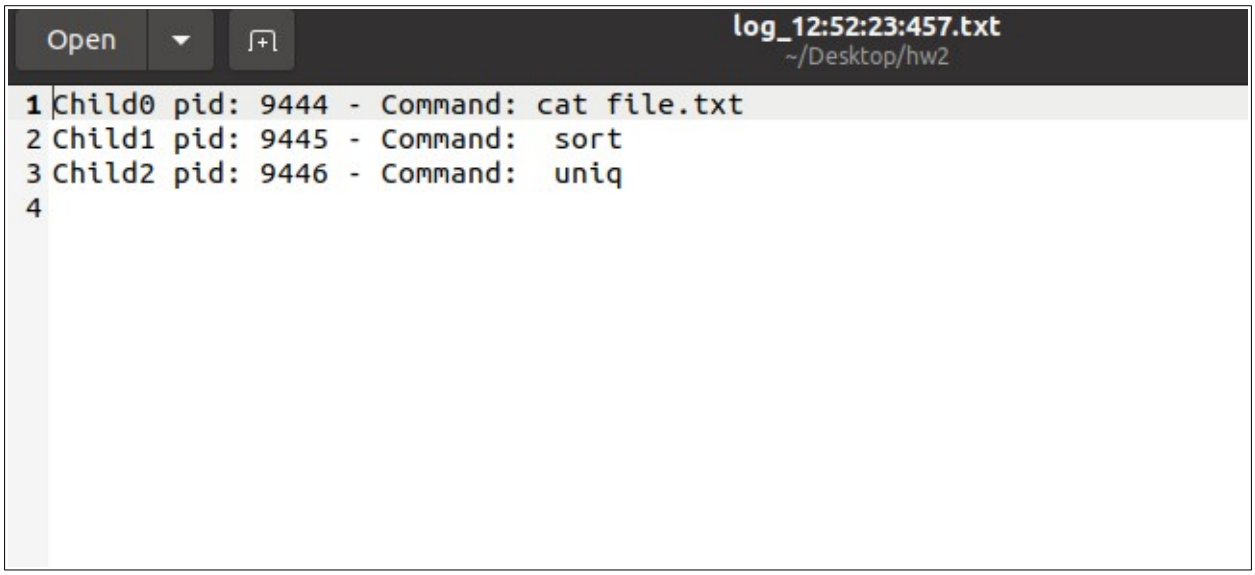

result.txt after execution:


## 2.3. cat file.txt | sort | uniq

Program output for command:



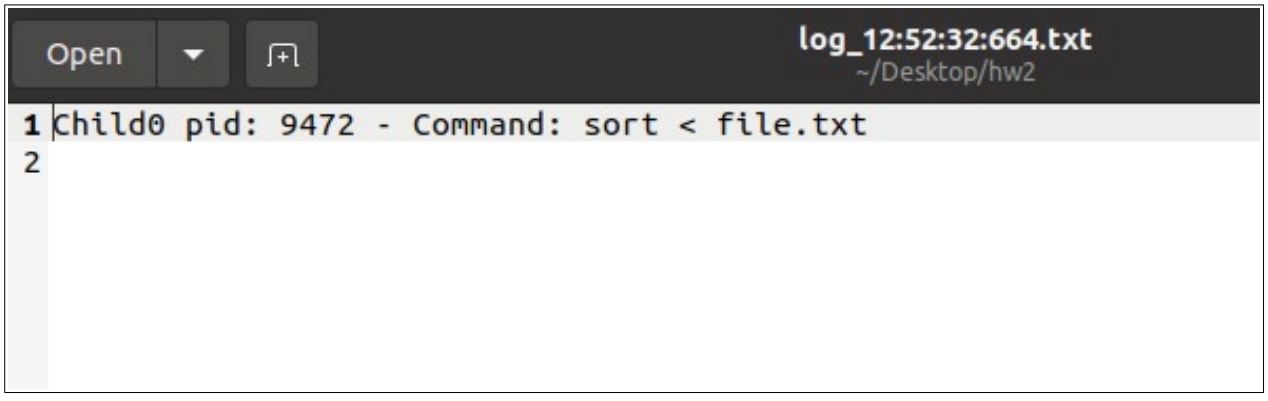
Log file of the command:




## 2.4. sort < file.txt

Program output for command:


Log file of the command:



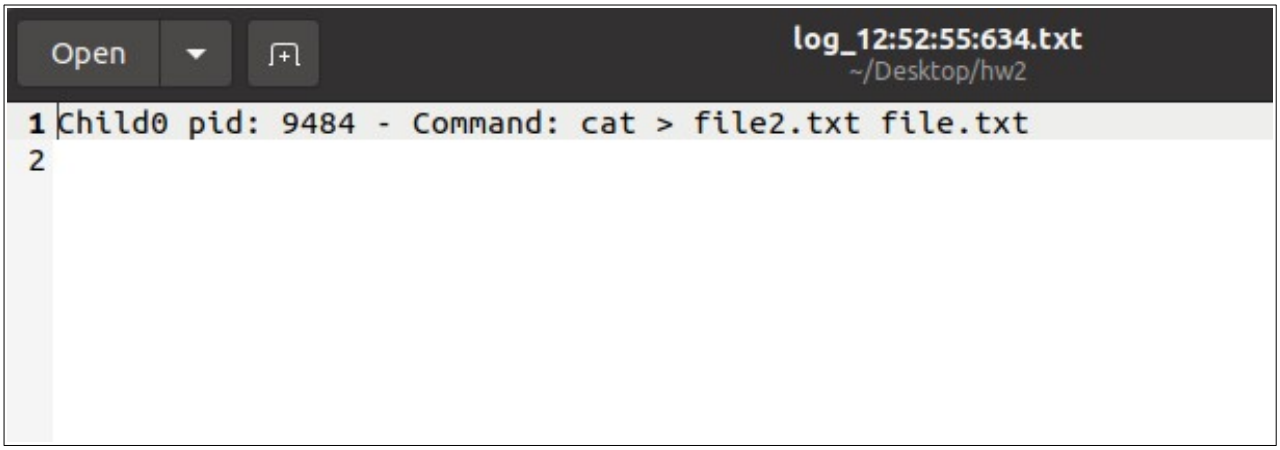
## 2.5. cat > file2.txt file.txt

Program output for command:

```
myShell> cat > file2.txt file.txt

myShell>
```

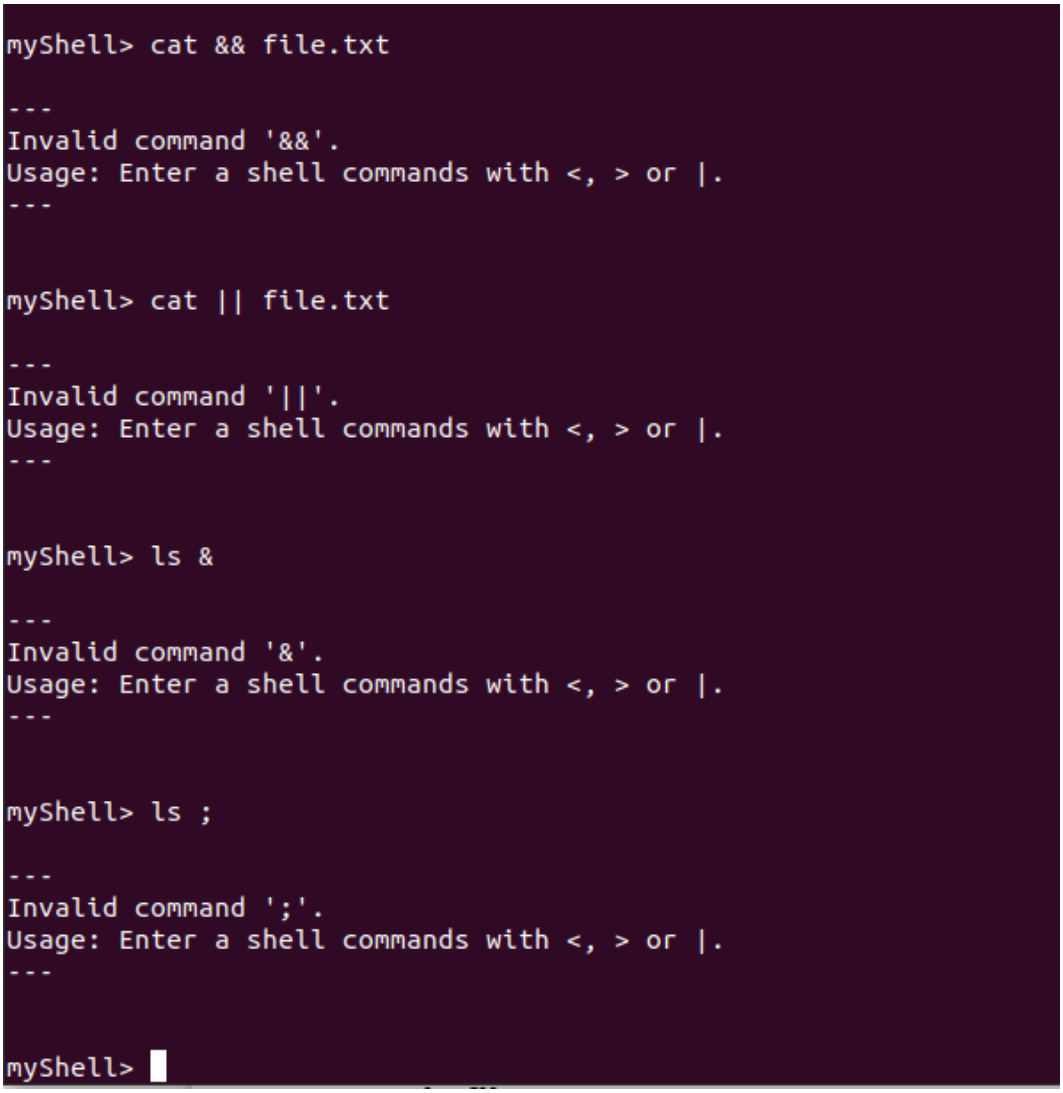
Log file of the command:




file2.txt after execution:

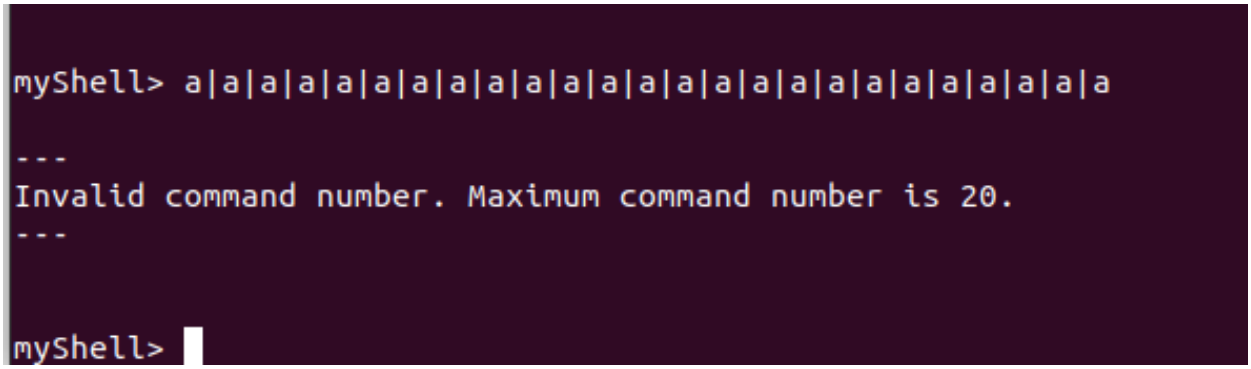
### 2.6. cat && file.txt,  cat || file.txt, ls &, ls ; (unsupported command, Invalid)

Program output for command:




### 2.7. a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a (25 commands, Invalid)

Program output for command:

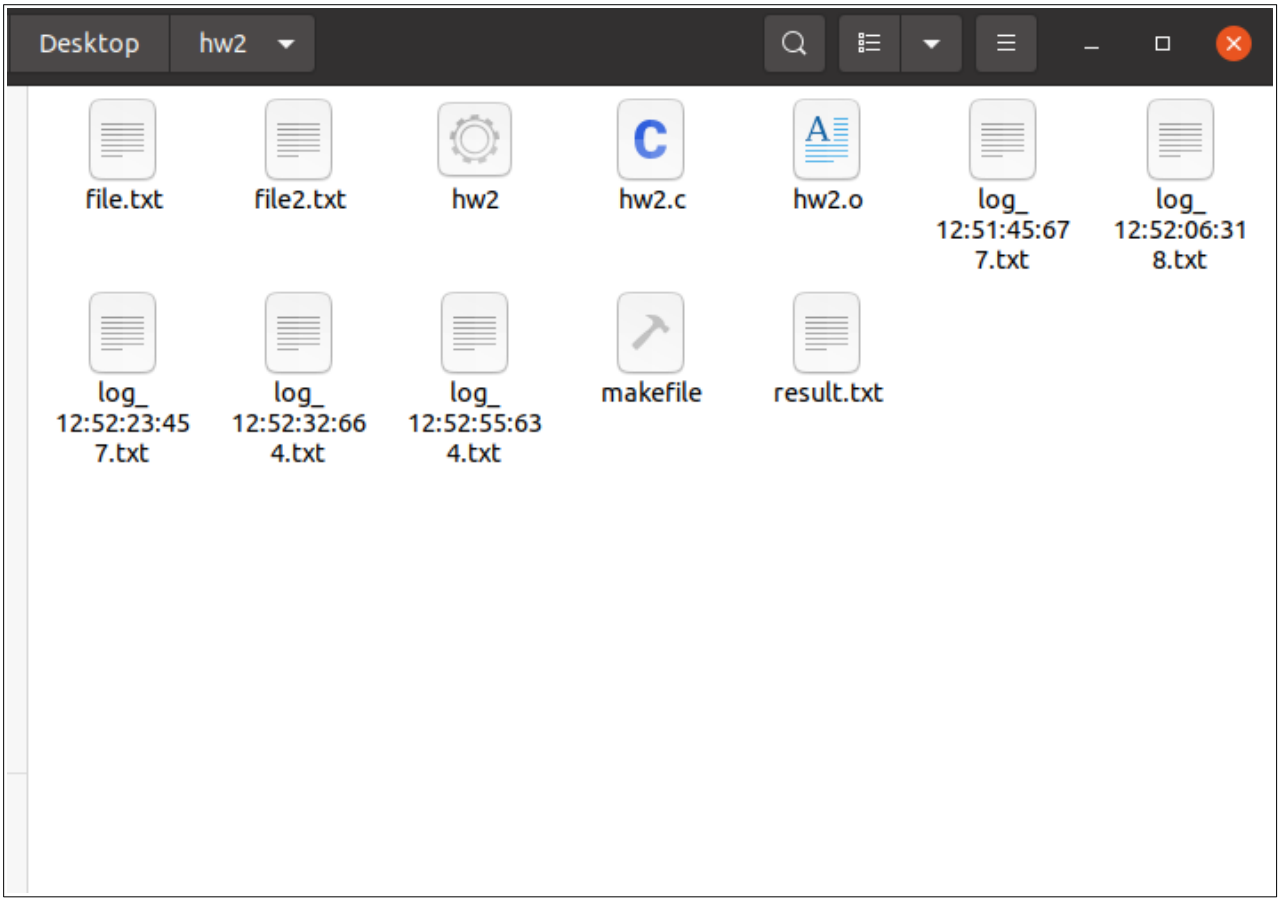


## 2.8. :q (quit command) and valgring result

Program output for command:

```
myShell> :q
==9372==
==9372== HEAP SUMMARY:
==9372==     in use at exit: 0 bytes in 0 blocks
==9372==   total heap usage: 20 allocs, 20 frees, 10,978 bytes allocated
==9372==
==9372== All heap blocks were freed -- no leaks are possible
==9372==
==9372== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
ezgi@ezgi-Lenovo-ThinkBook-16p-Gen-2:~/Desktop/hw2$
```

After the execution of the program, folder content:



## 3. Makefile

You can see makefile content down below. It only complies the codes with warning flags by “make” command and cleans .o files with “make clean” command.

*Note:* I didn’t add the valgrind because it doesnt work without the executable file. You can test it as “valgrind -s ./hw2” (Also the result of valgrind is in the part 2.8)