CSE344-SYSTEMS PROGRAMMING

HOMEWORK 2 REPORT

Yağız Hakkı Aydın
1901042612

PART I : EXPLANATION OF THE CODE

>>>>Program starts with the main function,when shell emulator is started:

-createLogFile function gets called.

This function creates an file with name with corresponding time in human readable format.Uses getTimeStr function to get time string in human readable format.Then by using this string,opens the log file.

To open log file , uses “fileDescriptorForLogFile” global variable to keep file descriptor for the log file.

-Then signal system call is used to handle SGKİLL when user enters ctrl+c by using handleSGKILL function.

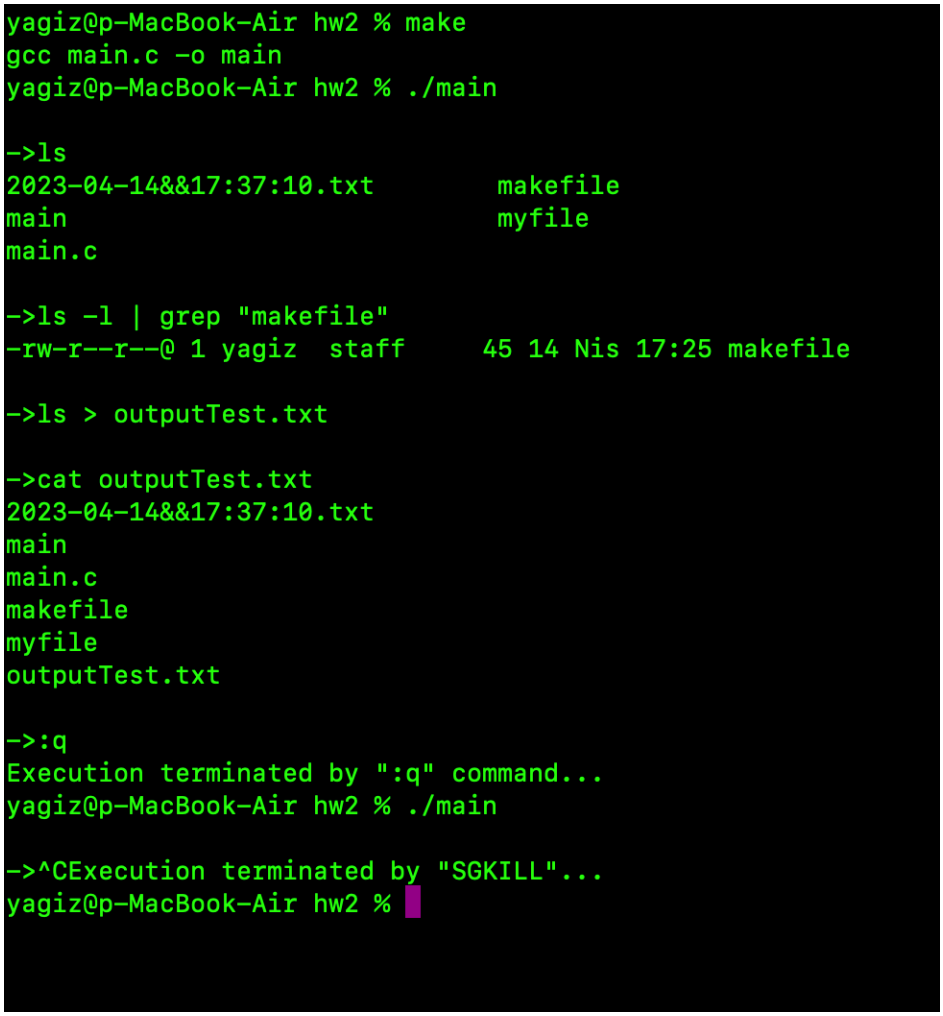
-Then infinite loop starts

At each step for loop,one line of the command is entered by the user.This command is limited with 4000 characters fixed size.

At each step,input string which contains commands those are seperated with pipe symbol is send to runCommands function to handle execution of the commands and pipe oeration between them.(Will be explained in detailed).

If user enters :q , shell gets terminated;if user presses ctrl+c,program gets terminated too.For each condition,file descriptor for log file gets closed.

Here is an example output how the terminal works



>>>>The main part of the program is runCommands function.

-This function mainly takes the entire command line.Tokenizes the command with tokenizeCommandLine function and obtains a 2d array which contains all commands seperated by pipe symbol.

-Created 2d int array for keeping file descriptors for pipe.

-Sets prev_pipe_read variable to STDIN_FILENO because it is initialized for first command which does not previous command to take it's output.

-Then strats a loop to execute commands.

For each step of this loop:

>Creates pipe with the array of file descriptors.

>Then creates a new process using fork for next command,and keeps return value of the fork in pid.

>Checks the value of pid.

-If -1 , error is handled.

-If ,0 , Previous command's output is connected to child(corresponding) process's input and then connects corresponding process's output to end of the pipe.Then closes file descriptors.

-If not 0 or -1, end of pipe is closed. prev_pipe_read is set for the next command.

-Process is written to the log file.

>>>>Other one of the main parts of the program is redirect function.

-This function tokenizes single command by using tokenizeSingleCommand function.

-Checks if given command is an redirection command by using checkIfCommandIsRedirection.

-Then finds index redirection symbol in tokenizedCommand array.

-Then concatenates left side of redirection symbol.

-Then opens the file with using right of the redirection symbol as name.

-By using dup2,redirects input or output to the given file.

-By using concatenated string,executes command with execl.

-----All of those functions checks returned values for operations for error check.

-----All memory allocated if also freed.

```c
#include <unistd.h>
#include <string.h>
#include <signal.h>
#include <sys/wait.h>
#include <time.h>




int fileDescriptorForLogFile = 0;  //This keeps file descriptor for log file,when shell starts,a new log file opened everytime



void redirect( char* command ); //This function makes redirecting if a command is redirecting command



void runCommands(char* commandLine); //This function takes command line that includes commands those are seperated
                                     //"|" pipe symbol and runs them with pipes,detailed explaination is at implementation

char** tokenizeCommandLine( char* commandLine );  //Takes command line and returns commands seperated with pipe symbol

char** tokenizeSingleCommand( const char* command );  //Takes a single command and tokenizes it

int checkIfCommandIsRedirection( const char* command );  //Checks if given command is redirection command

int createLogFile(); // Opens log file with the name corresponding to time stamp

void getTimeStr( char* timeStr ); //Sets timeStr to currant time with human readable time format

void writeToLogFile(int fd , int pid , char* command );  //Writes given command with given pid to the file at fd




void handleSGKILL()
{
    printf("Execution terminated by \"SGKILL\"...\n");
    close( fileDescriptorForLogFile );
    exit(1);
}
```

PART II : TESTS

```
yagiz@p-MacBook-Air hw2 % make
gcc main.c -o main
yagiz@p-MacBook-Air hw2 % ./main

->ls | sort | uniq | wc -l | awk '{print "There are " $1 " files in the current directory."}
'
There are 7 files in the current directory.

->ls -l
total 96
-rw-r--r--  1 yagiz  staff     71 14 Nis 19:13 2023-04-14&&19:13:53.txt
-rw-r--r--  1 yagiz  staff    968 14 Nis 19:15 2023-04-14&&19:14:39.txt
-rw-r--r--  1 yagiz  staff    278 14 Nis 19:15 2023-04-14&&19:15:40.txt
-rwxr-xr-x  1 yagiz  staff  34984 14 Nis 19:15 main
-rw-r--r--  1 yagiz  staff   8688 14 Nis 19:15 main.c
-rw-r--r--@ 1 yagiz  staff     45 14 Nis 17:25 makefile
-rw-r--r--  1 yagiz  staff    225 13 Nis 14:54 myfile

->cat makefile
final:
        gcc main.c -o main
clean:
        rm -f main
->echo "Hello, World!" > greeting.txt

->cat greeting.txt
Hello, World!

->ls
2023-04-14&&19:13:53.txt        2023-04-14&&19:15:40.txt        main
        makefile
2023-04-14&&19:14:39.txt        greeting.txt                    main.c
        myfile

->:q
Execution terminated by ":q" command...
yagiz@p-MacBook-Air hw2 %
```