SYSTEM PROGRAMMING HW2

TERMINAL CREATION

ELIFNUR KABALCI
1801042617

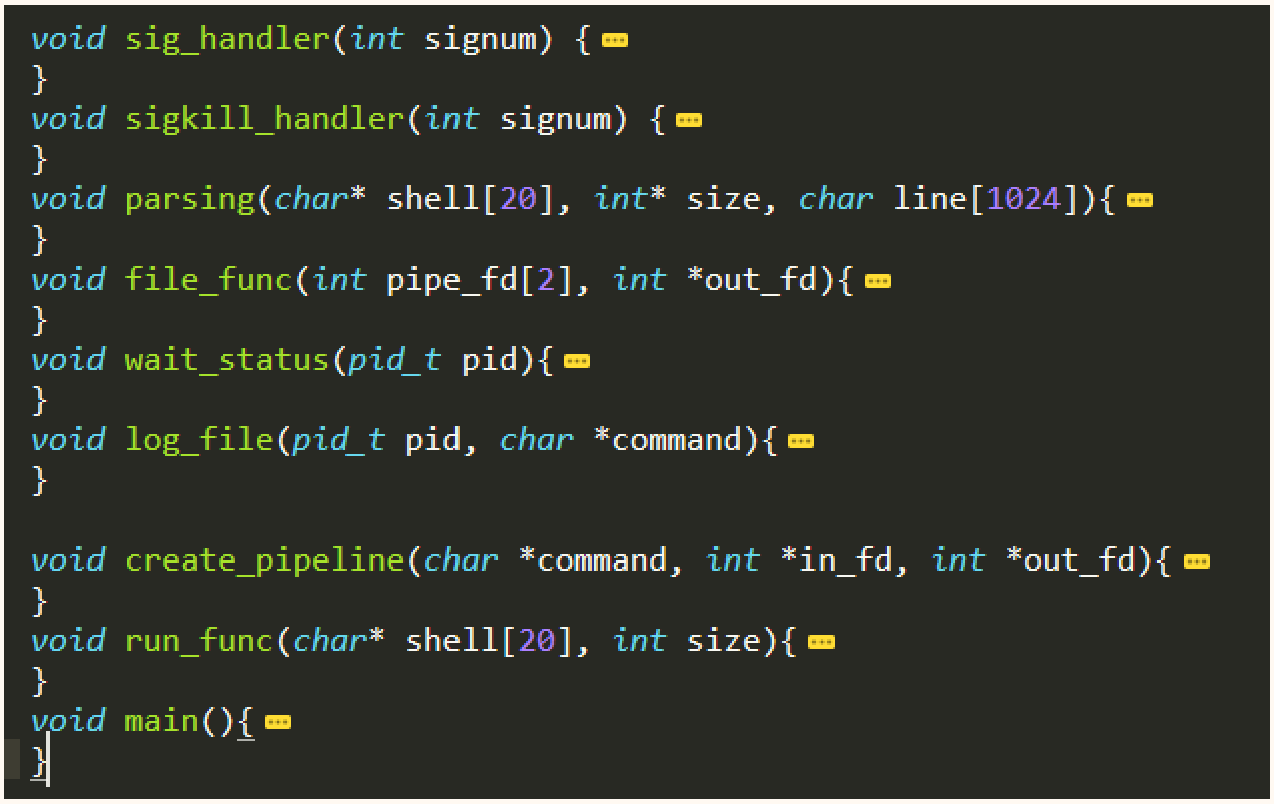
TABLE OF CONTENTS

ELIFNUR KABALCI

INTRODUCTION

The aim of this project is to design our own terminal with sh support. I made it possible for terminal commands to work and perform operations outside of Linux's own terminal. I dealt with commands in the bin/sh system.

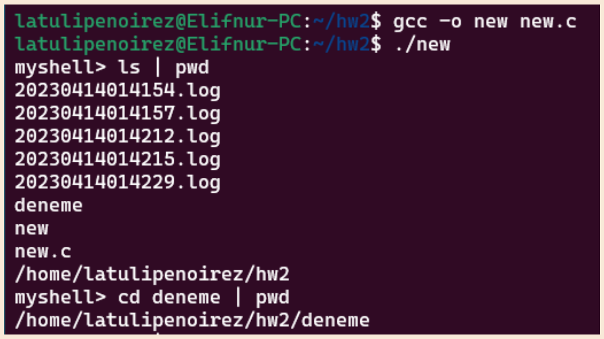
ABOUT FUNCTIONS



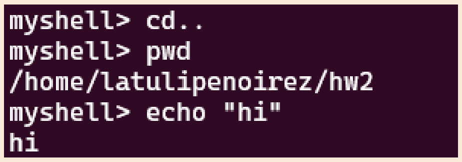
```
void sig_handler(int signum) {
}
void sigkill_handler(int signum) {
}
void parsing(char* shell[20], int* size, char line[1024]){
}
void file_func(int pipe_fd[2], int *out_fd){
}
void wait_status(pid_t pid){
}
void log_file(pid_t pid, char *command){
}

void create_pipeline(char *command, int *in_fd, int *out_fd){
}
void run_func(char* shell[20], int size){
}
void main(){
}
```

At first, I defined Signal Handler in Main. I designed a system that receives commands from the user until the user logs out. I got the data from the user with fgets because scanf ends in spaces and getline gave a pointer error. I did the fragmentation of the data received from the user with the parsing function. I transferred the data to a storage volume. Then I sent the number of commands to the function where I will execute the pipeline generator. Here I made the definition of the battery, created a child process, implemented it manually because there was no cd in the bin7sh commands. I duplicated the file definitions and linked the pipes. Then I deleted the pipes and ran my command with execl. I defined bin/sh, sh and -c in the parameter. I showed that it would continue to run with -c. Then I closed the pipe. If there is a "<" or ">" sign between the commands entered by the user, I ran the file operations. So here I wrote data to the file. In the meantime, I sent the process to wait. I created a log file with the log_file data function.

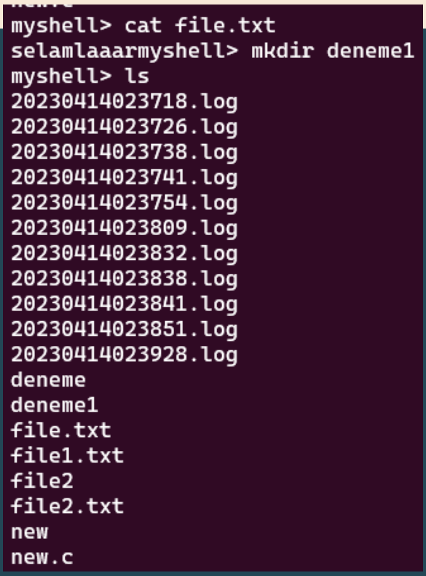
OUTPUTS



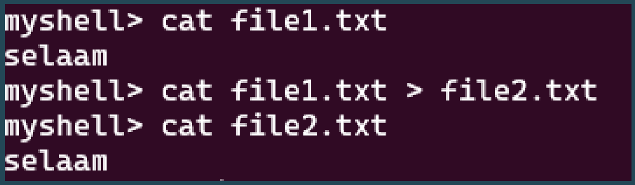
```
latulipenoirez@Elifnur-PC:~/hw2$ gcc -o new new.c
latulipenoirez@Elifnur-PC:~/hw2$ ./new
myshell> ls | pwd
20230414014154.log
20230414014157.log
20230414014212.log
20230414014215.log
20230414014229.log
deneme
new
new.c
/home/latulipenoirez/hw2
myshell> cd deneme | pwd
/home/latulipenoirez/hw2/deneme
```



```
myshell> cd..
myshell> pwd
/home/latulipenoirez/hw2
myshell> echo "hi"
hi
```

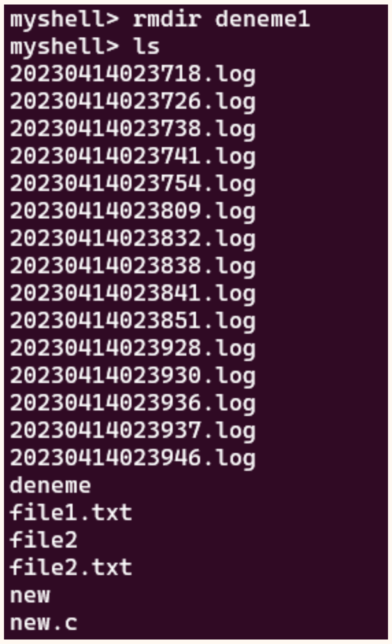


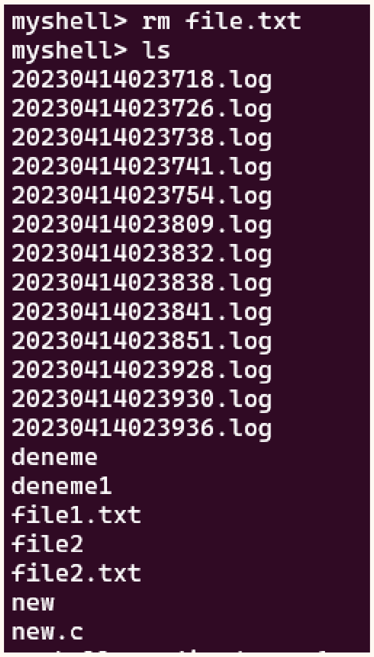
```
myshell> cat file.txt
selamlaaarmyshell> mkdir deneme1
myshell> ls
20230414023718.log
20230414023726.log
20230414023738.log
20230414023741.log
20230414023754.log
20230414023809.log
20230414023832.log
20230414023838.log
20230414023841.log
20230414023851.log
20230414023928.log
deneme
deneme1
file.txt
file1.txt
file2
file2.txt
new
new.c
```



```
myshell> cat file1.txt
selaam
myshell> cat file1.txt > file2.txt
myshell> cat file2.txt
selaam
```

1) LS, PWD, CD
2) CD.. , ECHO
3) CAT, MKDIR
4) CAT, > (FILE)


```
myshell> rmdir deneme1
myshell> ls
20230414023718.log
20230414023726.log
20230414023738.log
20230414023741.log
20230414023754.log
20230414023809.log
20230414023832.log
20230414023838.log
20230414023841.log
20230414023851.log
20230414023928.log
20230414023930.log
20230414023936.log
20230414023937.log
20230414023946.log
deneme
file1.txt
file2
file2.txt
new
new.c
```

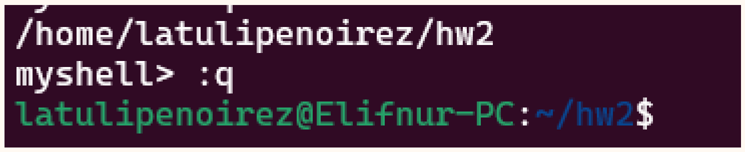

```
myshell> rm file.txt
myshell> ls
20230414023718.log
20230414023726.log
20230414023738.log
20230414023741.log
20230414023754.log
20230414023809.log
20230414023832.log
20230414023838.log
20230414023841.log
20230414023851.log
20230414023928.log
20230414023930.log
20230414023936.log
deneme
deneme1
file1.txt
file2
file2.txt
new
new.c
```

5) rmdir

6) rm

7) :q

- ➔ Other commands are working also. You can check this.
- ➔ Tests with using direction, I proof that with ls and pwd commands. You can look the ss's.
- ➔ This outputs also show that log file function is working.
- ➔ Zombie control made from waitpid().


```
/home/latulipenoirez/hw2
myshell> :q
latulipenoirez@Elifnur-PC:~/hw2$
```