CSE 344 HOMEWORK 2 REPORT

In this homework, I designed simple shell which is capable of followings.

• Each shell command should be executed via a newly created child process, meaning that multiple commands will result in multiple child processes.

• Proper handling of pipes ("|") and redirections ("<", ">") by redirecting the appropriate file descriptors.

• Usage information should be printed if the program is not called properly.

• Error messages and signals that occur during execution should be printed, and the program should return to the prompt to receive new commands.

• Aside from a SIGKILL (which also should be handled properly) the program must wait for ":q" to finalize its execution.

• Upon completion, all pids of child processes with their corresponding commands should be logged in a separate file. Each execution should create a new log file with a name corresponding to the current timestamp.

DEVELOPMENT

1)

```
int main(){
    while(1){
        get_command(); // get command from the user.

        if(fork() == 0){
            execute(command); // execute the command in the child.
        }
        else{
            wait(); // wait for the child process finished.
        }
    }
}
```

I started with this outline to accomplish each command should be executed via newly created child process instruction. According to this outline I can get input from the user and execute it in the child process and wait until child process finished. To satisfy the condition that terminal emulator capable of handling up to 20 shell commands in a single line, I removed execute(command) line from the child and add these lines.

```
for(int i = 0; i < commandsNumber - 1; i++){
    if(fork() == 0){
        execute()
    }
    else{
        wait();
    }
}
```

2)

Then I started to binding commands. Every command in a single line is separated with '|' sign. This sign provides to transfer output of the first command to the input of the second command. I create a pipe and make these arrangements in child to direct output of the command to write end of the command and input of the command to read end of the command.

```
dup2(fdin, STDIN_FILENO);

close(fdin);

dup2(fds[1], STDOUT_FILENO);

close(fds[1]);
```

3)

Then I close the write end of the pipe in the parent because of no need for write end, also read end of the pipe is assigned to fdin.

```
close(fds[1]);

fdin = fds[0];
```

4)

After all commands executed except the last command on the line, this code is added after the for loop to redirect input of the last command to read end of the pipe.

```
dup2(fdin, STDIN_FILENO);

close(fdin);
```

5)

```
typedef enum Direction{
    IN,
    OUT,
    NO
}Direction;

if(direction == IN){
    fdin = open(filename, O_RDONLY);

    if(fdin == -1){
        printf("No such a file or directory\n");
        exit(1);
    }
}


if(direction == OUT){
    int wfd = open(filename, O_WRONLY | O_CREAT| O_TRUNC, S_IWUSR | S_IRUSR);

    dup2(wfd, STDOUT_FILENO);

    close(wfd);
}
```

To achieve redirections, the code above is implemented. My program detects the '<' and '>' signs in the command and sets direction according to these signs. If the direction is OUT, the standard output of the current process is redirect to file. If the direction is IN, and file name is valid, the output of the file is redirect to standard input of the current process with additional codes.

6)

Signals are handled by signal, kill, and exit functions. When the SIGTERM or SIGINT signals are raised, the handler of the signal set the sigint or sigterm as 1. The if statements are checks whether the signals are raised or not, if the signals are raised, take necessary action.

```
int sigterm = 0;

signal(SIGTERM, sigtermHandler);

signal(SIGINT, sigintHandler);

if(sigterm == 1){
    printf("TERMINATING...\n");
    exit(1);
}

void sigintHandler(){
    exit(100);
}

void sigtermHandler(){
    sigterm = 1;
}
```

7)

To log all pids of child processes with their corresponding commands, my program contains the following code.

```
time_t current = time(NULL);

struct tm * local = localtime(&current);

char filename[50], pid[50];

strcat(path, args[0]);

strftime(filename, sizeof(filename), "%Y-%m-%d %H:%M:%S", local);

sprintf(pid, "%d", getpid());

int wfd = open(filename, O_WRONLY | O_CREAT| O_APPEND, S_IWUSR | S_IRUSR);

write(wfd, "PID: ", 5);

write(wfd, pid, strlen(pid));

write(wfd, "\nCommand: ", 10);

write(wfd, path, strlen(path));

write(wfd, "\n", 1);

close(wfd);
```

TESTS

1)

```
cagri@CSE344:~/Desktop/hw2$ make
gcc src.c -ansi -pedantic-errors -std=c99 -o hw2
cagri@CSE344:~/Desktop/hw2$ ./hw2
cagri@terminal$ ls
'2023-04-14 20:18:07'  '2023-04-14 20:18:34'  '2023-04-14 20:19:54'  '2023-04-14 20:21:38'  '2023-04-14 20:23:40'  '2023-04-14 20:24:37'   cayci.txt   odev.txt
'2023-04-14 20:18:24'  '2023-04-14 20:18:37'  '2023-04-14 20:21:12'  '2023-04-14 20:23:02'  '2023-04-14 20:23:50'   abc.txt                 hw2         readMe.txt
'2023-04-14 20:18:29'  '2023-04-14 20:19:38'  '2023-04-14 20:21:29'  '2023-04-14 20:23:06'  '2023-04-14 20:23:55'   cagri.txt               Makefile    src.c
cagri@terminal$ make clean
rm -f *.o hw2
rm -f *2023*
cagri@terminal$ ls
'2023-04-14 20:24:40'   abc.txt   cagri.txt   cayci.txt   Makefile   odev.txt   readMe.txt   src.c
cagri@terminal$ grep main src.c
int main(){
cagri@terminal$ grep main -n src.c
33:int main(){
cagri@terminal$ grp main -n src.c > cagri.txt
cagri@terminal$ cat cagri.txt
cagri@terminal$ grep main -n src.c
33:int main(){
cagri@terminal$ grep main -n src.c > cagri.txt
cagri@terminal$ cat cagri.txt
33:int main(){
cagri@terminal$ :q
EXITING...
cagri@CSE344:~/Desktop/hw2$
```

(SOME GENERAL TEST)

2)

```
cagri@CSE344:~/Desktop/hw2$ make
gcc src.c -ansi -pedantic-errors -std=c99 -o hw2
cagri@CSE344:~/Desktop/hw2$ ./hw2
cagri@terminal$ cat cayci.txt
merhaba
ben
merhaba
cagri
1901042629
ben
gtu
cagri
cagri@terminal$ cat cayci.txt | sort | uniq | wc -l
5
cagri@terminal$ :q
EXITING...
cagri@CSE344:~/Desktop/hw2$
```

(SOME GENERAL TESTS)

3)

```
cagri@CSE344:~/Desktop/hw2$ make
gcc src.c -ansi -pedantic-errors -std=c99 -o hw2
cagri@CSE344:~/Desktop/hw2$ ./hw2
cagri@terminal$ ls
'2023-04-14 20:18:07'  '2023-04-14 20:18:29'  '2023-04-14 20:18:37'  '2023-04-14 20:19:54'  '2023-04-14 20:21:29'  '2023-04-14 20:23:02'   cagri.txt   hw2        readMe.txt
'2023-04-14 20:18:24'  '2023-04-14 20:18:34'  '2023-04-14 20:19:38'  '2023-04-14 20:21:12'  '2023-04-14 20:21:38'   abc.txt             cayci.txt   Makefile   src.c
cagri@terminal$ ls | sort
2023-04-14 20:18:07
2023-04-14 20:18:24
2023-04-14 20:18:29
2023-04-14 20:18:34
2023-04-14 20:18:37
2023-04-14 20:19:38
2023-04-14 20:19:54
2023-04-14 20:21:12
2023-04-14 20:21:29
2023-04-14 20:21:38
2023-04-14 20:23:02
2023-04-14 20:23:06
abc.txt
cagri.txt
cayci.txt
hw2
Makefile
readMe.txt
src.c
cagri@terminal$ :q
EXITING...
cagri@CSE344:~/Desktop/hw2$
```

(USING PIPE)

4)

```
cagri@CSE344:~/Desktop/hw2$ make
gcc src.c -ansi -pedantic-errors -std=c99 -o hw2
cagri@CSE344:~/Desktop/hw2$ ./hw2
cagri@terminal$ ls
'2023-04-14 20:18:07'  '2023-04-14 20:18:29'  '2023-04-14 20:18:37'  '2023-04-14 20:19:54'  '2023-04-14 20:21:29'  '2023-04-14 20:23:02'  '2023-04-14 20:23:40'   cagri.txt   hw2        readMe.txt
'2023-04-14 20:18:24'  '2023-04-14 20:18:34'  '2023-04-14 20:19:38'  '2023-04-14 20:21:12'  '2023-04-14 20:21:38'  '2023-04-14 20:23:06'   abc.txt     cayci.txt   Makefile   src.c
cagri@terminal$ ls | sort > odev.txt
cagri@terminal$ cat odev.txt
2023-04-14 20:18:07
2023-04-14 20:18:24
2023-04-14 20:18:29
2023-04-14 20:18:34
2023-04-14 20:18:37
2023-04-14 20:19:38
2023-04-14 20:19:54
2023-04-14 20:21:12
2023-04-14 20:21:29
2023-04-14 20:21:38
2023-04-14 20:23:02
2023-04-14 20:23:06
2023-04-14 20:23:40
2023-04-14 20:23:50
abc.txt
cagri.txt
cayci.txt
hw2
Makefile
readMe.txt
src.c
cagri@terminal$ :q
EXITING...
cagri@CSE344:~/Desktop/hw2$
```

(USING PIPE)

5)

```
cagri@CSE344:~/Desktop/hw2$ make
gcc src.c -ansi -pedantic-errors -std=c99 -o hw2
cagri@CSE344:~/Desktop/hw2$ ./hw2
cagri@terminal$ ls
'2023-04-14 20:18:07'   abc.txt   hw2   Makefile   readMe.txt   src.c
cagri@terminal$ touch cagri.txt
cagri@terminal$ cat cagri.txt
cagri@terminal$ ls > cagri.txt
cagri@terminal$ cat cagri.txt
2023-04-14 20:18:07
2023-04-14 20:18:24
2023-04-14 20:18:29
2023-04-14 20:18:34
abc.txt
cagri.txt
hw2
Makefile
readMe.txt
src.c
cagri@terminal$ :q
EXITING...
cagri@CSE344:~/Desktop/hw2$
```

(DIRECTING TO FILE)

6)

```
cagri@CSE344:~/Desktop/hw2$ make
gcc src.c -ansi -pedantic-errors -std=c99 -o hw2
cagri@CSE344:~/Desktop/hw2$ make memoryLeak
valgrind --track-origins=yes --leak-check=full --show-leak-kinds=all ./hw2
==5579== Memcheck, a memory error detector
==5579== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5579== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==5579== Command: ./hw2
==5579==
cagri@terminal$ ls
'2023-04-14 15:48:36'  '2023-04-14 15:48:54'   hw2   Makefile   readMe.txt   src.c
cagri@terminal$ ps
    PID TTY          TIME CMD
   5526 pts/0    00:00:00 bash
   5578 pts/0    00:00:00 make
   5579 pts/0    00:00:00 memcheck-amd64-
   5581 pts/0    00:00:00 ps
cagri@terminal$ kill -SIGTERM 5578
TERMINATING...
==5579==
==5579== HEAP SUMMARY:
==5579==     in use at exit: 0 bytes in 0 blocks
==5579==   total heap usage: 2 allocs, 2 frees, 2,048 bytes allocated
==5579==
==5579== All heap blocks were freed -- no leaks are possible
==5579==
==5579== For lists of detected and suppressed errors, rerun with: -s
==5579== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
make: *** [Makefile:5: memoryLeak] Error 1
Terminated
cagri@CSE344:~/Desktop/hw2$
```

(MEMORY CHECK FOR SIGTERM)

7)

```
cagri@CSE344:~/Desktop/hw2$ make
gcc src.c -ansi -pedantic-errors -std=c99 -o hw2
cagri@CSE344:~/Desktop/hw2$ ./hw2
cagri@terminal$ cat > cayci.txt
merhaba
ben
cagri
TERMINATING(SIGINT)...
^Ccagri@CSE344:~/Desktop/hw2$ ./hw2
cagri@terminal$ ls
'2023-04-14 20:18:07'  '2023-04-14 20:18:29'  '2023-04-14 20:18:37'  '2023-04-14 20:19:54'  '2023-04-14 20:21:29'   cagri.txt   hw2        readMe.txt
'2023-04-14 20:18:24'  '2023-04-14 20:18:34'  '2023-04-14 20:19:38'  '2023-04-14 20:21:12'   abc.txt                 cayci.txt   Makefile   src.c
cagri@terminal$ cat cayci.txt
merhaba
ben
cagri
cagri@terminal$ :q
EXITING...
cagri@CSE344:~/Desktop/hw2$
```

(WRITING TO FILE)

8)

```
cagri@CSE344:~/Desktop/hw2$ make
gcc src.c -ansi -pedantic-errors -std=c99 -o hw2
cagri@CSE344:~/Desktop/hw2$ make memoryLeak
valgrind --track-origins=yes --leak-check=full --show-leak-kinds=all ./hw2
==5592== Memcheck, a memory error detector
==5592== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5592== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==5592== Command: ./hw2
==5592==
cagri@terminal$ ps
    PID TTY          TIME CMD
   5526 pts/0    00:00:00 bash
   5591 pts/0    00:00:00 make
   5592 pts/0    00:00:00 memcheck-amd64-
   5593 pts/0    00:00:00 ps
cagri@terminal$ ls
'2023-04-14 15:48:36'  '2023-04-14 15:48:54'  '2023-04-14 15:48:56'  '2023-04-14 15:49:07'  '2023-04-14 15:49:44'  '2023-04-14 15:49:46'   hw2   Makefile   readMe.txt   src.c
cagri@terminal$ ls | sort
2023-04-14 15:48:36
2023-04-14 15:48:54
2023-04-14 15:48:56
2023-04-14 15:49:07
2023-04-14 15:49:44
2023-04-14 15:49:46
2023-04-14 15:49:49
hw2
Makefile
readMe.txt
src.c
cagri@terminal$ touch abc.txt
cagri@terminal$ ls
'2023-04-14 15:48:36'  '2023-04-14 15:48:56'  '2023-04-14 15:49:44'  '2023-04-14 15:49:49'  '2023-04-14 15:50:03'   abc.txt   Makefile     src.c
'2023-04-14 15:48:54'  '2023-04-14 15:49:07'  '2023-04-14 15:49:46'  '2023-04-14 15:49:50'  '2023-04-14 15:50:05'   hw2       readMe.txt
cagri@terminal$ ls | sort > abc.txt
cagri@terminal$ cat abc.txt
2023-04-14 15:48:36
2023-04-14 15:48:54
2023-04-14 15:48:56
2023-04-14 15:49:07
2023-04-14 15:49:44
2023-04-14 15:49:46
2023-04-14 15:49:49
2023-04-14 15:49:50
2023-04-14 15:50:03
2023-04-14 15:50:05
2023-04-14 15:50:12
abc.txt
hw2
Makefile
readMe.txt
src.c
cagri@terminal$ :q
EXITING...
==5592==
==5592== HEAP SUMMARY:
==5592==     in use at exit: 0 bytes in 0 blocks
==5592==   total heap usage: 2 allocs, 2 frees, 2,048 bytes allocated
==5592==
==5592== All heap blocks were freed -- no leaks are possible
==5592==
==5592== For lists of detected and suppressed errors, rerun with: -s
==5592== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cagri@CSE344:~/Desktop/hw2$
```

(MEMORY CHECK FOR VALID COMMANDS)

9)

```
cagri@CSE344:~/Desktop/hw2$ make
gcc src.c -ansi -pedantic-errors -std=c99 -o hw2
cagri@CSE344:~/Desktop/hw2$ make memoryLeak
valgrind --track-origins=yes --leak-check=full --show-leak-kinds=all ./hw2
==5455== Memcheck, a memory error detector
==5455== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==5455== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==5455== Command: ./hw2
==5455==
cagri@terminal$ :Q
==5456==
==5456== HEAP SUMMARY:
==5456==     in use at exit: 0 bytes in 0 blocks
==5456==   total heap usage: 14 allocs, 14 frees, 8,000 bytes allocated
==5456==
==5456== All heap blocks were freed -- no leaks are possible
==5456==
==5456== For lists of detected and suppressed errors, rerun with: -s
==5456== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cagri@terminal$ :q
EXITING...
==5455==
==5455== HEAP SUMMARY:
==5455==     in use at exit: 0 bytes in 0 blocks
==5455==   total heap usage: 2 allocs, 2 frees, 2,048 bytes allocated
==5455==
==5455== All heap blocks were freed -- no leaks are possible
==5455==
==5455== For lists of detected and suppressed errors, rerun with: -s
==5455== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cagri@CSE344:~/Desktop/hw2$
```

(MEMORY CHECK FOR INVALID COMMANDS)

10)

```
cagri@CSE344:~/Desktop/hw2$ make
gcc src.c -ansi -pedantic-errors -std=c99 -o hw2
cagri@CSE344:~/Desktop/hw2$ ./hw2
cagri@terminal$ cat cagri.txt
2023-04-14 20:18:07
2023-04-14 20:18:24
2023-04-14 20:18:29
2023-04-14 20:18:34
abc.txt
cagri.txt
hw2
Makefile
readMe.txt
src.c
cagri@terminal$ sort -r < cagri.txt
src.c
readMe.txt
Makefile
hw2
cagri.txt
abc.txt
2023-04-14 20:18:34
2023-04-14 20:18:29
2023-04-14 20:18:24
2023-04-14 20:18:07
cagri@terminal$ :q
EXITING...
cagri@CSE344:~/Desktop/hw2$
```

(DIRECTING TO FILE)