CSE-344 HOMEWORK-1 REPORT

AHMET OKUR

1801042655

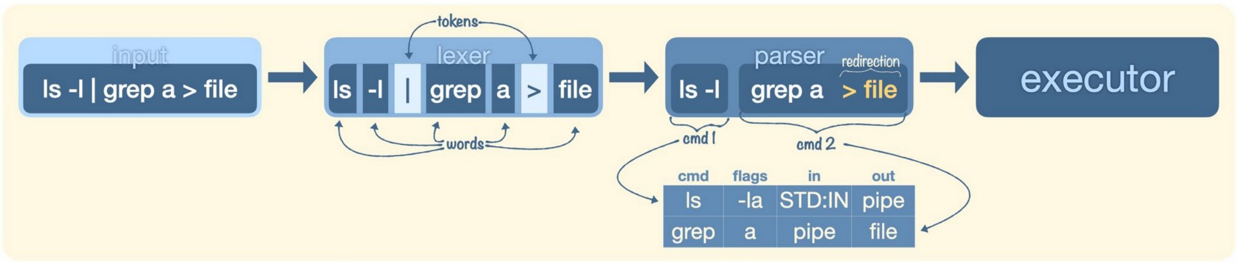
Problem Defination

In this homework, We develop a terminal emulator capable of handling up to 20 shell commands in a single line, without using the"system()"function from the standard C library. Instead, you should utilize the “fork()”, “execl()”, “wait()”, and “exit()”functions.

threads.

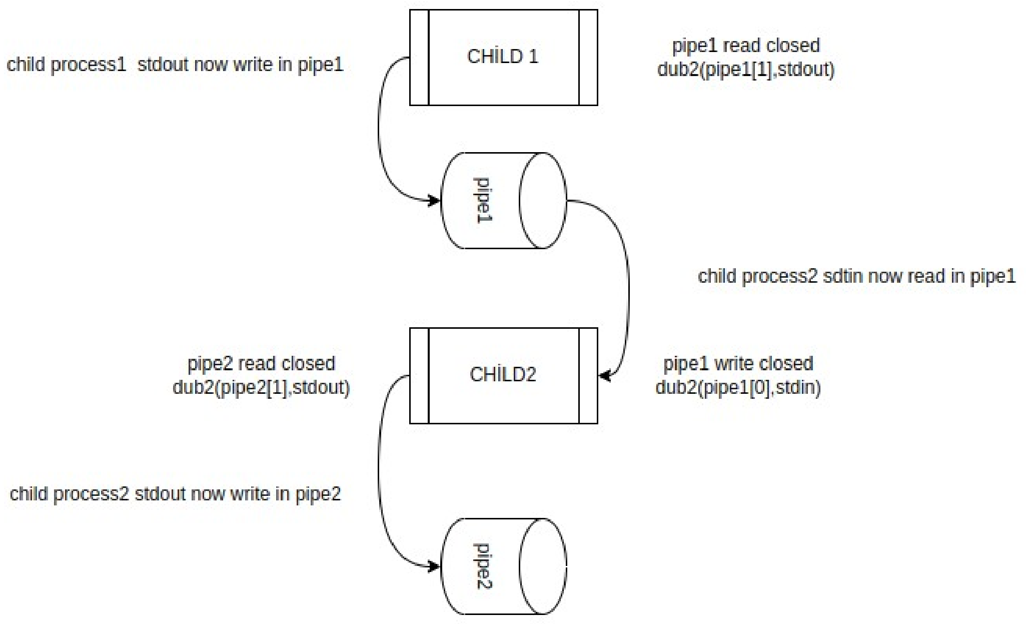
Plan & Design

- First i parse the entered command each allocated command represents a process
- I created a pipe for these
- I got process -1 pipe
- Below is a representative image

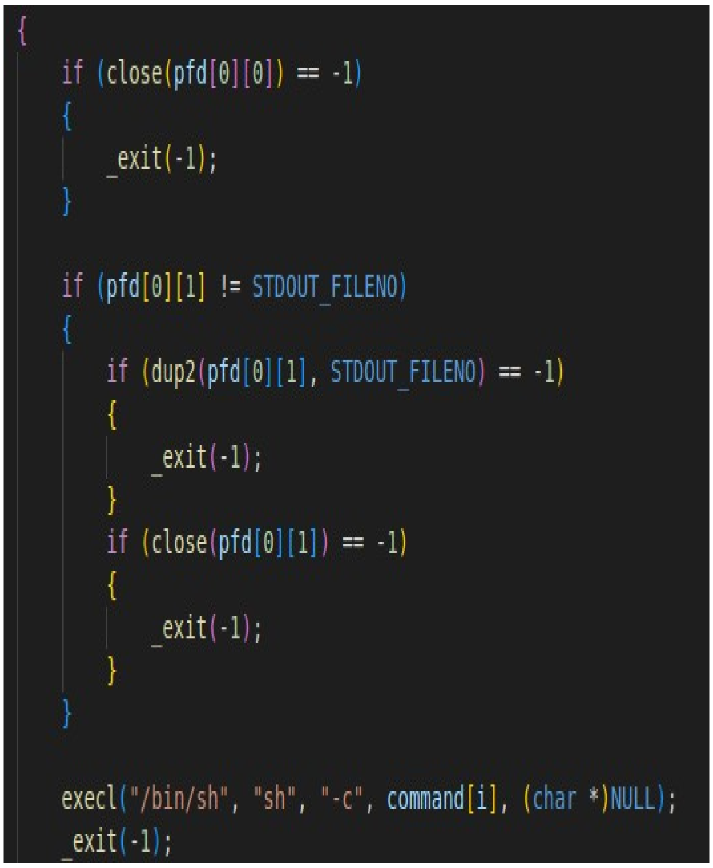


- When each process starts, it is checked to see if there is any redirection in it.
- I implemented the redirection command with execl, I will explain the detailed information below.
- The reading area of the pipe is closed when the process starts first
- The write area is duplicated using the stdout a dub 2 function.
- The child process started after it will read from the pipe and duplicate the write area for the next pipe. It repeats like this. I developed these commands by taking the lecture slides

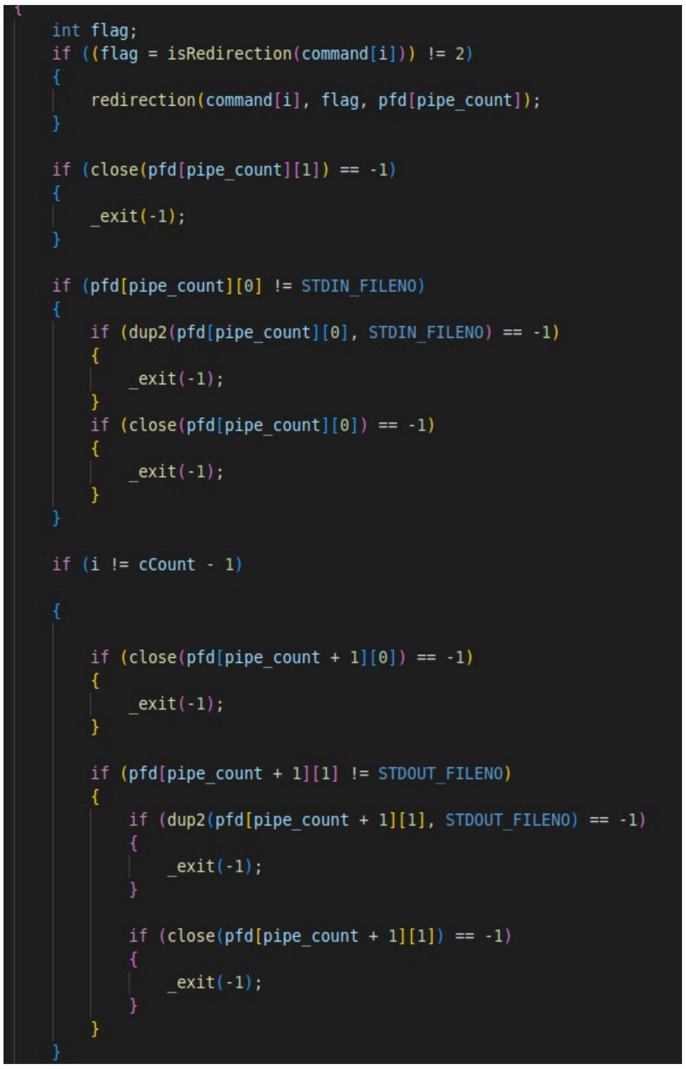
PIPE AND CHILD PROCESSES



This is my draft in a nutshell. works in a loop



```c
{
    if (close(pfd[0][0]) == -1)
    {
        _exit(-1);
    }

    if (pfd[0][1] != STDOUT_FILENO)
    {
        if (dup2(pfd[0][1], STDOUT_FILENO) == -1)
        {
            _exit(-1);
        }
        if (close(pfd[0][1]) == -1)
        {
            _exit(-1);
        }
    }

    execl("/bin/sh", "sh", "-c", command[i], (char *)NULL);
    _exit(-1);
```

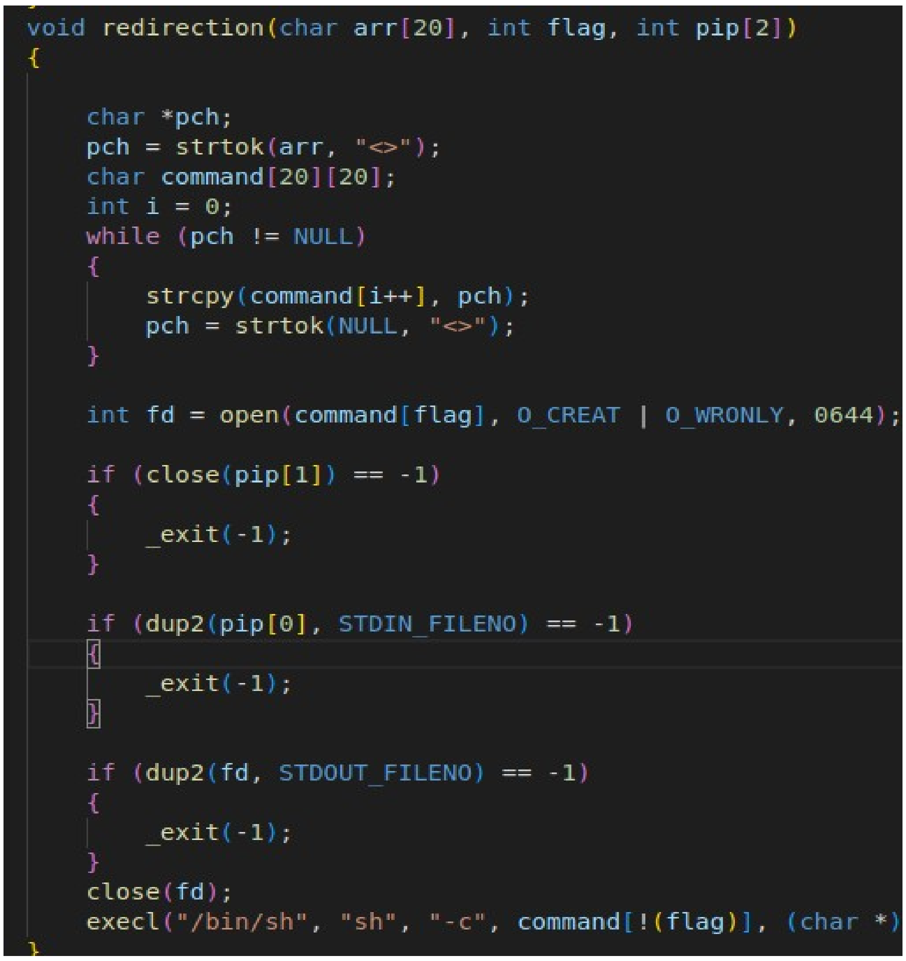


```c
{
    int flag;
    if ((flag = isRedirection(command[i])) != 2)
    {
        redirection(command[i], flag, pfd[pipe_count]);
    }

    if (close(pfd[pipe_count][1]) == -1)
    {
        _exit(-1);
    }

    if (pfd[pipe_count][0] != STDIN_FILENO)
    {
        if (dup2(pfd[pipe_count][0], STDIN_FILENO) == -1)
        {
            _exit(-1);
        }
        if (close(pfd[pipe_count][0]) == -1)
        {
            _exit(-1);
        }
    }

    if (i != cCount - 1)

    {

        if (close(pfd[pipe_count + 1][0]) == -1)
        {
            _exit(-1);
        }

        if (pfd[pipe_count + 1][1] != STDOUT_FILENO)
        {
            if (dup2(pfd[pipe_count + 1][1], STDOUT_FILENO) == -1)
            {
                _exit(-1);
            }

            if (close(pfd[pipe_count + 1][1]) == -1)
            {
                _exit(-1);
            }
        }
    }
```

Code implementation of the schema

REDIRECTION <> implementation

- Implemented the redirection myself using execl and open, definitely there is no automatic production of execl
- First we determine the direction
- First i opened the file with an open command
- I turned off the write area of the pipe
- Then I duplicated the stdin with the pipe's reading area
- Now the value read from the pipe can be written to the file
- I then dublice the file descriptor with stdout. The value from stdin will be written to the file with stdout
- Then I called execl and execl will have written what it received with stdin to the file with stdout.

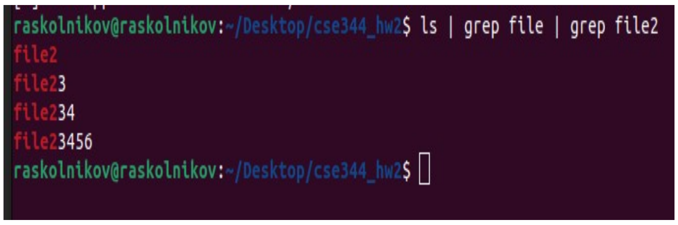


```c
void redirection(char arr[20], int flag, int pip[2])
{

    char *pch;
    pch = strtok(arr, "<>");
    char command[20][20];
    int i = 0;
    while (pch != NULL)
    {
        strcpy(command[i++], pch);
        pch = strtok(NULL, "<>");
    }

    int fd = open(command[flag], O_CREAT | O_WRONLY, 0644);

    if (close(pip[1]) == -1)
    {
        _exit(-1);
    }

    if (dup2(pip[0], STDIN_FILENO) == -1)
    {
        _exit(-1);
    }

    if (dup2(fd, STDOUT_FILENO) == -1)
    {
        _exit(-1);
    }
    close(fd);
    execl("/bin/sh", "sh", "-c", command[!(flag)], (char *)
}
```

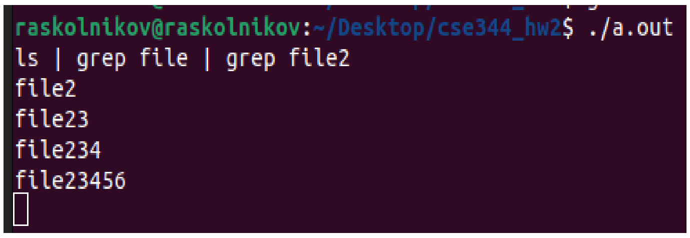
`This is my redirection implementation`

TEST

ls | grep file | grep file2



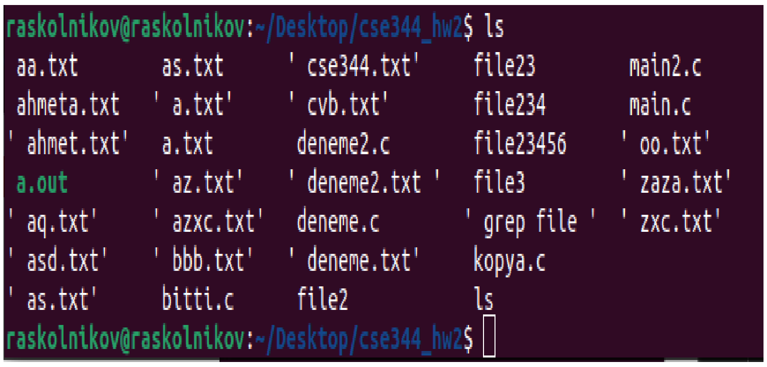
```
raskolnikov@raskolnikov:~/Desktop/cse344_hw2$ ls | grep file | grep file2
file2
file23
file234
file23456
raskolnikov@raskolnikov:~/Desktop/cse344_hw2$
```



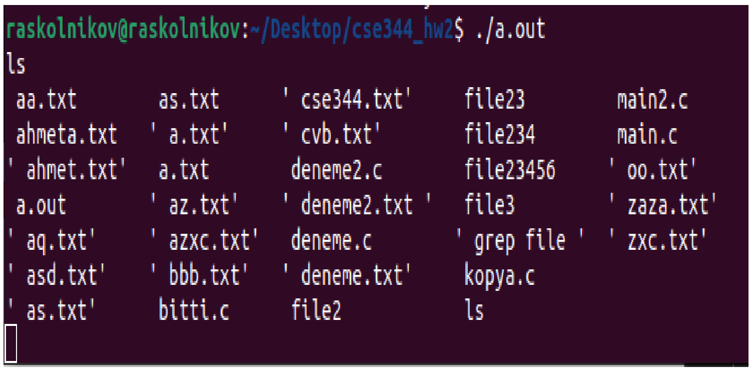
```
raskolnikov@raskolnikov:~/Desktop/cse344_hw2$ ./a.out
ls | grep file | grep file2
file2
file23
file234
file23456
```

Real shell **My shell**

ls



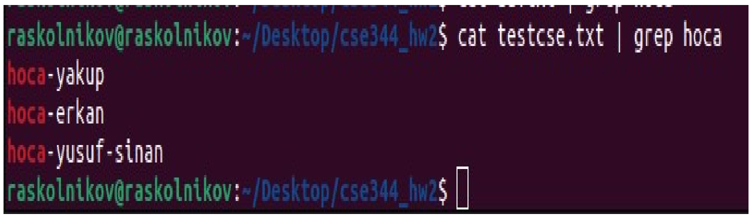
```
raskolnikov@raskolnikov:~/Desktop/cse344_hw2$ ls
 aa.txt        as.txt      ' cse344.txt'    file23        main2.c
 ahmeta.txt   ' a.txt'     ' cvb.txt'       file234       main.c
' ahmet.txt'   a.txt        deneme2.c       file23456    ' oo.txt'
 a.out        ' az.txt'    ' deneme2.txt '  file3        ' zaza.txt'
' aq.txt'     ' azxc.txt'   deneme.c       ' grep file ' ' zxc.txt'
' asd.txt'    ' bbb.txt'   ' deneme.txt'    kopya.c
' as.txt'      bitti.c      file2           ls
raskolnikov@raskolnikov:~/Desktop/cse344_hw2$
```



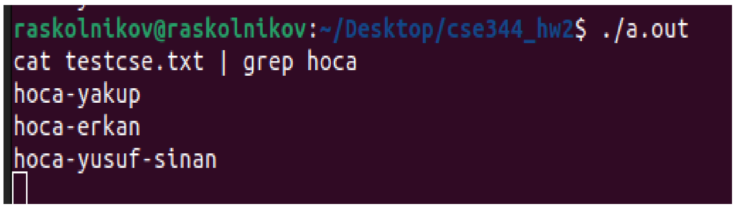
```
raskolnikov@raskolnikov:~/Desktop/cse344_hw2$ ./a.out
ls
 aa.txt        as.txt      ' cse344.txt'    file23        main2.c
 ahmeta.txt   ' a.txt'     ' cvb.txt'       file234       main.c
' ahmet.txt'   a.txt        deneme2.c       file23456    ' oo.txt'
 a.out        ' az.txt'    ' deneme2.txt '  file3        ' zaza.txt'
' aq.txt'     ' azxc.txt'   deneme.c       ' grep file ' ' zxc.txt'
' asd.txt'    ' bbb.txt'   ' deneme.txt'    kopya.c
' as.txt'      bitti.c      file2           ls
```

Real shell **My shell**

cat testcse.txt | grep hoca

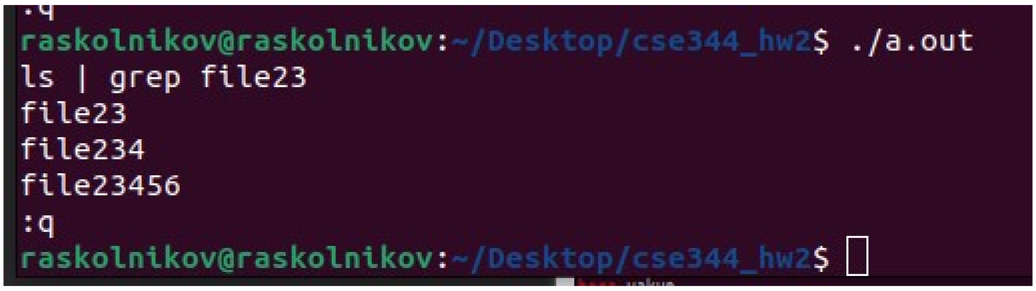


```
raskolnikov@raskolnikov:~/Desktop/cse344_hw2$ cat testcse.txt | grep hoca
hoca-yakup
hoca-erkan
hoca-yusuf-sinan
raskolnikov@raskolnikov:~/Desktop/cse344_hw2$
```



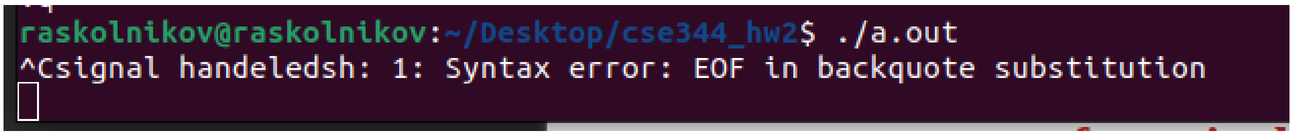
```
raskolnikov@raskolnikov:~/Desktop/cse344_hw2$ ./a.out
cat testcse.txt | grep hoca
hoca-yakup
hoca-erkan
hoca-yusuf-sinan
```

Real shell **My shell**



```
raskolnikov@raskolnikov:~/Desktop/cse344_hw2$ ./a.out
ls | grep file23
file23
file234
file23456
:q
raskolnikov@raskolnikov:~/Desktop/cse344_hw2$
```

for exit shell :q



```
raskolnikov@raskolnikov:~/Desktop/cse344_hw2$ ./a.out
^Csignal handeledsh: 1: Syntax error: EOF in backquote substitution
```

ctrl c handled