Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №1 по курсу

«Операционные системы»

Группа: М80-206Б-20

Студент: Ларченко А.О.

Преподаватель: Миронов Е.С.

Оценка: _____

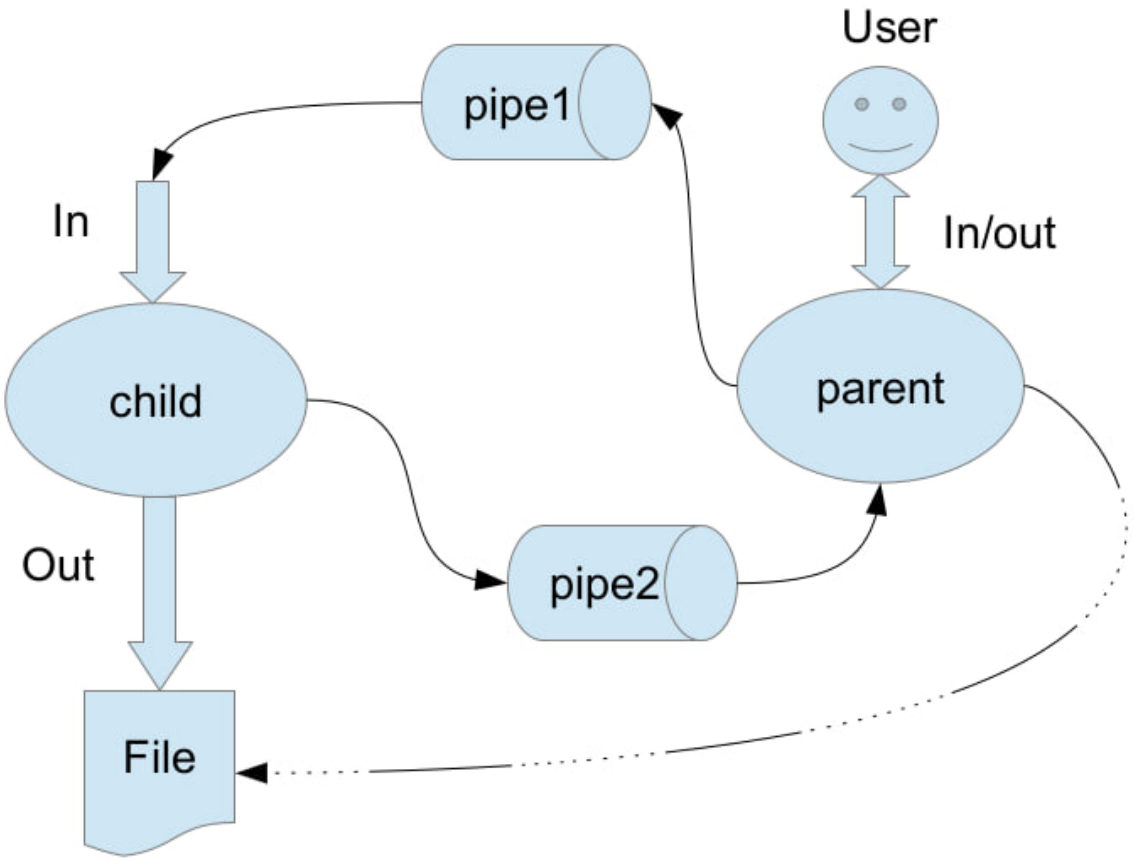
Дата: 06.10.23

Москва, 2023

Постановка задачи

Вариант 15.

Группа вариантов 4



Родительский процесс создает дочерний процесс. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child проверяет строки на валидность правилу. Если строка соответствует правилу, то она выводится в стандартный поток вывода дочернего процесса, иначе в pipe2 выводится информация об ошибке. Родительский процесс полученные от child ошибки выводит в стандартный поток вывода.

Правило проверки: строка должна начинаться с заглавной буквы

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork(void); – создает дочерний процесс.
- int pipe(int *fd); – создает канал, который используется для связи дочернего и родительского процессов
- ssize_t write(int fd, const void buf[count], size_t count) - записывает size_t count байт в указанный файловый дескриптор fd

- ssize_t read(int fd, void buf[.count], size_t count) - считывает size_t count байт в указанный файловый дескриптор fd
- int open(const char *pathname, int flags, mode_t mode) - открывает и создает файл(если мы укажем такой флаг)
- int close(int fd) - закрывает файловый дескриптор fd
- int dup2(int oldfd, int newfd) - дублирует файловый дескриптор newfd на место дескриптора oldfd
- int execl(const char *pathname, const char *arg, .../*, (char *) NULL */) - исполняет указанные файлы

Первым делом, моя программа получает на вход строку, которая будет являться именем файла, который будет создан или открыт(если такой есть). Но перед этим я обрабатываю вводную строку. Изначально я не знаю, какой длины будет строка, но прикинул, что имя файла не будет длиннее 100 символов. Поэтому введенные данные я помещаю в статический массив длинной 100(MAX_LEN). После этого я должен обработать входную строку и очистить её от ‘\n’, который тоже записывается в строку, а также игнорировать пустые байты. Это действие я выполняю с помощью функций “clean_name”, которая в свою очередь использует функцию “str_size” для определения настоящей длины исходной строки. Полученную строку я присваиваю динамической строке Filename.

Командой “open” я открываю/создаю файл с именем Filename и полным набором доступа.

Затем с помощью функций-оберток “pipe_creation” и “process_creation” я создаю 2 канала и 1 дочерний процесс.

Если созданный процесс - ребенок(“fork” вернул 0), то я закрываю ненужные для дочернего процесса дескрипторы (по заданию) и подменяю стандартные потоки(ввода, вывода и ошибок) для дочернего процесса с помощью функции “dub2” на “pipe_1” - новый поток ввода, “pipe_2” - новый поток ошибок и “f_input” (открытый в начале программы файл) - на новый поток вывода. После этого запускаю дочерний процесс.

Если процесс - взрослый, то закрываю дескрипторы, ненужные для родительского процесса, а после записываю в канал имя файла.

В дочернем процессе происходит проверка на правило(имя файла должно начинаться с заглавной латинской буквы). Если имя файла удовлетворяет условию, то в стандартный поток вывода(уже подмененный) записывается это имя файла. В противном случае - в поток ошибок отправляется пустая строка.

После выполнения дочернего процесса родительский проверяет поток вывода и ошибок. В зависимости от результата выводит соответствующее сообщение.

Код программы

main.c

```
#include <stdio.h>
#include "function.h"
```

```
int main(){
    char str_input[MAX_LEN];
    write(STDOUT_FILENO, "Enter filename with file extension: ", 37);
    read(STDIN_FILENO, str_input, MAX_LEN);
    char *Filename=NULL;
    if(clean_name(&Filename, str_input)==false){
        perror("Trying to create 0-value string: ");
        exit(-1);
    }
    int f_input=open(Filename, O_WRONLY | O_CREAT, 0777);
    // FILE* f_input =fopen(Filename, "w");
    if(f_input==-1){
        fprintf(stderr, "Can't open the file:  %s", Filename);
        exit(-1);
    }
    int pipe_1[2], pipe_2[2];
    pipe_creation(pipe_1);
    pipe_creation(pipe_2);
    int pid=process_creation();
    if(pid==0){
        // printf("Its child\n");
        close(pipe_1[1]); //unsed fd_pipe_1 for writing
        close(pipe_2[0]);//unsed fd_pipe_2 for reading
        if(dup2(pipe_1[0], STDIN_FILENO)==-1){

            perror("Call dup2 was ended with erorr: ");
            exit(-1);
        }
        if(dup2(f_input, STDOUT_FILENO)==-1){
            perror("Call dup2 was ended with erorr: ");
            exit(-1);
        }
        if(dup2(pipe_2[1], STDERR_FILENO)==-1){
            perror("Call dup2 was ended with erorr: ");
            exit(-1);
        }

        if(execl("./child", "./child", NULL)==-1){
            perror("Call execl was ended with erorr: ");
            exit(-1);
        }
        close(pipe_1[0]);
        close(pipe_2[1]);
        close(f_input);

    }else{ //it's parant
        // printf("Its parant");
        close(f_input);
        close(pipe_1[0]); //unsed fd_pipe_1 for reading
```

```
        close(pipe_2[1]); //unsed fd_pipe_2 for writing

        write(pipe_1[1], Filename, str_size(Filename));

        wait(NULL);
        check_res(pipe_2[0], STDOUT_FILENO);
        // write(STDOUT_FILENO, "\n", sizeof("\n"));
        write(STDOUT_FILENO, GREEN_COLOR, sizeof(GREEN_COLOR));
        write(STDOUT_FILENO, "\n", sizeof("\n"));
        write(STDOUT_FILENO, message_list[5], str_size(message_list[5]));
        write(STDOUT_FILENO, "\n", sizeof("\n"));
        write(STDOUT_FILENO, RESET_COLOR, sizeof(RESET_COLOR));
        close(pipe_1[1]);
        close(pipe_2[0]);
    }
}
```

function.h

```
#ifndef function_h
#define function_h
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdbool.h>
#include <fcntl.h>
#include <sys/wait.h>

#define MAX_LEN 100

#define RED_COLOR    "\x1b[31m"
#define GREEN_COLOR  "\x1b[32m"
#define RESET_COLOR    "\x1b[0m"

extern const char*  message_list[6];

void pipe_creation(int *fd);
int process_creation();
int str_size(const char *string);
bool clean_name(char **output_name, char* input_name);
void check_res(int fd_in, int fd_out);

#endif
```

function.c

```
#include <stdio.h>
#include "function.h"
```

```
void pipe_creation(int *fd){
    if(pipe(fd)==-1){
        perror("Call pipe was ended with error: ");
        exit(-1);
    }

}

int process_creation(){
    int pid =fork();
    if(pid==-1){
        perror("Call fork was ended with erorr: ");
        exit(-1);
    }
    return pid;
}

int str_size(const char *string){
    int len=0;
    for(int i=0; i<MAX_LEN; ++i){ // Fix reading '\n' bag and input string lenth
counting
        if(string[i]=='\n' || string[i]==EOF || string[i]=='\0'){
            break;
        }
        len++;
    }
    return len;
}

bool clean_name(char **output_name, char* input_name){
    int len=str_size(input_name);
    if(len==0){
        return false;
    }
    char tmp[len+1];
    for(int i=0; i<len;++i){
        tmp[i]=input_name[i];
    }
    tmp[len]='\0';
    free(*output_name);
    *output_name=tmp;
    return true;



}

void check_res(int fd_in, int fd_out){
    char str_input[MAX_LEN];
    read(fd_in, str_input, MAX_LEN);
```

```
    int len=str_size(str_input);
    // printf("%d", len);
    if(len!=0){
        char message[]="Correct filename.\n";
        write(fd_out, GREEN_COLOR, sizeof(GREEN_COLOR));
        write(fd_out, message, str_size(message));
        write(fd_out, RESET_COLOR, sizeof(RESET_COLOR));
    }else{
        write(fd_out, RED_COLOR, sizeof(RED_COLOR));
        write(fd_out, message_list[0], str_size(message_list[0]));
        write(fd_out, RESET_COLOR, sizeof(RESET_COLOR));
    }

}

const char*  message_list[]={
    //Errors:
    "Erorr!_Uncorrect input. Filename must begin with a capital letter!\n",
    "Call pipe was ended with error: ",
    "Call fork was ended with erorr: ",
    "Trying to create 0-value string: ",
    //Normal status
    "Enter filename with file extension: ",
    "Program was ended succsessfully!\n\n",

};
```

child.c

```
#include <stdio.h>
#include "function.h"

bool check_first_size(char a){
    if(a>='A' && a<='Z'){
        return true;
    }
    return false;
}

int main(){
    while(1){
        char input_name[MAX_LEN];
        read(STDIN_FILENO, input_name, sizeof(input_name));

        char* output_name=NULL;
        if(check_first_size(input_name[0])==true){

            clean_name(&output_name, input_name);
```

```
            write(STDOUT_FILENO, output_name, str_size(output_name));
            break;
        } else{
            write(STDERR_FILENO, "\0", sizeof("\0"));
            break;
        }
    }
    return 0;
}
```

Протокол работы программы

Тестирование:

```
arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/laba_1$ ls
child.c  function.c  function.h  main.c  Makefile  rm_txt.sh

arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/laba_1$ make
gcc -std=c99 -pedantic -Wall child.c function.c -o child
gcc -std=c99 -pedantic -Wall main.c function.c -o main

arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/laba_1$ ./main
Enter filename with file extension: Oo.txt
Correct filename.
Program was ended successfully!

arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/laba_1$ ls
child  child.c  function.c  function.h  main  main.c  Makefile  Oo.txt  rm_txt.sh

arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/laba_1$ cat Oo.txt
Oo.txt

================================================================================
arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/laba_1$ ./main
Enter filename with file extension: oo.txt
Error!_Uncorrect input. Filename must begin with a capital letter!
Program was ended successfully!

arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/laba_1$ ls
child  child.c  function.c  function.h  main  main.c  Makefile  oo.txt  Oo.txt  r
rm_txt.sh

arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/laba_1$ cat oo.txt
```

```
arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/laba_1$

================================================================================
arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/laba_1$ ./main
Enter filename with file extension: 000.txt
Error!_Uncorrect input. Filename must begin with a capital letter!
Program was ended successfully!

arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/laba_1$ ls
000.txt  child  child.c  function.c  function.h  main  main.c  Makefile  oo.txt  Oo.txt
rm_txt.sh

arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/laba_1$ cat 000.txt
arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/laba_1$
```

Strace:

```
arsenii@PC-Larcha14:~/Documents/VS_code_prog/OSI/laba_1$ strace -f ./main
execve("./main", ["./main"], 0x7fff59b2b2e8 /* 56 vars */) = 0
brk(NULL)                               = 0x5578b7997000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe7698bbd0) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
-1, 0) = 0x7f9983f3b000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=80191, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 80191, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f9983f27000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\"\233}\305\t\5?\344\337^)\350b\231\21\360"..., 68,
896) = 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2216304, ...}, AT_EMPTY_PATH) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f9983c00000
mmap(0x7f9983c28000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f9983c28000
mmap(0x7f9983dbd000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f9983dbd000
mmap(0x7f9983e15000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000) = 0x7f9983e15000
mmap(0x7f9983e1b000, 52816, PROT_READ|PROT_WRITE
, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f9983e1b000
close(3)                                = 0
```

mmap(NULL, 12288, PROT_READ|PROT_WRITE
, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f9983f24000
arch_prctl(ARCH_SET_FS, 0x7f9983f24740) = 0
set_tid_address(0x7f9983f24a10) = 13034
set_robust_list(0x7f9983f24a20, 24) = 0
rseq(0x7f9983f250e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7f9983e15000, 16384, PROT_READ) = 0
mprotect(0x5578b6075000, 4096, PROT_READ) = 0
mprotect(0x7f9983f75000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7f9983f27000, 80191) = 0
write(1, "Enter filename with file extensi"..., 37Enter filename with file extension:) = 37
read(0, O.txt
"O.txt\n", 100) = 6
openat(AT_FDCWD, "O.txt", O_WRONLY|O_CREAT, 0777) = 3
pipe2([4, 5], 0) = 0
pipe2([6, 7], 0) = 0
clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process
13037 attached
, child_tidptr=0x7f9983f24a10) = 13037
[pid 13034] close(3 <unfinished ...>
[pid 13037] set_robust_list(0x7f9983f24a20, 24 <unfinished ...>
[pid 13034] <... close resumed>) = 0
[pid 13037] <... set_robust_list resumed>) = 0
[pid 13034] close(4) = 0
[pid 13037] close(5 <unfinished ...>
[pid 13034] close(7) = 0
[pid 13034] write(5, "O.txt", 5 <unfinished ...>
[pid 13037] <... close resumed>) = 0
[pid 13034] <... write resumed>) = 5
[pid 13034] wait4(-1, <unfinished ...>
[pid 13037] close(6) = 0
[pid 13037] dup2(4, 0) = 0
[pid 13037] dup2(3, 1) = 1
[pid 13037] dup2(7, 2) = 2
[pid 13037] execve("./child", ["./child"], 0x7ffe7698bda8 /* 56 vars */) = 0
[pid 13037] brk(NULL) = 0x563157d7c000
[pid 13037] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe04d4f340) = -1 EINVAL (Invalid
argument)
[pid 13037] mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fe7e6001000
[pid 13037] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
[pid 13037] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 5
[pid 13037] newfstatat(5, "", {st_mode=S_IFREG|0644, st_size=80191, ...},

```
AT_EMPTY_PATH) = 0
[pid 13037] mmap(NULL, 80191, PROT_READ, MAP_PRIVATE, 5, 0) = 0x7fe7e5fed000
[pid 13037] close(5)                    = 0
[pid 13037] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6",
O_RDONLY|O_CLOEXEC) = 5
[pid 13037] read(5, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0"..., 832) =
832
[pid 13037] pread64(5, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784,
64) = 784
[pid 13037] pread64(5, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48,
848) = 48
[pid 13037] pread64(5,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\"\233}\305\t\5?\344\337^)\350b\231\21\360"..., 68, 896) = 68
[pid 13037] newfstatat(5, "", {st_mode=S_IFREG|0755, st_size=2216304, ...},
AT_EMPTY_PATH) = 0
[pid 13037] pread64(5, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784,
64) = 784
[pid 13037] mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 5,
0) = 0x7fe7e5c00000
[pid 13037] mmap(0x7fe7e5c28000, 1658880, PROT_READ|PROT_EXEC,
 MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 5, 0x28000) = 0x7fe7e5c28000
[pid 13037] mmap(0x7fe7e5dbd000, 360448, PROT_READ
, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 5, 0x1bd000) = 0x7fe7e5dbd000
[pid 13037] mmap(0x7fe7e5e15000, 24576, PROT_READ|PROT_WRITE
, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 5, 0x214000) = 0x7fe7e5e15000
[pid 13037] mmap(0x7fe7e5e1b000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fe7e5e1b000
[pid 13037] close(5)                    = 0
[pid 13037] mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fe7e5fea000
[pid 13037] arch_prctl(ARCH_SET_FS, 0x7fe7e5fea740) = 0
[pid 13037] set_tid_address(0x7fe7e5feaa10) = 13037
[pid 13037] set_robust_list(0x7fe7e5feaa20, 24) = 0
[pid 13037] rseq(0x7fe7e5feb0e0, 0x20, 0, 0x53053053) = 0
[pid 13037] mprotect(0x7fe7e5e15000, 16384, PROT_READ) = 0
[pid 13037] mprotect(0x563156ca0000, 4096, PROT_READ) = 0
[pid 13037] mprotect(0x7fe7e603b000, 8192, PROT_READ) = 0
[pid 13037] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
[pid 13037] munmap(0x7fe7e5fed000, 80191) = 0
[pid 13037] read(0, "O.txt", 100)       = 5
[pid 13037] write(1, "O.txt", 5)        = 5
[pid 13037] exit_group(0)               = ?
[pid 13037] +++ exited with 0 +++
<... wait4 resumed>NULL, 0, NULL)       = 13037
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=13037, si_uid=1000,
```

```
si_status=0, si_utime=0, si_stime=0} ---
read(6, "", 100)                        = 0
write(1, "\33[32m\0", 6)                = 6
write(1, "Correct filename.", 17Correct filename.)       = 17
write(1, "\33[0m\0", 5)                 = 5
write(1, "\33[32m\0", 6)                = 6
write(1, "\n\0", 2
)                       = 2
write(1, "Program was ended succsessfully!", 32Program was ended succsessfully!) = 32
write(1, "\n\0", 2
)                       = 2
write(1, "\33[0m\0", 5)                 = 5
close(5)                                = 0
close(6)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

В этой лабораторной работе я познакомился с системными вызовами, IPC(межпроцессорным взаимодействием), каналами и потоками. Было много новой информации, а вместе с ней много новых функций (pipe, dub2, execl, fork), но несмотря на это, я смог разобраться и осознать новые знания и реализовать их на практике.

Проблемы были, но они никак не были связаны с новой темой. Писал на си, а работать нужно было со входной строкой. Т.е. проблемы возникли только с обработкой входного текста. Но тут уже программист сам дурак…

В итоге у меня получился исправно работающий код, считаю, что с поставленной задачей справился успешно.