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.

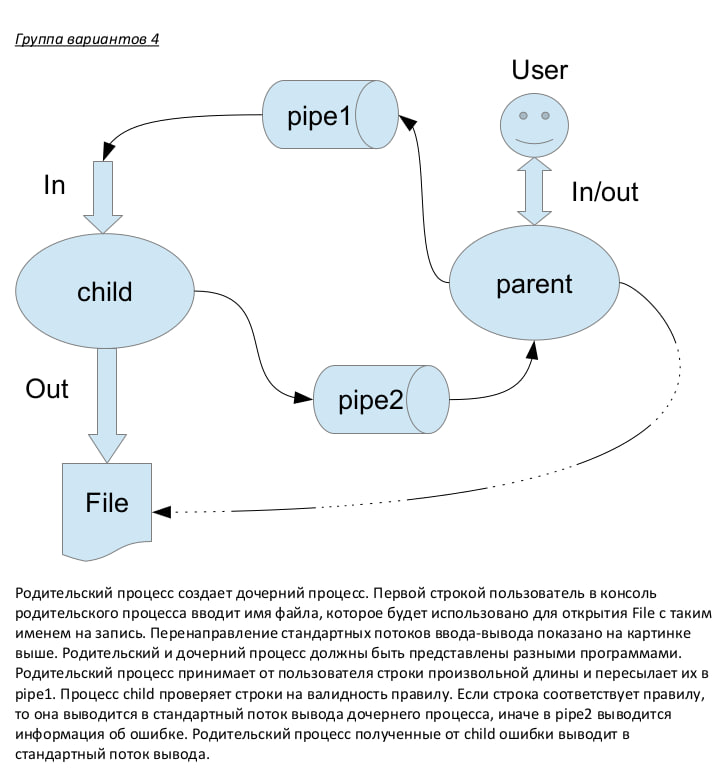
Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 06.10.23

Москва, 2023

**Постановка задачи**

**Вариант 15.**

****

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\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.