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-22

Студент: Сарайкин Н.С.

Преподаватель: Миронов Е.С.

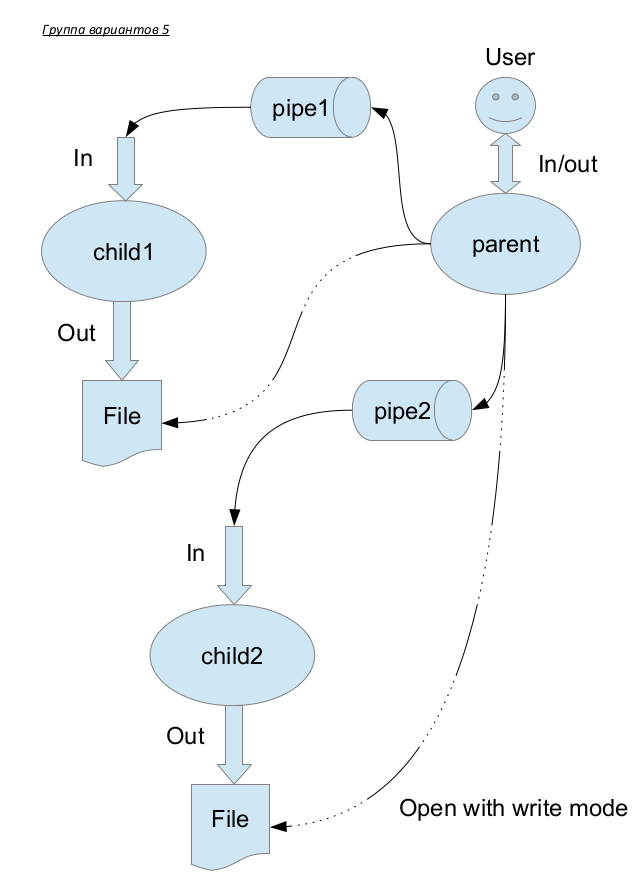
Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

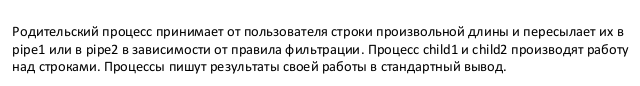
Дата: 03.11.23

Москва, 2023

**Постановка задачи**

**Вариант 22.**

****



Правило фильтрации: с вероятностью 80% строки отправляются в pipe1, иначе в pipe2. Дочерние процессы инвертируют строки.

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* pid\_t **fork**(void); – создает дочерний процесс, возвращает PID дочернего процесса, а процессу потомку возвращается 0, а в случае ошибки -1.
* int **pipe**(int \*fd); – создает канал, который используется для связи дочерних 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, после завершения возвращает количество записанных байтов, а в случае ошибки возвращает -1.
* ssize\_t read(int fd, void buf[.count], size\_t count) - считывает size\_t count байт в указанный файловый дескриптор fd, после завершения возвращает количество считанных байтов, а в случае ошибки возвращает -1.
* int open(const char \*pathname, int flags, mode\_t mode) - открывает и создает файл(если мы укажем такой флаг), возвращает файловый дескриптор, а в случае ошибки -1.
* int **close**(int fd) - закрывает файловый дескриптор fd
* int **dup2**(int oldfd, int newfd) - дублирует файловый дескриптор newfd на место дескриптора oldfd, возвращает новый дескриптор, а в случае ошибки -1.
* int **execl**(const char \*pathname, const char \*arg, .../\*, (char \*) NULL \*/) - исполняет указанные файлы

В самом начале программы нам потребуется ввести имя файла (с соответствующим расширением) c помощью функции ‘int inputing(char \*\*output\_name, int fd, int endl\_status)’, которая, в свою очередь, получает указатель на динамическую строчку, файловый дескриптор и 3-ую переменную. В зависимости от значения этой переменной, функция используется либо для получения названия файла/получения строки от родительского процесса через pipe, либо для последовательного считывания строчек (считывание будет осуществляться, пока не будет введен символ переноса строки (‘\n’), но если ввести (‘\n’) в случае получения названия файла - возникнет ошибка [ввод пустой строки]).

После этого функция open получает в аргументы уже введенную нами и обработанную строчку и открывает файл с этим именем, предварительно очищая его от содержимого, в случае если файла нет - он будет создан.

Функции ‘pipe\_creation’ и ‘process\_creation’ являются оболочками системных вызовов ‘pipe()’ и ‘fork()’ соответственно, в которых, одновременно с вызовом функций, выполняются проверки на ошибки, а в случае ошибок - программа аварийно завершается. С их помощью создаются каналы, необходимые для межпроцессорного взаимодейтсвия, и дочерние процессы.

В случае дочернего процесса (‘fork()’ вернул 0) программа закрывает ненужные для дочернего процесса дескрипторы и подменяет стандартные потоки (ввода, вывода и ошибок) с помощью системного вызова ‘dup2’. После этого дочерний процесс заменяет текущий процесс новой программой (‘child1.c’ или ‘child2.c’) с помощью системного вызова ‘execl’.

Родитель, в свою очередь, с помощью вышеописанной функции ‘inputing’ последовательно считывает все вводимые строчки и, согласно вероятности, записывает их либо в pipe\_1[1](стандартный поток ввода для 1-го дочернего процесса), либо в pipe\_2[1](стандартный поток ввода для 2-го дочернего процесса).

Дочерние процессы обрабатывают получаемые строки также с помощью ‘inputing’ и инвертируют их - для этого они используют функцию string\_invert(char \*\*output\_string, char\* input\_string, int len).

При вводе символа переноса строки (‘\n’) в пустую строку, либо при возникновении ошибки при переворачивании строк - дочерний процесс завершает работу, вместе с закрытием дочернего процесса закрываются все его файловые дескрипторы.

**Код программы**

**main.c**

#include "function.h"

int main(){

write(STDOUT\_FILENO, "Enter the first filename with file extension(.txt or .doc or .rtf): ", 68);

char \*Filename\_1=NULL;

char \*Filename\_2=NULL;

int f2\_output = 0;

pid\_t pid\_2 = 0;

if(inputing(&Filename\_1 ,STDIN\_FILENO, 0)<=0){

perror("Trying to create 0-value string: ");

exit(EXIT\_FAILURE);

}

int f1\_output=open(Filename\_1, O\_WRONLY | O\_CREAT | O\_TRUNC, S\_IWUSR);

free(Filename\_1);

if(f1\_output==-1){

fprintf(stderr, "Can't open the file: %s", Filename\_1);

exit(EXIT\_FAILURE);

}

int pipe1[2],pipe2[2];

pipe\_creation(pipe1);

pipe\_creation(pipe2);

pid\_t pid\_1 = process\_creation();

if (pid\_1 == 0)

{ // the 1st child

close(pipe1[1]); // fd\_pipe\_1[1] for writing

close(pipe2[0]); // fd\_pipe\_2[0] for reading

close(pipe2[1]); // fd\_pipe\_2[1] for writing

if(dup2(pipe1[0], STDIN\_FILENO)==-1){

perror("dup2 erorr ");

exit(EXIT\_FAILURE);

}

if(dup2(f1\_output, STDOUT\_FILENO)==-1){

perror("dup2 erorr ");

exit(EXIT\_FAILURE);

}

if(dup2(f1\_output, STDERR\_FILENO)==-1){

perror("dup2 erorr ");

exit(EXIT\_FAILURE);

}

if(execl("./child\_1", "./child\_1", NULL)==-1){

perror("execl erorr ");

exit(EXIT\_FAILURE);

}

}else {

// parent

write(STDOUT\_FILENO, "Enter the second filename with file extension(.txt or .doc or .rtf): ", 69);

if(inputing(&Filename\_2 ,STDIN\_FILENO, 0)<=0){

perror("Trying to create 0-value string: ");

exit(EXIT\_FAILURE);

}

int f2\_output=open(Filename\_2, O\_WRONLY | O\_CREAT | O\_TRUNC, S\_IWUSR);

free(Filename\_2);

if(f2\_output==-1){

fprintf(stderr, "Can't open the file: %s", Filename\_2);

exit(EXIT\_FAILURE);

}

pid\_t pid\_2=process\_creation();

if(pid\_2==0)

{ //the 2nd child

close(f1\_output);

close(pipe1[0]); // fd\_pipe\_1[0] for reading

close(pipe1[1]); // fd\_pipe\_1[1] for writing

close(pipe2[1]); // fd\_pipe\_2[1] for writing

if(dup2(pipe2[0], STDIN\_FILENO)==-1){

perror("dup2 erorr ");

exit(EXIT\_FAILURE);

}

if(dup2(f2\_output, STDOUT\_FILENO)==-1){

perror("dup2 erorr ");

exit(EXIT\_FAILURE);

}

if(dup2(f2\_output, STDERR\_FILENO)==-1){

perror("dup2 erorr ");

exit(EXIT\_FAILURE);

}

if(execl("./child\_2", "./child\_2", NULL)==-1){

perror(" execl erorr ");

exit(EXIT\_FAILURE);

}

}

else

{ // parent

close(pipe1[0]);

close(pipe2[0]);

write(STDOUT\_FILENO, "Enter something you want: ", 27);

while(true)

{

char \*s=NULL;

int s\_len=inputing(&s, STDIN\_FILENO, 1);

if(s\_len==-1){

free(s);

break;

}

int prob\_res=probability();

if(prob\_res==1){

if(write(pipe1[1], s, sizeof(char)\*s\_len)==-1){

perror("write error ");

exit(EXIT\_FAILURE);

}

if (write(pipe2[1], "-", sizeof("-"))==-1){

perror("write error ");

exit(EXIT\_FAILURE);

}

} else{

if (write(pipe2[1], s, s\_len\*sizeof(char))==-1){

perror("write error ");

exit(EXIT\_FAILURE);

}

if(write(pipe1[1], "-", sizeof("-"))==-1){

perror("write error ");

exit(EXIT\_FAILURE);

}

}

free(s);

}

}

}

close(pipe1[1]);

close(pipe2[1]);

close(f1\_output);

close(f2\_output);

kill(pid\_1, SIGTERM);

kill(pid\_2, SIGTERM);

write(STDOUT\_FILENO, "Programm was ended successfully!\n", 34);

}

**function.h**

#ifndef function\_h

#define function\_h

#include <stdio.h>

#include <fcntl.h> //files

#include <stdlib.h> //malloc, srand, rand

#include <stdbool.h>

#include <unistd.h>

#include <sys/types.h> //pid\_t

#include <signal.h> // kill

#include <time.h> //time(NULL)

#include <string.h>

#include <sys/stat.h>

#define MAX\_LEN 255 // max length for file's names

#define SIGTERM 15

/// @brief

void kill();

int inputing(char \*\*output\_name, int fd, int endl\_status);

void pipe\_creation(int \*fd);

int process\_creation();

bool string\_invert(char \*\*output\_string, char\* input\_string, int len);

int probability();

#endif

**function.c**

#include "function.h"

int inputing(char \*\*s\_output, int fd, int endl\_status){

int new\_l=MAX\_LEN;

char \*line=(char\*)malloc(sizeof(char)\*new\_l); // выделяем память под line размером MAX\_LEN = 255 байт

memset(line, 0 , new\_l); //заполняем line нулями

int i=0;

char ch; // выделили 1 байт, чтобы считывать STDIN\_FILENO посимвольно

read(fd, &ch, sizeof(ch));

if(ch=='\n'){ // проверка на \n

line[i]='\n';

\*s\_output=line;

return -1;

}

while(ch!=EOF && ch!='\0' && ch!='\n' ){

if(i>=new\_l){ // проверка не достигнута ли максимальная длина строки

new\_l=new\_l\*2;

line=(char \*)realloc(line, sizeof(char)\*new\_l); // увеличиваем объем выделенной памяти

}

line[i]=ch;

i++;

read(fd, &ch, sizeof(ch)); // продолжаем посимвольное считывание

}

if(endl\_status!=0){ // если нужно вводить строку НЕ один раз

if(i>=new\_l){

new\_l=new\_l\*2;

line=(char \*)realloc(line, sizeof(char)\*new\_l);

}

line[i]='\n';

i++;

}

if(i>=new\_l){

new\_l=new\_l\*2;

line=(char \*)realloc(line, sizeof(char)\*new\_l);

}

line[i] = '\0';

\*s\_output=line;

return i;

}

void kill()

{

}

void pipe\_creation(int \*fd){

if (pipe(fd) == -1){

perror("Call pipe was ended with error: ");

exit(EXIT\_FAILURE);

}

}

int process\_creation(){

pid\_t pid = fork();

if (pid == -1){

perror("Call fork was ended with erorr: ");

exit(EXIT\_FAILURE);

}

return pid;

}

int probability(){

srand(time(NULL)); //инициализация генератора случайных чисел и установка текущего времени в качестве его базы

int a =rand()%10+1; //случайные числа от 1 до 10

if(a<=8){

return 1;

} else{

return 2;

}

}

bool string\_invert(char \*\*output\_string, char\* input\_string, int len){ //fixed

char tmp[len+1];

for(int i=0; i<len;++i){

tmp[len-1-i]=input\_string[i];

}

tmp[len]='\0';

free(\*output\_string);

\*output\_string=tmp;

return true;

}

**child\_1.c**

#include "function.h"

int main(){

while(true){

char \*input\_strint=NULL;

int s\_len=inputing(&input\_strint, STDIN\_FILENO, 0);

char\* output\_string=NULL;

if ((input\_strint[0]=='-')){

continue;

} else if(s\_len<=0){

free(input\_strint);

break;

} else{

if(string\_invert(&output\_string, input\_strint, s\_len)==0){

write(STDOUT\_FILENO, " String\_invert Error2! ", 24);

break;

} else{

write(STDOUT\_FILENO, output\_string, s\_len\*sizeof(char));

}

}

free(input\_strint);

}

return 0;

}

**child\_2.c**

#include "function.h"

int main(){

while(true){

char \*input\_strint=NULL;

int s\_len=inputing(&input\_strint, STDIN\_FILENO, 0);

char\* output\_string=NULL;

if ((input\_strint[0]=='-')){

continue;

} else if(s\_len<=0){

free(input\_strint);

break;

} else{

if(string\_invert(&output\_string, input\_strint, s\_len)==0){

write(STDOUT\_FILENO, " String\_invert Error2! ", 24);

break;

} else{

write(STDOUT\_FILENO, output\_string, s\_len\*sizeof(char));

}

}

free(input\_strint);

}

return 0;

}

**Протокол работы программы**

**Тестирование:**

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab1/pr

ograms$ make

gcc -std=c11 -pedantic -Wall child\_1.c function.c -o child\_1

gcc -std=c11 -pedantic -Wall child\_2.c function.c -o child\_2

gcc -std=c11 -pedantic -Wall main.c function.c -o main

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab1/pr

ograms$ ls

Makefile child\_1 child\_1.c child\_2 child\_2.c function.c function.h main main.c

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab1/pr

ograms$ ./main

Enter the first filename with file extension(.txt or .doc or .rtf): 1.txt

Enter the second filename with file extension(.txt or .doc or .rtf): 2.txt

Enter something you want: one

two

three

four

five

six

seven

eight

nine

ten

Programm was ended successfully!

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab1/pr

ograms$ ls

1.txt 2.txt Makefile child\_1 child\_1.c child\_2 child\_2.c function.c function.h main main.c

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab1/pr

ograms$ cat 1.txt

owt eerht ruof evif xis neves thgie enin net

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab1

/programs$ cat 2.txt

eno

===============================================================================

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab1/prog

rams$ ./main

Enter the first filename with file extension(.txt or .doc or .rtf): 1.txt

Enter the second filename with file extension(.txt or .doc or .rtf): 2.txt

Enter something you want: smth with spacing

and smth else

Programm was ended successfully!

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab1/prog

rams$ cat 1.txt

gnicaps htiw htms esle htms

dnamattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab1/

programs$ cat 2.txt

===============================================================================

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab1/prog

rams$ ./main

Enter the first filename with file extension(.txt or .doc or .rtf): print1.doc

Enter the second filename with file extension(.txt or .doc or .rtf): print2.rtf

Enter something you want: looooooooong string with spaaaaaaaaaacing

one

two

three

four

Programm was ended successfully!

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab1/prog

rams$ cat print1.doc

gnicaaaaaaaaaaps htiw gnirts gnoooooooool eno owt eerht ruof

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab1/

programs$ cat print2.rtf

**Strace:**

execve("./main", ["./main"], 0x7ffcd66b0d18 /\* 36 vars \*/) = 0

brk(NULL) = 0x565070f07000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffe5544dce0) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1,

0) = 0x7f4d8f6b4000

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=19095, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 19095, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f4d8f6af000

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 =\340\2563\265?\356\25x\261\27\313A#\350"..., 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) =

0x7f4d8f487000

mmap(0x7f4d8f4af000, 1658880, PROT\_READ|PROT\_EXEC,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f4d8f4af000

mmap(0x7f4d8f644000, 360448, PROT\_READ,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7f4d8f644000

mmap(0x7f4d8f69c000, 24576, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7f4d8f69c000

mmap(0x7f4d8f6a2000, 52816, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f4d8f6a2000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1,

0) = 0x7f4d8f484000

arch\_prctl(ARCH\_SET\_FS, 0x7f4d8f484740) = 0

set\_tid\_address(0x7f4d8f484a10) = 140089

set\_robust\_list(0x7f4d8f484a20, 24) = 0

rseq(0x7f4d8f4850e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7f4d8f69c000, 16384, PROT\_READ) = 0

mprotect(0x565070598000, 4096, PROT\_READ) = 0

mprotect(0x7f4d8f6ee000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) =

0

munmap(0x7f4d8f6af000, 19095) = 0

write(1, "Enter the first filename with fi"..., 68Enter the first filename with file extension(.txt or

.doc or .rtf): ) = 68

getrandom("\x44\x70\x10\xe5\xe0\x87\x34\x53", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x565070f07000

brk(0x565070f28000) = 0x565070f28000

read(0, 1.txt

"1", 1) = 1

read(0, ".", 1) = 1

read(0, "t", 1) = 1

read(0, "x", 1) = 1

read(0, "t", 1) = 1

read(0, "\n", 1) = 1

openat(AT\_FDCWD, "1.txt", O\_WRONLY|O\_CREAT|O\_TRUNC, 0200) = 3

**pipe2([4, 5], 0) = 0**

**pipe2([6, 7], 0) = 0**

**clone(child\_stack=NULL,**

**flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process**

**140161 attached, child\_tidptr=0x7f4d8f484a10) = 140161**

[pid 140161] set\_robust\_list(0x7f4d8f484a20, 24 <unfinished ...>

[pid 140089] write(1, "Enter the second filename with f"..., 69 <unfinished ...>

Enter the second filename with file extension(.txt or .doc or .rtf): [pid 140161] <... set\_robust\_list

resumed>) = 0

[pid 140089] <... write resumed>) = 69

[pid 140161] close(5 <unfinished ...>

[pid 140089] read(0, <unfinished ...>

[pid 140161] <... close resumed>) = 0

[pid 140161] close(6) = 0

[pid 140161] close(7) = 0

**[pid 140161] dup2(4, 0) = 0**

**[pid 140161] dup2(3, 1) = 1**

**[pid 140161] dup2(3, 2) = 2**

**[pid 140161] execve("./child\_1", ["./child\_1"], 0x7ffe5544deb8 /\* 36 vars \*/) = 0**

[pid 140161] brk(NULL) = 0x560f7ab33000

[pid 140161] arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffe950d9cc0) = -1 EINVAL (Invalid

argument)

[pid 140161] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f4b38a42000

[pid 140161] access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

[pid 140161] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 5

[pid 140161] newfstatat(5, "", {st\_mode=S\_IFREG|0644, st\_size=19095, ...}, AT\_EMPTY\_PATH)

= 0

[pid 140161] mmap(NULL, 19095, PROT\_READ, MAP\_PRIVATE, 5, 0) = 0x7f4b38a3d000

[pid 140161] close(5) = 0

[pid 140161] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC)

= 5

[pid 140161] 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 140161] 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 140161] 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 140161] pread64(5, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0

=\340\2563\265?\356\25x\261\27\313A#\350"..., 68, 896) = 68

[pid 140161] newfstatat(5, "", {st\_mode=S\_IFREG|0755, st\_size=2216304, ...},

AT\_EMPTY\_PATH) = 0

[pid 140161] 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 140161] mmap(NULL, 2260560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 5, 0)

= 0x7f4b38815000

[pid 140161] mmap(0x7f4b3883d000, 1658880, PROT\_READ|PROT\_EXEC,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0x28000) = 0x7f4b3883d000

[pid 140161] mmap(0x7f4b389d2000, 360448, PROT\_READ,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0x1bd000) = 0x7f4b389d2000

[pid 140161] mmap(0x7f4b38a2a000, 24576, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0x214000) = 0x7f4b38a2a000

[pid 140161] mmap(0x7f4b38a30000, 52816, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f4b38a30000

[pid 140161] close(5) = 0

[pid 140161] mmap(NULL, 12288, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f4b38812000

[pid 140161] arch\_prctl(ARCH\_SET\_FS, 0x7f4b38812740) = 0

[pid 140161] set\_tid\_address(0x7f4b38812a10) = 140161

[pid 140161] set\_robust\_list(0x7f4b38812a20, 24) = 0

[pid 140161] rseq(0x7f4b388130e0, 0x20, 0, 0x53053053) = 0

[pid 140161] mprotect(0x7f4b38a2a000, 16384, PROT\_READ) = 0

[pid 140161] mprotect(0x560f79ce4000, 4096, PROT\_READ) = 0

[pid 140161] mprotect(0x7f4b38a7c000, 8192, PROT\_READ) = 0

[pid 140161] prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024,

rlim\_max=RLIM64\_INFINITY}) = 0

[pid 140161] munmap(0x7f4b38a3d000, 19095) = 0

[pid 140161] getrandom("\x97\x0b\x4c\xf9\x96\xa6\x4a\x3d", 8, GRND\_NONBLOCK) = 8

[pid 140161] brk(NULL) = 0x560f7ab33000

[pid 140161] brk(0x560f7ab54000) = 0x560f7ab54000

[pid 140161] read(0, 2.txt

<unfinished ...>

[pid 140089] <... read resumed>"2", 1) = 1

[pid 140089] read(0, ".", 1) = 1

[pid 140089] read(0, "t", 1) = 1

[pid 140089] read(0, "x", 1) = 1

[pid 140089] read(0, "t", 1) = 1

[pid 140089] read(0, "\n", 1) = 1

[pid 140089] openat(AT\_FDCWD, "2.txt", O\_WRONLY|O\_CREAT|O\_TRUNC, 0200) = 8

**[pid 140089] clone(child\_stack=NULL,**

**flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process**

**140168 attached, child\_tidptr=0x7f4d8f484a10) = 140168**

[pid 140168] set\_robust\_list(0x7f4d8f484a20, 24 <unfinished ...>

[pid 140089] close(4 <unfinished ...>

[pid 140168] <... set\_robust\_list resumed>) = 0

[pid 140168] close(3 <unfinished ...>

[pid 140089] <... close resumed>) = 0

[pid 140168] <... close resumed>) = 0

[pid 140089] close(6 <unfinished ...>

[pid 140168] close(4 <unfinished ...>

[pid 140089] <... close resumed>) = 0

[pid 140089] write(1, "Enter something you want: \0", 27 <unfinished ...>

Enter something you want: [pid 140168] <... close resumed>) = 0

[pid 140089] <... write resumed>) = 27

[pid 140168] close(5 <unfinished ...>

[pid 140089] read(0, <unfinished ...>

[pid 140168] <... close resumed>) = 0

[pid 140168] close(7) = 0

**[pid 140168] dup2(6, 0) = 0**

**[pid 140168] dup2(8, 1) = 1**

**[pid 140168] dup2(8, 2) = 2**

**[pid 140168] execve("./child\_2", ["./child\_2"], 0x7ffe5544deb8 /\* 36 vars \*/) = 0**

[pid 140168] brk(NULL) = 0x55a635099000

[pid 140168] arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffedb295600) = -1 EINVAL (Invalid

argument)

[pid 140168] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f96c3193000

[pid 140168] access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

[pid 140168] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

[pid 140168] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=19095, ...}, AT\_EMPTY\_PATH)

= 0

[pid 140168] mmap(NULL, 19095, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f96c318e000

[pid 140168] close(3) = 0

[pid 140168] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC)

= 3

[pid 140168] 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

[pid 140168] 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

[pid 140168] 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

[pid 140168] pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0

=\340\2563\265?\356\25x\261\27\313A#\350"..., 68, 896) = 68

[pid 140168] newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2216304, ...},

AT\_EMPTY\_PATH) = 0

[pid 140168] 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

[pid 140168] mmap(NULL, 2260560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0)

= 0x7f96c2f66000

[pid 140168] mmap(0x7f96c2f8e000, 1658880, PROT\_READ|PROT\_EXEC,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f96c2f8e000

[pid 140168] mmap(0x7f96c3123000, 360448, PROT\_READ,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7f96c3123000

[pid 140168] mmap(0x7f96c317b000, 24576, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7f96c317b000

[pid 140168] mmap(0x7f96c3181000, 52816, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f96c3181000

[pid 140168] close(3) = 0

[pid 140168] mmap(NULL, 12288, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f96c2f63000

[pid 140168] arch\_prctl(ARCH\_SET\_FS, 0x7f96c2f63740) = 0

[pid 140168] set\_tid\_address(0x7f96c2f63a10) = 140168

[pid 140168] set\_robust\_list(0x7f96c2f63a20, 24) = 0

[pid 140168] rseq(0x7f96c2f640e0, 0x20, 0, 0x53053053) = 0

[pid 140168] mprotect(0x7f96c317b000, 16384, PROT\_READ) = 0

[pid 140168] mprotect(0x55a6340f4000, 4096, PROT\_READ) = 0

[pid 140168] mprotect(0x7f96c31cd000, 8192, PROT\_READ) = 0

[pid 140168] prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024,

rlim\_max=RLIM64\_INFINITY}) = 0

[pid 140168] munmap(0x7f96c318e000, 19095) = 0

[pid 140168] getrandom("\xa2\x06\xa6\xa9\x90\xdc\x2d\xb8", 8, GRND\_NONBLOCK) = 8

[pid 140168] brk(NULL) = 0x55a635099000

[pid 140168] brk(0x55a6350ba000) = 0x55a6350ba000

[pid 140168] read(0, one

<unfinished ...>

[pid 140089] <... read resumed>"o", 1) = 1

[pid 140089] read(0, "n", 1) = 1

[pid 140089] read(0, "e", 1) = 1

[pid 140089] read(0, "\n", 1) = 1

[pid 140089] write(5, "one\n", 4) = 4

[pid 140161] <... read resumed>"o", 1) = 1

[pid 140089] write(7, "-\0", 2) = 2

[pid 140168] <... read resumed>"-", 1) = 1

[pid 140161] read(0, <unfinished ...>

[pid 140089] read(0, <unfinished ...>

[pid 140168] read(0, <unfinished ...>

[pid 140161] <... read resumed>"n", 1) = 1

[pid 140168] <... read resumed>"\0", 1) = 1

[pid 140161] read(0, <unfinished ...>

[pid 140168] read(0, <unfinished ...>

[pid 140161] <... read resumed>"e", 1) = 1

[pid 140161] read(0, "\n", 1) = 1

[pid 140161] write(1, "eno", 3) = 3

[pid 140161] read(0, two

<unfinished ...>

[pid 140089] <... read resumed>"t", 1) = 1

[pid 140089] read(0, "w", 1) = 1

[pid 140089] read(0, "o", 1) = 1

[pid 140089] read(0, "\n", 1) = 1

[pid 140089] write(5, "two\n", 4) = 4

[pid 140161] <... read resumed>"t", 1) = 1

[pid 140089] write(7, "-\0", 2) = 2

[pid 140168] <... read resumed>"-", 1) = 1

[pid 140161] read(0, <unfinished ...>

[pid 140089] read(0, <unfinished ...>

[pid 140161] <... read resumed>"w", 1) = 1

[pid 140168] read(0, <unfinished ...>

[pid 140161] read(0, <unfinished ...>

[pid 140168] <... read resumed>"\0", 1) = 1

[pid 140161] <... read resumed>"o", 1) = 1

[pid 140168] read(0, <unfinished ...>

[pid 140161] read(0, "\n", 1) = 1

[pid 140161] write(1, "owt", 3) = 3

[pid 140161] read(0, three

<unfinished ...>

[pid 140089] <... read resumed>"t", 1) = 1

[pid 140089] read(0, "h", 1) = 1

[pid 140089] read(0, "r", 1) = 1

[pid 140089] read(0, "e", 1) = 1

[pid 140089] read(0, "e", 1) = 1

[pid 140089] read(0, "\n", 1) = 1

[pid 140089] write(5, "three\n", 6) = 6

[pid 140161] <... read resumed>"t", 1) = 1

[pid 140089] write(7, "-\0", 2 <unfinished ...>

[pid 140161] read(0, "h", 1) = 1

[pid 140089] <... write resumed>) = 2

[pid 140168] <... read resumed>"-", 1) = 1

[pid 140089] read(0, <unfinished ...>

[pid 140168] read(0, <unfinished ...>

[pid 140161] read(0, "r", 1) = 1

[pid 140168] <... read resumed>"\0", 1) = 1

[pid 140168] read(0, <unfinished ...>

[pid 140161] read(0, "e", 1) = 1

[pid 140161] read(0, "e", 1) = 1

[pid 140161] read(0, "\n", 1) = 1

[pid 140161] write(1, "eerht", 5) = 5

[pid 140161] read(0, four

<unfinished ...>

[pid 140089] <... read resumed>"f", 1) = 1

[pid 140089] read(0, "o", 1) = 1

[pid 140089] read(0, "u", 1) = 1

[pid 140089] read(0, "r", 1) = 1

[pid 140089] read(0, " ", 1) = 1

[pid 140089] read(0, "\n", 1) = 1

[pid 140089] write(5, "four \n", 6) = 6

[pid 140161] <... read resumed>"f", 1) = 1

[pid 140161] read(0, <unfinished ...>

[pid 140089] write(7, "-\0", 2 <unfinished ...>

[pid 140161] <... read resumed>"o", 1) = 1

[pid 140089] <... write resumed>) = 2

[pid 140168] <... read resumed>"-", 1) = 1

[pid 140089] read(0, <unfinished ...>

[pid 140168] read(0, <unfinished ...>

[pid 140161] read(0, <unfinished ...>

[pid 140168] <... read resumed>"\0", 1) = 1

[pid 140161] <... read resumed>"u", 1) = 1

[pid 140168] read(0, <unfinished ...>

[pid 140161] read(0, "r", 1) = 1

[pid 140161] read(0, " ", 1) = 1

[pid 140161] read(0, "\n", 1) = 1

[pid 140161] write(1, " ruof", 5) = 5

[pid 140161] read(0,

<unfinished ...>

[pid 140089] <... read resumed>"\n", 1) = 1

[pid 140089] close(5) = 0

[pid 140161] <... read resumed>"", 1) = 0

[pid 140089] close(7 <unfinished ...>

[pid 140161] exit\_group(0 <unfinished ...>

[pid 140089] <... close resumed>) = 0

[pid 140168] <... read resumed>"", 1) = 0

[pid 140089] close(3 <unfinished ...>

[pid 140168] exit\_group(0 <unfinished ...>

[pid 140161] <... exit\_group resumed>) = ?

[pid 140089] <... close resumed>) = 0

[pid 140168] <... exit\_group resumed>) = ?

[pid 140089] close(0) = 0

[pid 140089] write(1, "Programm was ended successfully!"..., 34Programm was ended

successfully!

) = 34

**[pid 140168] +++ exited with 0 +++**

[pid 140089] --- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=140168,

si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

[pid 140089] exit\_group(0) = ?

**[pid 140161] +++ exited with 0 +++**

+++ exited with 0 +++

**Вывод**

В данной лабораторной работе я ознакомился с системными вызовами и межпроцессорным взаимодейтсвием, получил навыки работы с соответствующими функциями из библиотеки unistd.h - pipe, execl, fork, dub2.

Кроме того, смог реализовать программу, обеспечивающую обмен данными между процессами посредством каналов, согласно представленной схеме и заданию. В ходе работы пришлось приложить немало усилий, обрабатывая возникающие ошибки разного рода.