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: Бурунов М.А.

Преподаватель: Миронов Е.С.

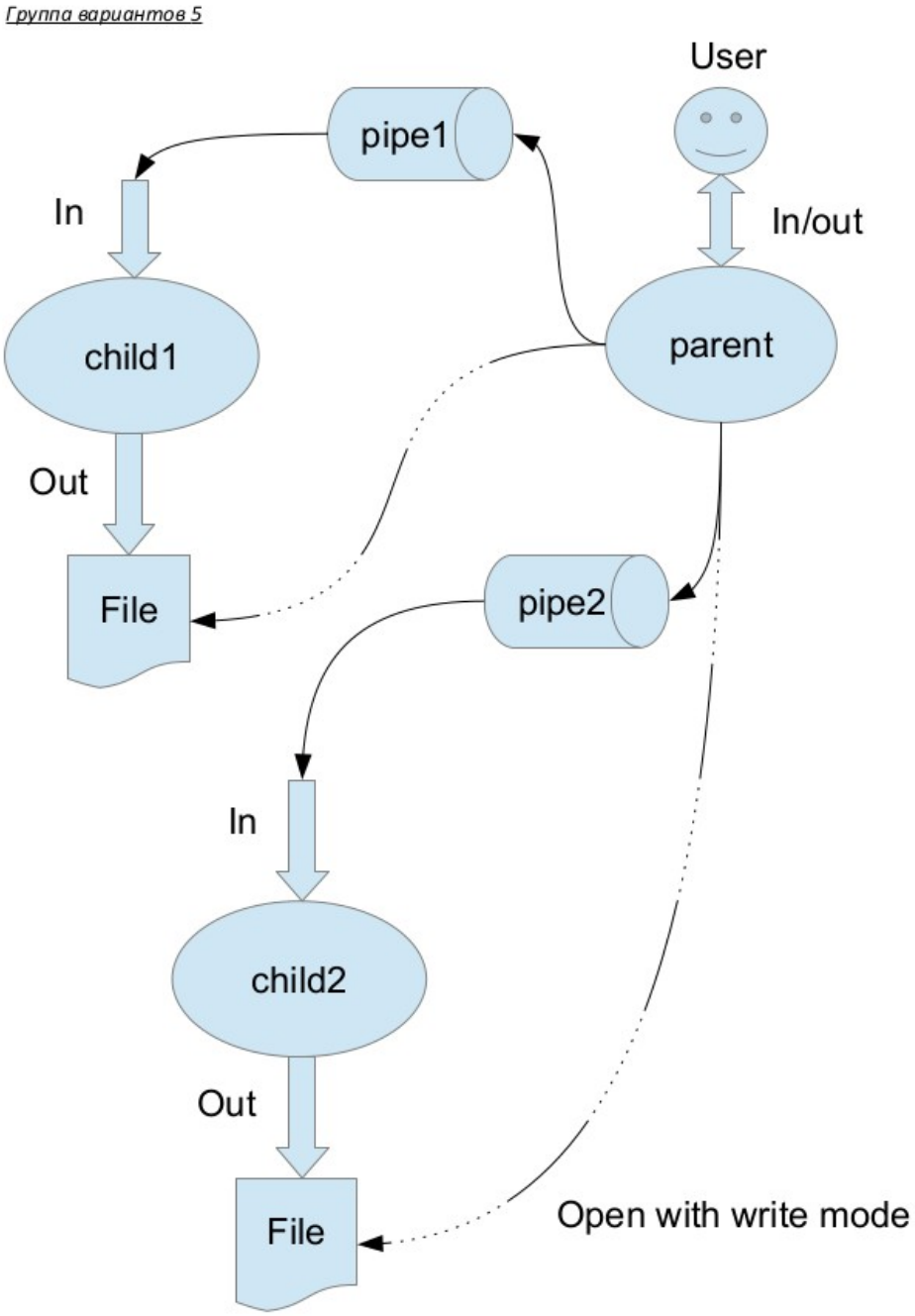
Оценка: _____

Дата: 01.03.24

Москва, 2024

Постановка задачи

Вариант 19.



Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1 или в pipe2 в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод.

Правило фильтрации: с вероятностью 80% строки отправляются в pipe1, иначе в pipe2.
Дочерние процессы удаляют гласные из строк.

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) с 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_changing(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_changing(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_changing(char **output_string, char* input_string, int len){ // from lab1
    char tmp[len+1];
    for(int i=0; i<len;++i){
        if ((input_string[i] == 'a' || input_string[i] == 'e' || input_string[i] == 'i' || input_string[i] == 'o' || input_string[i] == 'u'
        || input_string[i] == 'y' || input_string[i] == 'A' || input_string[i] == 'E' || input_string[i] == 'I'
        || input_string[i] == 'O' || input_string[i] == 'U' || input_string[i] == 'Y')){
            tmp[i]=' ';
        }else {
            tmp[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_changing(&output_string, input_strint, s_len)==0){
                write(STDOUT_FILENO, " String_changing 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_changing(&output_string, input_strint, s_len)==0){
              write(STDOUT_FILENO, " String_changing Error2! ", 24);
              break;
           } else{
              write(STDOUT_FILENO, output_string, s_len*sizeof(char));
           }
        }
        free(input_strint);
     }
     return 0;
}
```

Протокол работы программы

Тестирование:

```
vimi@HOME-PC:/mnt/c/Documents and Settings/vimi/Desktop/labsOS/3/programs$ ./main
Enter the first filename with file extension(.txt or .doc or .rtf): 1
Enter the second filename with file extension(.txt or .doc or .rtf): 2
one
two
three
four
five
vimi@HOME-PC:/mnt/c/Documents and Settings/vimi/Desktop/labsOS/3/programs$ cat 1.txt
cat: 1.txt: No such file or directory
vimi@HOME-PC:/mnt/c/Documents and Settings/vimi/Desktop/labsOS/3/programs$ cat 1
tw thr
f  r
f v
vimi@HOME-PC:/mnt/c/Documents and Settings/vimi/Desktop/labsOS/3/programs$ cat 2
 n vimi@HOME-PC:/mnt/c/Documents and Settings/vimi/Desktop/labsOS/3/programs$
```

```
vimi@HOME-PC:/mnt/c/Documents and Settings/vimi/Desktop/labsOS/3/programs$ ./main
Enter the first filename with file extension(.txt or .doc or .rtf): 1
Enter the second filename with file extension(.txt or .doc or .rtf): 2
hey
```

wait. I've gonna new complaint
Forever in debt to your priceless advice
She eyes me like a pieces when
I am week
vimi@HOME-PC:/mnt/c/Documents and Settings/vimi/Desktop/labsOS/3/programs$ cat 1
h w t. 'v g nn n w c mpl nt
F r v r n d bt t   r pr c l ss dv c
Sh   s m l k   p c s wh n
  m w k
vimi@HOME-PC:/mnt/c/Documents and Settings/vimi/Desktop/labsOS/3/programs$ cat 2
vimi@HOME-PC:/mnt/c/Documents and Settings/vimi/Desktop/labsOS/3/programs$

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"..., 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.