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.

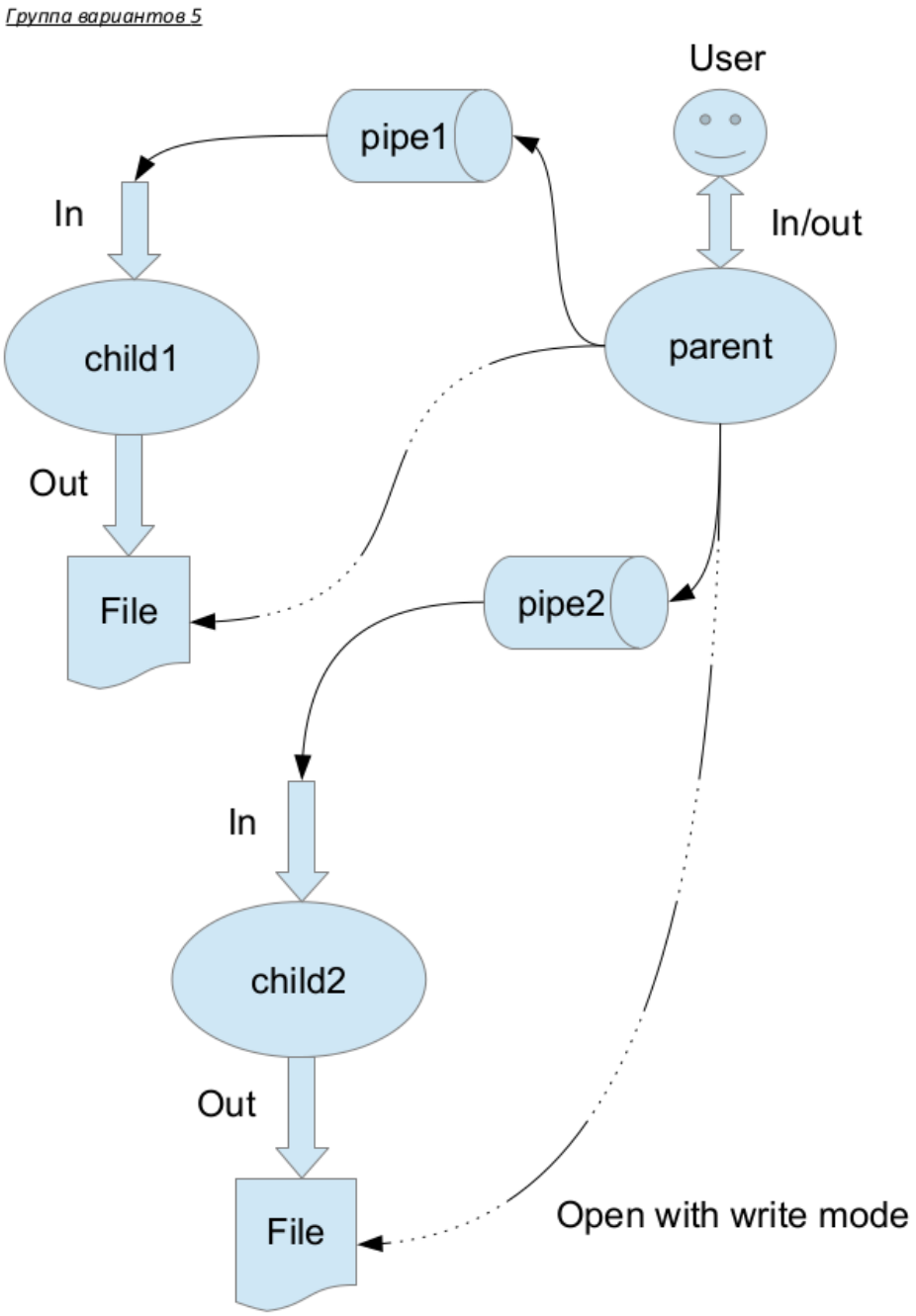
Оценка: _____

Дата: 03.11.23

Москва, 2023

Постановка задачи

Вариант 22.



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 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

После запуска программы, она потребует ввести имя файла(с соответствующим расширением) с помощью функции int inputing(char **output_name, int fd, int endl_status), которая, в свою очередь, получит указатель на динамическую строчку, файловый дескриптор для ввода символов и 3 переменную, которую мы используем для считывая строчек, не предназначенных для создания файла. Данная функция позволяет ввести строчку любой длины, ввод будет осуществляться, пока не будет введен символ переноса строки (‘\n’).

После этого функция open получает в аргументы уже введенную нами и обработанную строчку и открывает файл с этим именем, а если такого файла нет, то он будет создан.

Функции “pipe_creation” и “process_creation” являются оболочками функций pipe() и fork() соответственно, в которых одновременно с вызовом функций проверяется их корректность исполнения, а в случае ошибки программа аварийно завершится. С их помощью создается 2 канала, которые впоследствии будут использоваться для взаимодействия процессов между собой, а также создается первый дочерний процесс.

Если созданный процесс - дочерний(“fork()” вернул 0), то программа закрывает ненужные для дочернего процесса дескрипторы (по заданию) и подменяет для дочернего процесса стандартные потоки(ввода, вывода и ошибок) с помощью функции “dub2” на “pipe_1[0]” - новый поток ввода и “f1_output” (открытый в начале программы файл) - на новый поток вывода и

одновременно новый поток ошибок. После этого первый дочерний процесс запускает программу child_1.c, а программа main для этого дочернего процесса завершается.

Если созданный процесс - родитель, то также, как описано выше, подаётся на ввод еще 1 строчка и открывается/создается 2-ой файл f2_output и порождается второй дочерний процесс, у которого стандартные потоки будут: “pipe_2[0]” - поток ввода, f2_output - поток вывода и ошибок.

После вызова fork(), программа опять распараллеливается. Новый, второй дочерний процесс запускает программу child_2, программа main для этого дочернего процесса тоже завершается.

А родитель с помощью функции inputing(но уже с используемым 3 аргументом) считывает все входные строчки и согласно вероятности записывает их либо в pipe_1[1](стандартный поток ввода для первого процесса) либо в pipe_2[1](стандартный поток ввода для второго процесса). Я реализую бесконечный ввод строчек, поэтому, чтобы не было утечек памяти, я в обоих случаях записываю данные в оба “пайпа”, при этом запись вводимых строчек происходит согласно результату вероятности, в то время как в другой пайп записывается спец-символ, необходимый для корректный работы программы передачи строчек другим дочерним процессам, и для предотвращения потенциального непрекращающегося “залипания” программ, исполняющихся в дочерних процессах.

Для завершения работы с программой, нужно ввести ‘\n’ в пустую строку.

Дочерние процессы также обрабатывают строки с помощью функции inputing (но 3-ий параметр обязательно равен нулю). По заданию, дочерние процессы инвертируют строки, 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. И 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): ", 67);


   char *Filename_1=NULL;


   if(inputing(&Filename_1 ,STDIN_FILENO, 0)<=0){
      perror("Trying to create 0-value string: ");
      exit(-1);
   }
```

```
int f1_output=open(Filename_1, O_WRONLY | O_CREAT, 0777);


if(f1_output==-1){
    fprintf(stderr, "Can't open the file:  %s", Filename_1);
    exit(-1);
}


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(-1);
    }
    if(dup2(f1_output, STDOUT_FILENO)==-1){
        perror("dup2 erorr ");
        exit(-1);
    }
    if(dup2(f1_output, STDERR_FILENO)==-1){
        perror("dup2 erorr ");
        exit(-1);
    }


    if(execl("./child_1", "./child_1", NULL)==-1){
```

```
        perror("execl erorr ");
        exit(-1);
    }

    // close(pipe1[0]);
    // close(pipe2[1]);
    // close(f1_output);

}else {
    // parent
    write(STDOUT_FILENO, "\nEnter the second filename with file extension(.txt or .doc or .rtf): ", 71);
    char *Filename_2=NULL;

    if(inputing(&Filename_2 ,STDIN_FILENO, 0)<=0){
        perror("Trying to create 0-value string: ");
        exit(-1);
    }

    int f2_output=open(Filename_2, O_WRONLY | O_CREAT, 0777);
    if(f2_output==-1){
        fprintf(stderr, "Can't open the file:  %s", Filename_2);
        exit(-1);
    }

    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(-1);
    }
    if(dup2(f2_output, STDOUT_FILENO)==-1){
        perror("dup2 erorr ");
        exit(-1);
    }
    if(dup2(f2_output, STDERR_FILENO)==-1){
        perror("dup2 erorr ");
        exit(-1);
    }

    if(execl("./child_2", "./child_2", NULL)==-1){
        perror(" execl erorr ");
        exit(-1);
    }

    // close(pipe2[0]);
    // close(f2_output);

} 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){
            break;
        }
```

```
        int prob_res=probability();


        if(prob_res==1){
            if(write(pipe1[1], s, sizeof(char)*s_len)==-1){
                perror("write error ");
                exit(-1);
            }
            if (write(pipe2[1], "-", sizeof("-"))==-1){
                perror("write error ");
                exit(-1);
            }

        } else{
            if (write(pipe2[1], s, s_len*sizeof(char))==-1){
                perror("write error ");
                exit(-1);
            }
            if(write(pipe1[1], "-", sizeof("-"))==-1){
                perror("write error ");
                exit(-1);
            }
        }
    }
    write(STDOUT_FILENO, "\nProgramm was ended successfully!\n", 35);
    close(pipe1[1]);
    close(pipe2[1]);
    close(f1_output);
    close(f2_output);
      kill(pid_1, SIGTERM);
    kill(pid_2, SIGTERM);


}
```

```
    }
}
```

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)

#define MAX_LEN 255 // max length for file's names
#define SIGTERM 15

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 *tmp=NULL; //временный указатель для переопределения памяти
    char *line=(char*)malloc(sizeof(char)*new_l); // выделяем память под line размером MAX_LEN = 255 байт
    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;
      tmp=(char *)realloc(line, new_l); //увеличиваем объем выделенной памяти
      line=tmp;
      tmp=NULL;
      free(tmp);
   }
   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;
      tmp=(char *)realloc(line, new_l);
      line=tmp;
      tmp=NULL;
      free(tmp);
   }
   line[i]='\n';
   i++;
}
*s_output=line;
line=NULL;
free(line);
```

```
    return i;
}

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

int process_creation(){
    pid_t pid = fork();
    if (pid == -1){
        perror("Call fork was ended with erorr: ");
        exit(-1);
    }
    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 ){
            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));
            }
        }
    }
    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 ){
         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));
         }
      }
   }
   return 0;
}
```

Протокол работы программы

Тестирование:

```
mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/users/Никита/Desktop/Projects/labs3sem/lab1/programs$ ls
Makefile  child_1.c  child_2.c  function.c  function.h  main.c
mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/users/Никита/Desktop/Projects/labs3sem/lab1/programs$ make
gcc -std=c99 -pedantic -Wall  child_1.c function.c -o child_1
gcc -std=c99 -pedantic -Wall  child_2.c function.c -o child_2
gcc -std=c99 -pedantic -Wall  main.c function.c -o main
```

```
mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/users/Никита/Desktop/Projects/labs3sem/lab1/progra
ms$ ./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/labs3sem/lab1/progra
ms$ 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/labs3sem/lab1/progra
ms$ cat 1.txt
 eno owt ruof evif xis neves thgie enin net
mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/users/Никита/Desktop/Projects/labs3sem/lab1/progra
ms$ cat 2.txt
 eerht
```

==========

```
mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/users/Никита/Desktop/Projects/labs3sem/lab1/programs
$ rm *.txt
mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/users/Никита/Desktop/Projects/labs3sem/lab1/programs
$ ./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/labs3sem/lab1/programs
$ cat 1.txt
mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/users/Никита/Desktop/Projects/labs3sem/lab1/programs
```

```
$ cat 2.txt
gnicaps htiw htms esle htms dna
================================================================================
mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/users/Никита/Desktop/Projects/labs3sem/lab1/programs
$ rm *.txt
mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/users/Никита/Desktop/Projects/labs3sem/lab1/programs
$ ./main
Enter the first filename with file extension(.txt or .doc or .rtf):prin1.doc

Enter the second filename with file extension(.txt or .doc or .rtf): print2.rtf
Enter something you want: looooooooong string with spaaaaaaaaacing
one
two
three
four



Programm was ended successfully!
mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/users/Никита/Desktop/Projects/labs3sem/lab1/programs
$ cat prin1.doc
 gnicaaaaaaaaaaps htiw gnirts gnooooooooool eno owt eerht ruof
mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/users/Никита/Desktop/Projects/labs3sem/lab1/programs
$ cat print2.rtf
```

Strace:

```
execve("./main", ["./main"], 0x7ffefdd86558 /* 34 vars */) = 0
brk(NULL)                               = 0x562ce645b000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc9857a680) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
 0) = 0x7fbaf293b000
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=18175, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 18175, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbaf2936000
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\244;\374\204(\337f#\315I\214\234\f\256\271\32"...,
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) =
```

0x7fbaf270e000
mmap(0x7fbaf2736000, 1658880, PROT_READ|PROT_EXEC
, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7fbaf2736000
mmap(0x7fbaf28cb000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7fbaf28cb000
mmap(0x7fbaf2923000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000) = 0x7fbaf2923000
mmap(0x7fbaf2929000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fbaf2929000
close(3)   = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7fbaf270b000
arch_prctl(ARCH_SET_FS, 0x7fbaf270b740) = 0
set_tid_address(0x7fbaf270ba10)   = 4040
set_robust_list(0x7fbaf270ba20, 24)   = 0
rseq(0x7fbaf270c0e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7fbaf2923000, 16384, PROT_READ) = 0
mprotect(0x562ce5329000, 4096, PROT_READ) = 0
mprotect(0x7fbaf2975000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) =
0
munmap(0x7fbaf2936000, 18175)   = 0
write(1, "Enter the first filename with fi"..., 67Enter the first filename with file extension(.txt or
.doc or .rtf):) = 67
getrandom("\xa3\xe6\xad\xaf\x17\xf6\x25\x89", 8, GRND_NONBLOCK) = 8
brk(NULL)   = 0x562ce645b000
brk(0x562ce647c000)   = 0x562ce647c000
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, 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
4077
attached
<unfinished ...>
[pid 4077] set_robust_list(0x7fbaf270ba20, 24 <unfinished ...>
[pid 4040] <... clone resumed>, child_tidptr=0x7fbaf270ba10) = 4077
[pid 4077] <... set_robust_list resumed>) = 0
[pid 4040] write(1, "\nEnter the second filename with "..., 71

Enter the second filename with file extension(.txt or .doc or .rtf):) = 71
[pid 4077] close(5 <unfinished ...>
[pid 4040] read(0, <unfinished ...>
[pid 4077] <... close resumed>)   = 0
[pid 4077] close(6)   = 0
[pid 4077] close(7)   = 0
[pid 4077] dup2(4, 0)   = 0
[pid 4077] dup2(3, 1)   = 1
[pid 4077] dup2(3, 2)   = 2
[pid 4077] execve("./child_1", ["./child_1"], 0x7ffc9857a858 /* 34 vars */) = 0
[pid 4077] brk(NULL)   = 0x5603e4d5a000
[pid 4077] arch_prctl(0x3001 /* ARCH_??? */, 0x7fff6a98f5e0) = -1 EINVAL (Invalid argument)
[pid 4077] mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd5bd560000
[pid 4077] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
[pid 4077] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 5
[pid 4077] newfstatat(5, "", {st_mode=S_IFREG|0644, st_size=18175, ...}, AT_EMPTY_PATH) =
0
[pid 4077] mmap(NULL, 18175, PROT_READ, MAP_PRIVATE, 5, 0) = 0x7fd5bd55b000
[pid 4077] close(5)   = 0
[pid 4077] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) =
5
[pid 4077] 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 4077] 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 4077] 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 4077] pread64(5,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\244;\374\204(\337f#\315I\214\234\f\256\271\32"..., 68, 896) =
68
[pid 4077] newfstatat(5, "", {st_mode=S_IFREG|0755, st_size=2216304, ...},
AT_EMPTY_PATH) = 0
[pid 4077] 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 4077] mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 5, 0) =
0x7fd5bd333000
[pid 4077] mmap(0x7fd5bd35b000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 5, 0x28000) = 0x7fd5bd35b000
[pid 4077] mmap(0x7fd5bd4f0000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 5, 0x1bd000) = 0x7fd5bd4f0000
[pid 4077] mmap(0x7fd5bd548000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 5, 0x214000) = 0x7fd5bd548000
[pid 4077] mmap(0x7fd5bd54e000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd5bd54e000
[pid 4077] close(5)   = 0

[pid 4077] mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd5bd330000
[pid 4077] arch_prctl(ARCH_SET_FS, 0x7fd5bd330740) = 0
[pid 4077] set_tid_address(0x7fd5bd330a10) = 4077
[pid 4077] set_robust_list(0x7fd5bd330a20, 24) = 0
[pid 4077] rseq(0x7fd5bd3310e0, 0x20, 0, 0x53053053) = 0
[pid 4077] mprotect(0x7fd5bd548000, 16384, PROT_READ) = 0
[pid 4077] mprotect(0x5603e3c4d000, 4096, PROT_READ) = 0
[pid 4077] mprotect(0x7fd5bd59a000, 8192, PROT_READ) = 0
[pid 4077] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
[pid 4077] munmap(0x7fd5bd55b000, 18175) = 0
[pid 4077] getrandom("\x6a\xec\xe0\xec\xae\xe4\xc0\x8a", 8, GRND_NONBLOCK) = 8
[pid 4077] brk(NULL)   = 0x5603e4d5a000
[pid 4077] brk(0x5603e4d7b000)   = 0x5603e4d7b000
[pid 4077] read(0, 2.txt
 <unfinished ...>
[pid 4040] <... read resumed>"2", 1)   = 1
[pid 4040] read(0, ".", 1)   = 1
[pid 4040] read(0, "t", 1)   = 1
[pid 4040] read(0, "x", 1)   = 1
[pid 4040] read(0, "t", 1)   = 1
[pid 4040] read(0, "\n", 1)   = 1
[pid 4040] openat(AT_FDCWD, "2.txt", O_WRONLY|O_CREAT, 0777) = 8
[pid 4040] clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process
4078 attached
, child_tidptr=0x7fbaf270ba10) = 4078
[pid 4078] set_robust_list(0x7fbaf270ba20, 24 <unfinished ...>
[pid 4040] close(4 <unfinished ...>
[pid 4078] <... set_robust_list resumed>) = 0
[pid 4040] <... close resumed>)   = 0
[pid 4040] close(6 <unfinished ...>
[pid 4078] close(3 <unfinished ...>
[pid 4040] <... close resumed>)   = 0
[pid 4078] <... close resumed>)   = 0
[pid 4040] write(1, "Enter something you want: \0", 27 <unfinished ...>
Enter something you want: [pid 4078] close(4 <unfinished ...>
[pid 4040] <... write resumed>)   = 27
[pid 4078] <... close resumed>)   = 0
[pid 4040] read(0, <unfinished ...>
[pid 4078] close(5)   = 0
[pid 4078] close(7)   = 0
[pid 4078] dup2(6, 0)   = 0
[pid 4078] dup2(8, 1)   = 1
[pid 4078] dup2(8, 2)   = 2

[pid 4078] execve("./child_2", ["./child_2"], 0x7ffc9857a858 /* 34 vars */) = 0
[pid 4078] brk(NULL) = 0x5593cf42e000
[pid 4078] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffed8b57b60) = -1 EINVAL (Invalid argument)
[pid 4078] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd795075000
[pid 4078] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
[pid 4078] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
[pid 4078] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=18175, ...}, AT_EMPTY_PATH) = 0
[pid 4078] mmap(NULL, 18175, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fd795070000
[pid 4078] close(3) = 0
[pid 4078] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
[pid 4078] 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 4078] 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 4078] 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 4078] pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\244;\374\204(\337f#\315I\214\234\f\256\271\32"..., 68, 896) = 68
[pid 4078] newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2216304, ...}, AT_EMPTY_PATH) = 0
[pid 4078] 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 4078] mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fd794e48000
[pid 4078] mmap(0x7fd794e70000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7fd794e70000
[pid 4078] mmap(0x7fd795005000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7fd795005000
[pid 4078] mmap(0x7fd79505d000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000) = 0x7fd79505d000
[pid 4078] mmap(0x7fd795063000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd795063000
[pid 4078] close(3) = 0
[pid 4078] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd794e45000
[pid 4078] arch_prctl(ARCH_SET_FS, 0x7fd794e45740) = 0
[pid 4078] set_tid_address(0x7fd794e45a10) = 4078
[pid 4078] set_robust_list(0x7fd794e45a20, 24) = 0
[pid 4078] rseq(0x7fd794e460e0, 0x20, 0, 0x53053053) = 0
[pid 4078] mprotect(0x7fd79505d000, 16384, PROT_READ) = 0
[pid 4078] mprotect(0x5593ce8f2000, 4096, PROT_READ) = 0

```
[pid  4078] mprotect(0x7fd7950af000, 8192, PROT_READ) = 0
[pid  4078] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
[pid  4078] munmap(0x7fd795070000, 18175) = 0
[pid  4078] getrandom("\xec\x9e\xd7\x9b\x67\xb4\x89\xf0", 8, GRND_NONBLOCK) = 8
[pid  4078] brk(NULL)                   = 0x5593cf42e000
[pid  4078] brk(0x5593cf44f000)         = 0x5593cf44f000
[pid  4078] read(0, one
 <unfinished ...>
[pid  4040] <... read resumed>"o", 1)   = 1
[pid  4040] read(0, "n", 1)             = 1
[pid  4040] read(0, "e", 1)             = 1
[pid  4040] read(0, " ", 1)             = 1
[pid  4040] read(0, "\n", 1)            = 1
[pid  4040] write(7, "one \n", 5)       = 5
[pid  4078] <... read resumed>"o", 1)   = 1
[pid  4040] write(5, "-\0", 2 <unfinished ...>
[pid  4078] read(0,  <unfinished ...>
[pid  4040] <... write resumed>)        = 2
[pid  4078] <... read resumed>"n", 1)   = 1
[pid  4077] <... read resumed>"-", 1)   = 1
[pid  4078] read(0,  <unfinished ...>
[pid  4040] read(0,  <unfinished ...>
[pid  4078] <... read resumed>"e", 1)   = 1
[pid  4078] read(0,  <unfinished ...>
[pid  4077] read(0,  <unfinished ...>
[pid  4078] <... read resumed>" ", 1)   = 1
[pid  4077] <... read resumed>"\0", 1)  = 1
[pid  4078] read(0,  <unfinished ...>
[pid  4077] read(0,  <unfinished ...>
[pid  4078] <... read resumed>"\n", 1)  = 1
[pid  4078] write(1, " eno", 4)         = 4
[pid  4078] read(0, two
 <unfinished ...>
[pid  4040] <... read resumed>"t", 1)   = 1
[pid  4040] read(0, "w", 1)             = 1
[pid  4040] read(0, "o", 1)             = 1
[pid  4040] read(0, " ", 1)             = 1
[pid  4040] read(0, "\n", 1)            = 1
[pid  4040] write(5, "two \n", 5)       = 5
[pid  4077] <... read resumed>"t", 1)   = 1
[pid  4040] write(7, "-\0", 2 <unfinished ...>
[pid  4077] read(0,  <unfinished ...>
[pid  4040] <... write resumed>)        = 2
[pid  4078] <... read resumed>"-", 1)   = 1
[pid  4040] read(0,  <unfinished ...>
```

```
[pid  4078] read(0, "\0", 1)            = 1
[pid  4077] <... read resumed>"w", 1)   = 1
[pid  4078] read(0,  <unfinished ...>
[pid  4077] read(0, "o", 1)             = 1
[pid  4077] read(0, " ", 1)             = 1
[pid  4077] read(0, "\n", 1)            = 1
[pid  4077] write(1, " owt", 4)         = 4
[pid  4077] read(0, three
 <unfinished ...>
[pid  4040] <... read resumed>"t", 1)   = 1
[pid  4040] read(0, "h", 1)             = 1
[pid  4040] read(0, "r", 1)             = 1
[pid  4040] read(0, "e", 1)             = 1
[pid  4040] read(0, "e", 1)             = 1
[pid  4040] read(0, " ", 1)             = 1
[pid  4040] read(0, "\n", 1)            = 1
[pid  4040] write(5, "three \n", 7)     = 7
[pid  4077] <... read resumed>"t", 1)   = 1
[pid  4040] write(7, "-\0", 2 <unfinished ...>
[pid  4077] read(0, "h", 1)             = 1
[pid  4040] <... write resumed>)        = 2
[pid  4078] <... read resumed>"-", 1)   = 1
[pid  4040] read(0,  <unfinished ...>
[pid  4077] read(0,  <unfinished ...>
[pid  4078] read(0,  <unfinished ...>
[pid  4077] <... read resumed>"r", 1)   = 1
[pid  4078] <... read resumed>"\0", 1)  = 1
[pid  4077] read(0,  <unfinished ...>
[pid  4078] read(0,  <unfinished ...>
[pid  4077] <... read resumed>"e", 1)   = 1
[pid  4077] read(0, "e", 1)             = 1
[pid  4077] read(0, " ", 1)             = 1
[pid  4077] read(0, "\n", 1)            = 1
[pid  4077] write(1, " eerht", 6)       = 6
[pid  4077] read(0, four
 <unfinished ...>
[pid  4040] <... read resumed>"f", 1)   = 1
[pid  4040] read(0, "o", 1)             = 1
[pid  4040] read(0, "u", 1)             = 1
[pid  4040] read(0, "r", 1)             = 1
[pid  4040] read(0, " ", 1)             = 1
[pid  4040] read(0, "\n", 1)            = 1
[pid  4040] write(5, "four \n", 6)      = 6
[pid  4077] <... read resumed>"f", 1)   = 1
[pid  4040] write(7, "-\0", 2 <unfinished ...>
[pid  4077] read(0,  <unfinished ...>
```

```
[pid  4040] <... write resumed>)        = 2
[pid  4078] <... read resumed>"-", 1)   = 1
[pid  4077] <... read resumed>"o", 1)   = 1
[pid  4040] read(0,  <unfinished ...>
[pid  4078] read(0, "\0", 1)            = 1
[pid  4077] read(0,  <unfinished ...>
[pid  4078] read(0,  <unfinished ...>
[pid  4077] <... read resumed>"u", 1)   = 1
[pid  4077] read(0, "r", 1)             = 1
[pid  4077] read(0, " ", 1)             = 1
[pid  4077] read(0, "\n", 1)            = 1
[pid  4077] write(1, " ruof", 5)        = 5
[pid  4077] read(0,
 <unfinished ...>
[pid  4040] <... read resumed>"\n", 1)  = 1
[pid  4040] write(1, "\nProgramm was ended successfully"..., 35
Programm was ended successfully!
) = 35
[pid  4040] close(5)                    = 0
[pid  4077] <... read resumed>"", 1)    = 0
[pid  4040] close(7 <unfinished ...>
[pid  4077] exit_group(0 <unfinished ...>
[pid  4040] <... close resumed>)        = 0
[pid  4078] <... read resumed>"", 1)    = 0
[pid  4040] close(3 <unfinished ...>
[pid  4077] <... exit_group resumed>)   = ?
[pid  4040] <... close resumed>)        = 0
[pid  4078] exit_group(0 <unfinished ...>
[pid  4040] close(8)                    = 0
[pid  4040] kill(4077, SIGTERM <unfinished ...>
[pid  4078] <... exit_group resumed>)   = ?
[pid  4040] <... kill resumed>)         = 0
[pid  4040] kill(4078, SIGTERM)         = 0
[pid  4040] exit_group(0)               = ?
[pid  4040] +++ exited with 0 +++
[pid  4077] +++ 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.