**МИНОБРНАУКИ РОССИИ**

**САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра ВТ**

**ОТЧЕТ**

**по лабораторной работе №7**

**по дисциплине *«Организация процессов и программирование в среде Linux»***

**Тема: «ОБМЕН ДАННЫМИ ЧЕРЕЗ КАНАЛ»**

Cтудентка гр. 7306 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мищенко А.В.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Разумовский Г.В.

Санкт–Петербург

2020

**Цель работы:**

Знакомство с механизмом обмена данными через программный канал и системными вызовами, обеспечивающими такой обмен

**Задание:**

Написать программу, которая в качестве параметров принимает

имена трех текстовых файлов (2 входных и 1 выходной). Программа должна открыть канал и выходной файл, а затем породить двух потомков, которым передаются дескриптор канала для записи и имя входного файла. Каждый потомок выполняет свою программу, читая построчно текст из входного файла и записывая его в канал. Программа параллельно посимвольно читает данные из канала и записывает их в выходной файл, до тех пор пока оба потомка не закончат свою работу и канал будет пуст.

**Ход работы:**

В ходе выполнения работы была написана программа из двух файлов: main.cpp и childExec.cpp, исходный код которых представлен в Приложении 1.

При запуске приложения нужно передать три аргумента: файл in1.txt, из которого будет считать данные первый процесс-потомок, файл in2.txt, из которого будет считать данные второй процесс-потомок и выходной файл родителя. Содержимое файлов in1.txt, in2.txt и out.txt после запуска представлены на рисунках 1 - 3 соответственно.

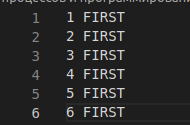


Рисунок 1. Содержимое файла in1.txt

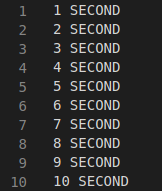


Рисунок 2. Содержимое файла in2.txt

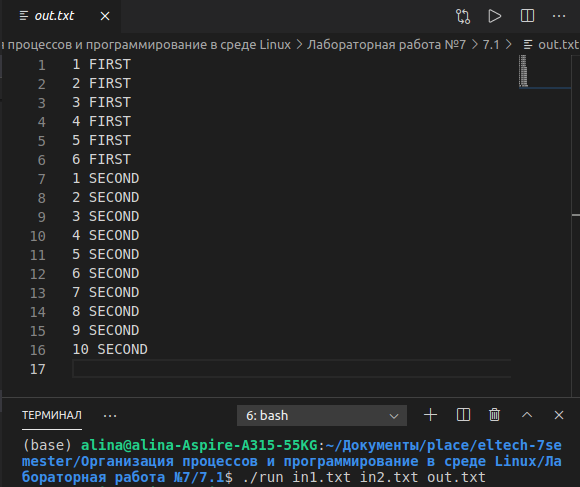


Рисунок 3. Содержимое выходного файла

**Вывод:**

В ходе лабораторной работы я научилась работать с механизмом каналов.

**Приложения**

Приложение 1.1. Файл main.cpp

#include <sys/wait.h>

#include <assert.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

// argv[1] - имя первого файла, argv[2] - второго файла, argv[3] - выходного файла

int main(int argc, char\* argv[]) {

FILE\* file = NULL; //выходной файл

// fd[0] - чтение из канала, fd[1] - запись в канал

int fd[2];

pid\_t pid\_1, pid\_2;

char ch;

assert(argc == 4);

//открываем канал

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

perror("open pipe");

exit(EXIT\_FAILURE);

}

//открываем файл для записи

if ((file = fopen(argv[3], "w")) == NULL) {

perror("open file");

exit(EXIT\_FAILURE);

}

//создаем первого процесс-потомка

if (!(pid\_1 = fork())) {

//закрываем канал для чтения

close(fd[0]);

//запускаем подпрограмму потомка с передачей дескриптора канала для записи и имени его файла

execl("childExec", "childExec", &fd[1], argv[1], NULL);

}

//создаем второго процесс-потомка

else if (!(pid\_2 = fork())) {

//закрываем канал для чтения

close(fd[0]);

//запускаем подпрограмму потомка с передачей дескриптора канала для записи и имени его файла

execl("childExec", "childExec", &fd[1], argv[2], NULL);

}

else {

// Закрытие канала на запись

close(fd[1]);

// Цикл чтения символов из канала и записи их в файл

while (read(fd[0], &ch, 1) > 0) {

fputc(ch, file);

}

// Ожидание завершения работы потомков

wait(&pid\_1);

wait(&pid\_2);

//закрываем канал на чтение и файл

close(fd[0]);

fclose(file);

}

}

Приложение 1.2. childExec.cpp

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <fstream>

using namespace std;

// argv[1] - дескриптор для записи в канал, argv[2] - имя файла

int main(int argc, char\* argv[])

{

ifstream file(argv[2]);

if (file.is\_open()) {

string s;

while(getline(file, s)){

s+="\n";

write(\*argv[1], &s[0], strlen(s.c\_str()));

}

file.close();

close(\*argv[1]);

}

exit(EXIT\_SUCCESS);

}