|  |
| --- |
| «Санкт-Петербургский государственный электротехнический университет |
| «ЛЭТИ» им. В.И.Ульянова(Ленина)» |
| (СПбГЭТУ «ЛЭТИ») Кафедра вычислительной техники |
|  |
| **Отчет**  **по лабораторным работам № 4**  **по дисциплине** **«Организация процессов и программирование в среде Linux»** |
| Тема: «Управление потоками» |
|  |

|  |
| --- |
|  |

|  |  |
| --- | --- |
| Студент гр. 9307 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_Брызгалова Е. А. |
|  |  |
|  |  |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Разумовский Г. В. |

Оглавление

[Цель работы: 3](#_Toc115542869)

[Задание 3](#_Toc115542870)

[Распечатка входных и выходных файлов. 3](#_Toc115542871)

[Тексты программ 5](#_Toc115542874)

[Текст программы main.cpp 5](#_Toc115542875)

[Вывод 8](#_Toc115542876)

Цель работы:

Знакомство с организацией потоков и способами синхронизации предков и потомков.

Задание

1. Написать программу, которая открывает входной файл и 2 выходных файла. Затем она должна в цикле построчно читать входной файл и порождать 2 потока. Одному потоку передавать нечетную строку, а другому – четную. Оба потока должны работать параллельно. Каждый поток записывает в свой выходной файл полученную строку и завершает работу. Программа должна ожидать завершения работы каждого потока и повторять цикл порождения потоков и чтения строк входного файла, пока не прочтет последнюю строку, после чего закрыть все файлы.
2. Откомпилировать программу и запустить ее.

## Распечатка входных и выходных файлов.

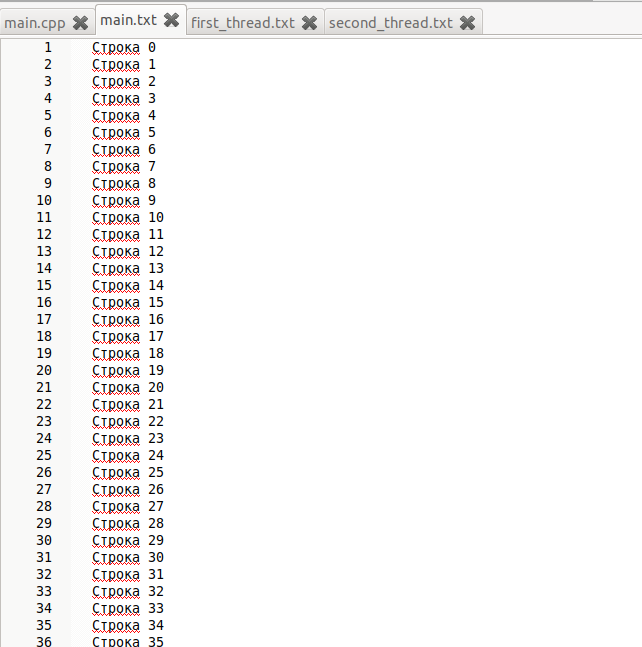


Рисунок 1. Распечатка файла main.txt

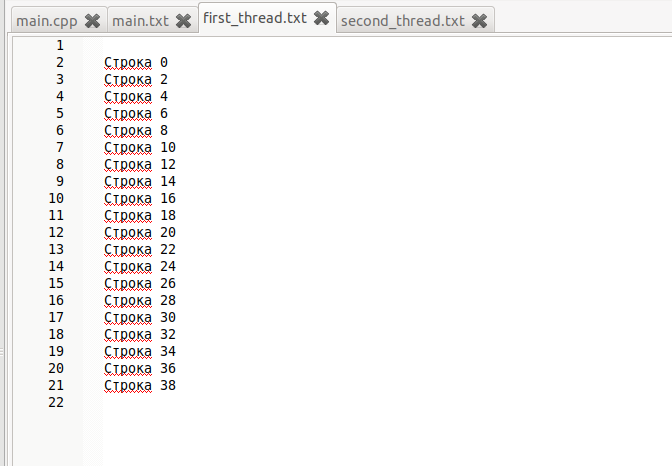


Рисунок 2. Распечатка файла first\_thread.txt

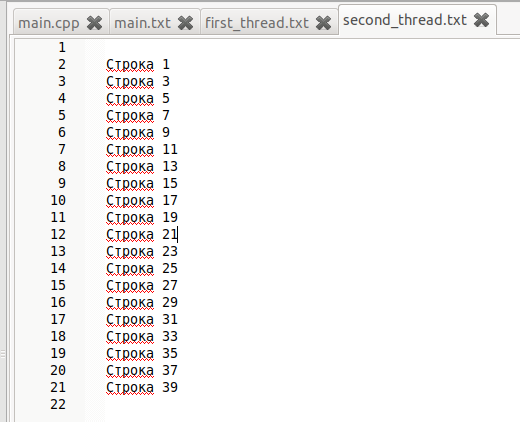


Рисунок 3. Распечатка файла second\_thread.txt

Тексты программ

### Текст программы main.cpp

#include <fstream>

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <iostream>

using namespace std;

struct thread\_arg

{

string file\_name;

string str;

};

void\* Thread\_func(void \*arg) {

ofstream file;

struct thread\_arg targ = \*(struct thread\_arg \*) arg;

file.open(targ.file\_name, std::ios::app);

file<<targ.str<<endl;

file.close();

return 0;

}

int main() {

pthread\_t thread1, thread2;

int status;

int status\_addr;

ifstream main\_file;

struct thread\_arg targ1,targ2;

bool f2 = 1;

main\_file.open("./main.txt");

targ2.file\_name= "./second\_thread.txt";

targ1.file\_name= "./first\_thread.txt";

while(f2){

if(getline(main\_file, targ1.str)){

status = pthread\_create(&thread1, NULL, Thread\_func, &targ1);

if (status != 0) {

cout<<"main error: can't create thread, status = "<< status;

exit(2);

}

status = pthread\_join(thread1, (void\*\*)&status\_addr);

if (status != 0) {

cout<<"main error: can't join thread, status = "<< status<<endl;

exit(1);

}

// cout<<"joined with address "<<status\_addr<<endl;

}

if(getline(main\_file, targ2.str)){

status = pthread\_create(&thread2, NULL, Thread\_func, &targ2);

if (status != 0) {

cout<<"main error: can't create thread, status = "<< status;

exit(2);

}

if (status != 0) {

cout<<"main error: can't join thread, status = "<< status;

exit(1);

}

//cout<<"joined with address "<<status\_addr;

status = pthread\_join(thread1, (void\*\*)&status\_addr);

}

else

f2 = false;

}

status = pthread\_join(thread1, (void\*\*)&status\_addr);

status = pthread\_join(thread2, (void\*\*)&status\_addr);

return 0;

}

Вывод

В процессе выполнения лабораторной работы познакомились с организацией потоков и способами синхронизации предков и потомков.

.