## МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

## «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

**Кафедра Вычислительной техники**

## ОТЧЕТ

**по лабораторной работе №4**

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

**Тема: Управление потоками**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9308 |  | Яловега Н.В. |
| Преподаватель |  | Разумовский Г.В. |

Санкт-Петербург

2022

# Цель работы.

Знакомство с организацией потоков и способами синхронизации предков и потомков.

# Задание.

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

# Выполнение работы

Компиляция программы:

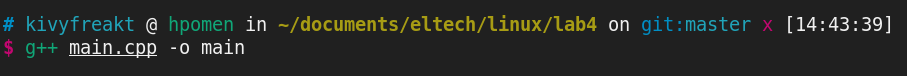


Рисунок 1. Компиляция программ

Запуск программы с входным файлом с 10 строками.

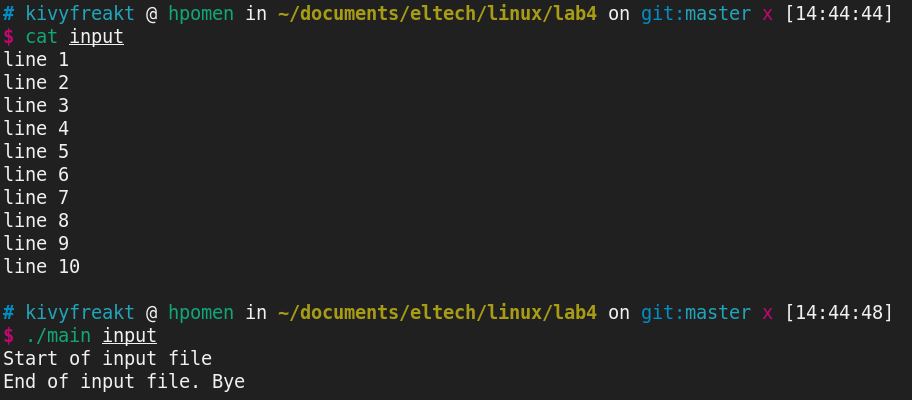
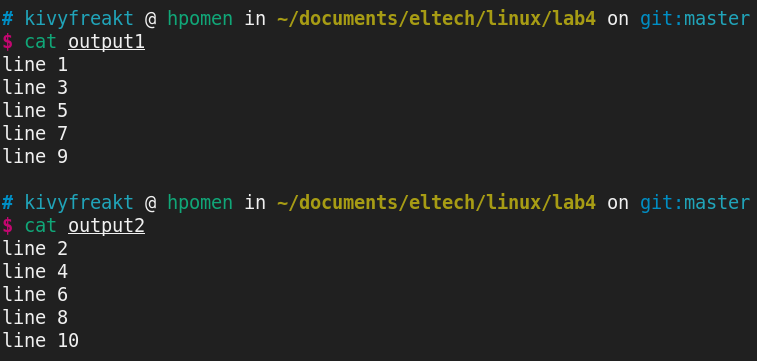


Рисунок 2. Запуск программы

Результат первого запуска представлен на рисунке 3.



*Рисунок 3. Результат выполнения*

Запуск программы с входным файлом с 5 строками.

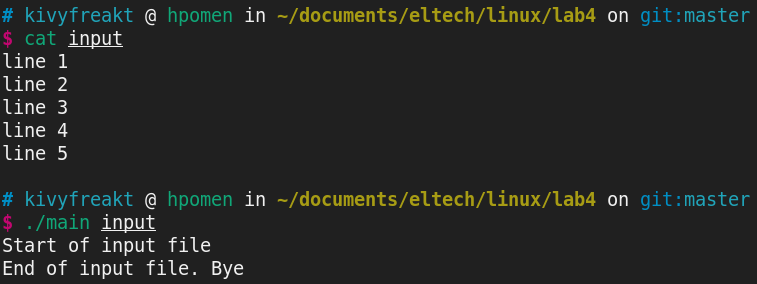


Рисунок 4. Запуск программы

Результат второго запуска представлен на рисунке 5.

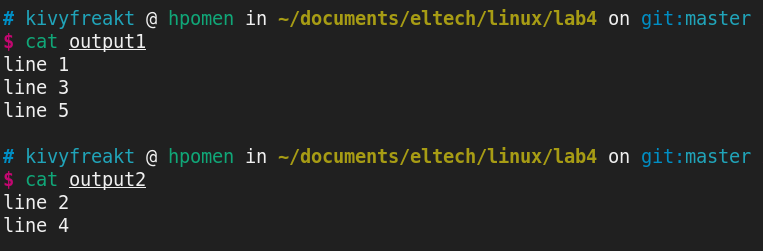


Рисунок 5. Результат выполнения

Запуск программы с пустым файлом.

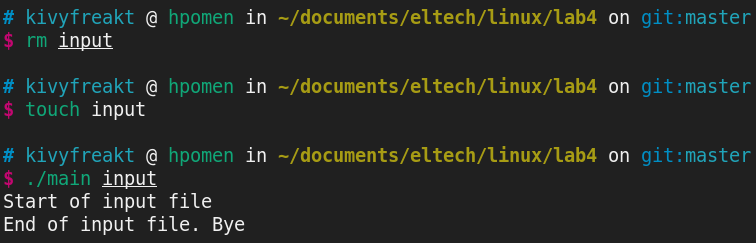


Рисунок 6. Запуск программы

Результат третьего запуска представлен на рисунке 7.

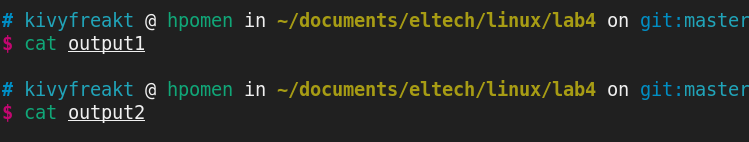


Рисунок 7. Результат выполнения

# Вывод

В ходе работы были изучены механизмы создания и управления потоков на примере параллельного переписывания входного файла в 2 выходных.

# Приложение

# main.cpp

#include <sys/types.h>

#include <pthread.h>

#include <iostream>

#include <fstream>

#include <string>

typedef struct \_thread\_args

{

std::fstream \*file;

const char \*str;

} thread\_args;

void \*thread\_routine(void \*args)

{

thread\_args \*arg = (thread\_args\*)args;

std::fstream &file\_out = \*(arg->file);

file\_out << arg->str << std::endl;

return NULL;

}

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

{

std::fstream input\_file;

std::fstream output1\_file;

std::fstream output2\_file;

if(argc == 2)

input\_file.open(argv[1], std::ios::in);

else

input\_file.open("input", std::ios::in);

output1\_file.open("output1", std::ios::out | std::ios::trunc);

output2\_file.open("output2", std::ios::out | std::ios::trunc);

thread\_args thr\_arg\_1,

thr\_arg\_2;

thr\_arg\_1.file = &output1\_file;

thr\_arg\_2.file = &output2\_file;

bool end\_of\_file = false;

std::string buffer1, buffer2;

pthread\_t thread\_1 = 0, thread\_2 = 0;

if(input\_file && output1\_file && output2\_file)

{

std::cout << "Start of input file\n";

while(!end\_of\_file)

{

if(getline(input\_file, buffer1))

{

thr\_arg\_1.str = buffer1.c\_str();

if(pthread\_create(&thread\_1, NULL, &thread\_routine, &thr\_arg\_1))

{

std::cout << "Thread\_1 ERROR\n";

return -1;

}

}

else

{

end\_of\_file = true;

continue;

}

if(getline(input\_file, buffer2))

{

thr\_arg\_2.str = buffer2.c\_str();

if(pthread\_create(&thread\_2, NULL, &thread\_routine, &thr\_arg\_2))

{

std::cout << "Thread\_2 ERROR\n";

return -1;

}

}

else

end\_of\_file = true;

pthread\_join(thread\_1, NULL);

pthread\_join(thread\_2, NULL);

}

std::cout << "End of input file. Bye\n";

input\_file.close();

output1\_file.close();

output2\_file.close();

}

else

std::cout << "ERROR: cant open file\n";

return 0;

}