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 программированиев 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:

```
# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab4 on git:master x [14:43:39]
$ g++ main.cpp -o main
```

Рисунок 1. Компиляция программ

Запуск программы с входным файлом с 10 строками.

```
# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab4 on git:master x [14:44:44]
$ cat input
line 1
line 2
line 3
line 4
line 5
line 6
line 7
line 8
line 9
line 10

# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab4 on git:master x [14:44:48]
$ ./main input
Start of input file
End of input file. Bye
```

Рисунок 2. Запуск программы

Результат первого запуска представлен на рисунке 3.

```
# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab4 on git:master
$ cat output1
line 1
line 3
line 5
line 7
line 9

# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab4 on git:master
$ cat output2
line 2
line 4
line 6
line 8
line 10
```

Рисунок 3. Результат выполнения

Запуск программы с входным файлом с 5 строками.

```
# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab4 on git:master
$ cat input
line 1
line 2
line 3
line 4
line 5

# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab4 on git:master
$ ./main input
Start of input file
End of input file. Bye
```

Рисунок 4. Запуск программы

Результат второго запуска представлен на рисунке 5.

```
# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab4 on git:master
$ cat output1
line 1
line 3
line 5

# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab4 on git:master
$ cat output2
line 2
line 4
```

Рисунок 5. Результат выполнения

Запуск программы с пустым файлом.

```
# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab4 on git:master
$ rm input

# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab4 on git:master
$ touch input

# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab4 on git:master
$ ./main input
Start of input file
End of input file. Bye
```

Рисунок 6. Запуск программы

Результат третьего запуска представлен на рисунке 7.

```
# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab4 on git:master
$ cat output1

# kivyfreakt @ hpomen in ~/documents/eltech/linux/lab4 on git:master
$ cat output2
```

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;
}
```