Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра информационных технологий автоматизированных систем

Факультет ИТиУ
Специальность АСОИ

Индивидуальная практическая работа по модулю 1 по дисциплине «Системное программное обеспечение», часть 1 «Процессы и потоки» Вариант №1

Выполнил:
Ст. Гр. 820601
Шведов А.Р
Зачетная книжка No 82060145

Минск 2020

1. Задание

Во всех вариантах заданий Написание программы, использующей системные объекты ОС для синхронизации потоков разных процессов

Имеется массив элементов типа Date в виде структуры struct Dates

```
{
int count = 0;//количество имеющихся элементов в массиве Date
dates[100];
} dts = {0};
```

Главный поток программы (функция *main()*) создает вторичный поток, передав в него указатель на *структуру dts*.

Вторичный поток запоминает значение из поля *count*, открывает файл и затем в цикле, если значение *count* изменилось, то записывает последний элемент массива *dates* в файл. Так продолжается до тех пор, пока *count* не достигнет некоторого максимального значения, после этого поток закрывает файл и завершается;

Далее главный поток организует цикл ввода дат следующим образом:

- инициализируется временная переменная tmp типа Date (ввод с клавиатуры);
- с помощью функции SuspendThread() приостанавливается поток;
- значение временной переменной заносится в массив;
- dts.dates[dts.count] = tmp;
- dts.count++;
- с помощью функции ResumeTread() поток запускается на выполнение.

Так продолжается до тех пор, пока *count* не достигнет некоторого максимального значения.

2. Ход работы

2.1. Теоретические сведения

Поток - последовательность инструкций, которые выполняются параллельно с другими потоками. Каждая программа создает по меньшей мере один поток: основной, который запускает функцию *main()*. Программа, использующая только главный поток, является однопоточной; если добавить один или более

потоков, она станет многопоточной.

Данная работу выполнялась на операционной системе Mac OS с помощью стандартных средств C++ 14 для работы с потоками.

2.2. Листинг программы

Файл “main.cpp”

```
#include <thread>
#include <iostream>
#include <fstream>
#include <mutex>
#include <condition_variable>

using namespace std;

static const int MAX_COUNT = 5;

struct Date {
    int year;
    int month;
    int day;
};

struct Dates {
    int count = 0; //количество имеющихся элементов в массиве
    Date dates[100]{};
} dts = {0};

mutex kLock;
condition_variable kCv;
bool kReady = false;
bool kProcessed = false;

void secondThread(Dates* dates){
    ofstream datesFile("dates.txt");

    int currentCount = dates->count;

    while (currentCount < MAX_COUNT) {
        // дождаться передачи управления от главного потока
        {
            unique_lock<std::mutex> lk(kLock);
```

```
            kCv.wait(lk, [] { return kReady; } );
        }

        if (dates->count != currentCount) {
            currentCount = dates->count;
            Date date = dates->dates[currentCount-1];
            datesFile << "date[" << currentCount << "]: { day: " << date.day << ", month:
" << date.month << ", year: " << date.year << " } \n" << endl;
        }

        {
            std::lock_guard<std::mutex> lk(kLock);
            kProcessed = true;
            kCv.notify_one();
        }
    }

    datesFile.close();
}

int main(int argc, const char * argv[]) {
    thread secondThr(secondThread, &dts);
    secondThr.detach();

    while (dts.count < MAX_COUNT) {
        Date tmp{};

        cout << "Enter year: ";
        cin >> tmp.year;
        cout << endl << "Enter month: ";
        cin >> tmp.month;
        cout << endl << "Enter day: ";
        cin >> tmp.day;
        cout << endl;

        dts.dates[dts.count] = tmp;
        dts.count++;

        {
            // передать управление второму потоку
            std::lock_guard<std::mutex> lk(kLock);
            kReady = true;
            kCv.notify_one();
        }
```

```
        // дождаться выполнения второго потока
        std::unique_lock<std::mutex> lk(kLock);
        kCv.wait(lk, []{return kProcessed;});
    }

    return 0;
}
```

3. Выводы

В ходе выполнения данной работы была написана программа, работающая с потоками средствами C++, были изучены теоретические сведения и практические сведения работы с многопоточными программами.