БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра информационных технологий автоматизированных систем

Факультет ФИТУ

Специальность АСОИ

Индивидуальная практическая работа по модулю 4

по дисциплине «Системное программное обеспечение», часть 1

ВАРИАНТ 1

Выполнил студент: Гриб В.А.

группа 820602

Зачетная книжка № 82060030

Минск 2020

**Индивидуальное задание**

Во всех вариантах заданий требуется синхронизировать потоки с помощью одного из следующих методов синхронизации:

• критическая секция;

• Mutex;

• событие;

• семафоры.

Задача для решения:

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.

**Решение**Для синхронизации потоков мною был использован мьютех.

Текст 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< 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;

}

{

**lock\_guard**<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**

**lock\_guard<**mutex**> lk(**kLock**);**

kReady = true;

kCv.notify\_one();

}

// дождаться выполнения второго потока

**unique\_lock**<mutex> **lk**(kLock);

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

}

**return** 0;

}

**Выводы:**

В ходе выполнения данной работы мною были изучены средства для синхронизации потоков в C++;