Министерство цифрового развития, связи и массовых коммуникаций

Российской Федерации

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский технический университет связи и информатики»

Кафедра «Системного программирования»

Отчет по лабораторной работе №3

на тему «Синхронизация процессов и потоков»

по дисциплине «Операционные системы»

Выполнил:

студент группы БВТ1903

Игнатов Д.В.

**Методы синхронизация потоков**

**Цель работы***:* получение практических навыков по использованию Win32 API для синхронизации потоков.

**Задание***:* Исследовать на конкретном примере следующих методов синхронизации потоков:

* критические секции
* мьютексы
* события

Задачу для синхронизации выбрать на свое усмотрение.

Задачи:

В первой задаче критические секции используются при работе с общим массивом. Один поток записывает значения, равные счетчику, в массив, второй поток меняет текущее значение массива по формуле.

Во второй программе используется мьютекс при работе двух процессов с общим кодом.

Исходный код:

// OC\_Lab3\_Part1.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <Windows.h>

#include <iostream>

#include <fstream>

#include "process.h"

using namespace std;

#define SIZE 5

int arr[SIZE] = {0, 0, 0, 0, 0};

CRITICAL\_SECTION critsect;

HANDLE hMutex;

HANDLE hEvent;

DWORD WINAPI thr1(LPVOID lpParam) //критическая секция

{

EnterCriticalSection(&critsect); //вход

for (int i = 0; i < SIZE; i++)

{

arr[i] = i;

cout << "Поток 1: " << "Элемент массива " << i << " = " << arr[i] << endl;

}

LeaveCriticalSection(&critsect); //выход

return 0;

}

DWORD WINAPI thr2(LPVOID lpParam) //критическая секция

{

EnterCriticalSection(&critsect); //вход

for (int i = 0; i < 5; i++)

{

arr[i] = 2 \* i + i \* i;

cout << "Поток 2: " << "Элемент массива " << i << " = " << arr[i] << endl;

}

LeaveCriticalSection(&critsect); //выход

return 0;

}

DWORD WINAPI fun1(LPVOID lpParam) //событие

{

HANDLE hOpenEvent;

hOpenEvent = OpenEvent(EVENT\_ALL\_ACCESS, FALSE, TEXT("Event"));

cout << "Событие открыто" << endl;

for (int i = 1; i <= 5; i++)

{

WaitForSingleObject(hEvent, INFINITE);

cout << " Свободно (1) " << i << endl;

}

return 0;

}

DWORD WINAPI fun2(LPVOID lpParam) //событие

{

cout << "Поток запущен" << endl; //запуск и установка события в сигнальное состояние 5 раз

for (int i = 1; i <= 5; i++)

{

cout << " Свободно (2) " << i << endl;

SetEvent(hEvent);

Sleep(2000);

}

return 0;

}

DWORD WINAPI ThreadProc(LPVOID lpParameter) {

DWORD y;

//cout << " " << endl;

for (int y = 0; y < 5; y++) {

WaitForSingleObject(hMutex, INFINITE);

cout << y << endl;

Sleep(500);

ReleaseMutex(hMutex);

}

Sleep(5000);

return 0;

}

void main()

{

setlocale (LC\_ALL,"russian");

int Z, i;

do

{

cout << "Выберите задание: \n 1. - Критическая секция \n 2. – Мьютекс

\n 3. - Событие \n 0. - Выход из программы" << endl;

cin >> Z;

switch (Z)

{

case 1:

{

// Задание 1. Критическая секция:

cout << "Критическая секция: " <<endl << endl;

InitializeCriticalSection(&critsect);

//инициализация критической секции

HANDLE th1 = CreateThread

(NULL, 0, thr1, NULL, 0, NULL);

HANDLE th2 = CreateThread

(NULL, 0, thr2, NULL, 0, NULL);

WaitForSingleObject(th1, INFINITE); // остановка первого потока при изменении его состояния

WaitForSingleObject(th2, INFINITE); // остановка второго потока при изменении его состояния

DeleteCriticalSection(&critsect); //удаление критической секции

cout << endl;

system("pause");

CloseHandle(th1);

CloseHandle(th2);

break;

}

case 2:

{

//Задание 2. Мьютекс:

cout << "Мьютекс: " << endl << endl;

cout << "Откройте Part2" << endl << endl;

system("pause");

HANDLE hThreads[2];

hMutex = CreateMutex(NULL, FALSE, TEXT("mtx"));

for(i = 0; i < 2; i++) {

hThreads[i] = CreateThread

(NULL, 0, ThreadProc, hMutex, 0, NULL);

cout << " Работает поток \n" << endl;

int iWait = ::WaitForSingleObject

( hThreads[i], INFINITE );

switch( iWait ) {

case WAIT\_OBJECT\_0: // Все нормально

break;

}

}

for(i = 0; i < 2; i++) {

CloseHandle(hThreads[i]);

}

CloseHandle(hMutex);

cout << endl;

system("pause");

break;

}

case 3:

{

//Задание 3. Событие:

cout << "Событие: " << endl << endl;

DWORD ThID1, ThID2;

hEvent = CreateEvent(NULL, FALSE, FALSE, TEXT("Event")); // несигнальное состояние

cout << "Событие создано" << endl;

HANDLE hTh1 = CreateThread(NULL, 0, fun1, NULL, 0, &ThID1);

HANDLE hTh2 = CreateThread(NULL, 0, fun2, NULL, 0, &ThID1);

WaitForSingleObject(hTh1, INFINITE);

//остановка, ожидание сигнального состояние fun1

WaitForSingleObject(hTh2, INFINITE);

CloseHandle(hTh1);

CloseHandle(hTh2);

CloseHandle(hEvent);

cout << endl;

system("pause");

break;

}

}

}

while (Z != 0);

}

Код Part2

// OC\_Lab3\_Part2.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <Windows.h>

#include <iostream>

#include <fstream>

#include "process.h"

using namespace std;

#define SIZE 5

int arr[SIZE] = {0, 0, 0, 0, 0};

int i = 0;

CRITICAL\_SECTION critsect;

HANDLE hMutex;

HANDLE hEvent;

DWORD WINAPI ThreadProc(LPVOID lpParameter) {

DWORD y;

//cout << " " << endl;

for (int y = 0; y < 5; y++) {

WaitForSingleObject(hMutex, INFINITE);

cout << y << endl;

Sleep(500);

ReleaseMutex(hMutex);

}

Sleep(5000);

return 0;

}

void main(){

setlocale (LC\_ALL,"russian");

//Задание 2. Мьютекс:

system("pause");

cout << "Мьютекс: " << endl << endl;

HANDLE hThreads[2];

hMutex = OpenMutex(NULL, FALSE, TEXT("mtx"));

for(i = 0; i < 2; i++) {

hThreads[i] = CreateThread(NULL, 0, ThreadProc, hMutex, 0, NULL);

cout << " Работает поток \n"<< endl;

int iWait = ::WaitForSingleObject( hThreads[i], INFINITE );

switch( iWait ) {

case WAIT\_OBJECT\_0: // Все нормально

break;

}

}

for(i = 0; i < 2; i++) {

CloseHandle(hThreads[i]);

}

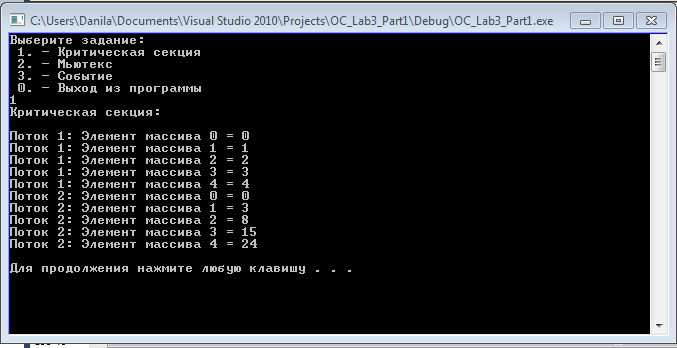
CloseHandle(hMutex);

cout << endl;

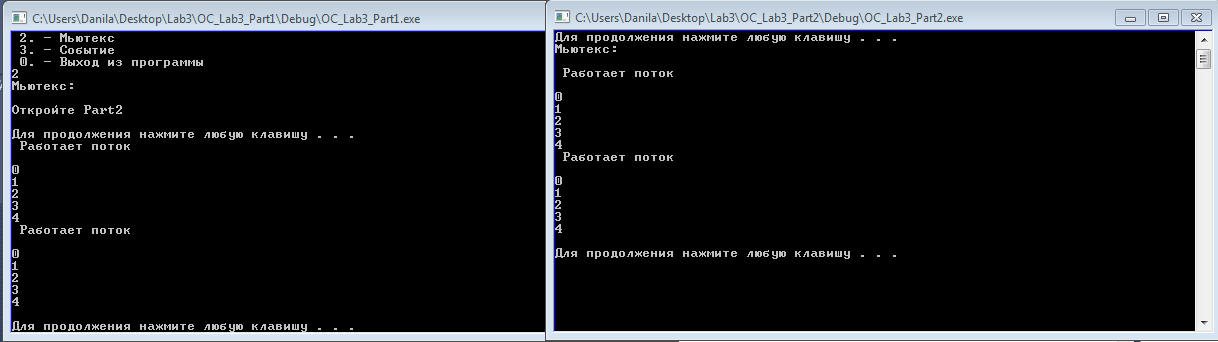
system("pause");

}

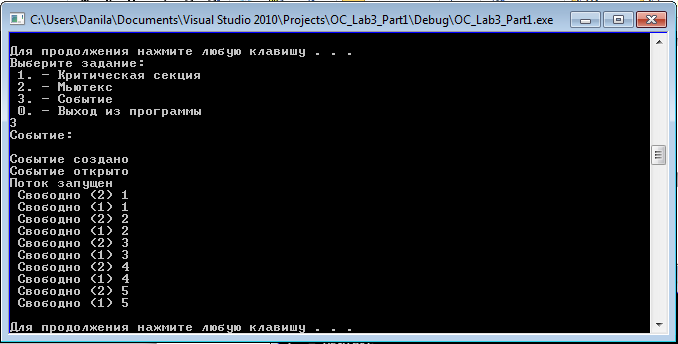
Результаты выполнения:



*Снимок экрана «Критические секции»*



*Снимок экрана «Мьютексы»*



*Снимок экрана «События»*