Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования «Московский технический университет связи и информатики»

Лабораторная №3

«Методы синхронизации потоков»

Выполнила студентка

группы БВТ1501

Козлова Анна

8 вариант

МОСКВА, 2018

**Цель работы:**

Получение практических навыков по использованию Win32 API для синхронизации потоков.

**Код:**

#include <windows.h>

#include <iostream>

#include <process.h>

#include <conio.h>

#include <sstream>

using namespace std;

#include <stdio.h>

CRITICAL\_SECTION cs; //для критических секций

int a = 1, k; //а - общая переменная, к - счетчик потоков

HANDLE event;

DWORD dwWaitResult;

void Thread1(void\* pParams)

{

EnterCriticalSection(&cs);

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

{

a = a \* 6 / 2;

printf("%d (a\*6/2)\n", a);

}

LeaveCriticalSection(&cs);

k++;

\_endthread();

}

void Thread2(void\* pParams)

{

EnterCriticalSection(&cs);

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

{

a += 1;

printf("%d (+1)\n", a);

}

LeaveCriticalSection(&cs);

k++;

\_endthread();

}

void Thread3(void\* pParams)

{

EnterCriticalSection(&cs);

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

{

a += 2 + a\*9/9 - 1;

printf("%d (2+a\*9/3-1)\n", a);

}

LeaveCriticalSection(&cs);

k++;

\_endthread();

}

//это поток, где всех почитают и молчат

void Thread12(void\* pParams)

{

WaitForSingleObject(event, INFINITE);

int m = 0;

cout << "Первый раз можно помолчать!" << endl;

while (m < 20) {

if (WaitForSingleObject(event, INFINITE) == WAIT\_OBJECT\_0) {

cout << "Молчим " << m << "-ый раз" << endl;

m++;

}

Sleep(500);

}

\_endthread();

}

//это поток, где люди выплескивают гнев

void Thread22(void\* pParams)

{

cout << "Пришли и начали кричать!" << endl;

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

for (int i = 0;i < 3;i++) {

cout << "Кричим!" << endl;

Sleep(500);

}

cout << "Ладно, помолчите..." << endl;

SetEvent(event);

Sleep(3000);

ResetEvent(event);

}

k++;

\_endthread();

}

int main3()

{

setlocale(LC\_ALL, "Russian");

HANDLE console = GetStdHandle(STD\_OUTPUT\_HANDLE);

HANDLE hMutex;

DWORD result;

int c;

do {

printf("1 - Критические секции\n2 - Мьютексы\n3 - События\n0 - Выход\n");

cin >> c;

switch (c)

{

case 1:

//3 потока, все они делают разные действия

//Если убрать крит секции, они будут делаться в разнобой

SetConsoleTextAttribute(console, 11);

printf("Критические секции\n");

SetConsoleTextAttribute(console, 15);

InitializeCriticalSection(&cs);

k = 0;

a = 0;

// массив элементов - из двух событий прочтения -

// по одному событию на файл

\_beginthread(Thread1, 0, NULL);

\_beginthread(Thread2, 0, NULL);

\_beginthread(Thread3, 0, NULL);

while (true) {

if (k == 3) break;

}

break;

case 2:

SetConsoleTextAttribute(console, 11);

printf("Мьютексы\n");

SetConsoleTextAttribute(console, 15);

hMutex = CreateMutex(NULL, FALSE, L"Mutex");

result = WaitForSingleObject(hMutex, 0);

if (result == WAIT\_OBJECT\_0)

{

cout << "Прыжок!" << endl;

Sleep(5500);

ReleaseMutex(hMutex);

}

else

{

cout << "Кто-то уже занял место для прыжка..." << endl;

}

CloseHandle(hMutex);

break;

case 3:

SetConsoleTextAttribute(console, 11);

printf("События\n");

SetConsoleTextAttribute(console, 15);

k = 0;

//создаем событие

event = CreateEvent(NULL, TRUE, FALSE, L"Event");

//создаем потоки

\_beginthread(Thread12, 0, NULL);

\_beginthread(Thread22, 0, NULL);

//закрываем событие

while (true) {

if (k == 1)

{

CloseHandle(event);

break;

}

}

break;

default:

break;

}

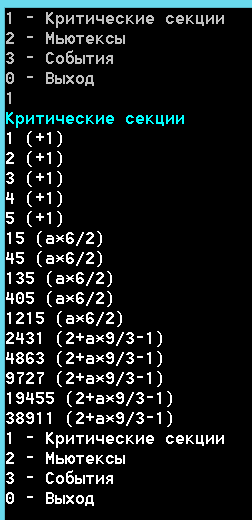
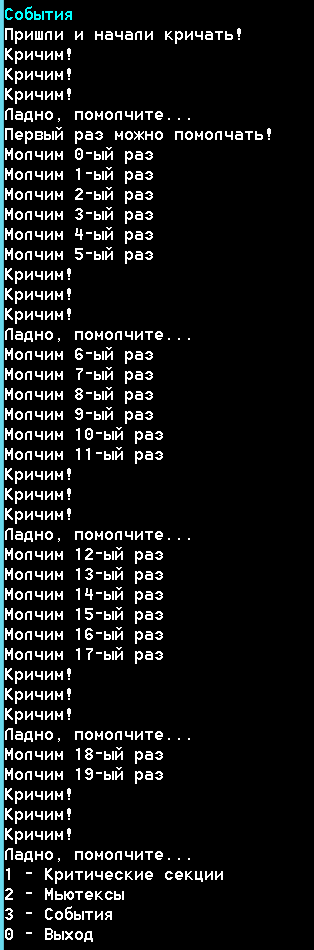
} while (c != 0);

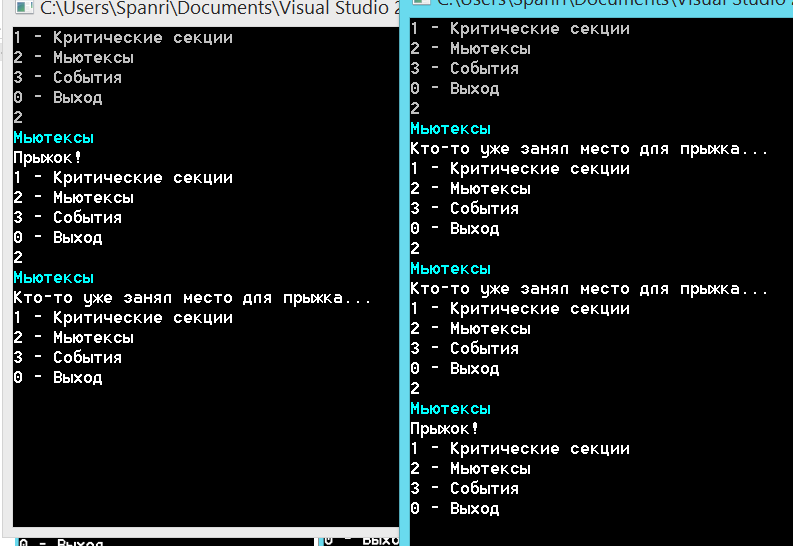
system("pause");

return 0;

}

**Пример работы:**





**Вывод:**

В результате проделанной работы научились синхронизировать потоки разными способами.