МИНИСТЕРСТВО ОБРАЗОВАНИЯ ОРЕНБУРГСКОЙ ОБЛАСТИ
Государственное автономное профессиональное образовательное учреждение
«ОРЕНБУРГСКИЙ КОЛЛЕДЖ ЭКОНОМИКИ И ИНФОРМАТИКИ»
(ГАПОУ ОКЭИ)

Лабораторная работа

ОКЭИ 09.02.07 7026 07 У

Тема: «*Методы синхронизации потоков*»

Выполнил: *Гадиляев Ислам Ильгамович*

Оренбург 2026

Содержание

1 Цель работы

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

2 Скриншот структуры готового проекта

На рисунке 1 представлен скриншот структуры готового проекта.

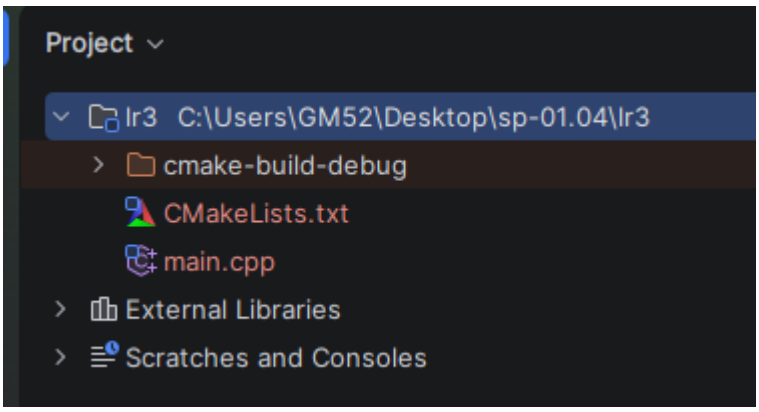

Рисунок 1 – Структура проекта

3 Листинг программы

Листинг первой программы представлен ниже.

```
#include <iostream>
#include <windows.h>
#include <stdio.h>

const int THREAD_COUNT = 5;
const int INCREMENTS = 1000;

long long counter = 0;
CRITICAL_SECTION cs;

DWORD WINAPI ThreadProc(LPVOID) {
    for (int i = 0; i < INCREMENTS; ++i) {
        EnterCriticalSection(&cs);
        counter++;
        LeaveCriticalSection(&cs);
    }
    return 0;
}

int main()
{
    SetConsoleOutputCP(CP_UTF8);

    InitializeCriticalSection(&cs);

    HANDLE threads[THREAD_COUNT];
```

```
    for (int i = 0; i < THREAD_COUNT; ++i) {
        threads[i] = CreateThread(NULL, 0, ThreadProc, NULL, 0, NULL);
    }

    WaitForMultipleObjects(THREAD_COUNT, threads, TRUE, INFINITE);
    for (int i = 0; i < THREAD_COUNT; ++i) CloseHandle(threads[i]);

    printf("Итоговое значение счетчика: %lld\n", counter);

    DeleteCriticalSection(&cs);

    return 0;
}
```

Листинг второй программы представлен ниже.

```
#include <iostream>
#include <windows.h>
#include <stdio.h>

const int THREAD_COUNT = 5;
const int ITERATIONS = 5;

HANDLE mutex;

DWORD WINAPI ThreadProc(LPVOID param) {
    int id = (int)(size_t)param;
    for (int i = 0; i < ITERATIONS; ++i) {
        WaitForSingleObject(mutex, INFINITE);
        printf("Поток %d: сообщение номер %d\n", id, i + 1);
        Sleep(100);  // имитация длительной работы
        ReleaseMutex(mutex);
    }
    return 0;
}

int main()
{
    SetConsoleOutputCP(CP_UTF8);

    mutex = CreateMutex(NULL, FALSE, NULL);

    HANDLE threads[THREAD_COUNT];
    for (int i = 0; i < THREAD_COUNT; ++i) {
        threads[i] = CreateThread(NULL, 0, ThreadProc, (LPVOID)(size_t)i, 0, NULL);
    }

    WaitForMultipleObjects(THREAD_COUNT, threads, TRUE, INFINITE);
    for (int i = 0; i < THREAD_COUNT; ++i) CloseHandle(threads[i]);

    CloseHandle(mutex);
    return 0;
}
```

Листинг третьей программы представлен ниже.

```
#include <iostream>
#include <windows.h>
#include <stdio.h>

const int THREAD_COUNT = 5;

HANDLE event;

DWORD WINAPI ThreadProc(LPVOID param) {
    int id = (int)(size_t)param;
    printf("Поток %d ждет сигнала...\n", id);
    WaitForSingleObject(event, INFINITE);
    printf("Поток %d получил сигнал и продолжает работу!\n", id);
    return 0;
```

```
}

int main()
{
    SetConsoleOutputCP(CP_UTF8);

    event = CreateEvent(NULL, TRUE, FALSE, NULL);

    HANDLE threads[THREAD_COUNT];
    for (int i = 0; i < THREAD_COUNT; ++i) {
        threads[i] = CreateThread(NULL, 0, ThreadProc, (LPVOID)(size_t)i, 0, NULL);
    }

    Sleep(3000);
    printf("Главный поток посылает сигнал...\n");
    SetEvent(event);

    WaitForMultipleObjects(THREAD_COUNT, threads, TRUE, INFINITE);
    for (int i = 0; i < THREAD_COUNT; ++i) CloseHandle(threads[i]);

    CloseHandle(event);
    return 0;
}
```

4 Результаты работы программы

Результат выполнения первой программы показан на рисунках 2.

Рисунок 2 – Результат работы первой программы

Таким образом, можно сказать, что несколько потоков одновременно увеличивают общий счётчик. Без синхронизации возникает состояние гонки (race condition), и итоговое значение будет меньше ожидаемого. С критической секцией счётчик защищён, и результат всегда правильный

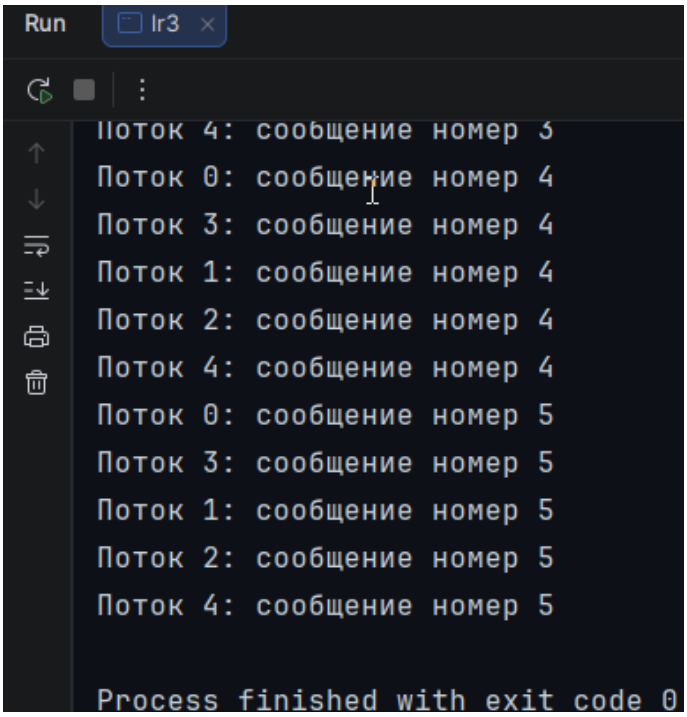

Рисунок 3 – Результат работы второй программы

С мьютексом вывод упорядочен, каждый поток печатает своё сообщение целиком. Это демонстрирует взаимоисключение при доступе к общему ресурсу (консоль). Для простоты используем анонимный мьютекс в одном процессе, но легко сделать именованный для нескольких процессов

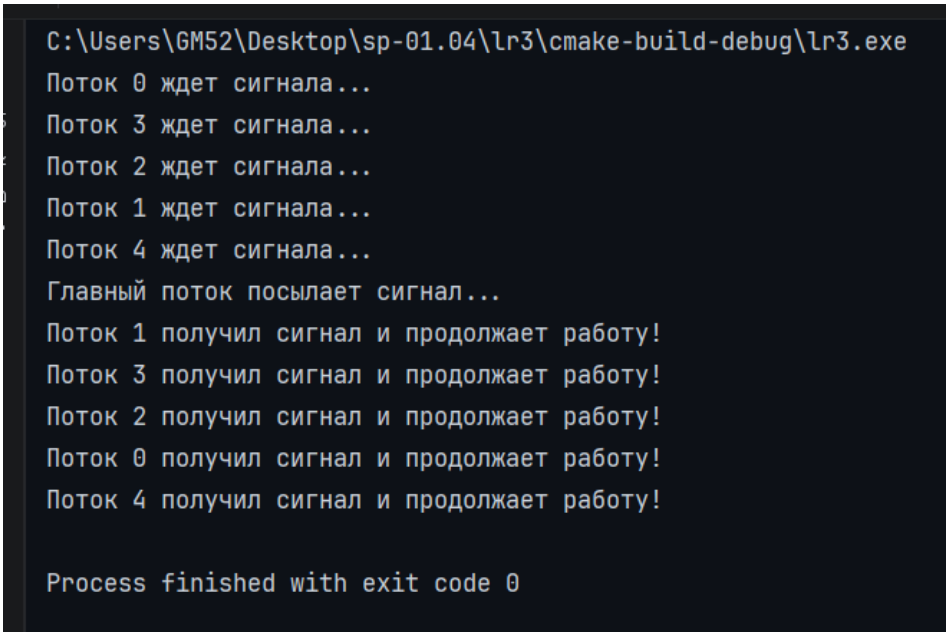

Рисунок 4 – Результат работы третьей программы

Главный поток ждёт 3 секунды (имитация подготовки), затем сигнализирует. 5 рабочих потоков ждут события и после получения печатают сообщение