

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

Институт цифровых интеллектуальных систем  
Кафедра компьютерных систем управления

Образовательная программа 15.03.04

«Автоматизация технологических процессов и производств»

Дисциплина «Программное обеспечение систем управления»

**Отчет**

**по практическим занятиям**

«Средства синхронизации на примере Windows NT»

Выполнил:

студент гр. АДБ-19-06 11.10.21 \_\_\_\_\_\_\_\_\_\_\_\_ Кулагин Г.В.

(дата) (подпись)

Принял:

Д.т.н., проф., заведующий кафедрой 11.10.21 \_\_\_\_\_\_\_\_ Мартинов Г.М.

(дата) (подпись)

Оглавление

[Критическая секция 3](#_Toc84426915)

[Семафор 6](#_Toc84426916)

[Мютекс 8](#_Toc84426917)

# Критическая секция

1. Написать программный код на языке C++ синхронизирующий 4-х потоков критической секцией.

#include <Windows.h>

#include <conio.h>

#include <ctime>

#include <iostream>

using namespace std;

int g\_array[100];

int g\_count = 0;

CRITICAL\_SECTION g\_hCriticalSection;

HANDLE hStdout;

int Lid = 0;

/// <summary>

/// Первый поток

/// </summary>

/// <param name="pv"></param>

/// <returns></returns>

static DWORD WINAPI Kulagin(void\* pv)

{

int i = 0;

int sync = 0;

COORD pos;

pos.X = 1;

while (i < 50)

{

::EnterCriticalSection(&g\_hCriticalSection);

srand(time(NULL));

g\_array[g\_count] = rand() % 256; //Гененрируем рандомное число

pos.Y = i;

SetConsoleCursorPosition(hStdout, pos);

if (!sync)

{

cout << "Kulagin" << endl;// Выводим фамилию в название столбца

sync++;

}

cout << g\_array[g\_count++];

Sleep(40); //Остановка потока на 40 мс

i++;

::LeaveCriticalSection(&g\_hCriticalSection);

}

if (!Lid)

Lid = 1;

return 0;

}

/// <summary>

/// Второй поток

/// </summary>

/// <param name="pv"></param>

/// <returns></returns>

static DWORD WINAPI Grigoriy(void\* pv)

{

int i = 0;

int sync = 0;

COORD pos;

pos.X = 18;

while (i < 50)

{

::EnterCriticalSection(&g\_hCriticalSection);

srand(time(NULL));

g\_array[g\_count] = rand() % 1000;

pos.Y = i;

SetConsoleCursorPosition(hStdout, pos);

if (!sync)

{

cout << "Grigoriy" << endl; // Выводим имя в название второго столбца

sync++;

}

cout << g\_array[g\_count++];

Sleep(40);

i++;

::LeaveCriticalSection(&g\_hCriticalSection);

}

if (!Lid)

Lid = 2;

return 0;

}

/// <summary>

/// Третий поток

/// </summary>

/// <param name="pv"></param>

/// <returns></returns>

static DWORD WINAPI Vladimirovich(void\* pv)

{

int i = 0;

int sync = 0;

COORD pos;

pos.X = 36;

while (i < 50)

{

::EnterCriticalSection(&g\_hCriticalSection);

srand(time(NULL));

g\_array[g\_count] = rand() % 256;

pos.Y = i;

SetConsoleCursorPosition(hStdout, pos);

if (!sync)

{

cout << "Vladimirovich" << endl; //Выводим отчество в название третьего столбца

sync++;

}

cout << g\_array[g\_count++];

Sleep(40);

i++;

::LeaveCriticalSection(&g\_hCriticalSection);

}

if (!Lid)

Lid = 3;

return 0;

}

/// <summary>

/// Четвертый поток

/// </summary>

/// <param name="pv"></param>

/// <returns></returns>

static DWORD WINAPI Gruppa(void\* pv)

{

int i = 0;

int sync = 0;

COORD pos;

pos.X = 54;

while (i < 50)

{

::EnterCriticalSection(&g\_hCriticalSection);

srand(time(NULL));

g\_array[g\_count] = rand() % 256;

pos.Y = i;

SetConsoleCursorPosition(hStdout, pos);

if (!sync)

{

cout << "Gruppa" << endl;//Выводим группу в название четвертого столбца

sync++;

}

cout << g\_array[g\_count++];

Sleep(40);

i++;

::LeaveCriticalSection(&g\_hCriticalSection);

}

if (!Lid)

Lid = 4;

return 0;

}

int main()

{

DWORD dw;

HANDLE hThreads[4];

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

::InitializeCriticalSection(&g\_hCriticalSection);

//Создание потоков

hThreads[0] = ::CreateThread(NULL, 0, Kulagin, NULL, 0, &dw);

hThreads[1] = ::CreateThread(NULL, 0, Grigoriy, NULL, 0, &dw);

hThreads[2] = ::CreateThread(NULL, 0, Vladimirovich, NULL, 0, &dw);

hThreads[3] = ::CreateThread(NULL, 0, Gruppa, NULL, 0, &dw);

::WaitForMultipleObjects(4, hThreads, TRUE, INFINITE);

//Закрытие потоков

::CloseHandle(hThreads[0]);

::CloseHandle(hThreads[1]);

::CloseHandle(hThreads[2]);

::CloseHandle(hThreads[3]);

::DeleteCriticalSection(&g\_hCriticalSection);

switch (Lid)

{

case 1: cout << "\n 1 thread finished first!\n"; break;

case 2: cout << "\n 2 thread finished first!\n"; break;

case 3: cout << "\n 3 thread finished first!\n"; break;

case 4: cout << "\n 4 thread finished first!\n"; break;

default: break;

}

cout << endl << "Press Any Key...";

\_getch();

return 0;

}

# Семафор

1. Написать программный код на языке C++ синхронизирующий 4-х потоков семафором, начальное значение счетчика = 2.

#include <Windows.h>

#include <conio.h>

#include <ctime>

#include <iostream>

using namespace std;

int g\_array[200];

int g\_count = 0;

HANDLE g\_hSemaphor = NULL;

HANDLE hStdout;

int Lid = 0;

/// <summary>

///Первый поток

/// </summary>

/// <param name="pv"></param>

/// <returns></returns>

static DWORD WINAPI Kulagin(void\* pv)

{

int i = 0;

int sync = 0;

while (i < 50)

{

WaitForSingleObject(&g\_hSemaphor, INFINITE);

srand(time(NULL));

g\_array[g\_count] = rand() % 256;

COORD pos;

pos.X = 1;

pos.Y = i;

SetConsoleCursorPosition(hStdout, pos);

if (!sync)

{

cout << "Kulagin" << endl; // Выводим фамилию в название столбца

sync++;

}

cout << g\_array[g\_count++];

Sleep(100);

i++;

ReleaseSemaphore(g\_hSemaphor, 1, NULL);

}

if (!Lid)

Lid = 1;

return 0;

}

/// <summary>

/// Второй поток

/// </summary>

/// <param name="pv"></param>

/// <returns></returns>

static DWORD WINAPI Grigoriy(void\* pv)

{

int i = 0;

int sync = 0;

while (i < 50)

{

WaitForSingleObject(&g\_hSemaphor, INFINITE);

srand(time(NULL));

g\_array[g\_count] = rand() % 1000;

COORD pos;

pos.X = 18;

pos.Y = i;

SetConsoleCursorPosition(hStdout, pos);

if (!sync)

{

cout << "Grigoriy" << endl; // Выводим имя в название второго столбца

sync++;

}

cout << g\_array[g\_count++];

Sleep(100);

i++;

ReleaseSemaphore(g\_hSemaphor, 1, NULL);

}

if (!Lid)

Lid = 2;

return 0;

}

/// <summary>

/// Третий поток

/// </summary>

/// <param name="pv"></param>

/// <returns></returns>

static DWORD WINAPI Vladimirovich(void\* pv)

{

int i = 0;

int sync = 0;

while (i < 50)

{

WaitForSingleObject(&g\_hSemaphor, INFINITE);

srand(time(NULL));

g\_array[g\_count] = rand() % 256;

COORD pos;

pos.X = 36;

pos.Y = i;

SetConsoleCursorPosition(hStdout, pos);

if (!sync)

{

cout << "Vladimirovich" << endl; //Выводим отчество в название третьего столбца

sync++;

}

cout << g\_array[g\_count++];

Sleep(100);

i++;

ReleaseSemaphore(g\_hSemaphor, 1, NULL);

}

if (!Lid)

Lid = 3;

return 0;

}

/// <summary>

/// Четвертый поток

/// </summary>

/// <param name="pv"></param>

/// <returns></returns>

static DWORD WINAPI Gruppa(void\* pv)

{

int i = 0;

int sync = 0;

while (i < 50)

{

WaitForSingleObject(&g\_hSemaphor, INFINITE);

srand(time(NULL));

g\_array[g\_count] = rand() % 256;

COORD pos;

pos.X = 54;

pos.Y = i;

SetConsoleCursorPosition(hStdout, pos);

if (!sync)

{

cout << "Gruppa" << endl; //Выводим группу в название четвертого столбца

sync++;

}

cout << g\_array[g\_count++];

Sleep(100);

i++;

ReleaseSemaphore(g\_hSemaphor, 1, NULL);

}

if (!Lid)

Lid = 4;

return 0;

}

int main()

{

DWORD dw;

HANDLE hThreads[4];

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

//Создание семафора

g\_hSemaphor = CreateSemaphore(NULL, 2, 1, NULL);

//Создание 4-х потоков

hThreads[0] = ::CreateThread(NULL, 0, Kulagin, NULL, 0, &dw);

hThreads[1] = ::CreateThread(NULL, 0, Grigoriy, NULL, 0, &dw);

hThreads[2] = ::CreateThread(NULL, 0, Vladimirovich, NULL, 0, &dw);

hThreads[3] = ::CreateThread(NULL, 0, Gruppa, NULL, 0, &dw);

::WaitForMultipleObjects(4, hThreads, TRUE, INFINITE);

::CloseHandle(hThreads[0]);

::CloseHandle(hThreads[1]);

::CloseHandle(hThreads[2]);

::CloseHandle(hThreads[3]);

::CloseHandle(g\_hSemaphor);

//Выводим какой поток финишировал первым

switch (Lid)

{

case 1: cout << "\n 1 thread finished first!\n"; break;

case 2: cout << "\n 2 thread finished first!\n"; break;

case 3: cout << "\n 3 thread finished first!\n"; break;

case 4: cout << "\n 4 thread finished first!\n"; break;

default: break;

}

cout << endl << "Press Any Key...";

\_getch();

return 0;

}

# Мютекс

1. Написать программный код на языке C++ синхронизирующий 4-х потоков мютексом.

#include <Windows.h>

#include <conio.h>

#include <ctime>

#include <iostream>

using namespace std;

int g\_array[100];

int g\_count = 0;

CRITICAL\_SECTION g\_hCriticalSection;

HANDLE hStdout;

int Lid = 0;

static DWORD WINAPI Kulagin(void\* pv)

{

int i = 0;

int sync = 0;

COORD pos;

pos.X = 1;

while (i < 50)

{

::EnterCriticalSection(&g\_hCriticalSection);

srand(time(NULL));

g\_array[g\_count] = rand() % 256;

pos.Y = i;

SetConsoleCursorPosition(hStdout, pos);

if (!sync)

{

cout << "Kulagin" << endl;

sync++;

}

cout << g\_array[g\_count++];

Sleep(40);

i++;

::LeaveCriticalSection(&g\_hCriticalSection);

}

if (!Lid)

Lid = 1;

return 0;

}

static DWORD WINAPI Grigoriy(void\* pv)

{

int i = 0;

int sync = 0;

COORD pos;

pos.X = 18;

while (i < 50)

{

::EnterCriticalSection(&g\_hCriticalSection);

srand(time(NULL));

g\_array[g\_count] = rand() % 1000;

pos.Y = i;

SetConsoleCursorPosition(hStdout, pos);

if (!sync)

{

cout << "Grigoriy" << endl;

sync++;

}

cout << g\_array[g\_count++];

Sleep(40);

i++;

::LeaveCriticalSection(&g\_hCriticalSection);

}

if (!Lid)

Lid = 2;

return 0;

}

static DWORD WINAPI Vladimirovich(void\* pv)

{

int i = 0;

int sync = 0;

COORD pos;

pos.X = 36;

while (i < 50)

{

::EnterCriticalSection(&g\_hCriticalSection);

srand(time(NULL));

g\_array[g\_count] = rand() % 256;

pos.Y = i;

SetConsoleCursorPosition(hStdout, pos);

if (!sync)

{

cout << "Vladimirovich" << endl;

sync++;

}

cout << g\_array[g\_count++];

Sleep(40);

i++;

::LeaveCriticalSection(&g\_hCriticalSection);

}

if (!Lid)

Lid = 3;

return 0;

}

static DWORD WINAPI Gruppa(void\* pv)

{

int i = 0;

int sync = 0;

COORD pos;

pos.X = 54;

while (i < 50)

{

::EnterCriticalSection(&g\_hCriticalSection);

srand(time(NULL));

g\_array[g\_count] = rand() % 256;

pos.Y = i;

SetConsoleCursorPosition(hStdout, pos);

if (!sync)

{

cout << "Gruppa" << endl;

sync++;

}

cout << g\_array[g\_count++];

Sleep(40);

i++;

::LeaveCriticalSection(&g\_hCriticalSection);

}

if (!Lid)

Lid = 4;

return 0;

}

int main()

{

DWORD dw;

HANDLE hThreads[4];

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

::InitializeCriticalSection(&g\_hCriticalSection);

hThreads[0] = ::CreateThread(NULL, 0, Kulagin, NULL, 0, &dw);

hThreads[1] = ::CreateThread(NULL, 0, Grigoriy, NULL, 0, &dw);

hThreads[2] = ::CreateThread(NULL, 0, Vladimirovich, NULL, 0, &dw);

hThreads[3] = ::CreateThread(NULL, 0, Gruppa, NULL, 0, &dw);

::WaitForMultipleObjects(4, hThreads, TRUE, INFINITE);

::CloseHandle(hThreads[0]);

::CloseHandle(hThreads[1]);

::CloseHandle(hThreads[2]);

::CloseHandle(hThreads[3]);

::DeleteCriticalSection(&g\_hCriticalSection);

switch (Lid)

{

case 1: cout << "\n 1 thread finished first!\n"; break;

case 2: cout << "\n 2 thread finished first!\n"; break;

case 3: cout << "\n 3 thread finished first!\n"; break;

case 4: cout << "\n 4 thread finished first!\n"; break;

default: break;

}

cout << endl << "Press Any Key...";

\_getch();

return 0;

}

**Вывод:** В ходе практических занятиях я ознакомился с понятиями критической секции, семафоры и мютекса, а также научился применять их в разработке.

**Список литературы:**