Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

[Кафедра информационных](https://www.belstu.by/fakultety/fit/vm) систем и технологий

Специальность 1-40 01 01 Программное обеспечение информационной безопасности мобильных систем

**Отчёт по лабораторной работе №6**

Дисциплина: Операционные системы

Тема: Синхронизация

Выполнила:

студентка 3 курса 5 группы

Городилина Анастасия Сергеевна

Минск 2024

**Задание 01**

Напишите ассемблерный код с применением команд BTS или BTR, демонстрирующий реализацию механизма синхронизации двух потоков одного процесса и поясните его работу.

#include <windows.h>

#include <iostream>cd

using namespace std;

int bit;

DWORD WINAPI thread(char lpParam) {

\_asm

{

getsem:

lock bts bit, 0;

jc getsem

}

for (int i = 0; i < 10; ++i) {

cout << lpParam << " TID: " << GetCurrentThreadId() << endl;

Sleep(500);

}

\_asm lock btr bit, 0

return 0;

}

int main() {

HANDLE hThread1, hThread2;

hThread1 = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)thread, (char\*)'1', 0, NULL);

hThread2 = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)thread, (char\*)'2', 0, NULL);

WaitForSingleObject(hThread1, INFINITE);

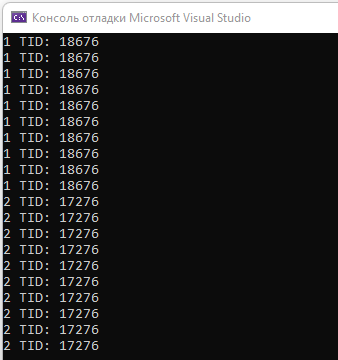
WaitForSingleObject(hThread2, INFINITE);

CloseHandle(hThread1);

CloseHandle(hThread2);

return 0;

}



**Задание 02**

Приложение **OS06\_02** синхронизирует выполнение потоков **main**, **A** и **B** с помощью механизма **critical section.**

Синхронизация должна обеспечивать поочередное выполнение итераций цикла с 30 по 60 в каждом потоке.

#include <windows.h>

#include <iostream>

using namespace std;

CRITICAL\_SECTION cs;

DWORD WINAPI thread(char\* lpParam) {

for (int i = 1; i <= 90; ++i) {

if (i == 30) {

EnterCriticalSection(&cs);

}

else if (i == 60) {

LeaveCriticalSection(&cs);

}

cout << lpParam << ": " << i << endl;

Sleep(100);

}

return 0;

}

int main() {

HANDLE hThread1, hThread2;

hThread1 = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)thread, (char\*)"A", 0, NULL);

hThread2 = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)thread, (char\*)"B", 0, NULL);

InitializeCriticalSection(&cs);

thread((char\*)"main");

WaitForSingleObject(hThread1, INFINITE);

WaitForSingleObject(hThread2, INFINITE);

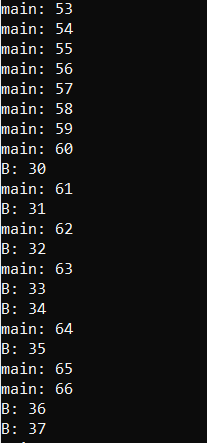
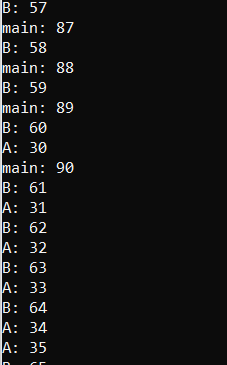
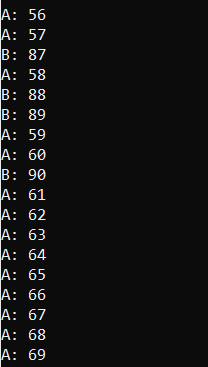
CloseHandle(hThread1);

CloseHandle(hThread2);

DeleteCriticalSection(&cs);

return 0;

}

**Задание 03**

Приложение **OS06\_03** синхронизирует выполнение процессов **OS06\_03**, **OS06\_03A** и **OS06\_03B** с помощью механизма **mutex.**

Синхронизация должна обеспечивать поочередное выполнение итераций цикла с 30 по 60.

OS06-03

#include <iostream>

#include <mutex>

#include "Windows.h"

using namespace std;

int main()

{

const wchar\_t\* path1 = L"C:\\AllMine\\Labs\\ОС\\lab6\\lab6\\Debug\\OS06\_03A.exe";

const wchar\_t\* path2 = L"C:\\AllMine\\Labs\\ОС\\lab6\\lab6\\Debug\\OS06\_03B.exe"; HANDLE mutex;

mutex = CreateMutex(NULL, false, L"OS06\_03");

STARTUPINFO si1, si2;

PROCESS\_INFORMATION pi1, pi2;

ZeroMemory(&si1, sizeof(STARTUPINFO));

ZeroMemory(&si2, sizeof(STARTUPINFO));

si1.cb = sizeof(STARTUPINFO);

si2.cb = sizeof(STARTUPINFO);

if (CreateProcessW(path1, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si1, &pi1))

cout << "Process OS06\_03A created\n";

else

cout << "Error(1)\n";

if (CreateProcessW(path2, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si2, &pi2))

cout << "Process OS06\_03B created\n";

else

cout << "Error(2)\n";

for (int i = 1; i <= 90; i++) {

if (i == 30) {

WaitForSingleObject(mutex, INFINITE);

}

else if (i == 60) {

ReleaseMutex(mutex);

}

cout << "Main: " << i << endl;

Sleep(100);

}

CloseHandle(mutex);

return 0;

}

}

OS06\_03A

#include <iostream>

#include <mutex>

#include "Windows.h"

using namespace std;

int main()

{

HANDLE mutex;

mutex = CreateMutex(NULL, false, L"OS06\_03");

for (int i = 1; i <= 90; i++) {

if (i == 30) {

WaitForSingleObject(mutex, INFINITE);

}

else if (i == 60) {

ReleaseMutex(mutex);

}

cout << "OS06\_03A: " << i << endl;

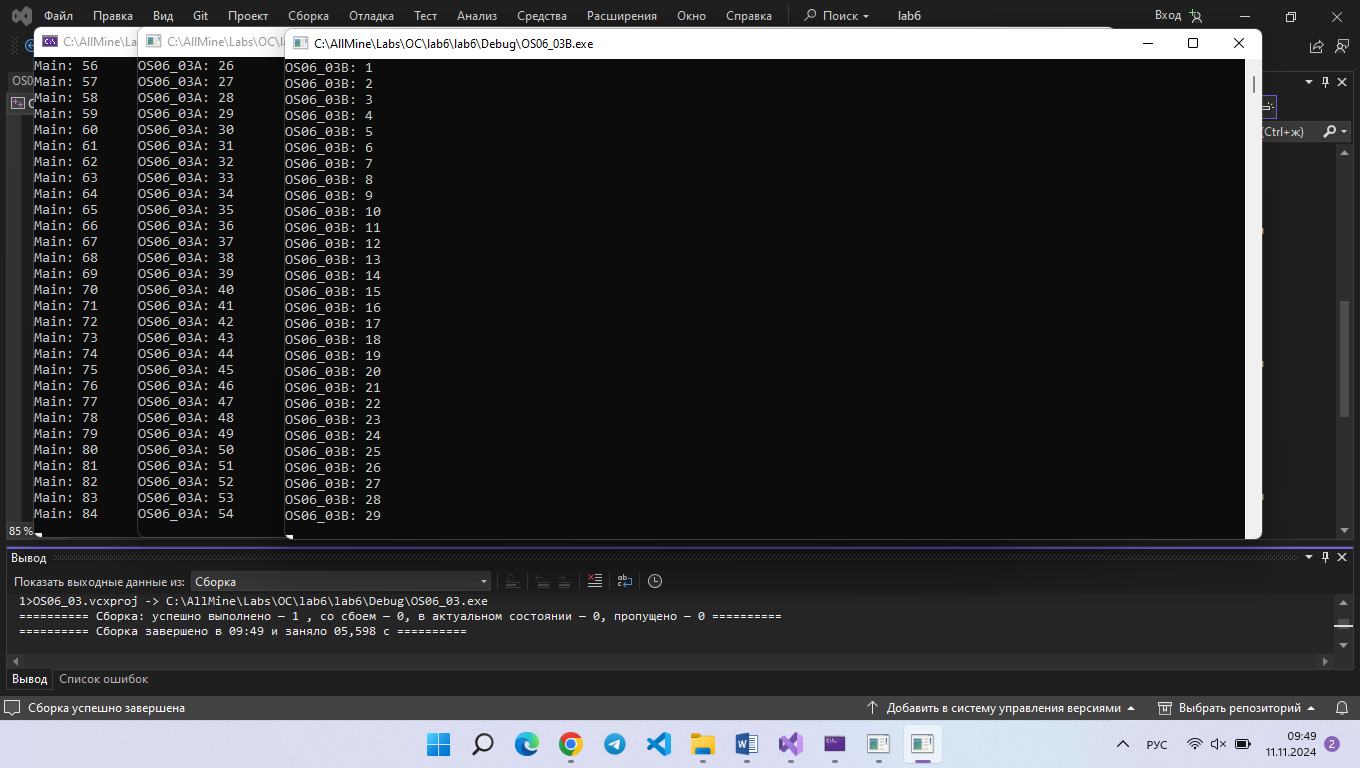
Sleep(100);

}

CloseHandle(mutex);

return 0;

}



**Задание 04**

Приложение **OS06\_04** синхронизирует выполнение процессов **OS06\_04**, **OS06\_04A** и **OS06\_04B** с помощью механизма **semaphore.**

Синхронизация должна обеспечивать поочередное выполнение итераций цикла с 30 по 60 одного (любого) процесса и двух других процессов. Другими словами, итерации с 30 по 60 должны одновременно выполняться только в двух из трех процессов.

OS06\_04

#include <iostream>

#include <mutex>

#include "Windows.h"

using namespace std;

int main()

{

const wchar\_t\* path1 = L"C:\\AllMine\\Labs\\ОС\\lab6\\lab6\\Debug\\OS06\_04A.exe";

const wchar\_t\* path2 = L"C:\\AllMine\\Labs\\ОС\\lab6\\lab6\\Debug\\OS06\_04B.exe";

HANDLE semaphore;

semaphore = CreateSemaphore(NULL,2 , 2, L"OS06\_04");

STARTUPINFO si1, si2;

PROCESS\_INFORMATION pi1, pi2;

ZeroMemory(&si1, sizeof(STARTUPINFO));

ZeroMemory(&si2, sizeof(STARTUPINFO));

si1.cb = sizeof(STARTUPINFO);

si2.cb = sizeof(STARTUPINFO);

if (CreateProcessW(path1, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si1, &pi1))

cout << "Process OS06\_04A created\n";

else

cout << "Error(1)\n";

if (CreateProcessW(path2, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si2, &pi2))

cout << "Process OS06\_04B created\n";

else

cout << "Error(2)\n";

for (int i = 1; i <= 90; i++) {

cout << "Main: " << i << endl;

if (i == 30) {

WaitForSingleObject(semaphore, INFINITE);

}

else if (i == 60) {

ReleaseSemaphore(semaphore, 1, NULL);

}

Sleep(100);

}

CloseHandle(semaphore);

return 0;

}

OS06\_04A

#include <iostream>

#include <mutex>

#include "Windows.h"

using namespace std;

int main()

{

HANDLE semaphore;

semaphore = CreateSemaphore(NULL, 2, 2, L"OS06\_04");

for (int i = 1; i <= 90; i++) {

cout << "OS06\_04A: " << i << endl;

if (i == 30) {

WaitForSingleObject(semaphore, INFINITE);

}

else if (i == 60) {

ReleaseSemaphore(semaphore, 1, NULL);

}

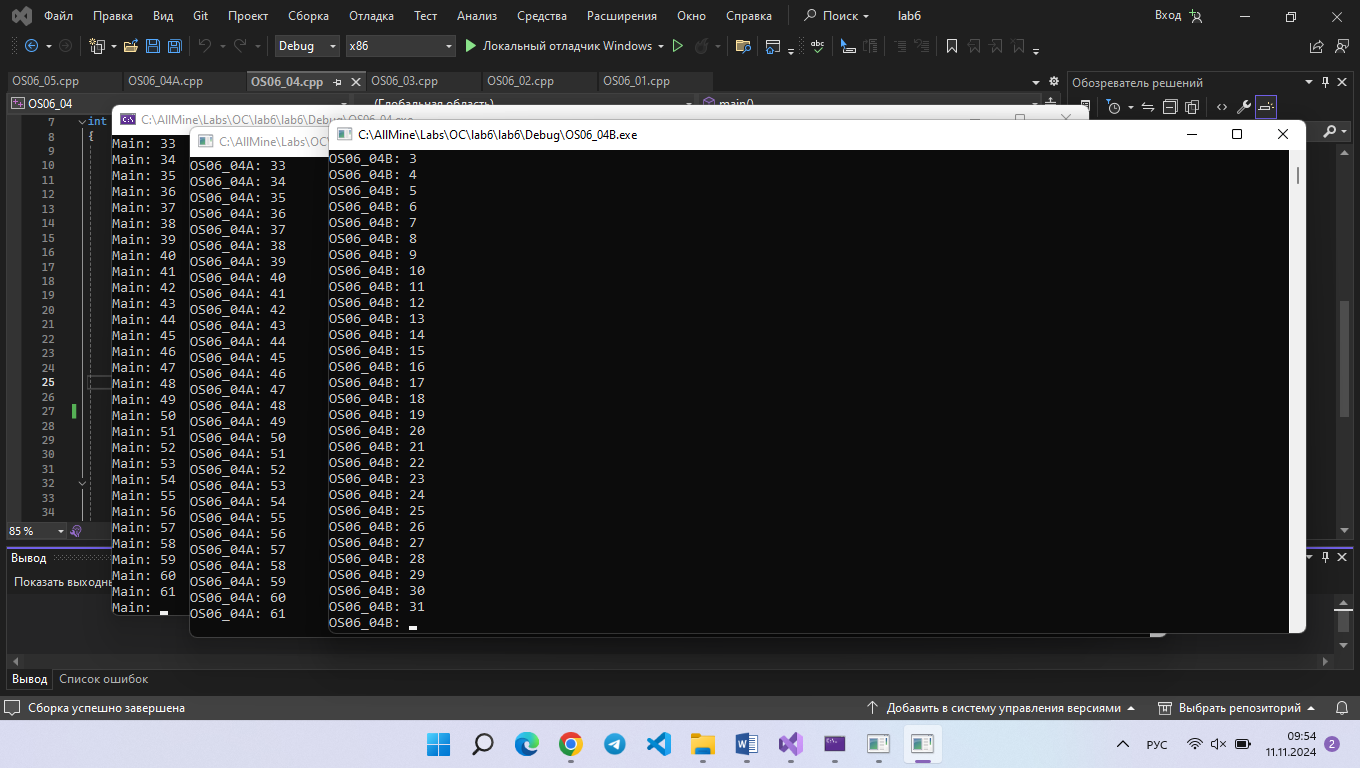
Sleep(100);

}

CloseHandle(semaphore);

return 0;

}



**Задание 05**

Приложение **OS06\_05** синхронизирует выполнение процессов **OS06\_05**, **OS06\_05A** и **OS06\_05B** с помощью механизма **event**.

Синхронизация должна обеспечивать выполнение приложения в два этапа: выполнение итераций с 1 по 15 процесса **OS06\_05**;

одновременное выполнение всех трех процессов: **OS06\_05** – продолжает выполнение итераций; процессы **OS06\_05A** и **OS06\_05B** выполняются начиная с первой итерации.

OS06\_05

#include <iostream>

#include <mutex>

#include "Windows.h"

using namespace std;

int main()

{

const wchar\_t\* path1 = L"C:\\AllMine\\Labs\\ОС\\lab6\\lab6\\Debug\\OS06\_05A.exe";

const wchar\_t\* path2 = L"C:\\AllMine\\Labs\\ОС\\lab6\\lab6\\Debug\\OS06\_05B.exe";

HANDLE event;

event = CreateEvent(NULL, FALSE, FALSE, L"OS06\_05");

STARTUPINFO si1, si2;

PROCESS\_INFORMATION pi1, pi2;

ZeroMemory(&si1, sizeof(STARTUPINFO));

ZeroMemory(&si2, sizeof(STARTUPINFO));

si1.cb = sizeof(STARTUPINFO);

si2.cb = sizeof(STARTUPINFO);

if (CreateProcessW(path1, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si1, &pi1))

cout << "Process OS06\_05A created\n";

else

cout << "Error(1)\n";

if (CreateProcessW(path2, NULL, NULL, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si2, &pi2))

cout << "Process OS06\_05B created\n";

else

cout << "Error(2)\n";

for (int i = 1; i <= 90; i++) {

cout << "Main: " << i << endl;

if (i == 15) {

SetEvent(event);

}

Sleep(1000);

}

CloseHandle(event);

return 0;

}

OS06\_05A

#include <iostream>

#include <mutex>

#include "Windows.h"

using namespace std;

int main()

{

HANDLE event;

event = CreateEvent(NULL, FALSE, FALSE, L"OS06\_05");

for (int i = 1; i <= 90; i++) {

WaitForSingleObject(event, INFINITE);

SetEvent(event);

cout << "OS06\_05A: " << i << endl;

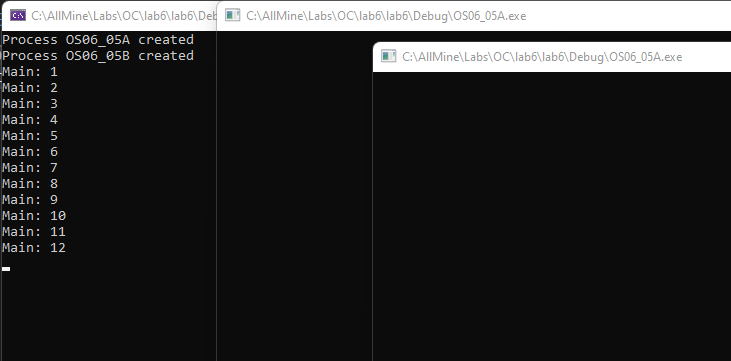
Sleep(100);

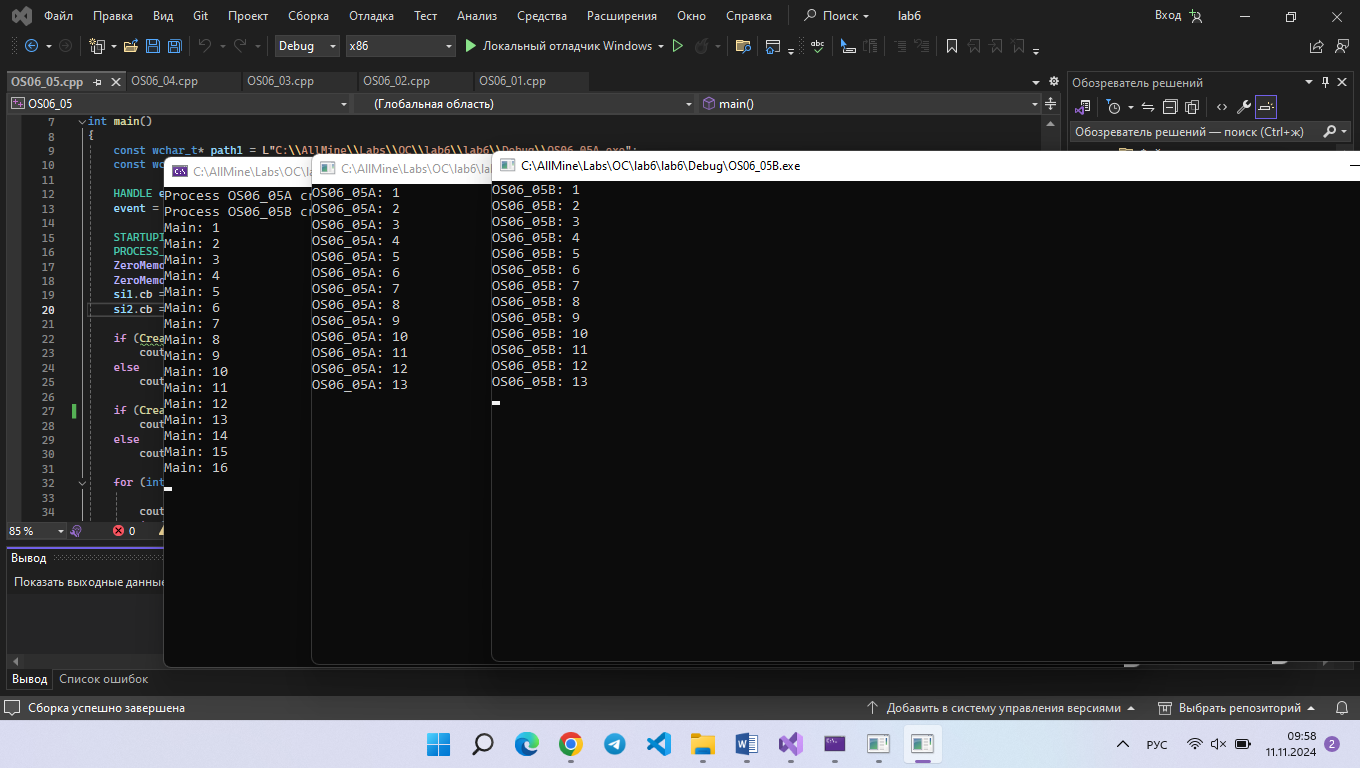
}

CloseHandle(event);

return 0;

}





**Задание 06**

Приложение **OS06\_06** синхронизирует выполнение потоков **main**, **A** и **B** с помощью механизма **mutex.**

Синхронизация должна обеспечивать поочередное выполнение итераций цикла с 30 по 60 в каждом потоке.

#include <stdio.h>

#include <time.h>

#include <unistd.h>

#include <pthread.h>

pthread\_mutex\_t mutex;

void\* thread(void\* name) {

for (int i = 1; i <= 90; ++i) {

if (i == 30) {

pthread\_mutex\_lock(&mutex);

} else if (i == 60) {

pthread\_mutex\_unlock(&mutex);

}

printf( "%s: %d \n", (char\*)name, i);

usleep(100000);

}

pthread\_exit("Child Thread");

}

int main() {

pthread\_t thread1, thread2;

pthread\_mutex\_init(&mutex, NULL);

pthread\_create(&thread1, NULL, thread, (void\*)"A");

pthread\_create(&thread2, NULL, thread, (void\*)"B");

thread((void\*)"main");

pthread\_join(thread1, NULL);

pthread\_join(thread2, NULL);

pthread\_mutex\_destroy(&mutex);

return 0;

}

