Белорусский государственный университет информатики и радиоэлектроники Факультет информационных технологий и управления

ОТЧЁТ

по лабораторной работе №4

**«Работа с потоками. Синхронизация потоков»**

Выполнил: Принял(а):

студент

группы № 980161 Герман Ю.О.

Алейчик И.Д.

Минск 2020

**Лабораторная работа №4**

1. **Изменение приоритета потока**.

В качестве программы, демонстрирующей изменение приоритета потока, рассмотреть следующую

DWORD WINAPI myThread1(void\* lpParameter)

{

int\* counterp = new int();

counterp = (int\*)lpParameter;

int counter = \*counterp;

while (counter < 20)

{

Sleep(500);

counter++;

printf("\n Counter= %d", counter);

}

return 0;

}

int main1()

{

int z = 5;

unsigned int myCounter = 0;

DWORD myThreadID;

HANDLE myHandle = CreateThread(0, 0, myThread1, (void\*)&z, 0, &myThreadID);

int CurPr = GetPriorityClass(GetCurrentProcess());

int iPriority1 = GetThreadPriority(myHandle);// + CurPr;

printf("\n Priority is: %d", iPriority1);

SetThreadPriority(myHandle, THREAD\_PRIORITY\_ABOVE\_NORMAL);

iPriority1 = GetThreadPriority(myHandle);// + CurPr;

printf("\n After Change Priority is: %d", iPriority1);

SetThreadPriority(myHandle, THREAD\_PRIORITY\_HIGHEST);

iPriority1 = GetThreadPriority(myHandle);// + CurPr;

printf("\nThe Highest Priority is: %d", iPriority1);

SetThreadPriority(myHandle, THREAD\_PRIORITY\_LOWEST);

iPriority1 = GetThreadPriority(myHandle);// + CurPr;

printf("\nThe Lowest Priority is: %d", iPriority1);

Sleep(5000);

printf("\n Main Process sagt Das ist Alles");

TerminateThread(myHandle, 0);

getchar();

return 0;

}



1. Запуск потока на запись в файл.

DWORD WINAPI myThread2(void\* lpParameter)

{

TCHAR Buffer[256] = TEXT("HELLO!");

HANDLE hOut;

DWORD dwBytes;

FILE\* fp;

TCHAR stdPath[30] = TEXT("C:\\Users\\ilyaa\\data.txt");

hOut = CreateFile(stdPath, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hOut == INVALID\_HANDLE\_VALUE)

{

printf("ERROR WRITING FILE");

return 2;

}

else

{

WriteFile(hOut, Buffer, 256, &dwBytes, NULL);

printf("Thread Written");

CloseHandle(hOut);

}

return 0;

}

int main2()

{

int z = 0;

DWORD myThreadID;

HANDLE myHandle = CreateThread(0, 0, myThread2, (void\*)&z, CREATE\_SUSPENDED, &myThreadID);

BOOL b = SetThreadPriorityBoost(myHandle, false);

if (b)

{

SetThreadPriority(myHandle, THREAD\_PRIORITY\_HIGHEST);

}

ResumeThread(myHandle);

Sleep(2000);

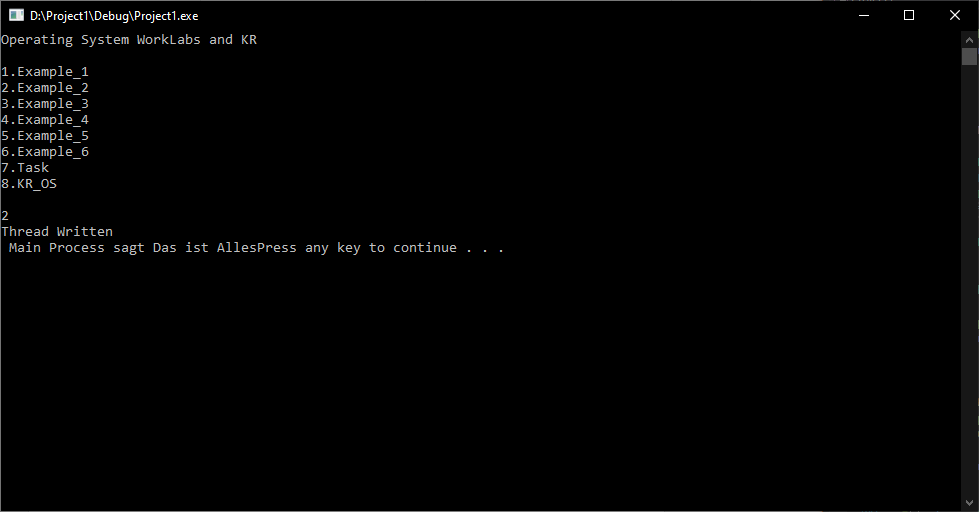
printf("\n Main Process sagt Das ist Alles");

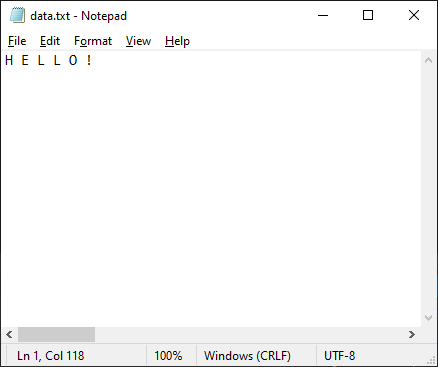
TerminateThread(myHandle, 0);

getchar();

return 0;

}





1. Теперь передадим записываемую строку в качестве параметра в поток:

DWORD WINAPI myThread3(void\* lpPar)

{

TCHAR\* b = new TCHAR();

b = (TCHAR\*)lpPar;

HANDLE hOut;

DWORD dwBytes;

FILE\* fp;

TCHAR stdPath[30] = TEXT("c:\\Users\\ilyaa\\data.txt");

hOut = CreateFile(stdPath, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hOut == INVALID\_HANDLE\_VALUE)

{

printf("ERROR WRITING FILE");

return 2;

}

else

{

WriteFile(hOut, b, 256, &dwBytes, NULL);

printf("Thread Written");

CloseHandle(hOut);

}

return 0;

}

int main3()

{

int z = 0;

DWORD myThreadID;

TCHAR Buff[256] = TEXT("HELLO! Students");

HANDLE myHandle = CreateThread(0, 0, myThread3, (void\*)Buff, CREATE\_SUSPENDED, &myThreadID);

BOOL b = SetThreadPriorityBoost(myHandle, false);

if (b)

{

SetThreadPriority(myHandle, THREAD\_PRIORITY\_HIGHEST);

}

ResumeThread(myHandle);

Sleep(2000);

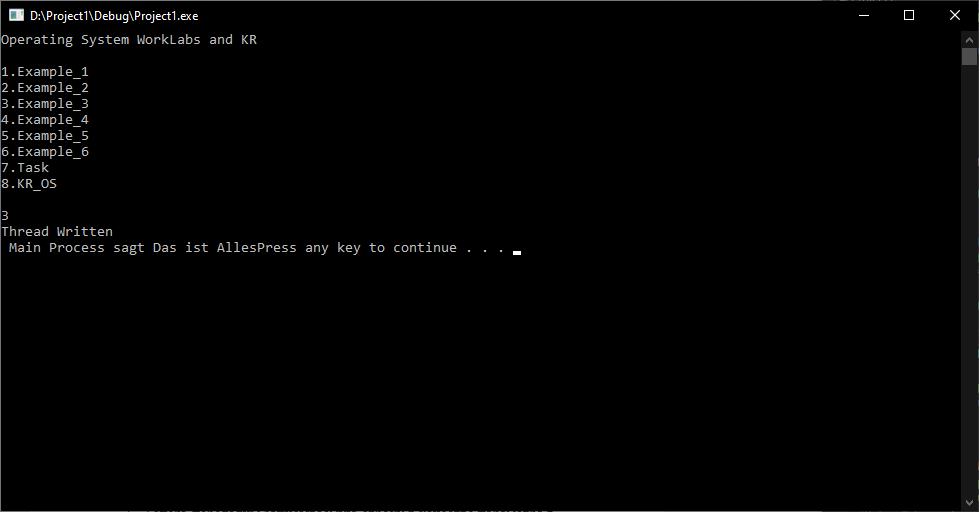
printf("\n Main Process sagt Das ist Alles");

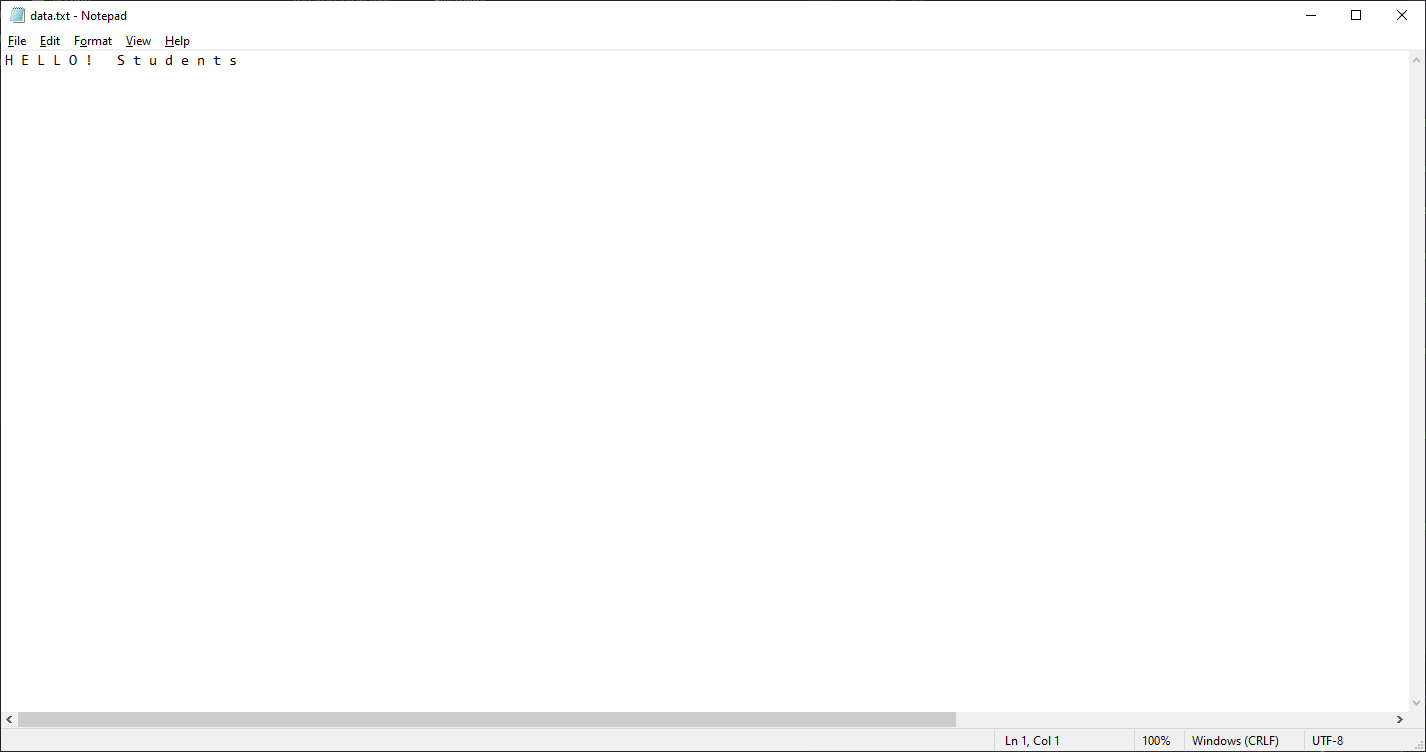
TerminateThread(myHandle, 0);

getchar();

return 0;

}





1. Теперь в главной программе дождемся завершения работы потока:

DWORD WINAPI myThread4(void\* lpPar)

{

TCHAR\* b = new TCHAR();

b = (TCHAR\*)lpPar;

HANDLE hOut;

DWORD dwBytes;

FILE\* fp;

TCHAR stdPath[30] = TEXT("C:\\Users\\ilyaa\\data.txt");

hOut = CreateFile(stdPath, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hOut == INVALID\_HANDLE\_VALUE)

{

printf("ERROR WRITING FILE");

return 2;

}

else

{

WriteFile(hOut, b, 256, &dwBytes, NULL);

printf("Thread Written");

CloseHandle(hOut);

}

return 0;

}

int main4()

{

int z = 0;

DWORD myThreadID;

TCHAR Buff[256] = TEXT("HELLO! Students 2");

HANDLE myHandle = CreateThread(0, 0, myThread4, (void\*)Buff, CREATE\_SUSPENDED, &myThreadID);

BOOL b = SetThreadPriorityBoost(myHandle, false);

if (b)

{

SetThreadPriority(myHandle, THREAD\_PRIORITY\_HIGHEST);

}

ResumeThread(myHandle);

WaitForSingleObject(myHandle, 10000);

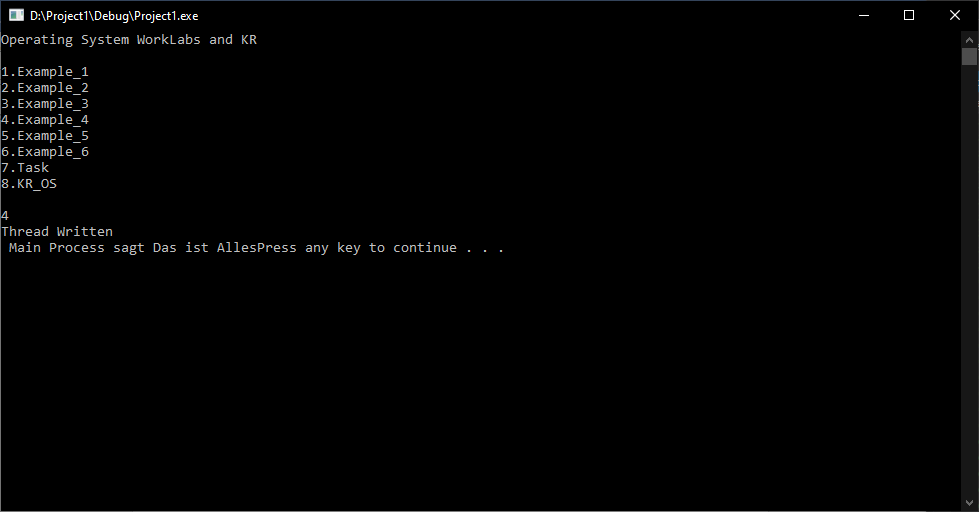
printf("\n Main Process sagt Das ist Alles");

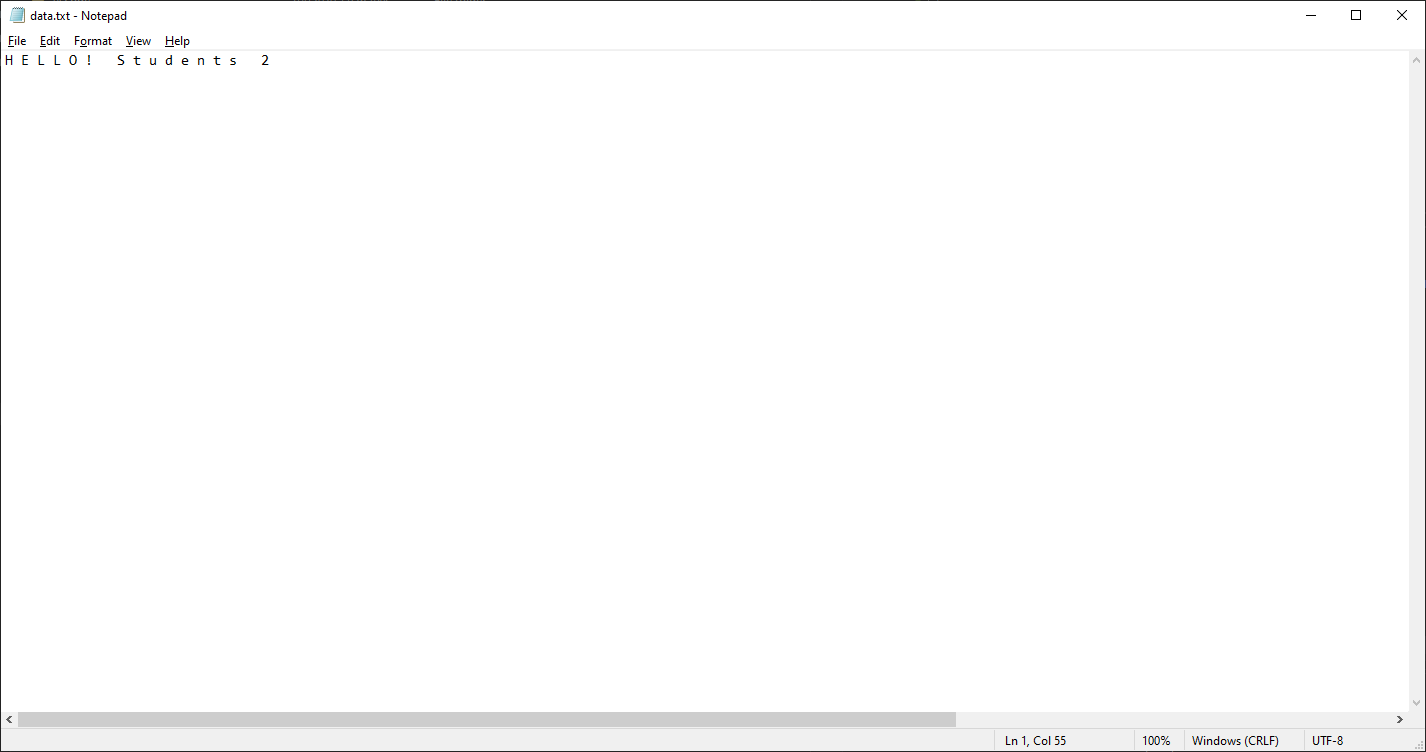
TerminateThread(myHandle, 0);

getchar();

return 0;

}





Здесь использована команда **WaitForSingleObject(myHandle,10000);**

которая заставляет ждать завершения потока либо завершение через 10000 миллисекунд.

Данная функция позволяет программе перейти в режим ожидания, пока не случится заданное событие

1. Теперь синхронизируем работу потока с файлом с помощью механизма семафоров (критических зон). Обратить внимание на использование команд

EnterCriticalSection( &cs ); - вход в критическую секцию

LeaveCriticalSection( &cs ); - выход из критической секции

#include "stdafx.h"

#include <windows.h>

#include <iostream>

#include "string.h"

CRITICAL\_SECTION cs;

DWORD WINAPI myThread5(void\* lpPar)

{

TCHAR\* b = new TCHAR();

b = (TCHAR\*)lpPar;

HANDLE hOut;

DWORD dwBytes;

FILE\* fp;

TCHAR stdPath[30] = TEXT("C:\\Users\\ilyaa\\data.txt");

EnterCriticalSection(&cs);

hOut = CreateFile(stdPath, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hOut == INVALID\_HANDLE\_VALUE)

{

printf("ERROR WRITING FILE");

return 2;

}

else

{

WriteFile(hOut, b, 256, &dwBytes, NULL);

printf("Thread Written");

CloseHandle(hOut);

}

LeaveCriticalSection(&cs);

return 0;

}

int main5()

{

int z = 0;

DWORD myThreadID;

TCHAR Buff[256] = TEXT("HELLO! Students 2");

InitializeCriticalSection(&cs);

HANDLE myHandle = CreateThread(0, 0, myThread5, (void\*)Buff, CREATE\_SUSPENDED, &myThreadID);

BOOL b = SetThreadPriorityBoost(myHandle, false);

if (b)

{

SetThreadPriority(myHandle, THREAD\_PRIORITY\_HIGHEST);

}

ResumeThread(myHandle);

WaitForSingleObject(myHandle, 10000);

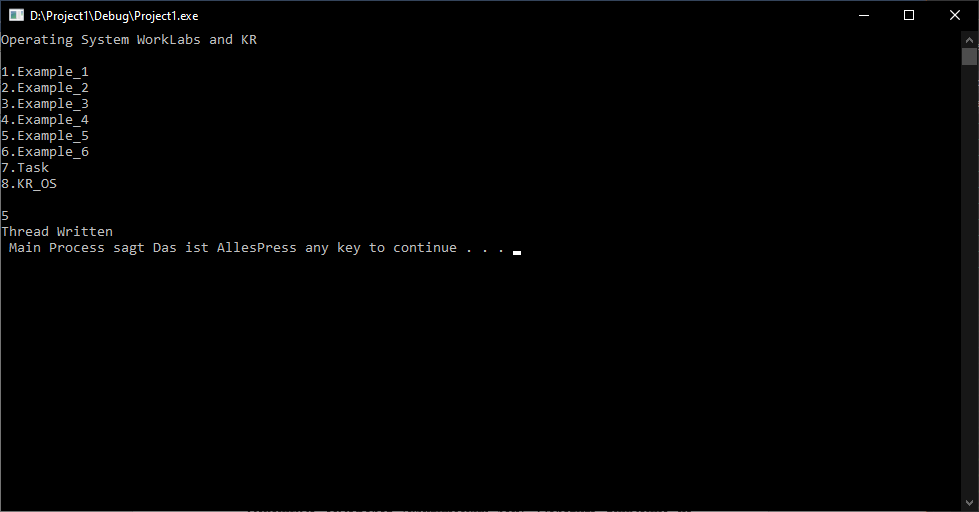
printf("\n Main Process sagt Das ist Alles");

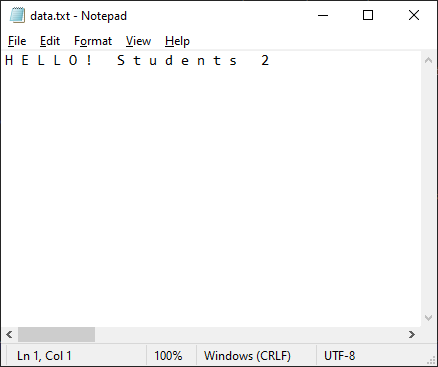
TerminateThread(myHandle, 0);

getchar();

return 0;

}





1. Наконец, сделаем синхронизированный доступ к одному и тому же файлу через механизм критических секций со стороны двух потоков.

#include "stdafx.h"

#include <windows.h>

#include <iostream>

#include "string.h"

CRITICAL\_SECTION cs;

DWORD WINAPI myThread6(void\* lpPar)

{

TCHAR\* b = new TCHAR();

b = (TCHAR\*)lpPar;

HANDLE hOut;

DWORD dwBytes;

FILE\* fp;

TCHAR stdPath[30] = TEXT("C:\\Users\\Ilyaa\\data.txt");

EnterCriticalSection(&cs);

hOut = CreateFile(stdPath, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hOut == INVALID\_HANDLE\_VALUE)

{

printf("ERROR WRITING FILE");

return 2;

}

else

{

WriteFile(hOut, b, 256, &dwBytes, NULL);

printf("Thread 1 Written");

CloseHandle(hOut);

}

LeaveCriticalSection(&cs);

return 0;

}

DWORD WINAPI myThread7(void\* lpPar)

{

TCHAR\* b = new TCHAR();

b = (TCHAR\*)lpPar;

HANDLE hOut;

DWORD dwBytes;

FILE\* fp;

TCHAR stdPath[30] = TEXT("C:\\Users\\Ilyaa\\data.txt");

EnterCriticalSection(&cs);

hOut = CreateFile(stdPath, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hOut == INVALID\_HANDLE\_VALUE)

{

printf("ERROR WRITING FILE");

return 2;

}

else

{

WriteFile(hOut, b, 256, &dwBytes, NULL);

printf("Thread 2 Written");

CloseHandle(hOut);

}

LeaveCriticalSection(&cs);

return 0;

}

int main6()

{

int z = 0;

DWORD myThreadID;

DWORD myThreadID2;

TCHAR Buff[256] = TEXT("HELLO! Students 2");

TCHAR Buff2[256] = TEXT("Gut Abend Liber Kollegen");

InitializeCriticalSection(&cs);

HANDLE myHandle = CreateThread(0, 0, myThread6, (void\*)Buff, CREATE\_SUSPENDED, &myThreadID);

BOOL b = SetThreadPriorityBoost(myHandle, false);

if (b)

{

SetThreadPriority(myHandle, THREAD\_PRIORITY\_HIGHEST);

}

HANDLE myHandle2 = CreateThread(0, 0, myThread7, (void\*)Buff2, CREATE\_SUSPENDED, &myThreadID2);

ResumeThread(myHandle);

ResumeThread(myHandle2);

WaitForSingleObject(myHandle, 10000);

WaitForSingleObject(myHandle2, 10000);

printf("\n Main Process sagt Das ist Alles");

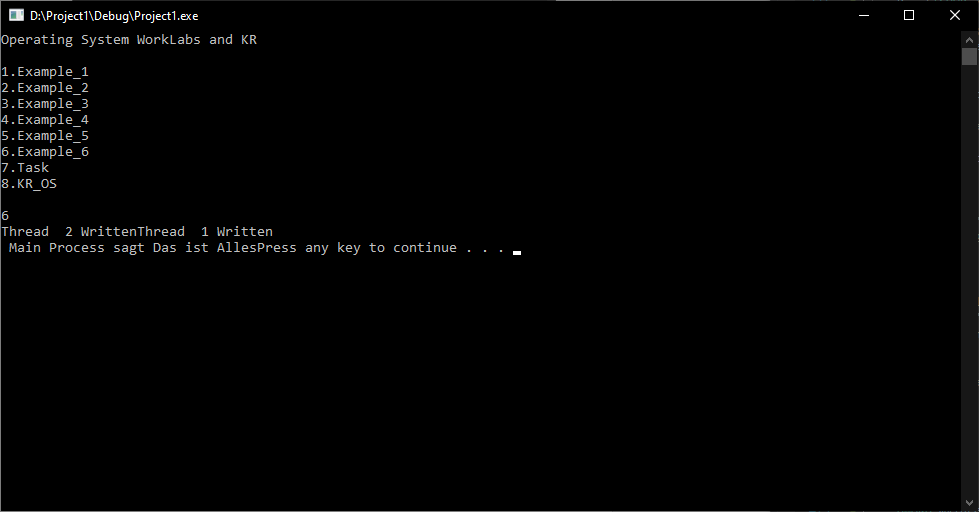
TerminateThread(myHandle, 0);

TerminateThread(myHandle2, 0);

getchar();

return 0;

}



**ЗАДАНИЕ.**

НАПИСАТЬ ПРОГРАММУ С ДВУМЯ ПОТОКАМИ. ОДИН ПИШЕТ В ФАЙЛ, ДРУГОЙ ЧИТАЕТ ИЗ ТОГО ЖЕ ФАЙЛА. ПРОДУМАТЬ СИНХРОНИЗАЦИЮ ДОСТУПА ЧЕРЕ МЕХАНИЗМ КРИТИЧЕСКИХ ЗОН.

Будем считать, что поток-писатель пишет числа. Для преобразования чисел в строки применим такую конструкцию

char\*u =new char();

itoa(25,u,10);

Строка u получает значение 25 в 10-ричной системе. Также может понадобиться соединение строк.

bool flag = false;

TCHAR stdPath[30] = TEXT("myfile.txt");

char\* u = new char(256);

TCHAR Buff[256] = TEXT("");

DWORD WINAPI myThread8(void\* lpPar)

{

HANDLE\* h = new HANDLE();

h = (HANDLE\*)lpPar;

HANDLE hh = \*h;

HANDLE hOut;

DWORD dwBytes;

FILE\* fp;

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

{

if (flag == false)

{

EnterCriticalSection(&cs);

hOut = CreateFile(stdPath, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

\_itoa\_s(i, u, 256, 10);

WriteFile(hOut, u, 256, &dwBytes, NULL);

printf("Thread 1 Written\n");

CloseHandle(hOut);

LeaveCriticalSection(&cs);

ResumeThread(hh);

Sleep(1000);

}

if (flag == true)

{

SuspendThread(hh);

flag = false;

}

}

return 0;

}

DWORD WINAPI myThread9(void\* lpPar)

{

TCHAR Buffer[256];

HANDLE hIn;

DWORD dwNumberOfBytes;

while (true)

{

if (flag == false)

{

EnterCriticalSection(&cs);

hIn = CreateFile(stdPath, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hIn == INVALID\_HANDLE\_VALUE)

{

printf("ERROR %x \n", GetLastError());

getchar();

return 2;

}

else

{

ReadFile(hIn, u, 256, &dwNumberOfBytes, NULL);

printf("Read: %s \n", u);

}

CloseHandle(hIn);

LeaveCriticalSection(&cs);

flag = true;

}

}

\_getch();

return 0;

}

int main7()

{

int z = 0;

DWORD myThreadID;

DWORD myThreadID2;

InitializeCriticalSection(&cs);

HANDLE myHandle2 = CreateThread(0, 0, myThread9, NULL, CREATE\_SUSPENDED, &myThreadID2);

HANDLE myHandle = CreateThread(0, 0, myThread8, (void\*)&myHandle2, NULL, &myThreadID);

BOOL b = SetThreadPriorityBoost(myHandle, false);

if (b)

{

SetThreadPriority(myHandle, THREAD\_PRIORITY\_HIGHEST);

}

WaitForSingleObject(myHandle, 10000);

WaitForSingleObject(myHandle2, 10000);

printf("\n Main Process sagt Das ist Alles");

TerminateThread(myHandle, 0);

TerminateThread(myHandle2, 0);

getchar();

return 0;

}

