МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра информационных систем и технологий

**ЛАБОРАТОРНАЯ РАБОТА** **№1.**

**РАБОТА С ПОТОКАМИ ПРОДОЛЖЕНИЕ**

Выполнили: Карпович П.И., Ральцевич Э.В., Коляго Л.В. ФИТ, 3 курс, 5 группа(ПОИТ)

Проверила: Герман Ю.О.

2017

Изменение приоритета потока.

#include "stdafx.h"

#include <windows.h>

#include <iostream>

#include "string.h"

DWORD WINAPI myThread(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 \_tmain(int argc, \_TCHAR\* argv[])

{

int z = 5;

unsigned int myCounter = 0;

DWORD myThreadID;

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

int CurPr = GetPriorityClass(GetCurrentProcess());

int iPriority1 = GetThreadPriority(myHandle);

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

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

iPriority1 = GetThreadPriority(myHandle);

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

SetThreadPriority(myHandle, THREAD\_PRIORITY\_HIGHEST);

iPriority1 = GetThreadPriority(myHandle);

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

SetThreadPriority(myHandle, THREAD\_PRIORITY\_LOWEST);

iPriority1 = GetThreadPriority(myHandle);

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

Sleep(5000);

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

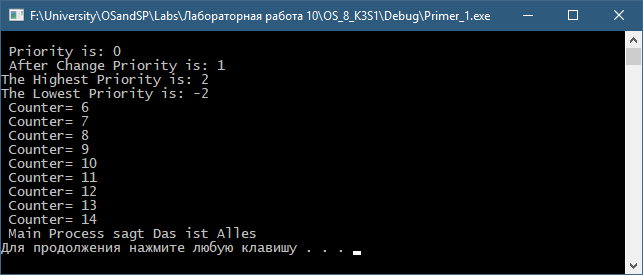
TerminateThread(myHandle, 0);

std::cout << std::endl;

system("pause");

return 0;

}



Запуск потока на запись в файл.

#include "stdafx.h"

#include <windows.h>

#include <iostream>

#include "string.h"

DWORD WINAPI myThread(void\* lpParameter)

{

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

HANDLE hOut;

DWORD dwBytes;

FILE \*fp;

TCHAR stdPath[30] = TEXT("f:\\myfile.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 \_tmain(int argc, \_TCHAR\* argv[])

{

int z = 0;

DWORD myThreadID;

HANDLE myHandle = CreateThread(0, 0, myThread, (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");

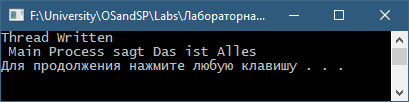
TerminateThread(myHandle, 0);

std::cout << std::endl;

system("pause");

return 0;

}



Теперь передадим записываемую строку в качестве параметра в поток:

#include "stdafx.h"

#include <windows.h>

#include <iostream>

#include "string.h"

DWORD WINAPI myThread(void\* lpPar)

{

TCHAR\* b = new TCHAR();

b = (TCHAR\*)lpPar;

HANDLE hOut;

DWORD dwBytes;

FILE \*fp;

TCHAR stdPath[30] = TEXT("f:\\myfile.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 \_tmain(int argc, \_TCHAR\* argv[])

{

int z = 0;

DWORD myThreadID;

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

HANDLE myHandle = CreateThread(0, 0, myThread, (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");

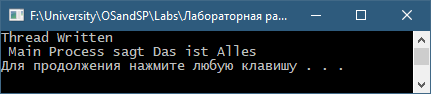
TerminateThread(myHandle, 0);

std::cout << std::endl;

system("pause");

return 0;

}



Теперь в главной программе дождемся завершения работы потока:

#include "stdafx.h"

#include <windows.h>

#include <iostream>

#include "string.h"

DWORD WINAPI myThread(void\* lpPar)

{

TCHAR\* b = new TCHAR();

b = (TCHAR\*)lpPar;

HANDLE hOut;

DWORD dwBytes;

FILE \*fp;

TCHAR stdPath[30] = TEXT("f:\\myfile.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 \_tmain(int argc, \_TCHAR\* argv[])

{

int z = 0;

DWORD myThreadID;

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

HANDLE myHandle = CreateThread(0, 0, myThread, (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");

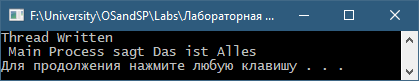
TerminateThread(myHandle, 0);

std::cout << std::endl;

system("pause");

return 0;

}



Теперь синхронизируем работу потока с файлом с помощью механизма семафоров (критических зон).

#include "stdafx.h"

#include <windows.h>

#include <iostream>

#include "string.h"

CRITICAL\_SECTION cs;

DWORD WINAPI myThread(void\* lpPar)

{

TCHAR\* b = new TCHAR();

b = (TCHAR\*)lpPar;

HANDLE hOut;

DWORD dwBytes;

FILE \*fp;

TCHAR stdPath[30] = TEXT("e:\\work\\myfile.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 \_tmain(int argc, \_TCHAR\* argv[])

{

int z = 0;

DWORD myThreadID;

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

InitializeCriticalSection(&cs);

HANDLE myHandle = CreateThread(0, 0, myThread, (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");

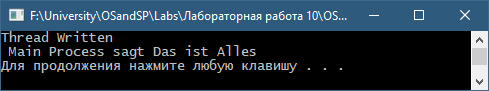
TerminateThread(myHandle, 0);

std::cout << std::endl;

system("pause");

return 0;

}



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 myThread(void\* lpPar)

{

TCHAR\* b = new TCHAR();

b = (TCHAR\*)lpPar;

HANDLE hOut;

DWORD dwBytes;

FILE \*fp;

TCHAR stdPath[30] = TEXT("f:\\myfile.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 myThread2(void\* lpPar)

{

TCHAR\* b = new TCHAR();

b = (TCHAR\*)lpPar;

HANDLE hOut;

DWORD dwBytes;

FILE \*fp;

TCHAR stdPath[30] = TEXT("f:\\myfile.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 \_tmain(int argc, \_TCHAR\* argv[])

{

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, myThread, (void\*)Buff, CREATE\_SUSPENDED, &myThreadID);

BOOL b = SetThreadPriorityBoost(myHandle, false);

if (b)

{

SetThreadPriority(myHandle, THREAD\_PRIORITY\_HIGHEST);

}

HANDLE myHandle2 = CreateThread(0, 0, myThread2, (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);

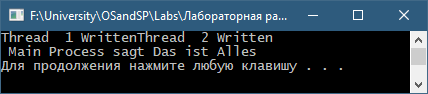
TerminateThread(myHandle2, 0);

std::cout << std::endl;

system("pause");

return 0;

}



**ЗАДАНИЕ.**

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

#include "stdafx.h"

#include <windows.h>

#include <iostream>

#include <fstream>

#include <string>

#include "string.h"

CRITICAL\_SECTION critical;

void writer(std::string filename, std::string text)

{

EnterCriticalSection(&critical);

std::ofstream file(filename);

file << text << std::endl;

LeaveCriticalSection(&critical);

}

void reader(std::string filename)

{

EnterCriticalSection(&critical);

std::ifstream file(filename);

std::string text;

file >> text;

std::cout << text << std::endl;

LeaveCriticalSection(&critical);

}

int main()

{

std::string filename = "f:\\myfile.txt";

std::string line = "test\_line";

if (!InitializeCriticalSectionAndSpinCount(&critical, 0x01))

{

throw;

}

writer(filename, line);

reader(filename);

DeleteCriticalSection(&critical);

system("pause");

return 0;

}

