ЛАБОРАТОРНАЯ РАБОТА №8.

**РАБОТА С ПОТОКАМИ ПРОДОЛЖЕНИЕ.**

Выполнила студентка  
 5 группы 3 курса ФИТ

Агапкина Диана

Минск, 2019

1. **Изменение приоритета потока.**

В качестве программы, демонстрирующей изменение приоритета потока, рассмотреть следующую

#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);// + 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.**

#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("c:\\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);

getchar();

return 0;

}

****

1. **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("c:\\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);

getchar();

return 0;

}



1. **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("c:\\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);

getchar();

return 0;

}

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

которая заставляет ждать завершения потока либо 10000 миллисекунд (что раньше).



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 myThread(void\* lpPar)

{

TCHAR\* b= new TCHAR();

b=(TCHAR\*) lpPar;

HANDLE hOut;

DWORD dwBytes;

FILE \*fp;

TCHAR stdPath[30]=TEXT("c:\\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);

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 myThread(void\* lpPar)

{

TCHAR\* b= new TCHAR();

b=(TCHAR\*) lpPar;

HANDLE hOut;

DWORD dwBytes;

FILE \*fp;

TCHAR stdPath[30]=TEXT("c:\\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("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 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);

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 применить такую конструкцию

char\*u =new char();

itoa(25,u,10);

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

#include "stdafx.h"

#include <windows.h>

#include <iostream>

#include "string.h"

#include <stdlib.h>

#include <fstream>

#pragma warning (disable : 4996)

CRITICAL\_SECTION cs;

DWORD WINAPI myThread(void\* lpPar)

{

//char info[256]="Hello student";

char info[256];

TCHAR\* b = new TCHAR();

b = (TCHAR\*)lpPar;

//std::wcout << b;

wcstombs(info, b, wcslen(b) + 1);

std::cout << "WRITING : " << info << "\n";

std::ofstream file("C:\\myfile.txt");

EnterCriticalSection(&cs);

file << info;

file.close();

LeaveCriticalSection(&cs);

return 0;

}

DWORD WINAPI myThread2(void\* lpPar)

{

EnterCriticalSection(&cs);

std::ifstream file("C:\\myfile.txt");

char info[256];

file.getline(info, 256);

file.close();

std::cout << "READ : " << info;

LeaveCriticalSection(&cs);

return 0;

}

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

{

DWORD myThreadID;

DWORD myThreadID2;

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

TCHAR Buff2[256] = TEXT("start");

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 end");

TerminateThread(myHandle, 0);

TerminateThread(myHandle2, 0);

getchar();

return 0;

}

