**山东师范大学**

**实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 院系部所 | 信息科学与工程学院 | | 课程名称 | | 计算机操作系统 |
| 所在班级 | 计工本1702 | 学号 | 201711010202 | 姓名 | 王汝芸 |
| 实验编号 | 03 | 题目 | 线程的互斥 | | |

一、实验目的和要求

（1）进一步掌握Windows系统环境下线程的创建与撤销。

（2）熟悉Windows系统提供的线程同步API。

（3）使用Windows系统提供的线程同步API解决实际问题。

二、实验环境

Windows 10

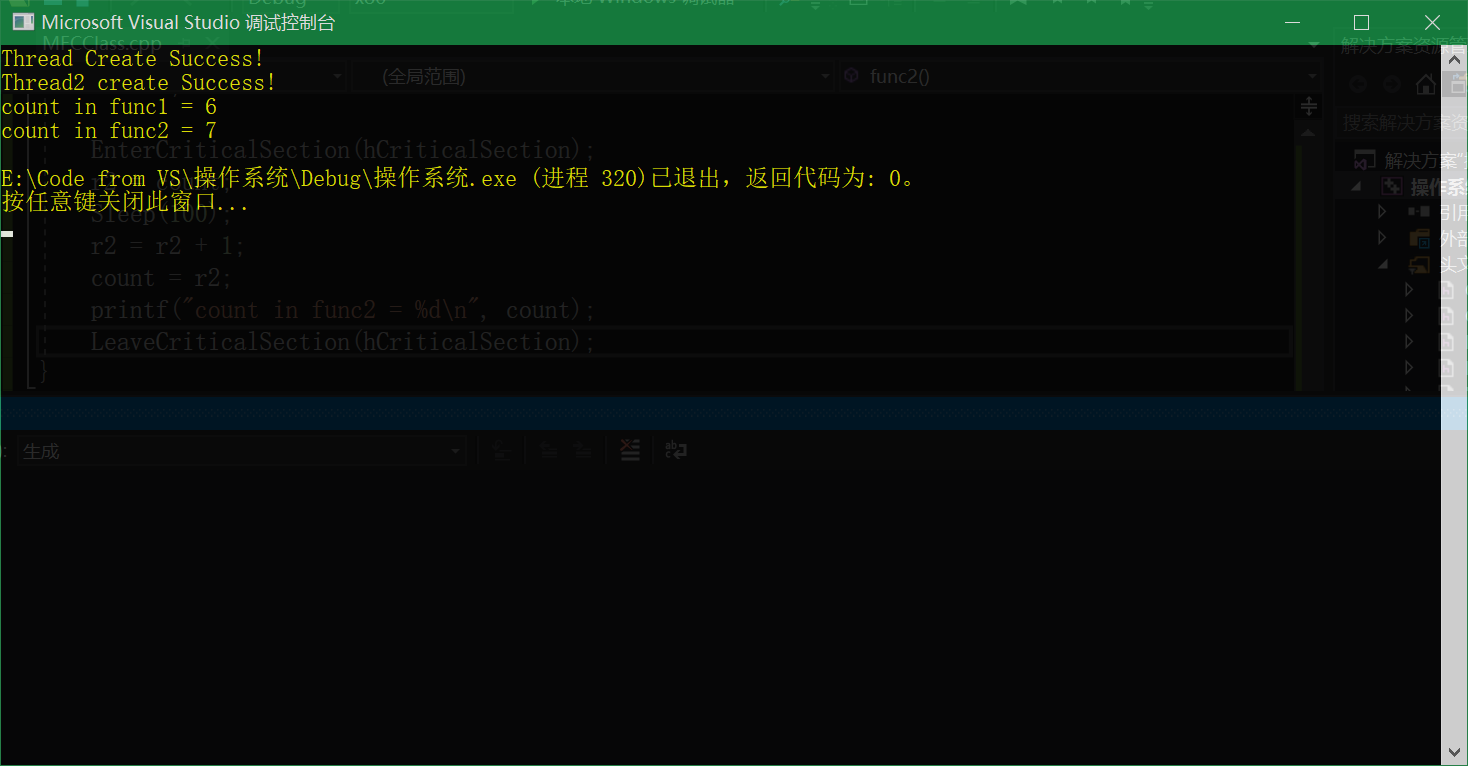
Visual Studio 2017

三、实验内容及实施

完成两个子线程间的互斥。在主线程中使用系统调用函数CreateThread()创建两个子线程，并使两个子线程互斥地使用全局变量count。

建立一个项目，在主线程中使用InitializeCriticalSection()初始化临界区，然后建立两个子线程。在两个子线程中使用全局变量count前、后分别使用EnterCriticalSection()进入临界区及使用LeaveCriticalSection()退出临界区，等两个子线程运行完毕，主线程使用DeleteCriticalSection()删除临界区并撤销线程。

四、实验结果



五、实验分析

该实验完成了两个子线程的互斥。若去掉互斥对象，观察全局变量count的变化，了解互斥对象的作用，进一步理解线程的互斥。本实验也可以使用互斥对象（Mutex）来完成两个线程的互斥，互斥对象（Mutex）的使用方法与信号量对象相似。

六、实验讨论（可选）

无。

七、附录 （可选）

//Semaphore.cpp:Define the entry point for the console application.

//实验三：线程的互斥

#include "stdafx.h"

#include "操作系统.h"

#ifdef \_DEBUG

#define new DUBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

//////////////////////////////////////////////////////////

//CWinApp theApp;

using namespace std;

static int count = 5;

static HANDLE h1;

static HANDLE h2;

LPCRITICAL\_SECTION hCriticalSection;

CRITICAL\_SECTION Critical;

void func1();

void func2();

int \_tmain(int argc, TCHAR\*atgv[], TCHAR\*envp[])

{

int nRetCode = 0;

DWORD dwThreadID1, dwThreadID2;

hCriticalSection = &Critical;

InitializeCriticalSection(hCriticalSection);

h1 = CreateThread((LPSECURITY\_ATTRIBUTES)NULL,

0,

(LPTHREAD\_START\_ROUTINE)func1,

(LPVOID)NULL,

0, &dwThreadID1);

if (h1 == NULL)printf("Thread Create Fail!\n");

else printf("Thread Create Success!\n");

h2 = CreateThread((LPSECURITY\_ATTRIBUTES)NULL,

0,

(LPTHREAD\_START\_ROUTINE)func2,

(LPVOID)NULL,

0, &dwThreadID2);

if (h2 == NULL)printf("Thread2 create Fail!\n");

else printf("Thread2 create Success!\n");

Sleep(1000);

CloseHandle(h1);

CloseHandle(h2);

DeleteCriticalSection(hCriticalSection);

ExitThread(0);

return nRetCode;

}

void func2()

{

int r2;

EnterCriticalSection(hCriticalSection);

r2 = count;

Sleep(100);

r2 = r2 + 1;

count = r2;

printf("count in func2 = %d\n", count);

LeaveCriticalSection(hCriticalSection);

}

void func1()

{

int r1;

EnterCriticalSection(hCriticalSection);

r1 = count;

Sleep(500);

r1 = r1 + 1;

count = r1;

printf("count in func1 = %d\n", count);

LeaveCriticalSection(hCriticalSection);

}