## 姓名：陈小龙 班级：2014级计算机2班 学号：20140031

## 实验三 Windows中的线程与线程同步现象

### 一、实验目的

1、掌握Windows中线程的操作。

2、熟悉线程不同步时的现象及环境因素。

3、掌握一种同步对象的使用。

### 二、实验理论基础及教材对应关系

1、线程和线程不同步的认识。

2、线程间的同步和通信。

### 三、实验内容与步骤

1、定义全局变量 int i = 0; 初始值置为 0。

2、创建两个线程，一个对 i 执行加 1 操作，另一个对 i 执行减 1 操作。两个线程执行相同的次数。

显然，正常情况下，i 的仍然保持为 0。

#include <stdio.h>

#include <windows.h>

#define MaxCount 9000000 // 循环次数要很大，可多次尝试一些值

DWORD \_\_stdcall fun1( LPVOID p1)

{

for( int j =0 ;j < MaxCount;j++){

i++;

}

return 0;

}

DWORD \_\_stdcall fun2( LPVOID p1)

{

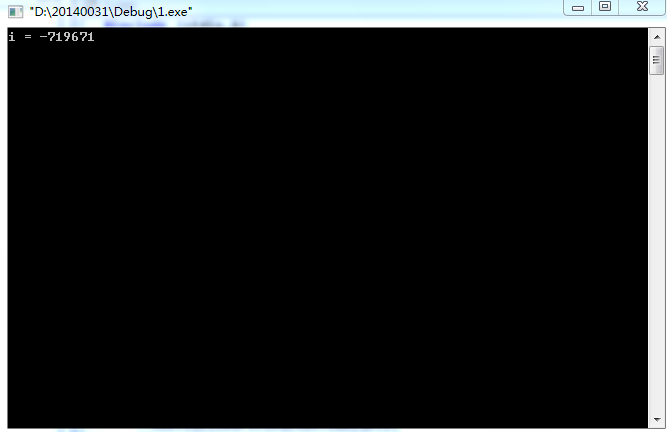
for( int j =0 ;j < MaxCount;j++){

i--;

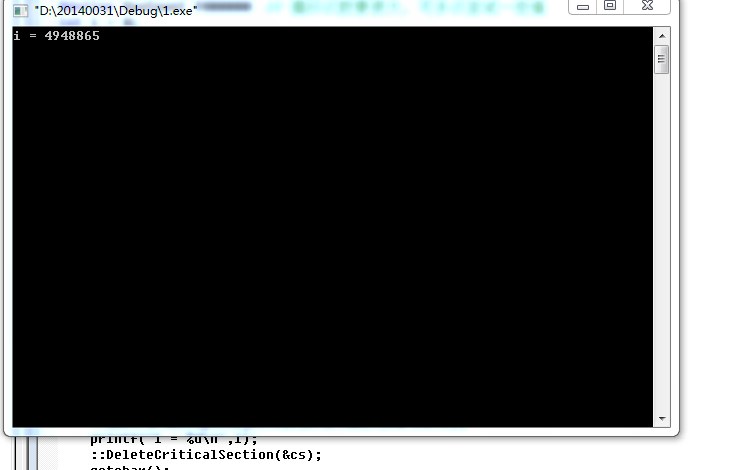
}

return 0;

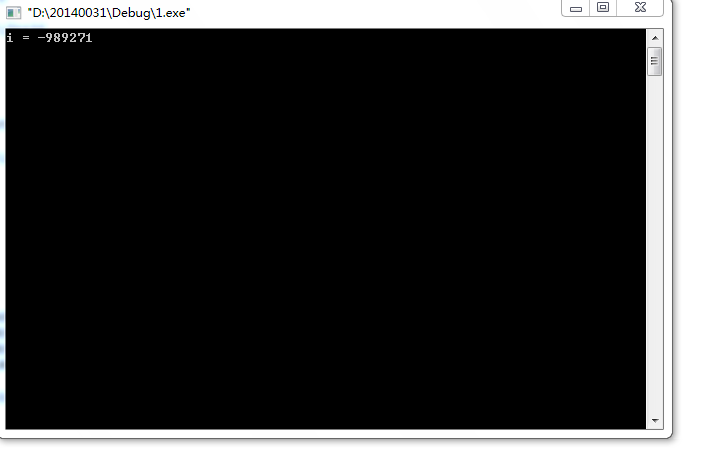
}



**I的值为-719671**



**I的值为4948865**



**I的值为-989271**

3、观察两个线程执行后的情况，可以发觉最后 i 的值不一定是 0， 有时是很大的正数，有时是很大的负数，这就是多个线程在操作同一个变量 i时，未同步时带来的严重问题。

还应该了解，在多个线程操作共享的变量时，才需要考虑同步问题。

1. 给这两个线程加上同步代码，再来观察对 i 值的影响。步骤2的函数稍微改动即可：

CRITICAL\_SECTION cs;

DWORD \_\_stdcall fun1( LPVOID p1)

{

for( int j =0 ;j < MaxCount;j++){

::EnterCriticalSection(&cs);

i++;

::LeaveCriticalSection(&cs);

}

}

DWORD \_\_stdcall fun2( LPVOID p1)

{

for( int j =0 ;j < MaxCount;j++){

::EnterCriticalSection(&cs);

i--;

::LeaveCriticalSection(&cs);

}

}

加入的同步代码的两个线程，无论如何执行，i 的值总是 0 ，结果是正确的。

1. 主函数的写法

int main()

{

DWORD id1,id2;

HANDLE hThread[2];

::InitializeCriticalSection(&cs);

hThread[0] = ::CreateThread(0,0,fun1,0,0,&id1);

hThread[1] = ::CreateThread(0,0,fun2,0,0,&id2);

::WaitForMultipleObjects(2,hThread,1,INFINITE);

printf("i = %d\n",i);

::DeleteCriticalSection(&cs);

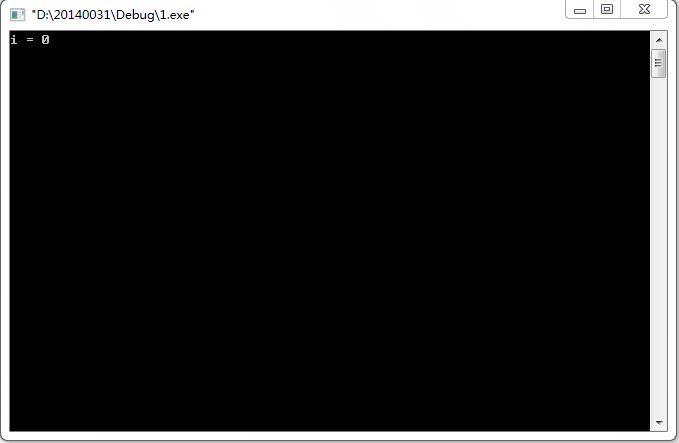
getchar();

return 0;

}

### 四、实验材料的提交与成绩评定

本实验的实验报告一份（电子版，格式参考统一实验报告）



**I的值为0**



**I的值为0**