[秒杀多线程第四篇 一个经典的多线程同步问题](http://blog.csdn.net/morewindows/article/details/7442333)

分类： [Windows多线程](http://blog.csdn.net/MoreWindows/article/category/1115271) [Windows编程](http://blog.csdn.net/MoreWindows/article/category/862060)2012-04-10 09:57 102165人阅读 [评论](http://blog.csdn.net/morewindows/article/details/7442333#comments)(49) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/morewindows/article/details/7442333#report)

[多线程](http://www.csdn.net/tag/%e5%a4%9a%e7%ba%bf%e7%a8%8b)[thread](http://www.csdn.net/tag/thread)[semaphore](http://www.csdn.net/tag/semaphore)[fun](http://www.csdn.net/tag/fun)[null](http://www.csdn.net/tag/null)[windows](http://www.csdn.net/tag/windows)

上一篇《[秒杀多线程第三篇原子操作 Interlocked系列函数](http://blog.csdn.net/morewindows/article/details/7429155)》中介绍了原子操作在多进程中的作用，现在来个复杂点的。这个问题涉及到线程的同步和互斥，是一道非常有代表性的多线程同步问题，如果能将这个问题搞清楚，那么对多线程同步也就打下了良好的基础。

程序描述：

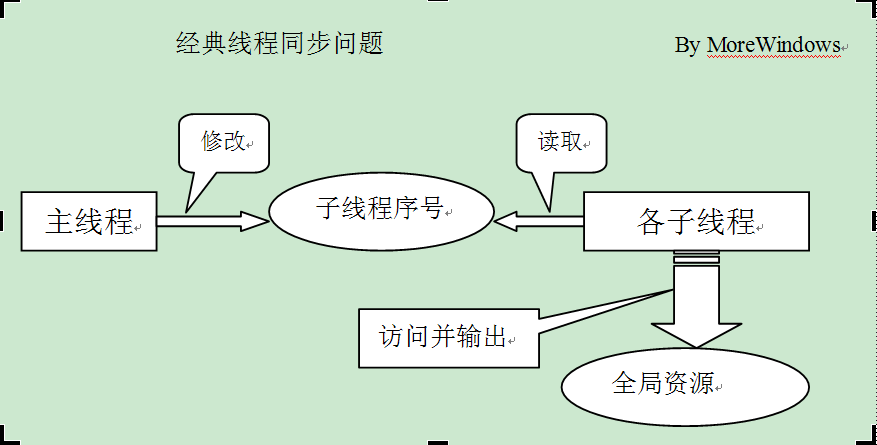
主线程启动10个子线程并将表示子线程序号的变量地址作为参数传递给子线程。子线程接收参数 -> sleep(50) -> 全局变量++ -> sleep(0) -> 输出参数和全局变量。

要求：

1．子线程输出的线程序号不能重复。

2．全局变量的输出必须递增。

下面画了个简单的示意图：



分析下这个问题的考察点，主要考察点有二个：

1．主线程创建子线程并传入一个指向变量地址的指针作参数，由于线程启动须要花费一定的时间，所以在子线程根据这个指针访问并保存数据前，主线程应等待子线程保存完毕后才能改动该参数并启动下一个线程。这涉及到**主线程与子线程之间的同步**。

2．子线程之间会互斥的改动和输出全局变量。要求全局变量的输出必须递增。这涉及到**各子线程间的互斥**。

下面列出这个程序的基本框架，可以在此代码基础上进行修改和验证。

**[cpp]** [view plaincopy](http://blog.csdn.net/morewindows/article/details/7442333)

1. //经典线程同步互斥问题
2. #include <stdio.h>
3. #include <process.h>
4. #include <windows.h>
6. **long** g\_nNum; //全局资源
7. unsigned **int** \_\_stdcall Fun(**void** \*pPM); //线程函数
8. **const** **int** THREAD\_NUM = 10; //子线程个数
10. **int** main()
11. {
12. g\_nNum = 0;
13. **HANDLE**  handle[THREAD\_NUM];
15. **int** i = 0;
16. **while** (i < THREAD\_NUM)
17. {
18. handle[i] = (**HANDLE**)\_beginthreadex(NULL, 0, Fun, &i, 0, NULL);
19. i++;//等子线程接收到参数时主线程可能改变了这个i的值
20. }
21. //保证子线程已全部运行结束
22. WaitForMultipleObjects(THREAD\_NUM, handle, TRUE, INFINITE);
23. **return** 0;
24. }
26. unsigned **int** \_\_stdcall Fun(**void** \*pPM)
27. {
28. //由于创建线程是要一定的开销的，所以新线程并不能第一时间执行到这来
29. **int** nThreadNum = \*(**int** \*)pPM; //子线程获取参数
30. Sleep(50);//some work should to do
31. g\_nNum++;  //处理全局资源
32. Sleep(0);//some work should to do
33. printf("线程编号为%d  全局资源值为%d\n", nThreadNum, g\_nNum);
34. **return** 0;
35. }

运行结果可以参考下列图示，强烈建议读者亲自试一试。

图1

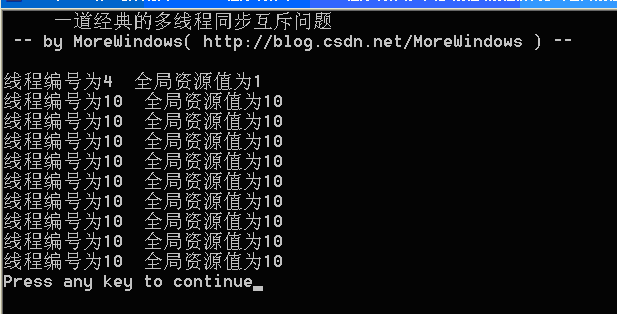


图2

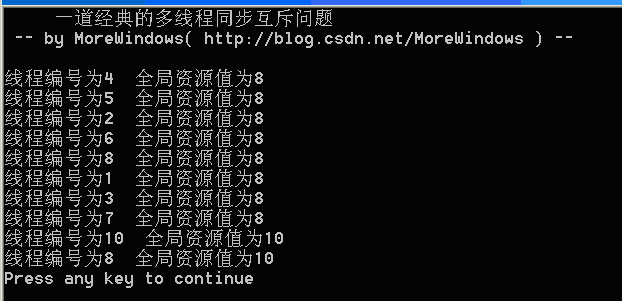


图3



可以看出，运行结果完全是混乱和不可预知的。本系列将会运用Windows平台下各种手段包括关键段，事件，互斥量，信号量等等来解决这个问题并作一份全面的总结，敬请关注。

刚学习线程，29行 int nThreadNum = \*(int \*)pPM ，怎么理解，为什么nThreadNum会自增，没有 nThreadNum++啊

30楼 [vine\_branches](http://blog.csdn.net/u014652390) 2015-07-08 17:29发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7442333#reply)

[](http://blog.csdn.net/u014652390)

**[cpp]** [view plaincopy](http://blog.csdn.net/morewindows/article/details/7442333)

1. #include "stdafx.h"
2. #include "stdio.h"
3. #include <process.h>
4. #include <windows.h>
6. #define THREAD\_NUM  10 // THREAD\_NUM > 64会有问题
7. **volatile** **long** g\_nLoginCount;
9. unsigned **int** \_stdcall ThreadFun4(**void** \* pPM)
10. {
11. **int** nThreadNum = \*(**int** \*)pPM;
12. g\_nLoginCount++;
13. printf("Thread number is %d,  g\_nLoginCount is %d\n", nThreadNum, g\_nLoginCount);
14. **return** 0;
15. }
16. **int** \_tmain(**int** argc, \_TCHAR\* argv[])
17. {
18. g\_nLoginCount = 0;
19. **HANDLE** handle[THREAD\_NUM];
21. **for** (**int** i = 0; i < THREAD\_NUM; i++)
22. {
23. handle[i] = (**HANDLE**)\_beginthreadex(NULL, 0, ThreadFun4, &i, 0, NULL);
24. }
25. WaitForMultipleObjects(THREAD\_NUM, handle, **true**, INFINITE);
26. **return** 0;
27. }

以上是我的代码，i值乱序可以理解，为什么nLoginCount 会出现递减的情况呀？

引用“wy223170wy”的评论：博主，这个主线程和子线程的同步问题不是十分了解，可否详细解释一下  
同问啊楼主。

参考;windows核心编程 和C++面向对象多线程编程

[](http://blog.csdn.net/skyandcode)

楼主，看到你这篇文章，按你的思路写了一个例子，确发现一个奇怪的现象：  
  
  
#include "stdafx.h"  
#include<Windows.h>  
#include <process.h>  
#include <iostream>  
using namespace std;  
  
int num=0;   
const int threadNum=10;  
unsigned int \_stdcall ThreadFunc(void \*lpParam)  
{  
//注意到一下两行注释的代码，按没注释的运行结果是正常的。 但是如果用注释的那两行，每次线程编号都几乎是10这个编号，初学多线程，望解惑。  
//int\*i=(int\*)lpParam;  
int i=\*(int\*)lpParam;  
Sleep(50);  
num++;  
Sleep(0);  
//cout<<"编号为："<<\*i<<"的线程输出"<<num<<endl;  
cout<<"编号为："<<i<<"的线程输出"<<num<<endl;  
return 0;  
}  
  
int \_tmain(int argc, \_TCHAR\* argv[])  
{  
HANDLE hThread[threadNum];  
int i=0;  
for(int i=0;i<threadNum;i++)  
{  
hThread[i]=(HANDLE)\_beginthreadex(NULL,0,ThreadFunc,&i,0,NULL);  
  
}  
  
WaitForMultipleObjects(threadNum,hThread,TRUE,INFINITE);  
  
return 0;  
}

我的理解：int i=\*(int\*)lpParam;保存的是指针所指向的值。  
int\*i=(int\*)lpParam;保存的是指针。而我们所要操作（显示）的是指针所指向的值。

如果保存的是指针的话，因为它所指向的内容可能会被其它线程所改变，所以在通过指针的间接访问时它已经不是原来的值了。

我是新手，想问为什么代码中没有把所有句柄关闭啊。不是一直说createThrea（）d后要调用closehandle（）吗或者biginthread后要exitthread啊

[](http://blog.csdn.net/wojiushiwo987)

**[cpp]** [view plaincopy](http://blog.csdn.net/morewindows/article/details/7442333)

1. **int** main()
2. {
3. g\_nNum = 0;
4. **HANDLE**  handle[THREAD\_NUM];
6. **int** i = 0;
7. **while** (i < THREAD\_NUM)
8. {
9. handle[i] = (**HANDLE**)\_beginthreadex(NULL, 0, Fun, &i, 0, NULL);
10. //  i++;//等子线程接收到参数时主线程可能改变了这个i的值
11. InterlockedIncrement((**PLONG**)&i);    //改动1,确保原子操作
12. Sleep(10);                          //改动2，实质这改变了原意，阻塞进程运行了.
13. }
14. //保证子线程已全部运行结束
15. WaitForMultipleObjects(THREAD\_NUM, handle, TRUE, INFINITE);
16. **return** 0;
17. }
19. unsigned **int** \_\_stdcall Fun(**void** \*pPM)
20. {
21. //由于创建线程是要一定的开销的，所以新线程并不能第一时间执行到这来
22. **int** nThreadNum = \*(**int** \*)pPM; //子线程获取参数
23. Sleep(50);//some work should to do
24. //  g\_nNum++;  //处理全局资源
25. InterlockedIncrement((**PLONG**)&g\_nNum);   //改动3确保原子操作.
26. Sleep(0);//some work should to do
27. printf("线程编号为%d  全局资源值为%d\n", nThreadNum, g\_nNum);
28. **return** 0;
29. }

在源代码是修改了三处操作（修改1，2，3）能完成题目要求的操作。只是应用楼主第3讲的interlocket确保原子操作。  
核心也是考点的应该是CrticalSection，semaphore，mutex，Event下面几章的内容。

[](http://blog.csdn.net/zwgdft)

如果采用i的值传递也就不会引起第一个问题了，采用地址传递则需要Event同步，建议LZ将两个做个对比，更有启发性。

Re: [ws\_thinker](http://blog.csdn.net/wangliuyuedeyu123) 2014-07-15 13:04发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7442333#reply)

[](http://blog.csdn.net/wangliuyuedeyu123)

回复zwgdft：你好，请问，线程间怎么实现值传递，而不是地址传递？谢谢

Re: [MoreWindows](http://blog.csdn.net/MoreWindows) 2012-04-11 09:12发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7442333#reply)