[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)

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)2012-04-09 09:08 68071人阅读 [评论](http://blog.csdn.net/morewindows/article/details/7429155#comments)(126) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/morewindows/article/details/7429155#report)

[thread](http://www.csdn.net/tag/thread)[Thread](http://www.csdn.net/tag/Thread)[原子操作](http://www.csdn.net/tag/%e5%8e%9f%e5%ad%90%e6%93%8d%e4%bd%9c)[多线程](http://www.csdn.net/tag/%e5%a4%9a%e7%ba%bf%e7%a8%8b)[汇编](http://www.csdn.net/tag/%e6%b1%87%e7%bc%96)

上一篇《[多线程第一次亲密接触 CreateThread与\_beginthreadex本质区别](http://blog.csdn.net/morewindows/article/details/7421759)》中讲到一个多线程报数功能。为了描述方便和代码简洁起见，我们可以只输出最后的报数结果来观察程序是否运行出错。这也非常类似于统计一个网站每天有多少用户登录，每个用户登录用一个线程模拟，线程运行时会将一个表示计数的变量递增。程序在最后输出计数的值表示有今天多少个用户登录，如果这个值不等于我们启动的线程个数，那显然说明这个程序是有问题的。整个程序代码如下：

**[cpp]** [view plaincopy](http://blog.csdn.net/morewindows/article/details/7429155)

1. #include <stdio.h>
2. #include <process.h>
3. #include <windows.h>
4. **volatile** **long** g\_nLoginCount; //登录次数
5. unsigned **int** \_\_stdcall Fun(**void** \*pPM); //线程函数   \_\_stdcall有什么用
6. **const** **int** THREAD\_NUM = 10; //启动线程数
7. unsigned **int** \_\_stdcall ThreadFun(**void** \*pPM)
8. {
9. Sleep(100); //some work should to do
10. g\_nLoginCount++;
11. Sleep(50);
12. **return** 0;
13. }
14. **int** main()
15. {
16. g\_nLoginCount = 0;
18. **HANDLE**  handle[THREAD\_NUM];
19. **for** (**int** i = 0; i < THREAD\_NUM; i++)
20. handle[i] = (**HANDLE**)\_beginthreadex(NULL, 0, ThreadFun, NULL, 0, NULL);
22. WaitForMultipleObjects(THREAD\_NUM, handle, TRUE, INFINITE);
23. printf("有%d个用户登录后记录结果是%d\n", THREAD\_NUM, g\_nLoginCount);
24. **return** 0;
25. }

程序中模拟的是10个用户登录，程序将输出结果：

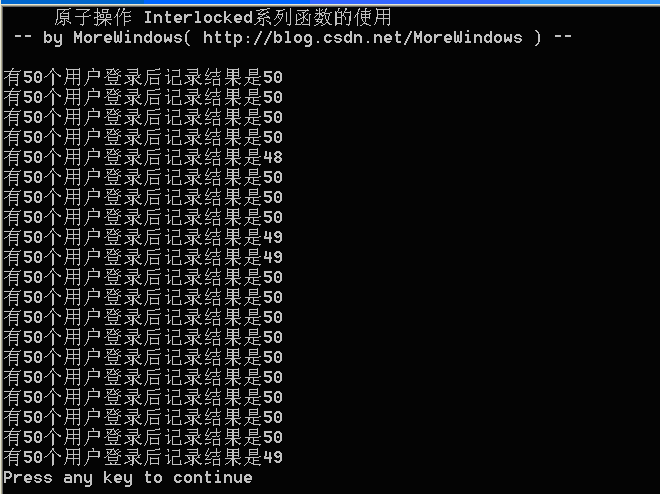


和[上一篇](http://blog.csdn.net/morewindows/article/details/7421759)的线程报数程序一样，程序输出的结果好象并没什么问题。下面我们增加点用户来试试，现在模拟50个用户登录，为了便于观察结果，在程序中将50个用户登录过程重复20次，代码如下：

**[cpp]** [view plaincopy](http://blog.csdn.net/morewindows/article/details/7429155)

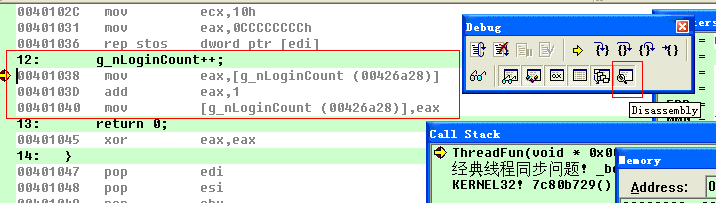
1. #include <stdio.h>
2. #include <windows.h>
3. **volatile** **long** g\_nLoginCount; //登录次数
4. unsigned **int** \_\_stdcall Fun(**void** \*pPM); //线程函数
5. **const** **DWORD** THREAD\_NUM = 50;//启动线程数
6. **DWORD** WINAPI ThreadFun(**void** \*pPM)
7. {
8. Sleep(100); //some work should to do
9. g\_nLoginCount++;
10. Sleep(50);
11. **return** 0;
12. }
13. **int** main()
14. {
15. printf("     原子操作 Interlocked系列函数的使用\n");
16. printf(" -- by MoreWindows( http://blog.csdn.net/MoreWindows ) --\n\n");
18. //重复20次以便观察多线程访问同一资源时导致的冲突
19. **int** num= 20;
20. **while** (num--)
21. {
22. g\_nLoginCount = 0;
23. **int** i;
24. **HANDLE**  handle[THREAD\_NUM];
25. **for** (i = 0; i < THREAD\_NUM; i++)
26. handle[i] = CreateThread(NULL, 0, ThreadFun, NULL, 0, NULL);
27. WaitForMultipleObjects(THREAD\_NUM, handle, TRUE, INFINITE);
28. printf("有%d个用户登录后记录结果是%d\n", THREAD\_NUM, g\_nLoginCount);
29. }
30. **return** 0;
31. }

运行结果如下图：



现在结果水落石出，明明有50个线程执行了g\_nLoginCount++;操作，但结果输出是不确定的，有可能为50，但也有可能小于50。

       要解决这个问题，我们就分析下g\_nLoginCount++;操作。在VC6.0编译器对g\_nLoginCount++;这一语句打个断点，再按F5进入调试状态，然后按下Debug工具栏的Disassembly按钮，这样就出现了汇编代码窗口。可以发现在C/C++语言中一条简单的自增语句其实是由三条汇编代码组成的，如下图所示。



讲解下这三条汇编意思：

第一条汇编将g\_nLoginCount的值从内存中读取到寄存器eax中。

第二条汇编将寄存器eax中的值与1相加，计算结果仍存入寄存器eax中。

第三条汇编将寄存器eax中的值写回内存中。

       这样由于线程执行的并发性，很可能线程A执行到第二句时，线程B开始执行，线程B将原来的值又写入寄存器eax中，这样线程A所主要计算的值就被线程B修改了。这样执行下来，结果是不可预知的——可能会出现50，可能小于50。

       因此在多线程环境中对一个变量进行读写时，我们需要有一种方法能够保证对一个值的递增操作是原子操作——即不可打断性，一个线程在执行原子操作时，其它线程必须等待它完成之后才能开始执行该原子操作。这种涉及到硬件的操作会不会很复杂了，幸运的是，Windows系统为我们提供了一些以Interlocked开头的函数来完成这一任务（下文将这些函数称为Interlocked系列函数）。

下面列出一些常用的Interlocked系列函数：

**1.增减操作**

LONG\_\_cdeclInterlockedIncrement(LONG volatile\* Addend);

LONG\_\_cdeclInterlockedDecrement(LONG volatile\* Addend);

返回变量执行增减操作之后的值。

LONG\_\_cdec InterlockedExchangeAdd(LONG volatile\* Addend, LONGValue);

返回运算后的值，注意！加个负数就是减。

**2.赋值操作**

LONG\_\_cdeclInterlockedExchange(LONG volatile\* Target, LONGValue);

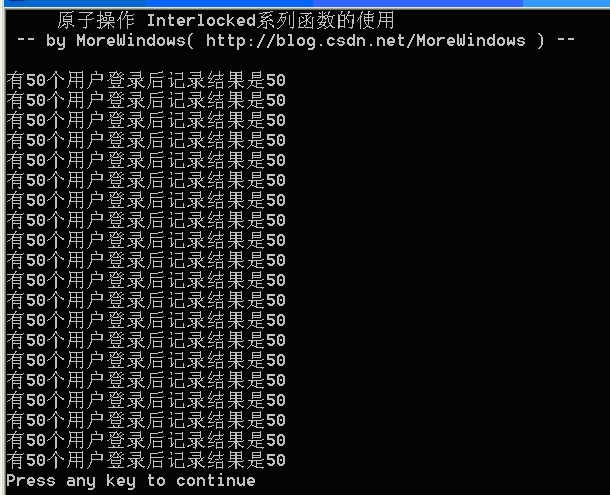
Value就是新值，函数会返回原先的值。

在本例中只要使用InterlockedIncrement()函数就可以了。将线程函数代码改成：

**[cpp]** [view plaincopy](http://blog.csdn.net/morewindows/article/details/7429155)

1. **DWORD** WINAPI ThreadFun(**void** \*pPM)
2. {
3. Sleep(100);//some work should to do
4. //g\_nLoginCount++;
5. InterlockedIncrement((**LPLONG**)&g\_nLoginCount);
6. Sleep(50);
7. **return** 0;
8. }

再次运行，可以发现结果会是唯一的。



       因此，在多线程环境下，我们对变量的自增自减这些简单的语句也要慎重思考，防止多个线程导致的数据访问出错。更多介绍，请访问MSDN上Synchronization Functions这一章节，地址为 <http://msdn.microsoft.com/zh-cn/library/aa909196.aspx>

看到这里，相信本系列首篇《[秒杀多线程第一篇 多线程笔试面试题汇总](http://blog.csdn.net/morewindows/article/details/7392749)》中选择题第一题（百度笔试题）应该可以秒杀掉了吧（知其然也知其所以然），正确答案是D。另外给个附加问题，程序中是用50个线程模拟用户登录，有兴趣的同学可以试下用100个线程来模拟一下（上机试试绝对会有意外发现^\_^）。

下一篇《秒杀多线程第四篇 一个经典多线程同步问题》将提出一个稍为复杂点但却非常经典的多线程同步互斥问题，这个问题会采用不同的方法来解答，从而让你充分熟练多线程同步互斥的“招式”。更多精彩，欢迎继续参阅。

转载请标明出处，原文地址：<http://blog.csdn.net/morewindows/article/details/7429155>

如果觉得本文对您有帮助，请点击‘顶’支持一下，您的支持是我写作最大的动力，谢谢。

62楼 [liema2000](http://blog.csdn.net/liema2000) 2014-08-29 07:18发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

[](http://blog.csdn.net/liema2000)

对于线程数多于64个的情况，我搜索资料，可以实现。（64 !!!!!）  
代码修改如下  
// WaitForMultipleObjects(THREAD\_NUM, handle, TRUE, INFINITE);   
//因而当线程数多于64时WaitForMultipleObjects应做如下更改：   
int tempNumThreads = THREAD\_NUM;   
int tempMax = 0;   
while( tempNumThreads >= MAXIMUM\_WAIT\_OBJECTS )   
{   
tempNumThreads -= MAXIMUM\_WAIT\_OBJECTS;   
WaitForMultipleObjects( MAXIMUM\_WAIT\_OBJECTS, &handle[ tempMax], TRUE, INFINITE);   
tempMax += MAXIMUM\_WAIT\_OBJECTS;   
}   
WaitForMultipleObjects( tempNumThreads, &handle[ tempMax ], TRUE, INFINITE);  
//但这种方法对第三个参数为FALSE即只要有一个事件为有信号时不适用。

1. #include <stdio.h>
2. #include <process.h>
3. #include <windows.h>
4. **volatile** **long** g\_nCount;
5. //子线程函数
6. unsigned **int** \_\_stdcall ThreadFun(**PVOID** pM)
7. {
8. //  Sleep(50);
9. InterlockedIncrement((**LPLONG**)&g\_nCount);
10. printf("线程ID号为%4d的子线程报数%d\n", GetCurrentThreadId(), g\_nCount);
11. **return** 0;
12. }
13. //主函数，所谓主函数其实就是主线程执行的函数。
14. **int** main()
15. {
16. printf("     子线程报数 \n");
17. printf(" -- by MoreWindows( http://blog.csdn.net/MoreWindows ) --\n\n");
19. **const** **int** THREAD\_NUM = 10;
20. **HANDLE** handle[THREAD\_NUM];
22. g\_nCount = 0;
23. **for** (**int** i = 0; i < THREAD\_NUM; i++)
24. handle[i] = (**HANDLE**)\_beginthreadex(NULL, 0, ThreadFun, NULL, 0, NULL);
25. WaitForMultipleObjects(THREAD\_NUM, handle, TRUE, INFINITE);
26. //为了代码的规范问题
27. **for** ( i = 0; i < THREAD\_NUM; i++)
28. CloseHandle(handle[i]);
29. **return** 0;
30. }

楼主您好，这是我结合上一节修改的，这里的原子操作好像没有什么作用，能解释一下吗？

Re: [FENGYEJINGXIANG](http://blog.csdn.net/FENGYEJINGXIANG) 2015-07-20 21:27发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

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

回复tswanggl1985：InterlockedIncrement((LPLONG)&g\_nCount);   
printf("线程ID号为%4d的子线程报数%d\n", GetCurrentThreadId(), g\_nCount);  
这两句之间是可能出现线程切换的，InterlockedIncrement只能保证g\_nCount在每个线程里面都加1，但是显示的顺序有可能是乱的。  
  
如果不用InterlockedIncrement，可能有重复的数字  
而使用后，数字不会重复了，只是顺序不对。

Re: [FENGYEJINGXIANG](http://blog.csdn.net/FENGYEJINGXIANG) 2015-07-20 21:26发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

[http://avatar.csdn.net/D/2/1/3_a28496647.jpg](http://blog.csdn.net/A28496647)

登陆人数有时是49，有时是50，使用InterlockedIncrement来自增之后就永远是50，但是，当我把THREAD\_NUM的值改大一些之后又出现这个问题，是什么情况？

Re: [MoreWindows](http://blog.csdn.net/MoreWindows) 2013-11-18 13:32发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

[http://avatar.csdn.net/C/F/7/3_morewindows.jpg](http://blog.csdn.net/MoreWindows)

回复A28496647：请看后面的多线程同步文章——《秒杀多线程第四篇 一个经典多线程同步问题》

勤学如初春之苗，不见其长，日有所增。

[http://avatar.csdn.net/B/6/7/3_tan523xt.jpg](http://blog.csdn.net/tan523xt)

const DWORD THREAD\_NUM = 100；我改成100以后，显示实际登录数为0，不知道什么原因。

Re: [halle1991](http://blog.csdn.net/halle1991) 2013-10-07 22:43发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

回复tan523xt：貌似WaitForMultipleObjects第一个参数最大是64

[http://avatar.csdn.net/F/5/E/3_daoluanxiaozi.jpg](http://blog.csdn.net/daoluanxiaozi)

这样由于线程执行的并发性，很可能线程A执行到第二句时，线程B开始执行，线程B将原来的值又写入寄存器eax中，这样线程A所主要计算的值就被线程B修改了。这样执行下来，结果是不可预知的——可能会出现50，可能小于50。  
  
这句有错误？  
  
因为线程是有自己独有的寄存器集合，在线程切换的时候是会保存现场的。  
  
出现错误可能的原因是，A，B线程都读取了同一个值，然后相加后，写入到 memory 中，这样实际上 A 和 B 是只对 global value 加了一次。

Re: [iooidadi](http://blog.csdn.net/iooidadi) 2014-10-31 11:01发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

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

回复无无无无无无无无：对，线程切换是会保护现场的，楼主原理写错了。

真的是印象深刻啊。虽然从网上查到WaitForMultipleObjects（）第一个参数最大是64，但是当自己去试的时候就忘了这茬了，还总在想为什么会输出的总是0。看了评论才明白。估计以后用到肯定会记住！

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

是我的机器配置太强悍了吗，我怎么运行不出现登录数和++数不相等的情况呢。。。

[http://avatar.csdn.net/C/F/7/3_morewindows.jpg](http://blog.csdn.net/MoreWindows)

回复yingsun：那你每个线程执行10次++吧

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

纠错!  
楼主的下面内容  
【下面列出一些常用的Interlocked系列函数：  
1.增减操作  
LONG\_\_cdecl InterlockedAdd(LONG volatile\* Addend, LONGValue);】  
没有InterlockedAdd这个函数，应该是下面函数：  
LONG InterlockedExchangeAdd ( LPLONG volatile Addend, // addend  
LONG Value // increment value);  
估计是楼主笔误。

[http://avatar.csdn.net/C/F/7/3_morewindows.jpg](http://blog.csdn.net/MoreWindows)

回复：这个sleep()主要是起延时的作用，不然示例代码中的线程将很快执行完毕，从而很难观察到线程之间互相干扰。

20楼 [无涯Andy](http://blog.csdn.net/xuanjian6) 2012-06-12 12:36发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

[](http://blog.csdn.net/xuanjian6)

改成100后，所有输出都乱了  
于是我把for循环里面改成了  
handles[i]=(HANDLE)\_beginthreadex(NULL,0,ThreadFunc,NULL,0,NULL);  
WaitForSingleObject(handles[i],INFINITE);  
去掉了WaitForMultipleObjects(THREAD\_NUM,handles,true,INFINITE);  
发现即使不用InterlockedIncrement(&log\_count);  
输出的结果也都是100。我对多线程不了解，不知道这样做对不对，请博主解释一下。谢谢。

Re: [MoreWindows](http://blog.csdn.net/MoreWindows) 2012-06-12 15:15发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

[http://avatar.csdn.net/C/F/7/3_morewindows.jpg](http://blog.csdn.net/MoreWindows)

回复无涯Andy：呵呵，你这样是一个线程执行完毕后再执行下一个线程，没有并发操作。之所以不能用100，是因为MSDN上对WaitForMultipleObjects()函数第一个参数作了说明：The maximum number of object handles is MAXIMUM\_WAIT\_OBJECTS（64）。现在你亲身试了下应该会印象深刻些了。

Re: [MoreWindows](http://blog.csdn.net/MoreWindows) 2012-06-14 15:20发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

[http://avatar.csdn.net/C/F/7/3_morewindows.jpg](http://blog.csdn.net/MoreWindows)

回复MoreWindows：肯定不会，我记得操作系统书上讲过多线程的执行是无序的，不可再现的。

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

楼主, 在release模式下, ++操作只有一句汇编代码:  
++g\_nLoginCount;   
008F100B inc dword ptr [g\_nLoginCount (8F3370h)]   
看上去应该算是原子操作了.  
但运行结果依然是有49, 48的.

Re: [MoreWindows](http://blog.csdn.net/MoreWindows) 2012-05-21 12:33发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

[http://avatar.csdn.net/C/F/7/3_morewindows.jpg](http://blog.csdn.net/MoreWindows)

回复ison\_jaro：我对汇编不是很了解，查了下资料，inc虽然只是一条汇编语句，但处理器也会分（1）读操作数（2）计算（3）写计算结果这三步来运算的，所以在多CPU情况下仍然可以视为非原子操作。

[http://avatar.csdn.net/5/A/E/3_chence19871.jpg](http://blog.csdn.net/chence19871)

楼主，请教：  
以下多线程对int型变量x的操作，哪几个需要进行同步：   
A. x=y; B. x++; C. ++x; D. x=1;  
B和C语句需要访问x原来的值，这个好理解。A为什么要同步啊，想不明白~

Re: [MoreWindows](http://blog.csdn.net/MoreWindows) 2012-05-17 18:54发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

[http://avatar.csdn.net/C/F/7/3_morewindows.jpg](http://blog.csdn.net/MoreWindows)

回复夏夜de星星：bird57521说的对，另外用VC看汇编代码很方便的，建议你学习下。

Re: [bird57521](http://blog.csdn.net/bird57521) 2012-05-16 11:08发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

[http://avatar.csdn.net/0/B/1/3_bird57521.jpg](http://blog.csdn.net/bird57521)

回复夏夜de星星：调试的时候 alt+8,跟反汇编就明白啦，x = y；分两步：先把y的值mov到寄存器eax，再把eax mvo 到x。学习了！

Re: [coryee](http://blog.csdn.net/coryee) 2013-04-10 10:21发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

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

回复bird57521：每个线程都有自己的寄存器，所以最后应该不会改变到x的值。除非x的最后结果又被另一个线程覆盖掉，是这样的吗

12楼 [winnelc](http://blog.csdn.net/winnelc) 2012-05-03 17:24发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

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

超过64个线程就会出现异常状况

Re: [MoreWindows](http://blog.csdn.net/MoreWindows) 2012-05-03 18:53发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

[http://avatar.csdn.net/C/F/7/3_morewindows.jpg](http://blog.csdn.net/MoreWindows)

回复winnelc：对的，系统作了限制。

11楼 [lanzhengpeng2](http://blog.csdn.net/lanzhengpeng2) 2012-04-12 08:47发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

[](http://blog.csdn.net/lanzhengpeng2)

Interlocked系列最重要的不是++,--;而是double compare and swap系列函数----DCAS.  
你走得太快了

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

博主，问下，一般来说，在创建后就应该释放。如下：

**[cpp]** [view plaincopy](http://blog.csdn.net/morewindows/article/details/7429155)

1. h[i] = ::CreateThread(NULL, 0, ThreadFun, NULL, 0, NULL);
2. ::CloseHandle(h[i]):

但我用你的例子，不加释放，结果和你的一样，如果加上了CloseHandle，结果就会变的看不明白了，这个是为什么啊？

Re: [MoreWindows](http://blog.csdn.net/MoreWindows) 2012-04-10 12:19发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

[http://avatar.csdn.net/C/F/7/3_morewindows.jpg](http://blog.csdn.net/MoreWindows)

回复inkb\_hyc：你在调用CloseHandle(h[i])后又访问了这个h[i]。

Re: [inkb\_hyc](http://blog.csdn.net/inkb_hyc) 2012-04-10 13:08发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

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

回复MoreWindows：什么意思？访问了这个h又会执行ThreadFun么？

Re: [MoreWindows](http://blog.csdn.net/MoreWindows) 2012-04-10 14:29发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

[http://avatar.csdn.net/C/F/7/3_morewindows.jpg](http://blog.csdn.net/MoreWindows)

回复inkb\_hyc：不好意思，没说清楚。你看代码：  
24. HANDLE handle[THREAD\_NUM];   
25. for (i = 0; i < THREAD\_NUM; i++)   
26. handle = CreateThread(NULL, 0, ThreadFun, NULL, 0, NULL);   
27. WaitForMultipleObjects(THREAD\_NUM, handle, TRUE, INFINITE);   
因为第27句中用到了handle数组，所以你在第26句之后不能立即调用CloseHandle(handle);而是应该在第27句之后再调用。

Re: [inkb\_hyc](http://blog.csdn.net/inkb_hyc) 2012-04-10 16:20发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

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

回复MoreWindows：谢谢~。  
看明白了。

8楼 [夜影如歌](http://blog.csdn.net/scorpiuseol) 2012-04-09 23:37发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7429155#reply)

[http://avatar.csdn.net/A/D/A/3_scorpiuseol.jpg](http://blog.csdn.net/scorpiuseol)

volatile 这个修饰符，有什么特别么？用在这里有什么作用？

回复夜影如歌：显示告诉编译器，这个值不知道在什么地方就会改变，请不要删除- -

回复夜影如歌：jianshiku说的对，volatile int i就表示每次读取i的值都强制从内存中读取，可以防止编译器将它优化成将i的值保存到寄存器中，后面再访问i时直接从寄存器中读取。

InterlockedIncrement((LPLONG)&g\_nLoginCount);   
这句中的(LPLONG)如果去掉会有问题吗？

回复jianshiku：可以去掉，这个看个人习惯。

回复MoreWindows：我用VC++6.0，去掉后编译不过，楼主能解释下吗，你的blog很不错

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

100个线程为什么会出现0或大于100的结果，我试了最多64个线程能产生“比较正常”的结果，是不是跟系统的线程数限制有关？

回复ahery：是的，MSDN上对WaitForMultipleObjects()函数第一个参数作了说明：The maximum number of object handles is MAXIMUM\_WAIT\_OBJECTS（64）。不过亲身试试一定会印象深刻。

是因为这个吗？？？？DWORD nCount 句柄的数量 最大值为MAXIMUM\_WAIT\_OBJECTS（64）

回复merlin\_q：是的，MSDN上对WaitForMultipleObjects()函数第一个参数作了说明：The maximum number of object handles is MAXIMUM\_WAIT\_OBJECTS.