[秒杀多线程第八篇 经典线程同步 信号量Semaphore](http://blog.csdn.net/morewindows/article/details/7481609)

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)2012-05-03 09:30 58489人阅读 [评论](http://blog.csdn.net/morewindows/article/details/7481609#comments)(99) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/morewindows/article/details/7481609#report)

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

阅读本篇之前推荐阅读以下姊妹篇：

《[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)》

《[秒杀多线程第五篇经典线程同步关键段CS](http://blog.csdn.net/morewindows/article/details/7442639)》

《[秒杀多线程第六篇经典线程同步事件Event](http://blog.csdn.net/morewindows/article/details/7445233)》

《[秒杀多线程第七篇经典线程同步互斥量Mutex](http://blog.csdn.net/morewindows/article/details/7470936)》

前面介绍了[关键段CS](http://blog.csdn.net/morewindows/article/details/7442639)、[事件Event](http://blog.csdn.net/morewindows/article/details/7445233)、[互斥量Mutex](http://blog.csdn.net/morewindows/article/details/7470936)在经典线程同步问题中的使用。本篇介绍用信号量Semaphore来解决这个问题。

首先也来看看如何使用信号量，信号量Semaphore常用有三个函数，使用很方便。下面是这几个函数的原型和使用说明。

第一个 CreateSemaphore

函数功能：创建信号量

函数原型：

HANDLE CreateSemaphore(

  LPSECURITY\_ATTRIBUTES lpSemaphoreAttributes,

  LONG lInitialCount,

  LONG lMaximumCount,

  LPCTSTR lpName

);

函数说明：

第一个参数表示安全控制，一般直接传入NULL。

第二个参数表示初始资源数量。

第三个参数表示最大并发数量。

第四个参数表示信号量的名称，传入NULL表示匿名信号量。

第二个 OpenSemaphore

函数功能：打开信号量

函数原型：

HANDLE OpenSemaphore(

  DWORD dwDesiredAccess,

  BOOL bInheritHandle,

  LPCTSTR lpName

);

函数说明：

第一个参数表示访问权限，对一般传入SEMAPHORE\_ALL\_ACCESS。详细解释可以查看MSDN文档。

第二个参数表示信号量句柄继承性，一般传入TRUE即可。

第三个参数表示名称，不同进程中的各线程可以通过名称来确保它们访问同一个信号量。

第三个 ReleaseSemaphore

函数功能：递增信号量的当前资源计数

函数原型：

BOOL ReleaseSemaphore(

  HANDLE hSemaphore,

  LONG lReleaseCount,

  LPLONG lpPreviousCount

);

函数说明：

第一个参数是信号量的句柄。

第二个参数表示增加个数，必须大于0且不超过最大资源数量。

第三个参数可以用来传出先前的资源计数，设为NULL表示不需要传出。

注意：**当前资源数量大于0，表示信号量处于触发，等于0表示资源已经耗尽故信号量处于末触发。**在对信号量调用等待函数时，等待函数会检查信号量的当前资源计数，如果大于0(即信号量处于触发状态)，减1后返回让调用线程继续执行。一个线程可以多次调用等待函数来减小信号量。

最后一个 信号量的清理与销毁

由于信号量是内核对象，因此使用CloseHandle()就可以完成清理与销毁了。

在经典多线程问题中设置一个信号量和一个关键段。用信号量处理主线程与子线程的同步，用关键段来处理各子线程间的互斥。详见代码：

**[cpp]** [view plaincopy](http://blog.csdn.net/morewindows/article/details/7481609)

1. #include <stdio.h>
2. #include <process.h>
3. #include <windows.h>
4. **long** g\_nNum;
5. unsigned **int** \_\_stdcall Fun(**void** \*pPM);
6. **const** **int** THREAD\_NUM = 10;
7. //信号量与关键段
8. **HANDLE**            g\_hThreadParameter;
9. CRITICAL\_SECTION  g\_csThreadCode;
10. **int** main()
11. {
12. printf("     经典线程同步 信号量Semaphore\n");
13. printf(" -- by MoreWindows( http://blog.csdn.net/MoreWindows ) --\n\n");
15. //初始化信号量和关键段
16. g\_hThreadParameter = CreateSemaphore(NULL, 0, 1, NULL);//当前0个资源，最大允许1个同时访问
17. InitializeCriticalSection(&g\_csThreadCode);
19. **HANDLE**  handle[THREAD\_NUM];
20. g\_nNum = 0;
21. **int** i = 0;
22. **while** (i < THREAD\_NUM)
23. {
24. handle[i] = (**HANDLE**)\_beginthreadex(NULL, 0, Fun, &i, 0, NULL);
25. WaitForSingleObject(g\_hThreadParameter, INFINITE);//等待信号量>0
26. ++i;
27. }
28. WaitForMultipleObjects(THREAD\_NUM, handle, TRUE, INFINITE);
30. //销毁信号量和关键段
31. DeleteCriticalSection(&g\_csThreadCode);
32. CloseHandle(g\_hThreadParameter);
33. **for** (i = 0; i < THREAD\_NUM; i++)
34. CloseHandle(handle[i]);
35. **return** 0;
36. }
37. unsigned **int** \_\_stdcall Fun(**void** \*pPM)
38. {
39. **int** nThreadNum = \*(**int** \*)pPM;
40. ReleaseSemaphore(g\_hThreadParameter, 1, NULL);//信号量++
42. Sleep(50);//some work should to do
44. EnterCriticalSection(&g\_csThreadCode);
45. ++g\_nNum;
46. Sleep(0);//some work should to do
47. printf("线程编号为%d  全局资源值为%d\n", nThreadNum, g\_nNum);
48. LeaveCriticalSection(&g\_csThreadCode);
49. **return** 0;
50. }

运行结果如下图：



可以看出来，信号量也可以解决线程之间的同步问题。

由于信号量可以计算资源当前剩余量并根据当前剩余量与零比较来决定信号量是处于触发状态或是未触发状态，因此信号量的应用范围相当广泛。本系列的《秒杀多线程第十篇 生产者消费者问题》将再次使用它来解决线程同步问题，欢迎大家参阅。

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

网上搜索了一下,感觉对理解信号量挺有帮助的:  
以一个停车场的运作为例。简单起见，假设停车场只有三个车位，一开始三个车位都是空的。这时如果同时来了五辆车，看门人允许其中三辆直接进入，然后放下车拦，剩下的车则必须在入口等待，此后来的车也都不得不在入口处等待。这时，有一辆车离开停车场，看门人得知后，打开车拦，放入外面的一辆进去，如果又离开两辆，则又可以放入两辆，如此往复。  
在这个停车场系统中，车位是公共资源，每辆车好比一个线程，看门人起的就是信号量的作用。  
抽象的来讲，信号量的特性如下：信号量是一个非负整数（车位数），所有通过它的线程/进程（车辆）都会将该整数减一（通过它当然是为了使用资源），当该整数值为零时，所有试图通过它的线程都将处于等待状态。在信号量上我们定义两种操作： Wait（等待） 和 Release（释放）。当一个线程调用Wait操作时，它要么得到资源然后将信号量减一，要么一直等下去（指放入阻塞队列），直到信号量大于等于一时。Release（释放）实际上是在信号量上执行加操作，对应于车辆离开停车场，该操作之所以叫做“释放”是因为释放了由信号量守护的资源。

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

**[cpp]** [view plaincopy](http://blog.csdn.net/morewindows/article/details/7481609)

1. **HANDLE** CreateSemaphore(
2. LPSECURITY  ATTRIBUTES lpSemaphoreAttributes,  //安全属性
3. **LONG** lInitialCount，                           //信号量对象的初始值
4. **LONG** lMaximumCount，                           //信号量的最大值
5. **LPCTSTR**  lpName                                //信号量名
6. ）；
7. 参数说明：
8. （1）lpSemaphoreAttributes：指定安全属性，为NULL时，信号量得到一个
9. 默认的安全描述符。
10. （2） lInitialCount：指定信号量对象的初始值。该值必须大于等于0，小于等于lMaximumCount。当其值大于0时，信号量被唤醒。当该函数释放了一个等待该信号量的线程时，lInitialCount值减1，当调用函数ReleaseSemaphore（）时，按其指定的数量加一个值。
11. （3） lMaximumCount：指出该信号量的最大值，该值必须大于0。
12. （4） lpName：给出信号量的名字。
13. 返回值：
14. 信号量创建成功，将返回该信号量的句柄。如果给出的信号量名是系统已经存在的信号量，将返回这个已存在信号量的句柄。如果失败，系统返回NULL，可以调用函数GetLastError（）查询失败的原因。

28楼 [亚细亚](http://blog.csdn.net/yaxiya) 2013-06-04 14:08发表 [[回复]](http://blog.csdn.net/morewindows/article/details/7481609#reply)

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

俺应该好好品味一下博主讲解的:Mutex,Event,Semaphore之间的区别;特别是ReleaseMutex()与ReleaseSemaphore()之间的区别,他们与等待函数WaitForSingleObject()之间的关系,事件对象中的触发与置位(包括手动与自动).哦...这些知识点掌握了,受益匪浅啊...!

回复亚细亚：mutex的release就是把资源从本线程释放出去，再让其他线程去争抢，releasesemaphore则是通知waitforxxobject的线程可以运行了~

[http://avatar.csdn.net/8/9/9/3_zhushangshan.jpg](http://blog.csdn.net/zhushangshan)

老师您好，我现在在进行多线程编程的学习，有个问题我一直没有调出来，就是两个子线程交替输出，同样的输入时输出不稳定，麻烦您帮忙看一下吧，我把代码的链接贴出来了。  
http://topic.csdn.net/u/20121016/19/1e81cd1d-ca6b-4ad4-a7ac-f5a1b62d4eac.html?85388

[](http://blog.csdn.net/liuxy1987)??????????确实是个问题

当ReleaseSemaphore过程中，可用资源数大于最大资源数的时候，ReleaseSemaphore并不会阻塞自己，貌似是返回了0表示错误？其实当当前资源数已经达到最大资源数的时候，ReleaseSemaphore是不是应该阻塞自己才比较合理？

[](http://blog.csdn.net/zhaoabc139)

windows老师，对于八，请教几个问题：  
1）子线程fun中进入关键区后的sleep（0）这是延迟吗，能不写吗？  
2）关键区可以换成Mutex来保证子线程之间的互斥吧？  
3）如果用Semaphor+Mutex来解决生产者与消费者问题，是不是要将fun变为一个producer和一个consumer子线程，然后再主线程中加入生成两个子线程的语句？

回复zhaoabc139：1。sleep()只是用来模拟一些费时的操作，可以去掉。  
2。关键段和互斥量都可以用于线程的互斥，因此在本例中可以用互斥量来代替关键段。  
3。对的，肯定要有生产者线程和消费者线程这二个功能上不一样的线程。《秒杀多线程第十篇 生产者消费者问题》将会有详细介绍，欢迎继续参阅。

[http://avatar.csdn.net/2/A/B/3_sy5837387.jpg](http://blog.csdn.net/sy5837387)

回复MoreWindows：LZ您好 我想请问下 主线程里面的WaitForSingleObject函数是不是还起到使信号量-1的作用？不然信号量都是在子线程里面自增而没有减少过？

回复sy5837387：对的，如果信号量为0，主线程调用WaitForSingleObject进入阻塞，直到有子线程调用ReleaseSemaphore后。主线程才能继续执行。

勿在浮沙筑高台。