事件（Event）是WIN32提供的最灵活的线程可同步方式，根据状态变迁方式的不同，事件可分为两类

手动设置：这种对象只可能用程序手动设置在需要该事件或者事件发生时，采用 SetEvent及ResetEvent来进行设置。

自动恢复：一旦事件发生并被处理后，自动恢复到没有事件状态，不需要再次设置。

**1. CreateEvent()**

[CreateEventA function (synchapi.h) - Win32 apps | Microsoft Docs](https://docs.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-createeventa)

HANDLE CreateEventA(

LPSECURITY\_ATTRIBUTES lpEventAttributes,

BOOL bManualReset,

BOOL bInitialState,

LPCSTR lpName

);

**2. SetEvent**

[SetEvent function (synchapi.h) - Win32 apps | Microsoft Docs](https://docs.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-setevent)

BOOL SetEvent(

HANDLE hEvent

);

**3. WaitForSingleobject()**

[WaitForSingleObject function (synchapi.h) - Win32 apps | Microsoft Docs](https://docs.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-waitforsingleobject)

DWORD WaitForSingleObject(

HANDLE hHandle,

DWORD dwMilliseconds

);

该函数需要传递一个内核对象句柄，如果该内核对象处于未通知状态，则该函数导致线程进入阻塞状态；如果该内核对象处于已通知状态，则该函数立即返回 WAIT\_OBJECT()第二个参数指明要等待的时间（毫秒），INFINITE表示无限等待，如果第二个参数为0，那么函数立即返回。如果等待超时，该函数返 WAIT\_TIMEOUT如果该函数失败，返回 WAIT\_FAILED。

该函数需要传递一个内核对象句柄，如果该内核对象处于未通知状态，则该函数导致线程进入阻塞状态；如果该內核对象处于已通知状态，则该函数立即返回 WAIT\_OBJECT()。第二个数指明要等待的时间（毫秒），INFINITE表示无限等待，如果第二个参数为0，那么函数立即返回。如果等待超时，该函数返 WAIT\_TIMEOUT。  
如果该函数失败，返回 WAIT\_FAILED

**4. CloseHandle()**

[CloseHandle function (handleapi.h) - Win32 apps | Microsoft Docs](https://docs.microsoft.com/en-us/windows/win32/api/handleapi/nf-handleapi-closehandle)

BOOL CloseHandle(

HANDLE hObject

);

**5. 示例演示**

#include<process.h>

#include<windows.h>

#include<stdio.h>

void \_\_cdecl SellThread1(void\* param);

void \_\_cdecl SellThread2(void\* param);

//100张票

int tickets = 100;

HANDLE hEvent = INVALID\_HANDLE\_VALUE;

int main()

{

//创建事件，此刻为有信号状态

//自动重置信号状态, 初始化为有信号状态，线程可以直接获取

hEvent = CreateEvent(NULL, FALSE, TRUE ,L"事件对象");

Sleep(1000);

//主线程休眠3秒之后，将信号量设置为无信号状态

//ResetEvent(hEvent);

printf("开始卖票了！\n");

//创建两个售票窗口

uintptr\_t t1 = \_beginthread(SellThread1, 0, "售口窗口A");

uintptr\_t t2 = \_beginthread(SellThread2, 0, "售口窗口B");

//无限等待两个线程全部执行完毕

HANDLE hArr[] = { (HANDLE)t1, (HANDLE)t2 };

WaitForMultipleObjects(2, hArr, true, INFINITE);

printf("卖票结束！\n");

return 0;

}

void \_\_cdecl SellThread1(void\* param)

{

char \*name = (char \*)param;

while (tickets>0)

{

//如果事件对象为有信号状态（没有线程拥有它），则线程可以获取它后继续执行

//自动重置的事件对象,调用了WaitForSingleObject函数之后，自动重置为无信号

WaitForSingleObject(hEvent, INFINITE);

if (tickets > 0)

{

Sleep(10);

//CPU恰好执行到这里，这个时候线程时间片到了，并且此时还剩最后一张票

printf("%s卖出第%d张票！\n", name, tickets--);

}

//SetEvent让事件对象变成有信号状态

SetEvent(hEvent);

}

}

void \_\_cdecl SellThread2(void\* param)

{

char \*name = (char \*)param;

while (tickets > 0)

{

//如果事件对象为有信号状态（没有线程拥有它），则线程可以获取它后继续执行

//自动重置的事件对象,调用了WaitForSingleObject函数之后，自动重置为无信号

WaitForSingleObject(hEvent, INFINITE);

if (tickets > 0)

{

Sleep(10);

//CPU恰好执行到这里，这个时候线程时间片到了，并且此时还剩最后一张票

printf("%s卖出第%d张票！\n", name, tickets--);

}

//SetEvent让事件对象变成有信号状态

SetEvent(hEvent);

}

}