WSACreateEvent

创建一个初始状态为失信的匿名的需要手动重置的[事件对象](http://baike.baidu.com/view/2309178.htm" \t "_blank)。

**#include <winsock2.h>**

**WSAEVENT WSAAPI WSACreateEvent( VOID );**

**返回值**：

如果函数成功，则返回值即是[事件对象](http://baike.baidu.com/view/2309178.htm" \t "_blank)的句柄。

如果函数失败，返回WSA\_INVALID\_EVENT。应用程序可通过调用WSAGetLastError()函数获取进一步的[错误信息](http://baike.baidu.com/view/627766.htm" \t "_blank)。

[错误代码](http://baike.baidu.com/view/1600019.htm" \t "_blank)：

WSANOTINITIALISED 在调用本API之前应成功调用WSAStartup()。

WSAENETDOWN 网络子系统失效。

WSA\_NOT\_ENOUGH\_MEMORY 无足够内存创建[事件对象](http://baike.baidu.com/view/2309178.htm" \t "_blank)。

WSAEventSelect

调用[WSAEventSelect()](http://baike.haosou.com/doc/710997-752637.html" \t "_blank)函数，将一个[事件对象](http://baike.haosou.com/doc/2074876-2194904.html" \t "_blank)与网络事件集合关联在一起。当网络事件发生时，应用程序以事件的形式接收网络事件通知。

**int WSAEventSelect(**

**SOCKET s,  // 定义的套接字**

**WSAEVENT hEventObject, // 与套接字关联在一起的事件对象句柄**

**long lNetworkEvent // 应用程序感兴趣的各种网络事件类型的一个组合**

**//(FD\_READ|FD\_WRITE|FD\_ClOSE)**

   );

WSAResetEvent()

WSAResetEvent()，计算机函数。功能为将指定的[事件对象](http://baike.baidu.com/view/2309178.htm" \t "_blank)状态重新设置为未置信号。

**#include <winsock2.h>**

**BOOL WSAAPI WSAResetEvent( WSAEVENT hEvent );**

hEvent：标识一个开放的[事件对象](http://baike.baidu.com/view/2309178.htm" \t "_blank)句柄。

返回值：

如果函数成功，返回真TRUE。如果函数失败，返回假FALSE。可调用WSAGetLastError()来获取进一步的[错误信息](http://baike.baidu.com/view/627766.htm" \t "_blank)。

[错误代码](http://baike.baidu.com/view/1600019.htm" \t "_blank)：

WSA\_INVALID\_HANDLE hEvent不是一个合法的[事件对象](http://baike.baidu.com/view/2309178.htm" \t "_blank)句柄。

WSAWaitForMultipleEvents

DWORD WSAWaitForMultipleEvents(

          DWORD cEvents,  //指定的是事件对象的数量，最大值64个

          const WSAEVENT FAR \* lphEvents, //指针，用于直接引用该数组

          BOOL fWaitAll, // WSAWaitForMultipleEvents如何等待在事件数组中的对象，

//若设为TRUE，那么只有等lphEvents数组内包含的所有事件对象都已进

//入“已传信”状态，函数才会返回；

//但若设为FALSE，任何一个事件对象进入“已传信”状态，函数就会返回。

          DWORD dwTimeout, //最多可等待一个网络事件发生有多长时间，以毫秒为单位

          BOOL fAlertable //该参数主要用于在重叠式I/O模型中，在完成例程的处理过程中使用。

     );

例：

nIndex = WSAWaitForMultipleEvents(......);

    hEvent = hEventArray[nIndex - WSA\_WAIT\_EVENT\_0];

   知道了造成网络事件发生的套接字后，调用WSAEnumNetworkEvents函数，调查发生了什么类型的网络事件，定义如下：

WSAEnumNetworkEvents

 int WSAEnumNetworkEvents(

       SOCKET s,  // 网络事件发生相关联的套接字

       WSAEVENT hEventObject, //重设事件对象，将处在“已传信”改为“未传信”，亦

//WSAResetEvent替代

       LPWSAMETWORKEVENTS lpNetworkEvents // **接收套接字上发生的网络事件类型以及可能**

**//出现的错误代码**

   );

typedef struct \_WSANETWORKEVENTS {

          long lNetworkEvent; //指定了一个值，对应于套接字上发生的所有网络事件类型

          int iErrorCode[FD\_MAX\_EVENTS]; //指定的是一个错误代码数组

    }WSANETWORKEVENTS, FAR \* LPWSANETWORKEVENTS;

 例：

  if (NetworkEvents.lNetworkEvents & FD\_READ) {

        if (NetworkEvents.iErrorCode[FD\_READ\_BIT] != 0) {

        }

    }

例：

 case FD\_WRITE:  // 可以发送数据了  
   {   
      // 进入无限循环  
      while(TRUE)  
      {  
          // 从文件中读取数据，保存到packet.data里面.  
          in.read((char\*)&packet.data，MAX\_PACKET\_SIZE);

          // 发送数据  
          if (send(wparam, (char\*)(&packet), sizeof(PACKET), 0) == SOCKET\_ERROR)  
          {  
              if (WSAGetLastError() == WSAEWOULDBLOCK)  
              {  
                  // 发送缓冲区已经满了, 退出循环.  
                  break;  
              }  
              else // 其他错误  
              {  
                 // 显示出错信息然后退出.  
                 CleanUp();  
                 return(0);  
              }  
           }  
        }  
     } break;