判断非阻塞SOCKET是否已经断开的几种方法

注意要区分不同操作系统分别进行测试， 包括WINDOWS, LINUX和UNIX会各有不同

在WINDOWS下比较简单，可以使用FD\_CLOSE事件判断SOCKET是否已经断开

bool IsSocketClosed(SOCKET clientSocket)  
{  
 bool ret = false;  
 HANDLE closeEvent = WSACreateEvent();  
 WSAEventSelect(clientSocket, closeEvent, FD\_CLOSE);

 DWORD dwRet = WaitForSingleObject(closeEvent, 0);  
   
 if(dwRet == WSA\_WAIT\_EVENT\_0)  
  ret = true;  
 else if(dwRet == WSA\_WAIT\_TIMEOUT)  
  ret = false;

 WSACloseEvent(closeEvent);  
 return ret;  
}

在UNIX/LINUX下，非阻塞模式SOCKET可以采用recv+MSG\_PEEK的方式进行判断，其中MSG\_PEEK保证了仅仅进行状态判断，而不影响数据接收

对于主动关闭的SOCKET, recv返回-1，而且errno被置为9（#define EBADF   9 /\* Bad file number \*/）

或104 （#define ECONNRESET 104 /\* Connection reset by peer \*/）

对于被动关闭的SOCKET,recv返回0，而且errno被置为11（#define EWOULDBLOCK EAGAIN /\* Operation would block \*/）

对正常的SOCKET, 如果有接收数据，则返回>0, 否则返回-1，而且errno被置为11（#define EWOULDBLOCK EAGAIN /\* Operation would block \*/）

因此对于简单的状态判断（不过多考虑异常情况），  
    recv返回>0，   正常  
    返回-1，而且errno被置为11  正常  
    其它情况    关闭

#include <errno.h>

bool IsSocketClosed(int clientSocket)  
{  
 char buff[32];  
 int recvBytes = recv(clientSocket, buff, sizeof(buff), MSG\_PEEK);  
   
 int sockErr = errno;  
   
 //cout << "In close function, recv " << recvBytes << " bytes, err " << sockErr << endl;  
   
 if( recvBytes > 0) //Get data  
  return false;  
   
 if( (recvBytes == -1) && (sockErr == EWOULDBLOCK) ) //No receive data  
  return false;  
     
 return true;  
}

此外，所有操作系统上还可以通过TCP\_KEEPLIVE实行心跳检测