套接字函数
（Socket API）

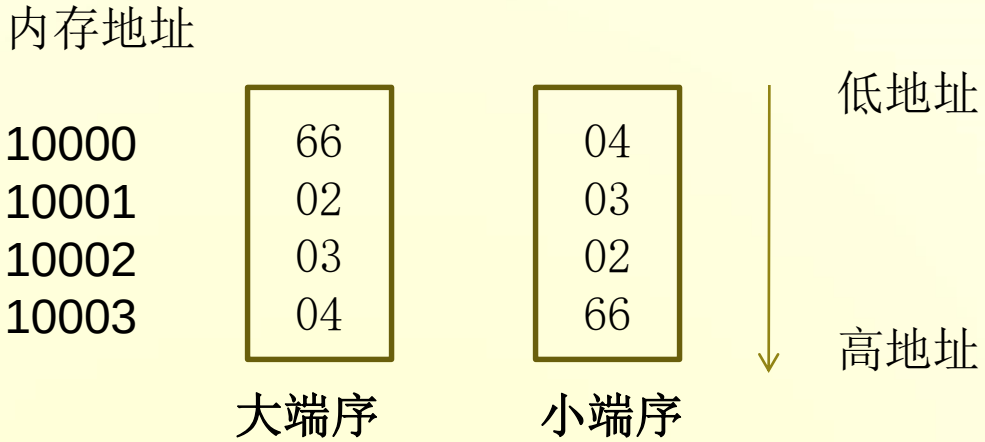
https://docs.microsoft.com/zh-cn/windows/desktop/api/winsock2/
https://www.cnblogs.com/hgwang/p/6074038.html

中山大学 计算机系 张永民
2019年3月7日

网络字节序函数

- 多字节数在内存有**大端序**(big-endian)和**小端序**(little-endian)，为**主机字节序**(host byte order)。

0x66020304存放
在内存的两种次序：

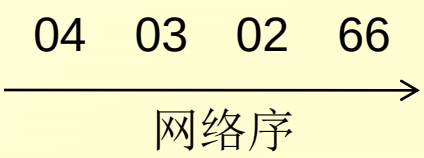


Motorola的微处理器
IBM大型机

Intel CPU
大部分小型机

- 因特网的**网络字节序**(network byte order)采用大端序，即先发送高位字节。

发送0x66020304



- 当把多字节数从主机(host)发往网络(network)或从网络(network) 发往主机(host)时，就需要用函数进行转换，以免出现顺序错误.
- 函数**ntohs**(u_short)把一个16位数从网络字节序转换到主机字节序。
  函数**htons**(u_short)把一个16位数从主机字节序转换到网络字节序。
- 函数格式:

```
u_short htons(u_short host)
u_short ntohs(u_short net)
u_long htonl(u_long host)  //32位数，主机到网络
u_long ntohl(u_long net)   //32位数，网络到主机
```

地址结构

- sockaddr结构

```
// sockaddr表示一般地址，与sockaddr_in字节数相同，可以相互转换。
struct  sockaddr {
    ushort sa_family;            //地址簇
    char sa_data[14];            // 地址
};
```

- sockaddr_in结构

```
//用来存放因特网的IP地址和端口号
struct   sockaddr_in {
        short sin_family;
        u_short sin_port;            //端口号
        struct in_addr sin_addr;     //IP地址
        char sin_zero[8];
};
```

IP地址表示

```
//IP地址是32位数，采用点分十进制数表示，例如，66.2.3.4。
//IP地址的3种表示方法：四个字节，两个字，一个32位整数
typedef  struct in_addr {
      union {
            struct {
                  u_char s_b1,s_b2,s_b3,s_b4;   //四个字节
            } s_un_b;
            struct {
                  u_short s_w1,s_w2;                  //两个字
            } s_un_w;
            u_long s_addr;                            //一个32位整数
      } S_un;
} IN_ADDR, *PIN_ADDR, FAR *LPIN_ADDR;
```

■ 地址转换函数

(1) atoi把字符串表示的数字(如，"80")转换成整数。

int atoi(const char *nptr);

(2) inet_addr把点分十进制的IP地址(例，"192.168.1.1")转化为32位IP地址。

unsigned long inet_addr(const char* cp)

(3) inet_ntoa把32位IP地址转化为点分十进制的IP地址。

char* FAR inet_ntoa(struct in_addr in)

函数gethostbyname

- gethostbyname函数把域名(如，www.sysu.edu.cn)映射成IP地址(202.116.64.9)。

```
struct hostent FAR * gethostbyname(
        const char FAR*name;              /* 主机名或域名,如, www.sohu.com  */
)
```

- **hostent结构**

```
struct  hostent {
    char    FAR * h_name;              /* 主机官方名或域名, 如, www.sohu.com */
    char    FAR * FAR * h_aliases;     /* alias list, null-terminated */
    short   h_addrtype;                /* host address type */
    short   h_length;                  /* length of address */
    char    FAR * FAR * h_addr_list;   /* list of addresses */
   #define h_addr  h_addr_list[0]      /* 获得IP地址(网络字节序) */
};
```

- getaddrinfo已取代gethostbyname。用getpeername可以已连接套接字获得对方地址，getsockname获得本地地址。 int getpeername(SOCKET s, struct sockaddr* name, int* namelen)

函数getservbyname

- **getservbyname**把服务名解释为端口号

```
struct servent FAR  * getservbyname(          80        13        23
    const char FAR * name;          /* 服务名，如, HTTP, DayTime; Telnet */
    const char FAR * proro;         /*  协议(可选), 如，udp, tcp */
)
struct  servent {
      char    FAR * s_name;          /* official service name */
      char    FAR * FAR * s_aliases; /* alias list */
      char    FAR * s_proto;         /* protocol to use */
      short   s_port;                /* 端口号 */
};
```

函数getprotobyname

- 把协议名转化为协议号

```
struct protoent FAR  * getprotobyname(
     const char FAR * name;          /* 协议名，如, udp, tcp */
)                                                    17，6

struct  protoent {
        char    FAR * p_name;           /* official protocol name */
        char    FAR * FAR * p_aliases;  /* alias list */
        short   p_proto;                /* 协议号17(udp), 6(tcp)*/
};
```

SOCKET socket(int af, int type, int protocol)

功能： 建立一个绑定到特定服务提供者的套接字。

参数： 地址簇(Address Family)或协议簇(Protocol Family)
套接字类型
地址簇中的协议号

返回： 要监听的套接字的描述符或INVALID_SOCKET。出错时可以采用GetLastError()取到错误码。

参数取值：

af: AF_UNSPEC(未定义)，AF_INET(IPv4)，AF_INET6(IPv6)
*与PF_开头相同

type: SOCK_STREAM(流式)，SOCK_DGRAM(数据报)
SOCK_RAW(原始)

protocol: IPPROTO_TCP(TCP的协议号6)
IPPROTO_UDP(UDP的协议号17)
IPPROTO_ICMP(ICMP的协议号1)

- int bind(SOCKET s,
  const struct sockaddr* name,
  int namelen);

功能：把一个本地地址(IP地址和端口号)与套接字关联。

参数

s	一个未绑定的套接字描述符
name	一个绑定到套接字的本地地址(地址簇，本地IP地址，本地端口号)。*参数INADDR_ANY绑定所有地址。
namelen	地址长度（字节数）

返回值

无错误，返回0，否则，返回SOCKET_ERROR，调用函数WSAGetLastError取得错误代码。

说明

(1) 如果绑定的端口号为0，系统从1024 - 5000分配一个唯一的端口号。

(2) 函数getsockname可以在连接以后获得绑定的地址和端口号。不鼓励客户端绑定端口号，因为可能会发生冲突。

(3) SO_EXCLUSIVEADDR(独占绑定) 或者 SO_REUSEADDR(可重用)选项要在bind之前执行才有效。

int listen(SOCKET s, int backlog)

功能：把一个本地地址与套接字关联。

参数

s　　一个已绑定未被连接的套接字描述符

backlog　连接请求队列(**queue of pending connections**)的最大长度(一般取值2到 4) 。用SOMAXCONN则由系统确定。

返回值

无错误，返回0，否则，返回SOCKET_ERROR，可以调用函数WSAGetLastError取得错误代码。

说明

(1) 执行listen之后套接字进入被动模式。

(2) 队列满了以后，将拒绝新的连接请求。客户端将出现连接错误WSAECONNREFUSED。

(3) 在正在listen的套接字上执行listen不起作用。

例子：
```
if (listen( ListenSocket, 3) == SOCKET_ERROR)
    printf("Error listening on socket: %d。\n", WSAGetLastError());
```

SOCKET accept(SOCKET s, struct sockaddr* addr, int* addrlen)

功能： 接受到来的一个套接字的连接请求。

参数

s	一个处于监听状态的套接字描述符（in）
addr	(可选)指向存放请求连接的客户端地址（out）
addrlen	(可选)指向存放上述地址长度(字节数)的整数和返回实际地址长度(inout)

返回值

无错时，返回新套接字描述符，否则，返回INVALID_SOCKET，可以调用函数 WSAGetLastError取得错误代码。

说明

(1) accept函数从连接请求队列抽取第一个连接，并返回新套接字，其特性与s相同。新套接字用来处理实际连接。

(2) 如果该套接字的请求连接的队列为空并且套接字设为阻塞方式，accept函数将阻塞调用者， 套接字被标志为阻塞转态。如果该队列为空且设置为 非阻塞状态，WSAGetLastError()返回错误代码WSAEWOULDBLOCK。

(3) 如果返回的addr或addrlen为NULL，则表示没有远端地址返回。

- int connect(SOCKET s, const struct sockaddr* name, int namelen);

功能： 建立到一个指定套接字的连接，并指明远端地址(服务器IP和端口号)。

参数：

s	一个未连接的套接字描述符（in）
name	指向存放请求连接的服务器端地址（in）
namelen	存放上述地址长度(字节数)(in)

返回值：

无错时，返回0，

否则，返回SOCKET_ERROR，可以调用函数 WSAGetLastError取得错误代码。

说明：

(1) connect用name来指明远端地址。如果s没有绑定，则由系统自动分配本地地址，并进入绑定状态。

(2) 对于面向连接无阻塞套接字，如果不能立即连接，则返回错误WSAEWOULDBLOCK，然后继续向前执行。使用select时，会通知select的writefd。

(3) 对于无连接套接字，connect将设置默认目的地址供以后send和recv使用。接收的任何来自非默认目的地址的数据报将被丢弃。sendto 和 recvfrom 总是可以使用，因为它们自带目的地址。

(4) 对于无连接套接字，*name可以指出任何有效地址，包括广播地址*。可是，为了连接到一个广 播地址，必须用 setsockopt设置SO_BROADCAST选项有效，否则，connect将返回错误 WSAEACCES。

(5) bind可以用来指定客户端口号。

举例：

```
// sockaddr_in 结构说明了要连接的地址簇、IP地址和端口号
   sockaddr_in clientService;
   clientService.sin_family = AF_INET;
   clientService.sin_addr.s_addr = inet_addr( "127.0.0.1" );
   clientService.sin_port = htons( 27015 );
   if ( connect( ConnectSocket, (SOCKADDR*) &clientService,
                        sizeof(clientService) ) ==SOCKET_ERROR) {
       printf( "Failed to connect.\n" );
       WSACleanup();   return;}
```

- int send(SOCKET s, const char* buf, int len,int flags)

功能： 在一个已连接套接字上发送数据。

参数

s	一个已连接的套接字描述符（in）
buf	指向要传送数据的缓冲区 （in）
len	上述缓冲区的长度(字节数) (in)
flags	指明发送方式： MSG_DONTROUTE --不进行路由，0 --忽略该选项，MSG_OOB – 用于发送带外数据。

返回值

无错时，返回实际发送的字节数(不大于len), 否则，返回SOCKET_ERROR ，用函数 WSAGetLastError取得错误代码。

说明

(1) 对于面向消息的套接字(AF_INET + SOCK_DGRAM + IPPROTO_UDP)，注意不要超过提供者最大包的大小。 用参数SO_MAX_MSG_SIZE调用getsockopt可以得到该最大值。如果太长， 将返回错误WSAEMSGSIZE，而且不会发出任何数据。

(2) send的成功只是数据被成功发出，并不意味数据已经被接收方成功接收。如果传输层没有缓冲区空间放置要发送的数据，send将被阻塞，除非采用的是非阻塞套接字。对非阻塞面向流的套接字，send实际所写字节数取决于客户和服务器的缓冲区空闲块大小。select函数可以用来确定是否可以发送更多的数据。

(3) len为0是允许的，对于面向流的模式，立即执行并返回成功，而对于面向消息的套接字，将发送数据部分为空的数据段。

(4) flags 参数一般取0值。

```
//---------------------
// Send an initial buffer
iResult = send( ConnectSocket, sendbuf, (int)strlen(sendbuf), 0 );
if (iResult == SOCKET_ERROR) {
    printf("send() failed with error: %d\n", WSAGetLastError());
    closesocket(ConnectSocket);
    WSACleanup();
    return 1;
}
printf("Bytes Sent: %d\n", iResult);
```

➢int *recv(SOCKET s, char* buf, int len, int flags);*

功能： 从已连接套接字或已绑定无连接套接字。

参数

s	一个标识的已连接套接字的描述符（in）
buf	指向要接收数据的缓冲区 （out）
len	上述缓冲区的长度(字节数) (in)
flags	影响接收方式的标志 (in)

返回：

无错时，返回实际接收的字节数(不大于len)，缓冲区中包含所接收的数据。如果连接关闭，将返回0。

否则，返回SOCKET_ERROR，可以调用函数 WSAGetLastError取得错误代码。

说明

(1) 用于无连接套接字时必须先执行bind，用于面向连接的套接字时必须先建立连接。

(2) 必须知道套接字的本地地址。服务器采用bind和accept获得本地地址。客户端在connect或sendto时由系统自动分配。客户端不鼓励bind本地地址。

(3)recv只接收具有指定远端地址的报文，到达绑定端口的其它报文被默默地丢弃。

(4)对于面向连接的套接字，调用recv将返回尽可能多的数据，但不会超出缓冲区大小。

(5)对于无连接的套接字，数据从connect函数指定的目的地址的队列的第一个数据报中取出。如果报文太大，缓冲区装不下，则只取报文的前面部分并返回错误WSAEMSGSIZE。对于不可靠协议，多出部分将被抛弃。对于可靠协议，服务提供者(如TCP)将保留剩余部分直到被下一次recv取走。

(6)如果没有数据，recv会被阻塞，直到数据到达。

(7)如果套接字是面向连接的，远端已经正常shutdown，所有数据已经接收完，则recv用0字节返回。如果连接被复位，则recv返回SOCKET_ERROR，并设置错误WSAECONNRESET。

(8)flags默认取值为0。

➢int closesocket(SOCKET s);

功能： 关闭套接字，释放其资源。

参数

s　　一个套接字的描述符（in）

返回：

无错时，将返回0。

否则，返回SOCKET_ERROR ，可以调用函数 WSAGetLastError取得错误代码。

说明：

如果这是最后一个对基本套接字的引用，所关联的命名信息和排队队列将被丢弃。进程中任何线程所发布的任何异步调用将被取消而不会发出通告消息。

- **int sendto(SOCKET s, const char* buf,**
  int len, int flags,
  const struct sockaddr* to, int tolen);

功能： 发送数据到指定地址。

参数

s	一个已标识的（可能已连接的）套接字描述符（in）
buf	指向要传送数据的缓冲区 （in）
len	上述缓冲区的长度(字节数) (in)
flags	指明发送方式 (in)，与send用法相同。
to	一个可选指针，指向包含目标套接字地址的sockaddr结构
tolen	to所指向结构的大小(字节数)。

返回：

无错时，返回实际发送的字节数(不大于len) 。

否则，返回SOCKET_ERROR，可以调用函数 WSAGetLastError取得错误代码。

➢int recvfrom(SOCKET s, char* buf, int len, int flags, struct sockaddr* from, int* fromlen)

功能：从已连接套接字或已绑定无连接套接字接收数据。

参数：

s	一个标识的已绑定套接字的描述符（in）
buf	指向要接收数据的缓冲区 （out）
len	上述缓冲区的长度(字节数) (in)
flags	影响接收方式的标志 (in)
from	指向存放源地址(sockaddr结构)的缓冲区。(可选)
fromlen	指向上述源地址的大小(字节数)。 （可选）

返回：

无错时，返回实际接收的字节数(不大于len)，缓冲区中包含所接收的数据。如果连接关闭，将返回0。否则，返回SOCKET_ERROR ，可以调用函数 WSAGetLastError取得错误代码。

说明

(1)recvfrom函数可以读取有连接和无连接套接字上到来的数据，并捕捉发送数据一方的地址。recvfrom函数一般用于无连接套接字。在服务器上采用bind显式绑定本地地址，在客户机上用sendto隐式绑定本地地址。

(2)对于面向流的套接字(TCP)，调用recvfrom将返回尽可能多的数据，但不会超出缓冲区大小。对于面向消息的套接字(UDP)，数据从connect函数指定的目的地址的队列的第一个数据报中取出。如果报文太大，缓冲区装不下，则只取报文的前面部分并返回错误WSAEMSGSIZE。对于不可靠协议(UDP)，多出部分将被抛弃。对于是UDP，如果数据段的数据部分为空，返回值为0。

(4)如果没有数据，recvfrom会被阻塞，直到数据到达。如果设置了非阻塞模式，将会返回SOCKET_ERROR，并设置错误WSAEWOULDBLOCK。 select可以用来判断何时数据到达。

(5)如果套接字是无连接的，并且参数from非零，则返回时from将包含源地址。

(6)如果套接字是面向连接的，远端已经正常shutdown，所有数据已经接收完，则recvfrom用0字节返回。如果连接被复位，则recvfrom返回SOCKET_ERROR，并设置错误WSAECONNRESET。

(7)Flags可以取值0。其它取值为：

MSG_PEEK	读回数据，但是不从输入缓冲区中删除。
MSG_OOB	处理带外数据。
MSG_WAITALL	等待缓冲区满。

函数getsockname

格式：int PASCAL FAR getsockname(SOCKET s, struct sockaddr FAR* name, int FAR* namelen);

功能：取得一个已绑定或已连接套接字s的本地地址。

参数： s：标识一个已捆绑套接口的描述字。
name：接收套接口的地址（名字）。
namelen：名字缓冲区长度。

返回值：若无错误发生，getsockname()返回0。否则的话，返回SOCKET_ERROR错误，应用程序可通过WSAGetLastError()获取相应错误代码。

setsockopt

格式： int PASCAL FAR setsockopt(SOCKET s,int level,int optname, const char FAR *optval,int optlen);

参数：

s	标识一个套接字的描述符。
level	选项定义的层次；目前仅支持SOL_SOCKET和IPPROTO_TCP层次。
optname	需设置的选项。
optval	指针，指向存放选项值的缓冲区。
optlen	optval缓冲区长度。

返回值

若无错误发生，setsockopt（）返回0。否则的话，返回SOCKET_ERROR错误，应用程序可通过WSAGetLastError（）获取相应错误代码。

错误代码

WSANOTINITIALISED：在使用此API之前应首先成功地调用WSAStartup（）。

WSAENETDOWN：WINDOWS套接口实现检测到网络子系统失效。

WSAEFAULT：optval不是进程地址空间中的一个有效部分。

WSAEINPROGRESS：一个阻塞的WINDOWS套接口调用正在运行中。

WSAEINVAL：level值非法，或optval中的信息非法。

WSAENETRESET：当SO_KEEPALIVE设置后连接超时。

WSAENOPROTOOPT：未知或不支持选项。其中，SOCK_STREAM类型的套接口不支持SO_BROADCAST选项，SOCK_DGRAM类型的套接口不支持SO_DONTLINGER 、SO_KEEPALIVE、SO_LINGER和SO_OOBINLINE选项。

WSAENOTCONN：当设置SO_KEEPALIVE后连接被复位。

WSAENOTSOCK：描述字不是一个套接口。

SO_BROADCAST BOOL	允许套接口传送广播信息。
SO_DEBUG BOOL	记录调试信息。
SO_DONTLINER BOOL	不要因为数据未发送就阻塞关闭操作。设置本选项相当于将SO_LINGER的l_onoff元素置为零。
SO_DONTROUTE BOOL	禁止选径；直接传送。
SO_KEEPALIVE BOOL	发送“保持活动”包。
SO_LINGER struct linger FAR*	如关闭时有未发送数据，则逗留。
SO_OOBINLINE BOOL	在常规数据流中接收带外数据。
SO_RCVBUF int	为接收确定缓冲区大小。
SO_REUSEADDR BOOL	允许套接口和一个已在使用中的地址捆绑。
SO_SNDBUF int	指定发送缓冲区大小。
TCP_NODELAY BOOL	禁止发送合并的Nagle算法。

setsockopt（）不支持的BSD选项有：

SO_ACCEPTCONN BOOL	套接口在监听。
SO_ERROR int	获取错误状态并清除。
SO_RCVLOWAT int	接收低级水印。
SO_RCVTIMEO int	接收超时。
SO_SNDLOWAT int	发送低级水印。
SO_SNDTIMEO int	发送超时。
SO_TYPE int	套接口类型。
IP_OPTIONS	在IP头中设置选项。

1.对于UDP当一个IP地址和端口绑定到某个套接口上时，还允许此IP地址和端口捆绑到另一个套接口上。一般来说，这个特性仅在支持多播的系统上才有，而且只对UDP套接口而言（TCP不支持多播）。。

```
BOOL bReuseaddr = TRUE;
setsockopt(s,SOL_SOCKET,SO_REUSEADDR,(const
        char*)&bReuseaddr,sizeof(BOOL));
```

2. 如果要已经处于连接状态的socket在调用closesocket()后强制关闭，不经历TIME_WAIT的过程：

```
BOOL bDontLinger = FALSE;
setsockopt(s,SOL_SOCKET,SO_DONTLINGER,(const
        char*)&bDontLinger,sizeof(BOOL));
```

3.在send(),recv()过程中有时由于网络状况等原因，收发不能预期进行，可以设置收发时限：

```
int nNetTimeout = 1000; //1秒
setsockopt(socket,SOL_SOCKET,SO_SNDTIMEO,(char
        *)&nNetTimeout,sizeof(int)); //发送时限
setsockopt(socket,SOL_SOCKET,SO_RCVTIMEO,(char
        *)&nNetTimeout,sizeof(int)); //接收时限
```

4. 在send()的时候，返回的是实际发送出去的字节(同步)或发送到socket缓冲区的字节(异步)；系统默认的发送和接收缓冲区为8688字节；如果发送或是接收的数据量比较大，可以改变socket缓冲区的大小，减少send(),recv()的执行次数：

```
int nRecvBuf = 32 * 1024; //设置为32K
setsockopt(s,SOL_SOCKET,SO_RCVBUF,(const
    char*)&nRecvBuf,sizeof(int));  // 接收缓冲区
int nSendBuf = 32 * 1024; //设置为32K
setsockopt(s,SOL_SOCKET,SO_SNDBUF,(const
    char*)&nSendBuf,sizeof(int));  //发送缓冲区
```

5. 在发送数据的时候，不执行由系统缓冲区到socket缓冲区的拷贝，以提高程序的性能：

```
int nZero = 0;
setsockopt(socket,SOL_SOCKET,SO_SNDBUF,(char
                *)&nZero,sizeof(nZero));
```

6. 在接收数据时，不执行将socket缓冲区的内容拷贝到系统缓冲区：

```
int nZero = 0;
setsockopt(s,SOL_SOCKET,SO_RCVBUF,(char *)&nZero,sizeof(int));
```

7. 一般在发送UDP数据报的时候，希望该socket发送的数据具有广播特性：

```
BOOL bBroadcast = TRUE;
setsockopt(s,SOL_SOCKET,SO_BROADCAST,(const
                char*)&bBroadcast,sizeof(BOOL));
```

8. 在client连接服务器过程中，如果处于非阻塞模式下的socket在connect()的过程中可以设置connect（）延时，直到accpet（）被调用（此设置只有在非阻塞的过程中有显著的作用，在阻塞的函数调用中作用不大）

```
BOOL bConditionalAccept = TRUE;
setsockopt(s,SOL_SOCKET,SO_CONDITIONAL_ACCEPT,(const
                char*)&bConditionalAccept,sizeof(BOOL));
```

9. 如果在发送数据的过程中send()没有完成，还有数据没发送，而调用了closesocket()，以前一般采取的措施是shutdown(s,SD_BOTH),但是数据将会丢失。

```
struct linger {
  u_short l_onoff;
  u_short l_linger;
};
linger m_sLinger;
m_sLinger.l_onoff = 1; //在调用closesocket()时还有数据未发送完，
                       允许等待。若m_sLinger.l_onoff=0；则调用
                       closesocket()后强制关闭
m_sLinger.l_linger = 5; //设置等待时间为5秒
setsockopt(s, SOL_SOCKET, SO_LINGER,  (const
            char*)&m_sLinger,sizeof(linger));
```

函数getsockopt()

功能

```
int getsockopt(int sockfd, int level, int optname, void *optval,
   socklen_t *optlen);
```

参数

sockfd：一个标识套接口的描述字。

level：选项定义的层次。支持的层次仅有SOL_SOCKET和IPPROTO_TCP。

optname：需获取的套接口选项。

optval：指针，指向存放所获得选项值的缓冲区。

optlen：指针，指向optval缓冲区的长度值。

返回值

若无错误发生，getsockopt()返回0。否则的话，返回SOCKET_ERROR错误，应用程序可通过WSAGetLastError()获取相应错误代码。

SO_ACCEPTCONN BOOL 套接口正在用listen()监听。
SO_BROADCAST BOOL 套接口设置为传送广播信息。
SO_DEBUG BOOL 允许调试。
SO_DONTLINER BOOL 若为真，则SO_LINGER选项被禁止。
SO_DONTROUTE BOOL 禁止选径。
SO_ERROR int 获取错误状态并清除。
SO_KEEPALIVE BOOL 发送“保持活动”信息。
SO_LINGER struct linger FAR* 返回当前各linger选项。
SO_OOBINLINE BOOL 在普通数据流中接收带外数据。
SO_RCVBUF int 接收缓冲区大小。
SO_REUSEADDR BOOL 套接口能和一个已在使用中的地址捆绑。
SO_SNDBUF int 发送缓冲区大小。
SO_TYPE int 套接口类型（如SOCK_STREAM）。
TCP_NODELAY BOOL 禁止发送合并的Nagle算法。

getsockopt()不支持的BSD选项有：

选项名	类型	意义
SO_RCVLOWAT	int	接收低级水印。
SO_RCVTIMEO	int	接收超时。
SO_SNDLOWAT	int	发送低级水印。
SO_SNDTIMEO	int	发送超时。
IP_OPTIONS		获取IP头中选项。
TCP_MAXSEG	int	获取TCP最大段的长度。

用一个未被支持的选项去调用getsockopt()将会返回一个WSAENOPROTOOPT错误代码（可用WSAGetLastError()获取）。