<https://blog.csdn.net/m0_37776493/article/details/78190587>

#include <sys/socket.h>

     int setsockopt( int socket, int level, int option\_name,

                         const void \*option\_value, size\_t option\_len);

    第一个参数socket是套接字描述符。第二个参数level是被设置的选项的级别，如果想要在套接字级别上设置选项，就必须把level设置为 **SOL\_SOCKET**。 option\_name指定准备设置的选项，option\_name可以有哪些取值，这取决于level，以linux 2.6内核为例（在不同的平台上，这种关系可能会有不同），在套接字级别上(**SOL\_SOCKET**)，option\_name可以有以下取 值：

1. SO\_DEBUG，打开或关闭调试信息。  
       当option\_value不等于0时，打开调试信息，否则，关闭调试信息。它实际所做的工作是在sock->sk->sk\_flag中置 SOCK\_DBG(第10)位，或清SOCK\_DBG位。
2. SO\_REUSEADDR，打开或关闭地址复用功能。  
       当option\_value不等于0时，打开，否则，关闭。它实际所做的工作是置sock->sk->sk\_reuse为1或0。
3. SO\_DONTROUTE，打开或关闭路由查找功能。  
       当option\_value不等于0时，打开，否则，关闭。它实际所做的工作是在sock->sk->sk\_flag中置或清SOCK\_LOCALROUTE位。
4. SO\_BROADCAST，允许或禁止发送广播数据。  
       当option\_value不等于0时，允许，否则，禁止。它实际所做的工作是在sock->sk->sk\_flag中置或清SOCK\_BROADCAST位。
5. SO\_SNDBUF，设置发送缓冲区的大小。  
       发送缓冲区的大小是有上下限的，其上限为256 \* (sizeof(struct sk\_buff) + 256)，下限为2048字节。该操作将sock->sk->sk\_sndbuf设置为val \* 2，之所以要乘以2，是防  
   止大数据量的发送，突然导致缓冲区溢出。最后，该操作完成后，因为对发送缓冲的大小 作了改变，要检查sleep队列，如果有进程正在等待写，将它们唤醒。
6. SO\_RCVBUF，设置接收缓冲区的大小。  
       接收缓冲区大小的上下限分别是：256 \* (sizeof(struct sk\_buff) + 256)和256字节。该操作将sock->sk->sk\_rcvbuf设置为val \* 2。
7. SO\_KEEPALIVE，套接字保活。  
       如果协议是TCP，并且当前的套接字状态不是侦听(listen)或关闭(close)，那么，当option\_value不是零时，启用TCP保活定时 器，否则关闭保活定时器。对于所有协议，该操  
   作都会根据option\_value置或清 sock->sk->sk\_flag中的 SOCK\_KEEPOPEN位。
8. SO\_OOBINLINE，紧急数据放入普通数据流。  
       该操作根据option\_value的值置或清sock->sk->sk\_flag中的SOCK\_URGINLINE位。
9. SO\_NO\_CHECK，打开或关闭校验和。  
       该操作根据option\_value的值，设置sock->sk->sk\_no\_check。
10. SO\_PRIORITY，设置在套接字发送的所有包的协议定义优先权。Linux通过这一值来排列网络队列。  
        这个值在0到6之间（包括0和6），由option\_value指定。赋给sock->sk->sk\_priority。
11. SO\_LINGER，如果选择此选项, close或 shutdown将等到所有套接字里排队的消息成功发送或到达延迟时间后>才会返回. 否则, 调用将立即返回。  
        该选项的参数（option\_value)是一个linger结构：  
            struct linger {  
                int   l\_onoff;     
                int   l\_linger;    
            };  
    如果linger.l\_onoff值为0(关闭），则清 sock->sk->sk\_flag中的SOCK\_LINGER位；否则，置该位，并赋sk->sk\_lingertime值为 linger.l\_linger。
12. SO\_PASSCRED，允许或禁止SCM\_CREDENTIALS 控制消息的接收。  
        该选项根据option\_value的值，清或置sock->sk->sk\_flag中的SOCK\_PASSCRED位。
13. SO\_TIMESTAMP，打开或关闭数据报中的时间戳接收。  
        该选项根据option\_value的值，清或置sock->sk->sk\_flag中的SOCK\_RCVTSTAMP位，如果打开，则还需设sock->sk->sk\_flag中的SOCK\_TIMESTAMP位，同时，将全局变量  
    netstamp\_needed加1。
14. SO\_RCVLOWAT，设置接收数据前的缓冲区内的最小字节数。  
        在Linux中，缓冲区内的最小字节数是固定的，为1。即将sock->sk->sk\_rcvlowat固定赋值为1。
15. SO\_RCVTIMEO，设置接收超时时间。  
        该选项最终将接收超时时间赋给sock->sk->sk\_rcvtimeo。
16. SO\_SNDTIMEO，设置发送超时时间。  
        该选项最终将发送超时时间赋给sock->sk->sk\_sndtimeo。
17. SO\_BINDTODEVICE，将套接字绑定到一个特定的设备上。  
        该选项最终将设备赋给sock->sk->sk\_bound\_dev\_if。
18. SO\_ATTACH\_FILTER和SO\_DETACH\_FILTER。  
        关于数据包过滤，它们最终会影响sk->sk\_filter。  
        以上所介绍的都是在**SOL\_SOCKET**层的一些套接字选项，如果超出这个范围， 给出一些不在这一level的选项作为参数，最终会得到- ENOPROTOOPT的返回值。但以上的分析仅限  
    于这些选项对sock-sk的值的影响，这些选项真正如何发挥作用，我们的探索道路 将漫漫其修远。

**1.**closesocket（一般不会立即关闭而经历TIME\_WAIT的过程）后想继续重用该socket：  
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soket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，发收不能预期进行,而设置收发时限：  
int nNetTimeout=1000;//1秒  
//发送时限  
**setsockopt**(socket，SOL\_S0CKET,SO\_SNDTIMEO，(char \*)&nNetTimeout,sizeof(int));  
//接收时限  
**setsockopt**(socket，SOL\_S0CKET,SO\_RCVTIMEO，(char \*)&nNetTimeout,sizeof(int));

**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);系统默认的状态发送和接收一次为8688字节(约为8.5K)；在实际的过程中发送数据  
和接收数据量比较大，可以设置socket缓冲区，而避免了send(),recv()不断的循环收发：  
// 接收缓冲区  
int nRecvBuf=32\*1024;//设置为32K  
**setsockopt**(s,SOL\_SOCKET,SO\_RCVBUF,(const char\*)&nRecvBuf,sizeof(int));  
//发送缓冲区  
int nSendBuf=32\*1024;//设置为32K  
**setsockopt**(s,SOL\_SOCKET,SO\_SNDBUF,(const char\*)&nSendBuf,sizeof(int));

**5.** 如果在发送数据的时，希望不经历由系统缓冲区到socket缓冲区的拷贝而影响  
程序的性能：  
int nZero=0;  
**setsockopt**(socket，SOL\_S0CKET,SO\_SNDBUF，(char \*)&nZero,sizeof(nZero));

**6.**同上在recv()完成上述功能(默认情况是将socket缓冲区的内容拷贝到系统缓冲区)：  
int nZero=0;  
**setsockopt**(socket，SOL\_S0CKET,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()被呼叫(本函数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(),以前我们  
一般采取的措施是"从容关闭"shutdown(s,SD\_BOTH),但是数据是肯定丢失了，如何设置让程序满足具体  
应用的要求(即让没发完的数据发送出去后在关闭socket)？  
struct linger {  
u\_short l\_onoff;  
u\_short l\_linger;  
};  
linger m\_sLinger;  
m\_sLinger.l\_onoff=1;//(在closesocket()调用,但是还有数据没发送完毕的时候容许逗留)  
// 如果m\_sLinger.l\_onoff=0;则功能和2.)作用相同;  
m\_sLinger.l\_linger=5;//(容许逗留的时间为5秒)  
**setsockopt**(s,SOL\_SOCKET,SO\_LINGER,(const char\*)&m\_sLinger,sizeof(linger));

设置套接口的选项。  
   #include <winsock.h>  
   int PASCAL FAR **setsockopt**( SOCKET s, int level, int optname,  
   const char FAR\* optval, int optlen);  
   s：标识一个套接口的描述字。  
   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缓冲区的长度。  
注释：  
**setsockopt**()函数用于任意类型、任意状态套接口的设置选项值。尽管在不同协议层上存在选项，但本函数仅定义了最高的“套接口”层次上的选项。选项影响套接口的操作，诸如加急数据是否在普通数据流中接收，广播数据是否可以从套接口发送等等。  
   有两种套接口的选项：一种是布尔型选项，允许或禁止一种特性；另一种是整形或结构选项。允许一个布尔型选项，则将optval指向非零整形数；禁止一个选项optval指向一个等于零的整形数。对于布尔型选项，optlen应等于sizeof(int)；对其他选项，optval指向包含所需选项的整形数或结构，而optlen则为整形数或结构的长度。SO\_LINGER选项用于控制下述情况的行动：套接口上有排队的待发送数据，且 closesocket()调用已执行。参见closesocket()函数中关于SO\_LINGER选项对closesocket()语义的影响。应用程序通过创建一个linger结构来设置相应的操作特性：  
   struct linger {  
int l\_onoff;  
int l\_linger;  
   };  
   为了允许SO\_LINGER，应用程序应将l\_onoff设为非零，将l\_linger设为零或需要的超时值（以秒为单位），然后调用**setsockopt**()。为了允许SO\_DONTLINGER（亦即禁止SO\_LINGER），l\_onoff应设为零，然后调用**setsockopt**()。  
   缺省条件下，一个套接口不能与一个已在使用中的本地地址捆绑（参见bind()）。但有时会需要“重用”地址。因为每一个连接都由本地地址和远端地址的组合唯一确定，所以只要远端地址不同，两个套接口与一个地址捆绑并无大碍。为了通知WINDOWS套接口实现不要因为一个地址已被一个套接口使用就不让它与另一个套接口捆绑，应用程序可在bind()调用前先设置SO\_REUSEADDR选项。请注意仅在bind()调用时该选项才被解释；故此无需（但也无害）将一个不会共用地址的套接口设置该选项，或者在bind()对这个或其他套接口无影响情况下设置或清除这一选项。  
   一个应用程序可以通过打开SO\_KEEPALIVE选项，使得WINDOWS套接口实现在TCP连接情况下允许使用“保持活动”包。一个WINDOWS套接口实现并不是必需支持“保持活动”，但是如果支持的话，具体的语义将与实现有关，应遵守RFC1122“Internet主机要求－通讯层”中第 4.2.3.6节的规范。如果有关连接由于“保持活动”而失效，则进行中的任何对该套接口的调用都将以WSAENETRESET错误返回，后续的任何调用将以WSAENOTCONN错误返回。  
   TCP\_NODELAY选项禁止Nagle算法。Nagle算法通过将未确认的数据存入缓冲区直到蓄足一个包一起发送的方法，来减少主机发送的零碎小数据包的数目。但对于某些应用来说，这种算法将降低系统性能。所以TCP\_NODELAY可用来将此算法关闭。应用程序编写者只有在确切了解它的效果并确实需要的情况下，才设置TCP\_NODELAY选项，因为设置后对网络性能有明显的负面影响。TCP\_NODELAY是唯一使用IPPROTO\_TCP层的选项，其他所有选项都使用SOL\_SOCKET层。  
   如果设置了SO\_DEBUG选项，WINDOWS套接口供应商被鼓励（但不是必需）提供输出相应的调试信息。但产生调试信息的机制以及调试信息的形式已超出本规范的讨论范围。  
**setsockopt**()支持下列选项。其中“类型”表明optval所指数据的类型。  
选项        类型   意义  
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（参见bind()）。  
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：  
选项名    类型 意义  
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。  
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。