<https://blog.csdn.net/hnlyyk/article/details/51143256>

版权声明：【博主微信公众号：不忘初心的行者】【本文为博主原创,未经博主允许不得转载】 https://blog.csdn.net/u010193457/article/details/51143256

recv函数  
int recv( SOCKET s, char FAR \*buf, int len, int flags);  
不论是客户还是服务器应用程序都用recv函数从TCP连接的另一端接收数据。  
该函数的第一个参数指定接收端套接字描述符；   
第二个参数指明一个缓冲区，该缓冲区用来存放recv函数接收到的数据；   
第三个参数指明buf的长度； 第四个参数一般置0。  
这里只描述同步Socket的recv函数的执行流程。当应用程序调用recv函数时，  
（1）recv先等待s的发送缓冲中的数据被协议传送完毕，如果协议在传送s的发送缓冲中的数据时出现网络错误，那么recv函数返回SOCKET\_ERROR，  
（2）如果s的发送缓冲中没有数据或者数据被协议成功发送完毕后，recv先检查套接字s的接收缓冲区，  
如果s接收缓冲区中没有数据或者协议正在接收数 据，那么recv就一直等待，直到协议把数据接收完毕。  
当协议把数据接收完毕，recv函数就把s的接收缓冲中的数据copy到buf中  
（**注意协议接收到的数据可能大于buf的长度，所以 在这种情况下要调用几次recv函数才能把s的接收缓冲中的数据copy完**。  
recv函数仅仅是copy数据，真正的接收数据是协议来完成的）， recv函数返回其实际copy的字节数。  
如果recv在copy时出错，那么它返回SOCKET\_ERROR；  
如果recv函数在等待协议接收数据时网络中断了，那么它返回0。  
**默认 socket 是阻塞的 解阻塞与非阻塞recv返回值没有区分，都是 <0 出错 ；=0 连接关闭 ；>0 接收到数据大小，**  
特别：  
**返回值<0时并且(errno == EINTR || errno == EWOULDBLOCK || errno == EAGAIN)的情况下认为连接是正常的，继续接收**。  
只是阻塞模式下recv会阻塞着接收数据，非阻塞模式下如果没有数据会返回，不会阻塞着读，因此需要循环读取）。  
返回说明：   
成功执行时，返回接收到的字节数。  
另一端已关闭则返回0。  
失败返回-1，  
errno被设为以下的某个值   
**EAGAIN：套接字已标记为非阻塞，而接收操作被阻塞或者接收超时**  
EBADF：sock不是有效的描述词   
ECONNREFUSE：远程主机阻绝网络连接   
EFAULT：内存空间访问出错   
EINTR：操作被信号中断   
EINVAL：参数无效   
ENOMEM：内存不足   
ENOTCONN：与面向连接关联的套接字尚未被连接上   
ENOTSOCK：sock索引的不是套接字 当返回值是0时，为正常关闭连接；  
思考：  
当对侧没有send，即本侧的套接字s的接收缓冲区无数据，返回值是什么（EAGAIN，原因为超时，待测）