[linux send与recv函数详解](http://www.cnblogs.com/blankqdb/archive/2012/08/30/2663859.html)

#include <sys/socket.h>

ssize\_t recv(int sockfd, void \*buff, size\_t nbytes, int flags);

ssize\_t send(int sockfd, const void \*buff, size\_t nbytes, int flags);

recv 和send的前3个参数等同于read和write。

flags参数值为0或：

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| flags | 说明 | recv | send |
| MSG\_DONTROUTE | 绕过路由表查找 |  |  |
| MSG\_DONTWAIT | 仅本操作非阻塞 |  |  |
| MSG\_OOB | 发送或接收带外数据 |  |  |
| MSG\_PEEK | 窥看外来消息 |  |  |
| MSG\_WAITALL |  |  |  |

ssize\_t send(int sockfd, const void \*buff, size\_t nbytes, int flags);

第一个参数是客户端对应的套接字（千万注意），第二个参数指向发送的信息所在的缓冲区（内存），第三个参数是缓冲区的大小，第四个参数一般设置为0.

 send函数的返回值非常重要，如果成功，则返回发送的字节数，如果失败，则返回对应的失败码。

 1) send先比较发送数据的长度nbytes和套接字sockfd的发送缓冲区的长度，如果nbytes > 套接字sockfd的发送缓冲区的长度, 该函数返回SOCKET\_ERROR;

 2) 如果nbtyes <= 套接字sockfd的发送缓冲区的长度，那么send先检查协议是否正在发送sockfd的发送缓冲区中的数据，如果是就等待协议把数据发送完，如果协议还没有开始发送sockfd的发送缓冲区中的数据或者sockfd的发送缓冲区中没有数据，那么send就比较sockfd的发送缓冲区的剩余空间和nbytes

 3) 如果 nbytes > 套接字sockfd的发送缓冲区剩余空间的长度，send就一起等待协议把套接字sockfd的发送缓冲区中的数据发送完

 4) 如果 nbytes < 套接字sockfd的发送缓冲区剩余空间大小，send就仅仅把buf中的数据copy到剩余空间里(注意并不是send把套接字sockfd的发送缓冲区中的数据传到连接的另一端的，而是协议传送的，send仅仅是把buf中的数据copy到套接字sockfd的发送缓冲区的剩余空间里)。

 5) 如果send函数copy成功，就返回实际copy的字节数，如果send在copy数据时出现错误，那么send就返回SOCKET\_ERROR; 如果在等待协议传送数据时网络断开，send函数也返回SOCKET\_ERROR。

 6) send函数把buff中的数据成功copy到sockfd的改善缓冲区的剩余空间后它就返回了，但是此时这些数据并不一定马上被传到连接的另一端。如果协议在后续的传送过程中出现网络错误的话，那么下一个socket函数就会返回SOCKET\_ERROR。（每一个除send的socket函数在执行的最开始总要先等待套接字的发送缓冲区中的数据被协议传递完毕才能继续，如果在等待时出现网络错误那么该socket函数就返回SOCKET\_ERROR）

 7) 在unix系统下，如果send在等待协议传送数据时网络断开，调用send的进程会接收到一个SIGPIPE信号，进程对该信号的处理是进程终止。

2.recv函数

sockfd: 接收端套接字描述符

buff：   用来存放recv函数接收到的数据的缓冲区

nbytes: 指明buff的长度

flags:   一般置为0

 1) recv先等待s的发送缓冲区的数据被协议传送完毕，如果协议在传送sock的发送缓冲区中的数据时出现网络错误，那么recv函数返回SOCKET\_ERROR

 2) 如果套接字sockfd的发送缓冲区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sockfd的接收缓冲区，如果sockfd的接收缓冲区中没有数据或者协议正在接收数据，那么recv就一起等待，直到把数据接收完毕。当协议把数据接收完毕，recv函数就把s的接收缓冲区中的数据copy到buff中（注意协议接收到的数据可能大于buff的长度，所以在这种情况下要调用几次recv函数才能把sockfd的接收缓冲区中的数据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）

 3) 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。

 4) 在unix系统下，如果recv函数在等待协议接收数据时网络断开了，那么调用 recv的进程会接收到一个SIGPIPE信号，进程对该信号的默认处理是进程终止。