<http://blog.sina.com.cn/s/blog_4b6f7840010128gx.html>

Linux学习笔记之网络的send和recv函数

1、  函数原型

Int send(int socket, const void \* buff, int length, int flags)

Int recv(int socket, void \*buff, int length, int flags)

2、  对于正常的tcp通信来说，参数flags应设置为0

3、  Send（）函数不会阻塞程序的执行（当套接字是阻塞的并且套接字的TCP发送缓冲区容不下应用程序缓冲区中的所有数据时，send()函数将会阻塞）。缓冲区的数据被发送到系统中的底层的TCP发送缓冲区，然后函数将返回。有可能在send()函数中定义的缓冲区中的数据不会全部被发送出去。Send()函数的返回值表示有多少个字节的数据已被发送到TCP发送缓冲区。确认这个返回值匹配缓冲区的大小以确保所有的数据都已被发送出去时非常重要的。

4、  Recv()函数是一个阻塞式函数。当文件描述符fd用fcntl()设置为非阻塞后，会理解返回-1，errno为resource temporarily unavailable ，或者可以用标志MSG\_DONTWAIT来设置recv()当次读取数据为非阻塞，程序的执行将暂停直到函数从远程客户端接收到数据或者远程客户端明确的断开TCP会话。如果客户端断开TCP会话，recv()函数返回值为0。如果客户端发送数据包，recv()函数将接收到的数据放入定义的缓冲区并返回接收到的字节数。

Len = recv(fd, buff, len, 0)

Buff[len] = ‘\0’当接收到数据时，recv()函数返回的缓冲区长度被用于在缓冲区中放置一个NULL结束符。这确保了每个接收到的字符串都被正确的终止了。

                            while((n = recv(connfd, buff,sizeof(buff), 0)) > 0)   //回发

                            {

                                   buff[n] = '\0';

                                   printf("recv : %s\n",buff);

                                   n = send(connfd,buff, n, 0);

                                   if(n < 0)

                                   {

                                         perror("write error\n");

                                         return -1;

                                   }

                                   buff[n] = '\0';

                                   printf("write : %s\n",buff);

                            }当远程客户端关闭TCP会话连接时，recv()将返回0，用于退出while()循环。

5、  如果服务器和客户端同时等待在recv()函数调用上，它们将产生死锁，并且不会有通信发生。

6、  Send()函数发送的数据将被放入一个字节流中通过网络传递给接收主机，接收主机使用recv()函数读取字节流中的部分数据，但并没有办法控制读取字节流中哪一部分的数据，你不能保证通过一次recv()函数调用就可以读取由send()函数发送的所有数据。

7、  TCP中没有消息边界的概念。当消息使用send()函数并使用recv()函数接收时，它们不一定在发送和接收时使用相同的边界，因为TCP有处理数据的内部缓冲区。由于TCP数据包可能会发生重传，所以应用程序发送给套接字的数据会在LINUX系统内部缓存。

8、  由两次send()函数调用发送的数据可以被一次recv()函数调用接收。

9、  标记消息边界：i. 使用消息长度 ii. 使用消息结束标记。使用消息长度有两种方法：i. 每个消息的长度都相同 ii. 每个消息中都包含其长度的含义。结束标记是一个预先确定的字符或一组字符，它用于界定一个消息的结尾。当主机接收到每个数据包时，必须检查数据包中的每一个字节。如果它不是消息结束标记，就把它添加到临时缓冲区中，当发现消息结束标记时，临时缓冲区中的数据就被传输到永久缓冲区中以便使用，然后临时缓冲区被清空以用于重组下一个消息。

                      ---------------------------摘自《Linux 高级程序设计》

                  [英] Jon Masters [美] Richard Blum 陈键 等 翻译