预备知识：每个TCP或UDP会话在内核中都是由一个五元组标识唯一性。因此内核中每个五元组必定不能相同

五元组：{传输层协议、源IP地址、源端口、目的IP地址、目的端口}

用户态的Socket编程中默认做了更严格的限制。通过SO\_REUSEADDR可以打开一些限制。下图是没设置和没设置SO\_REUSEADDR调用bind()函数执行情况（参考资料1 2）。

|  |  |  |  |
| --- | --- | --- | --- |
| SO\_REUSEADDR | socketA | socketB | Result |
| ON/OFF | 192.168.0.1:21 | 192.168.0.1:21 | Error (EADDRINUSE) |
| ON/OFF | 192.168.0.1:21 | 10.0.0.1:21 | OK |
| ON/OFF | 10.0.0.1:21 | 192.168.0.1:21 | OK |
| OFF | 0.0.0.0:21 | 192.168.1.0:21 | Error (EADDRINUSE) |
| OFF | 192.168.1.0:21 | 0.0.0.0:21 | Error (EADDRINUSE) |
| ON | 0.0.0.0:21 | 192.168.1.0:21 | OK |
| ON | 192.168.1.0:21 | 0.0.0.0:21 | OK |
| ON/OFF | 0.0.0.0:21 | 0.0.0.0:21 | Error (EADDRINUSE) |

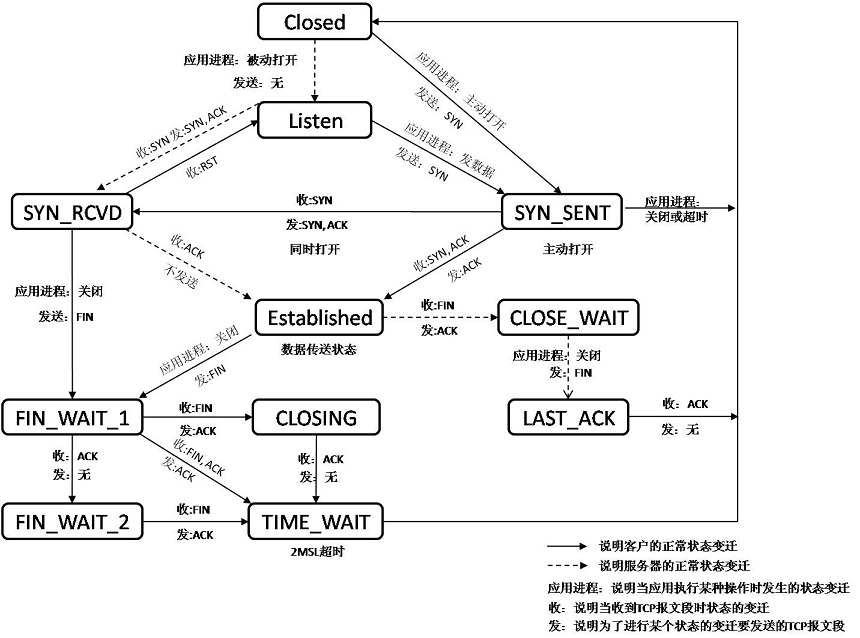
但实际上服务程序用SO\_REUSEADDR的主要目的不是绑定IP和端口的问题，主要是为了：即使近段时间内建立的链接没有完全关闭的情况下（包括内核态），依然允许在一个指定IP和端口上调用bind()函数。

发生这种情况的一种场景是(参考资料3)：

1. 启动一个监听服务器；
2. 连接请求到达，派生一个子进程来处理这个客户；
3. 监听服务器终止，但子进程继续为现有连接上的客户提供服务；
4. 重启监听服务器。

另外，对于TCP来说，用户态调用了close()函数并成功返回了，不代表这个连接在内核态中已经完全关闭了，TCP状态机的实际上会在TIME\_WAIT状态等待一段时间后才会将这个连接置为关闭状态。如果没有设置SO\_REUSEADDR，这段时间内bind()函数调用会失败。

注：为什么TIME\_WAIT需要等待一段时间呢？因为TCP需要做到连接可靠，有重传机制。关闭socket时无法确定自己发出的响应对端FIN的ACK报文被接收到了，如果ACK没有被对端接收到对端会重发FIN报文，本端就能响应对端重发的FIN报文了。



SO\_REUSEADDR还有一些其他作用，目前不太关心，详情可参考参考资料3

参考资料：

1. SO\_REUSEADDR和SO\_REUSEPORT异同，http://blog.chinaunix.net/uid-28587158-id-4006500.html
2. Socket options SO\_REUSEADDR and SO\_REUSEPORT, how do they differ? Do they mean the same across all major operating systems?，http://stackoverflow.com/questions/14388706/socket-options-so-reuseaddr-and-so-reuseport-how-do-they-differ-do-they-mean-t
3. UNIX网络编程卷1：套接字联网API 第三版，P165~P167