情景：

在实现 http 服务器的时候，发现 浏览器发过来的请求，服务器很久都没收到

结论：

这个socket 内核缓存机制，缓存太大了，数据一时不会发到用户层，

做法：

使用 setsockopt 将缓存设置成 最小值，就很快收到请求了，但是还不确定有没有什么别的问题

情景：

服务器程序结束后再重新bind，会bind err

隐患 3．地址使用错误（EADDRINUSE)

您可以使用 bind API 函数来绑定一个地址（一个接口和一个端口）到一个套接字端点。可以在服务器设置中使用这个函数，以便限制可能有连接到来的接口。也可以在客户端设置中使用这个函数，以便限制应当供出去的连接所使用的接口。bind 最常见的用法是关联端口号和服务器，并使用通配符地址（INADDR_ANY），它允许任何接口为到来的连接所使用。

bind 普遍遭遇的问题是试图绑定一个已经在使用的端口。该陷阱是也许没有活动的套接字存在，但仍然禁止绑定端口（bind 返回 EADDRINUSE），它由 TCP 套接字状态 TIME_WAIT 引起。该状态在套接字关闭后约保留 2 到 4 分钟。

在 TIME_WAIT 状态退出之后，套接字被删除，该地址才能被重新绑定而不出问题。

等待 TIME_WAIT 结束可能是令人恼火的一件事，特别是如果您正在开发一个套接字服务器，就需要停止服务器来做一些改动，然后重启。幸运的是，有方法可以避开 TIME_WAIT 状态。可以给套接字应用 SO_REUSEADDR 套接字选项，以便端口可以马上重用。

考虑清单 3 的例子。在绑定地址之前，我以 SO_REUSEADDR 选项调用 setsockopt。为了允许地址重用，我设置整型参数（on）为 1 （不然，可以设为 0 来禁止地址重用）。

清单 3．使用 SO_REUSEADDR 套接字选项避免地址使用错误

```
int sock, ret, on;
struct sockaddr_in servaddr;
/* Create a new stream (TCP) socket */
sock = socket( AF_INET, SOCK_STREAM, 0 ):
/* Enable address reuse */
on = 1;
ret = setsockopt( sock, SOL_SOCKET, SO_REUSEADDR, &on, sizeof(on) );
/* Allow connections to port 8080 from any available interface */
memset( &servaddr, 0, sizeof(servaddr) );
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl( INADDR_ANY );
servaddr.sin_port = htons( 45000 );
/* Bind to the address (interface/port) */
ret = bind( sock, (struct sockaddr *)&servaddr, sizeof(servaddr) );
```

在应用了 SO_REUSEADDR 选项之后，bind API 函数将允许地址的立即重用。