TCP连接时（下面是socket）

**单一进程** 可以同时打开文件数量，每个TCP都创建一个socket，每个socket句柄同时也是一个文件句柄。ulimit，查看当前用户进程，所允许打开的文件数量限制



当前用户的每个进程，可以 最多同时打开1024个文件（包括stdout, stdin, stderr，进程间通讯的unix域socket），剩下的，可用于客户端socket连接的文件数也就剩下 1024-10=1014。所以说：

缺省情况下，基于linux的通讯程序，最多允许 1014 个同时的TCP并发连接

怎么破？

破除linux对“当前用户的进程，同时可打开文件数量的限制”。两个soft limit和hard limit

hard limit指：系统根据硬件资源情况（主要是内存）计算出的系统最多可同时打开的数量

1. 到文件：**/etc/security/**limits.conf 中去修改
2. /etc/pam.d/login，添加新行

session required /lib/security/pam\_limits.so

告诉Linux，用户登陆完成后，调用 pam\_limits.so模块，以设置“此用户对各种资源数量的最大限制”。pam\_limits.so就是从/etc/security/limits.conf中读取限制值的

1. 查看linux系统级的 最大打开文件数目 的限制



系统最多允许所有用户总共同时打开的文件数量。这是 系统级的 hard limit ，用户级上，如果想设置一个比这还大的值。就需要先把这个值改了

修改：/etc/rc.local 在里面添加：echo *newvalue* > /proc/sys/fs/file-max

这样：linux启动后，强行将 系统级 hard limit设置为*newvalue*

上面改完后，重启系统。ulimit -n应该看到设置的值。但是，如果没看到，则可能是 用户登陆脚本 /etc/profile 中，使用ulimit -n 已经作了限制。由于 ulimit -n 时，新值<=上次ulimit -n设置的值，所以，用这个命令就无法增大这个限制了。怎么办呢？ /etc/profile 中，找到此命令，将其删除。然后重启系统

到此：为了支持高并发的TCP，同时打开文件数量的限制 就被解除了

网络内核对TCP连接的限制，这个也要修改

“打开文件数量的限制”解除后，仍会出现 TCP连接数增加到一定数量时，再也无法成功建立新的TCP连接。原因有多种

1. Linux网络内核对本地端口号范围有限制
2. Linux网络内核的 IP\_TABLE防火墙对 最大跟踪的TCP连接数有限制

系统本身就搞完了。接下来，写程序的要注意了

使用合适的 网络I/O技术 和 I/O事件分派机制

**网络I/O技术**

同步I/O（不合适）

非阻塞 同步I/O（反应式I/O）- - select(), poll(), epoll等机制

异步I/O - - AIO

**I/O事件分派机制**

从I/O事件分派机制来看

select()不合适。它所支持并发有限（通常1024之内）

如果考虑性能，poll()也不合适。虽然支持较高的TCP并发数。但，“轮询”机制，并非数高时，效率相当低，并且可能有 I/O事件分派不均，导致 部分 TCP连接上的I/O出现 饥饿。

epoll，AIO好。