Lesson 17

1.server 端指定的 port，不是用来通信，只用来监听请求。一旦 TCP 链接建立，那么双方协定好一个新的端口号进行通信。

2.server 刚启动时，port 阻塞在 accept 上，此时 TCP 处于 LISTEN 状态。一旦连接建立，那么连接的双方立刻处于 ESTABLISHED 状态，原来的端口号仍处于 LISTEN 状态。

3.一旦 server 自行关闭，那么建立的连接都被关闭，而 TCP 所处的 LISTEN 状态转为 TIME_WAIT 状态。这个状态大概维持 2min，此时 port 被占用。解决方案就是设置 socket 的端口复用。

4.server 的版本一：这是一个基本的 TCPserver 端。当它自行关闭的时候，TCP 进入 TIME_WAIT 状态，此时无法再次启动。

5.server 的版本二：设置了端口复用，解决了这一问题。

6.client 版本一：与 server 可以连接，但是存在如下的问题：

a) 客户端关闭，服务器可以及时接收到。

b) 服务器关闭，客户端无法感应。

7.原因分析：

a) server 阻塞在 read 上，一旦 client 关闭，read 立刻返回 0.

b) client 阻塞在 fgets 上，即使 server 关闭，client 也无法感知。

c) 解决方案是：client 采用 IO 复用。

8.逻辑上以字节为单位的，不需要处理大小端问题。

9.TCP 的粘包问题：

a) 如果某一方发送过快，例如快速发了长度为 400、600 的报文。

b) 接收方可能接收到一个长度为 1000 的消息，此时接收方无法从中判定正确的消息边界。

10.粘包问题的解决方案：

a) 发送报文前，先发送报文的长度 len，然后使用 readn、writen 进行接收。发送 len 注意处理大小端的问题。

b) 每个报文以\n 作为结束符号，这样接收端采用 readline 即可，发送时采用 writen。实际中的 HTTP、FTP 都是以\r\n 作为结束标志。

c) 上面第二种方法的问题是 readline 效率低。

11.recv 的 MSG_PEEK 选项表示可以从 fd 中预览数据，而不是把数据真的取走。

12.使用 recv 的 MAG_PEEK 包装一个 recv_peek 函数，然后使用它写一个 readline。这个 readline 只能用于网络通信。