Lesson 18

1.我们使用 select 解决了客户端不能及时感应 server 关闭的问题。采用 select 同时监听 STDIN_FILENO 和 peerfd，我们只阻塞在 select 调用上，而决不阻塞在 readline 和 fgets 上。

2.select 执行完毕后，会修改 readset 的值，使其只包含准备好的 fd，所以每次都要去重新设置 readset。

3.在 select 中，“准备好”的含义：

a) 若对 readset 中的某 fd 进行 read 不会阻塞，则认为该 fd 是准备好的。
b) 若对 writeset 中的某 fd 进行 write 不会阻塞，则认为该 fd 是准备好的。
c) 对于读写集合，普通文件的文件描述符总是准备好的。

4.select 实现了一种简单的并发。

5.read 或者 recv 并不是接收 TCP 连接的数据，而是去内核缓冲区请求数据，拷贝到用户空间。

6.实验：server 端 sleep 5s，client 快速发送两个数据，close 掉 fd，然后睡眠。server 接收第一个数据，然后 writen 给 fd，因为此时的 fd 已经 close，所以 server 会收到一个 RST 复位报文。然后 server 继续接收第二个报文，然后再次 writen 给 fd，此时就触发了 SIGPIPE 信号。

7.SIGPIPE 是往已经 RST 的 fd 写入数据所导致的。

8.close 与 shutdown 的区别：

a) close 采用引用计数关闭的。而 shutdown 则是直接关闭。

b) close 是同时关闭读写两端，而 shutdown 可以指定关闭哪个方向。

9.针对 client 连续发两次数据后 close 连接，导致 TCP 服务器崩溃的解决办法：

a) client 采用 shutdown 关闭写端，保留读端。

b) server 要去处理 SIGPIPE。

10.使用 select 编写客户端程序的一般步骤：

初始化参数，包括 readset、maxfd、nready

while(1)

{

1.先重新设置 readset

2.执行 select 调用，包括检查返回值

3.依次检查 stdin 和 peerfd，如果是前者就从键盘读取数据，如果是后者，就使用 readline 接收网络数据。

}

11.使用 poll 编写客户端的一般步骤：

准备数组（这里为 2），填入相应的 fd 以及 events。还有 maxi、nready

while(1)

{

1.执行 poll，以及检查返回值

2.检查两个 fd，通过 revents 字段

}

12. 使用 epoll 编写客户端程序的一般步骤：

使用 epoll_create 创建 epoll 的 fd

往 epollfd 中注册需要监听的两个 fd，使用 epoll_ctl

准备一个数组 events，接收需要处理的事件

while(1)

{

执行 epoll_wait

根据返回值 nready，依次遍历 events 这个数组

}