

POSIX API

本节内容：

image-20240824104021127

image-20240824104049138

1.socket () :

image-20240824104601197

2.bind () :


image-20240824105846208

3.listen():

image-20240825090211114

1.建立连接

三次握手：

image-20240825092138661

问题：

image-20240825092303702

a.第一次握手创建tcb的时候就开始了

b.通过数据包提供的源ip，源port，目的ip，目的port，可以匹配

c.listen第二个参数**backlog**的版本迭代

image-20240825092803504

问题：两边同时发syn怎么办？（p2p场景）

image-20240825215332917

优缺点： **

1.syn队列 优点：一定程度上防止泛洪

缺点：随着防火墙技术的深入，已经在防火墙就进行了隔离，这个越来越鸡肋

2.syn+accept队列


防止accept不处理而导致堆积？

3.accept队列：（最现代化）

大大提升建链的吞吐量？


4.accept () :

accept处理更适合使用水平触发LT (多个节点同时接入的时候可以不停触发)


 image-20240825093704631

如果是使用ET呢?

答: 用一个循环并且设置成非阻塞


 image-20240825094118205

实现:

 image-20240825215520941

2.传输数据

send():

 image-20240825210002728

send只copy到内核, 至于什么时候发给对方, 那取决与tcp协议栈的实现 (可能和数据量、时间有关系)


recv():

 image-20240825210318801

recv只从协议栈copy出来

mut:

最大传输单元, 可修改


 image-20240825210626531

问题:

以下两种方式有区别吗?

 image-20240825211333186

3.断开连接

 image-20240825213113799

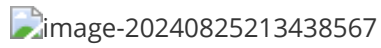
close():

1.把fd回收

2.send一个fin的空包过去

问题：

1.

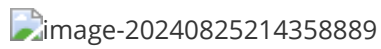
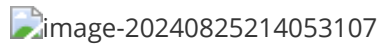


答案：没有关系，recv有个返回值，ret = 0代表有fin

问题：

1. ack没有收到，先收到fin

2. 两边同时调用close（现象：服务器出现大量time-wait现象）



网络协议栈

内容：(还有个快重传，收到**三个ack**的话)

