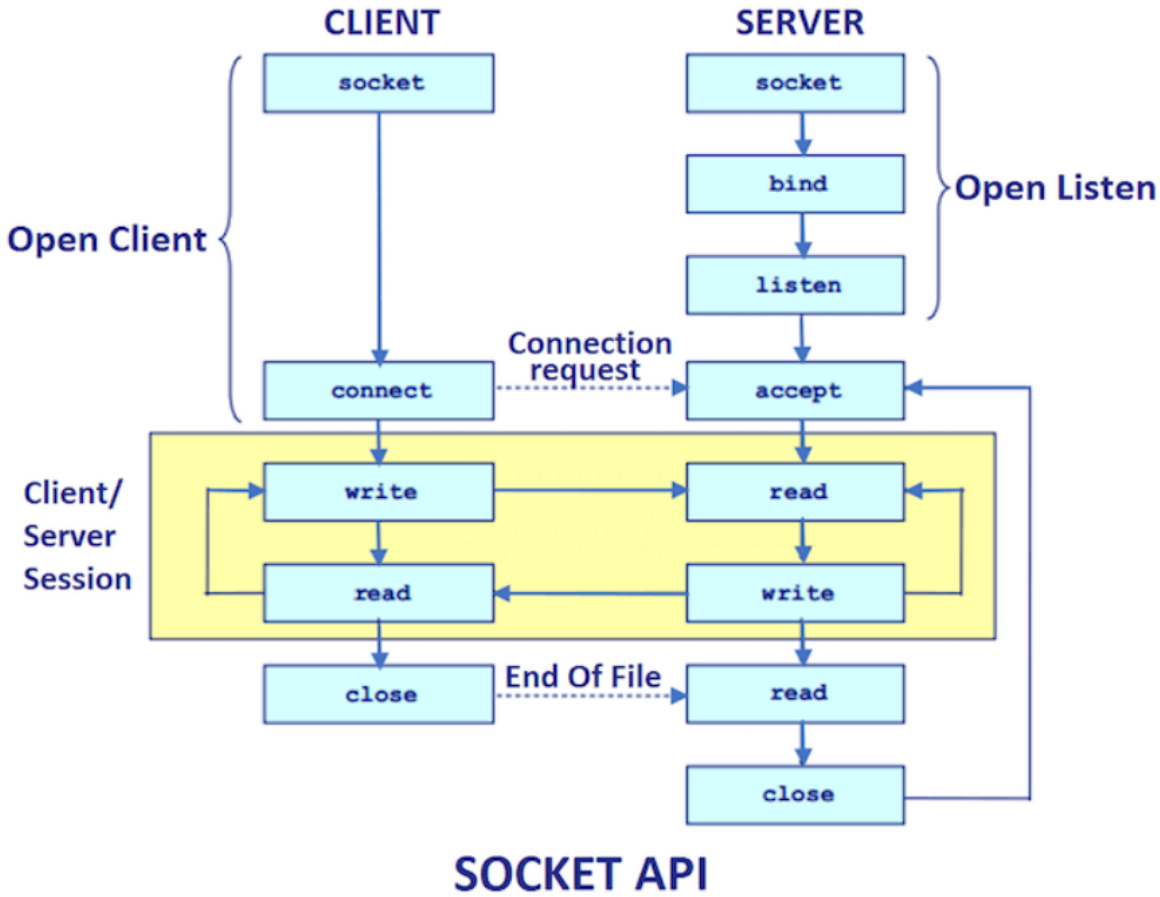
实践代码结构说明

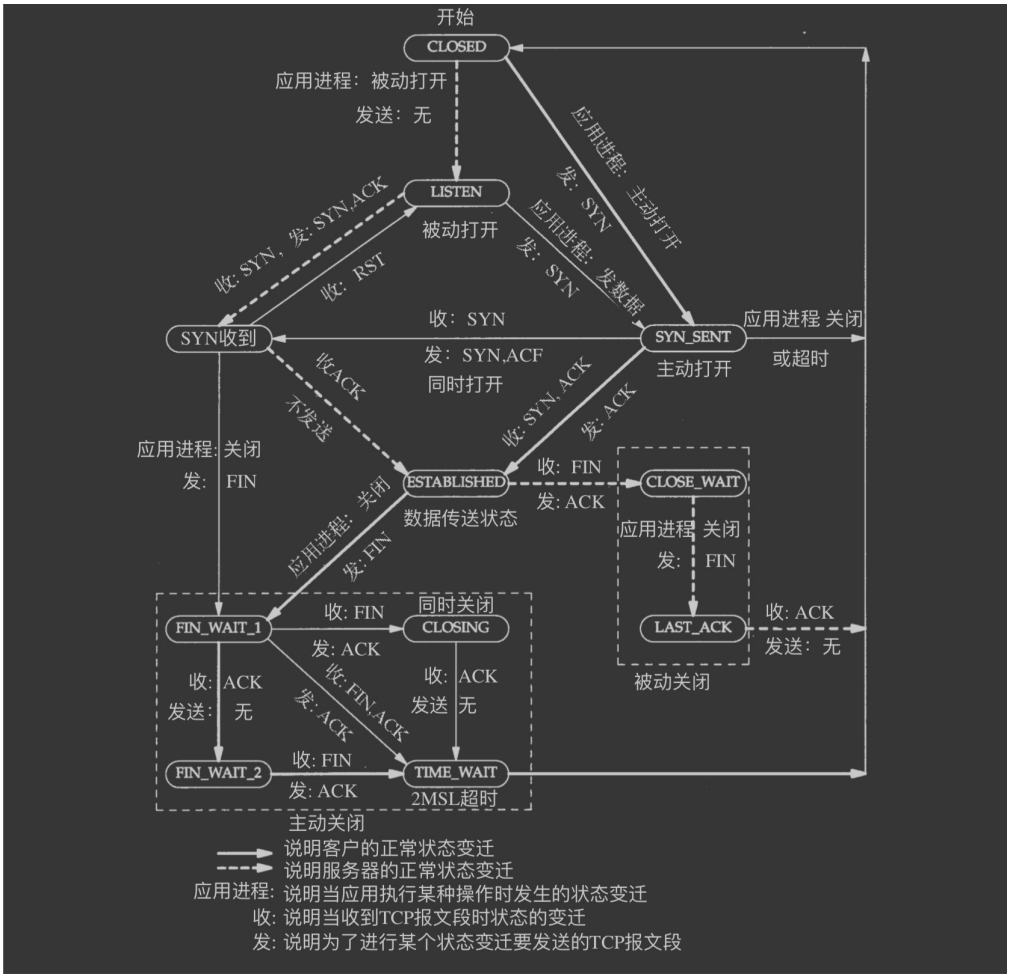
文件结构

- **tju_tcp:** 项目根目录
  - **build:** 存放所有编译的中间文件
  - **inc:** 存放所有头文件
    - global.h: 定义一些全局都会用到的变量和结构
    - kernel.h: 模拟一部分linux内核行为 比如如何发送数据到下一层 根据五元组查找socket等
    - tju_packet.h: 定义TCP所用到的数据包格式 提供各种数据包的操作(创建 获得字段等)
    - tju_tcp.h: 需要实现的TCP的各种结构和功能的定义
  - **src:** 存放所有源代码文件
    - client.c
    - server.c
    - kernel.c
    - tju_packet.c
    - tju_tcp.c
  - Makefile

为了理解代码的结构，需要了解LInux上真正的 socket 是如何运作的

真正的 socket 是如何运作的





linux内核中维护着三张哈希表

- 监听socket表 lhash
- 建立连接socket表 ehash
- 端口分配表 bhash

每当linux收到一个TCP报文

根据报文内容获取一个五元组(源IP 源PORT 目的IP 目的PORT 协议)

根据五元组计算哈希值 查找ehash 中是否有对应的socket

没有的话 根据 (目的IP 目的PORT 协议) 查找lhash中是否有对应的socket

还没有的话 丢弃数据包 (可能向对方发一个RST)

同时 linux给每一个处于LISTEN状态的socket 维护两个列表

- 半连接列表: 存放收到SYN但还没收到ACK的socket
- 全连接列表: 存放完成三次握手 建立起正常连接的socket

Socket API

- **socket**

  创建socket结构体 分配内存空间
  此时socket的状态为CLOSED

- **bind**

  linux内核中保存着一张端口分配表
  任何socket调用bind的时候
  linux都会检查该端口是否被占用并进行对应的处理

- **listen**

  socket进入LISTEN状态
  该socket会被加入lhash
  此时linux接收到TCP报文的时候 就能根据报文的内容 查找到lhash中的socket了
  实际上当linux收到SYN并找到对应的socket时
  就会新建一个全新的socket 并把它放到LISTEN的socket的半连接队列中
  还会将该新建的socket放到ehash中
  之后服务端发来的ACK 就能根据五元组找到这个新建的socket
  并将其从半连接队列取出 放到全连接队列中
  (ehash中的条目保持不变)

- **accept**

  如果LISTEN的socket全连接队列有条目
  就取出返回
  如果没有 就阻塞等待

- **connect**

  首先分配一个空闲的端口给调用的socket
  然后发送一个SYN给服务端
  设置自己的状态为TCP_SYN_SENT
  把调用的socket放到ehash中
  然后进入while循环 等待当前socket的状态变为ESTABLISHED
  对外表现就是阻塞等待建立连接
  而socket状态的改变来自收到服务端的SYNACK 是在其他函数中处理的

- **send/recv**

  linux给每个建立了连接的socket分配一个发送缓冲区 一个接受缓冲区
  用户调用send和recv时 本质上是写入和读取发送缓冲区和接受缓冲区
  而TCP发送数据时

  - 如何从缓冲区构造一个个数据包
  - 构造的数据包长度都是多少
  - 什么时候发送数据包
  - 发送多少数据包

  TCP接收数据时

  - 要不要发 要怎么发ACK
  - 如何整理接收到的TCP包
  - 如何把接收到的一个个分开的包含头部的TCP包去掉header 将可靠连续的数据写入接收缓冲区 等待用户读取

  这些内容涉及到可靠数据传输 流量控制 拥塞控制等多种机制影响

  也是本次实践的重点部分