**模型选择:** 1.多线程reactor2.多进程reactor 3.多线程reator(处理消息收发)+多进程worker(业务逻辑处理)

**1.**

主线程reactor负责监听连接,验证连接,将连接fd放到线程池中(unix socket?), fd%threadpoolLength

每个线程持有线程对象有监听fd列表,有主线程添加

主线程持有一个线程池对象threadpool,控制线程池,与线程池中线程通信

Threadpool{

1. pthread\_mutex\_t queue\_lock;
2. pthread\_cond\_t queue\_ready;
4. /\*链表结构，线程池中所有等待任务\*/
5. CThread\_worker \*queue\_head;
7. /\*是否销毁线程池\*/
8. **int** shutdown;
9. pthread\_t \*threadid;
10. /\*线程池中允许的活动线程数目\*/
11. **int** max\_thread\_num;
12. /\*当前等待队列的任务数目\*/
13. **int** cur\_queue\_size;

}

线程池中逻辑

如果监听fd为空,则置于空闲状态 sleep

否则建立一个reactor循环监听所有fd的读写,执行逻辑

问题:

1.是否需要线程池 im应用多个reactor线程常驻cpu运行的,不是创建销毁型任务

2.主线程怎么发送fd给线程池中线程 通过管道fd放到reactor监听;

3.一个线程崩掉,整个进程崩掉这种情况如何处理?

4.主线程通过管道发送消息管理子线程,如发送EXIT\_CODE,在需要整个服务端中止运行,往各个子线 程发送exit\_code,使其退出,主线程终止事件循环 什么情况终止整个服务运行?

技术需求:分包处理 协议制定 心跳检测

健壮性需求:

当cpu,内存达到一定边界,限制子线程继续接收fd并监听

一个线程挂了,如果..

预先派生子进程/线程比现场创建子进程/线程要复杂很多，不仅要对池中进程/线程数量进行动态管理，还要解决多进程/多线程对accept的“抢” 问题

业务需求:

基本,核心:私聊 群聊 群组管理 好友管理 个人信息管理 动态管理(公共,类似微博.私有,类似微信朋友圈)

其他:文件传输 截屏