

Buffer: 数据的读写

InetAddress: 封装 IP 地址

TcpConnection: 封装一次 TCP 连接

TcpServer 为新连接创建对应的 TcpConnection 对象

TcpServer: 网络服务器, 接受客户的连接

EventLoopThreadPool: 创建 IO 线程池, 用于把 TcpConnection 分派到某个 EventLoop 线程上

EventLoopThread: 启动一个线程, 在其中运行 EventLoop::loop()

Connector: 用于发起 TCP 连接

Socket: 一个 RAIIhandle, 封装一个 filedescriptor, 并在析构时关闭 fd。

Acceptor: 用于接收客户端请求

Channel: 负责注册与响应 IO 事件

TcpConnection构造函数中创建Channel对象, 保存客户端套接字fd和关心的事件(可读)

EventLoop 相当于 Channel 和 Poller 之间的桥梁, Channel 和 Poller 之间并不之间沟通, 而是借助着 EventLoop 这个类。

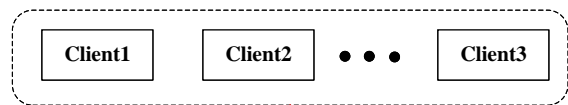
EpollPoller: Poller 的具体实现, Epoll 的具体封装

Poller: 负责负责监听文件描述符事件及返回件

抽象类

EventLoop: 负责 IO 和定时器事件的分派

Callbacks: 回调函数



mainLoop

subLoop1

subLoop2

subLoop3

TimerQueue: 定时器队列实现定时

一个EventLoop只持有一个TimerQueue对象, 而TimerQueue通过std::set持有多多个Timer对象, 但只会设置一个Channel。

事件驱动循环

1.runInLoop: 果当前线程是创建当前EventLoop对象的线程, 那么立即执行用户任务;

2.queueInLoop: 如果不是, 那么在loop循环中排队执行(本次循环末尾)

- 1) 监听套接字, 即本地ip地址&端口;
- 2) 监听通道事件, 读事件。

更新监听事件

监听多个 Channel

activeChannels_

回调函数都是在 EventLoop 所在线程执行

绑定 timerfd, 交由 Poller 监听, 发生可读事件(代表超时)后加入激活通道列表, 然后 EventLoop::loop() 逐个 Channel 调用对应的回调, 从而处理超时事件。