sync.pool 用过吗，了解的话讲一下

主要是为了保存和复用一些临时对象，避免频繁的内存分配，减少GC压力

go 里面互斥锁和读写锁有什么区别

普通互斥锁读写都会阻塞，读写锁在纯读操作的时候不会阻塞，写操作之间互斥，读写之间也会互斥

使用互斥锁的时候会不会出现某个 goroutine 一直取不到锁的情况？互斥锁有哪些状态？如何保证公平？

不会，互斥锁有正常状态和饥饿状态两种状态。

正常状态下，所有等待锁的 goroutine 会按 FIFO 的顺序等待，等待队列中被唤醒的 goroutine 不会立刻持有锁，而是会和最新请求锁的 goroutine 竞争，不过因为新请求的 goroutine 正在占用 cpu，而且可能有很多，所以一般都竞争不过，这时候最新请求的 goroutine 会拿到锁，被唤醒的 goroutine 会回到队首继续等待。

如果队首的等待超过 1ms，那么就会进入饥饿模式，饥饿模式下最新来的 goroutine 不会和等待队列里的竞争，总是优先把锁给等待队列里的第一个。

当等待队列里的 goroutine 获取了锁并且它在队列末尾时，或者某个等待 goroutine 等待时间小于 1ms 时，那么互斥锁的状态会回到正常模式，如此往复。