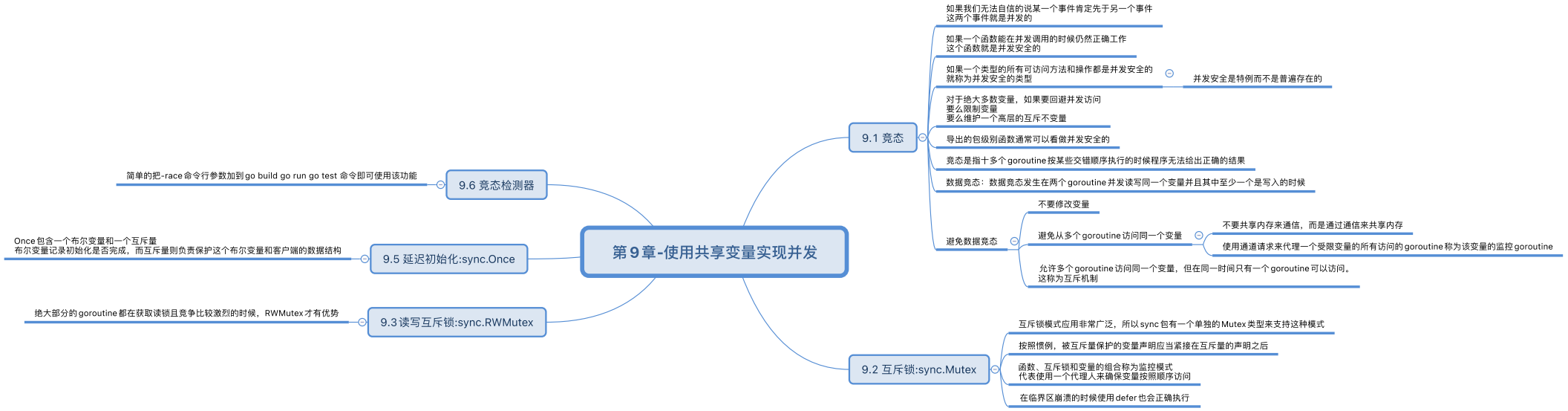
第9章-使用共享变量实现并发
9.1 竞态
如果我们无法自信的说某一个事件肯定先于另一个事件
这两个事件就是并发的
如果一个函数能在并发调用的时候仍然正确工作
这个函数就是并发安全的
如果一个类型的所有可访问方法和操作都是并发安全的
就称为并发安全的类型
并发安全是特例而不是普遍存在的
对于绝大多数变量，如果要回避并发访问
要么限制变量
要么维护一个高层的互斥不变量
导出的包级别函数通常可以看做并发安全的
竞态是指十多个goroutine按某些交错顺序执行的时候程序无法给出正确的结果
数据竞态：数据竞态发生在两个goroutine并发读写同一个变量并且其中至少一个是写入的时候
避免数据竞态
不要修改变量
避免从多个goroutine访问同一个变量
不要共享内存来通信，而是通过通信来共享内存
使用通道请求来代理一个受限变量的所有访问的goroutine称为该变量的监控goroutine
允许多个goroutine访问同一个变量，但在同一时间只有一个goroutine可以访问。
这称为互斥机制
9.2 互斥锁:sync.Mutex
互斥锁模式应用非常广泛，所以sync包有一个单独的Mutex类型来支持这种模式
按照惯例，被互斥量保护的变量声明应当紧接在互斥量的声明之后
函数、互斥锁和变量的组合称为监控模式
代表使用一个代理人来确保变量按照顺序访问
在临界区崩溃的时候使用defer也会正确执行
9.3 读写互斥锁:sync.RWMutex
绝大部分的goroutine都在获取读锁且竞争比较激烈的时候，RWMutex才有优势
9.5 延迟初始化:sync.Once
Once包含一个布尔变量和一个互斥量
布尔变量记录初始化是否完成，而互斥量则负责保护这个布尔变量和客户端的数据结构
9.6 竞态检测器
简单的把-race命令行参数加到go build go run go test 命令即可使用该功能