golang虽然相对其它语言来说算是一门新的编程语言，但是其实自它诞生以来也已经有了十年以上的历史了。最近这两年，golang语言变得异常火爆。

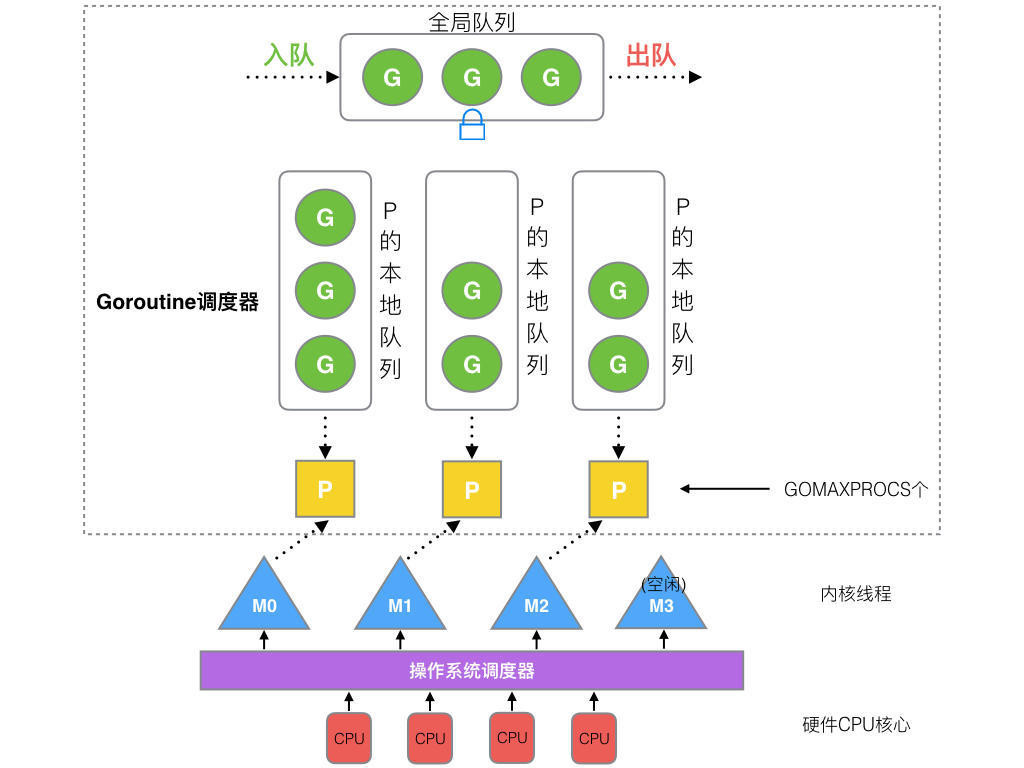
**内存管理**

golang中的内存管理和其它语言基本一样，也是分为堆区和栈区，较c语言进步的地方是它的内存垃圾回收是自动的，不需要程序员手动释放内存。下面总结几点golang中内存分配到堆区和栈区的几点特性

* Golang的运行时为每个goroutine分配一个栈区
* 当一个goroutine执行退出的时候，golang运行时并不立即清理栈区，它只是将它标记为不使用，当其它或者goroutine使用的时候在分配给它们使用。
* golang的运行时非常智能，如果一个变量的引用仍然存在，它会把它保存到堆上，这样其它的goroutine就可以访问它。
* 栈上的内存函数返回直接释放，堆上的内存需要垃圾回收
* golang的编译器会进行逃逸分析就是它会自主决定变量分配到堆上还是栈上。

**Goroutines**

golang语言最引以为傲的点应该就是goroutine了，它通过csp模型，解决了并发问题，而且它实现起来非常的简单优雅。

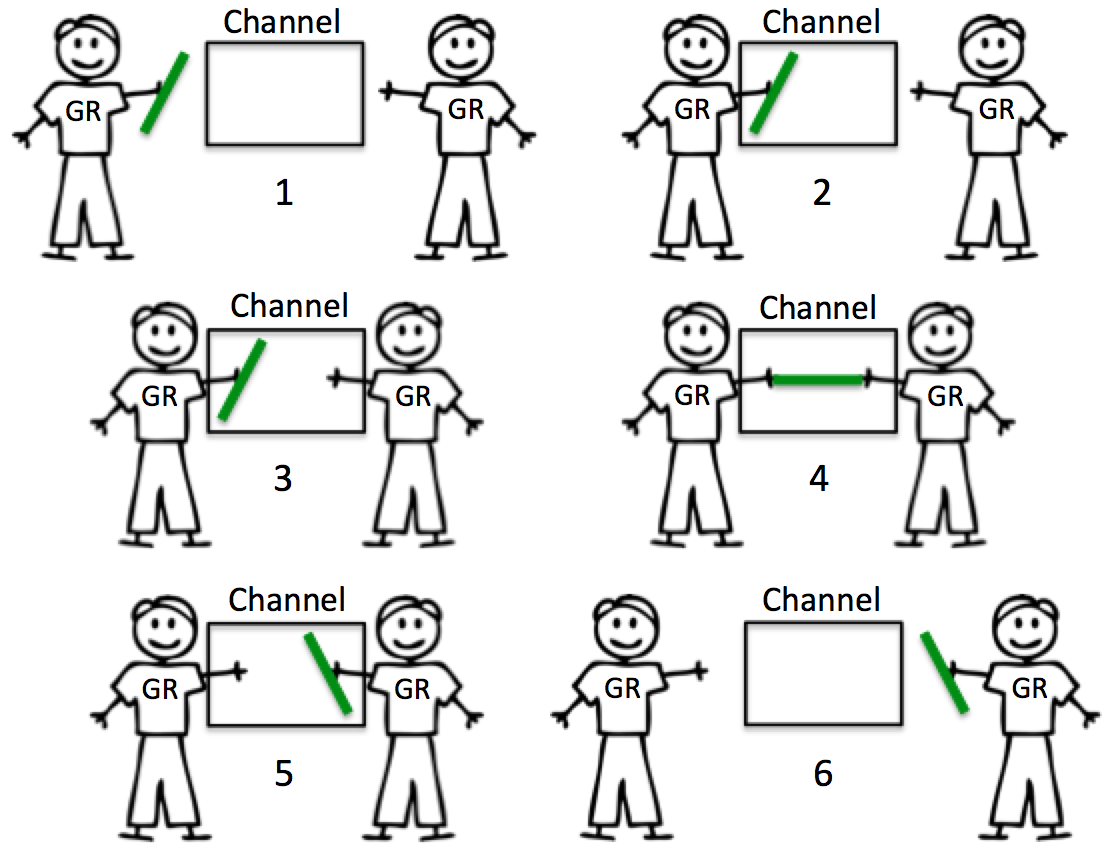


关于goroutine也有几点需要说明的。

* 每个程序至少包含一个goroutine，那就是main goroutine,当程序启动的时候，它会被自动创建出来。
* goroutine不是线程，它是用户级别的线程，是由用户自己管理的，不是操作系统管理的。
* goroutine很轻量，因此涉及到并发的时候，我们可以非常方便地启动多个goroutine。
* 一个goroutine里面还可以启动goroutine。
* goroutine之间通过管道进行通信。

**Channel**

channel是golang语言一个特殊的数据结构，goroutine就是通过通道进行通讯，通道类似于一个队列，遵循先进先出原则。

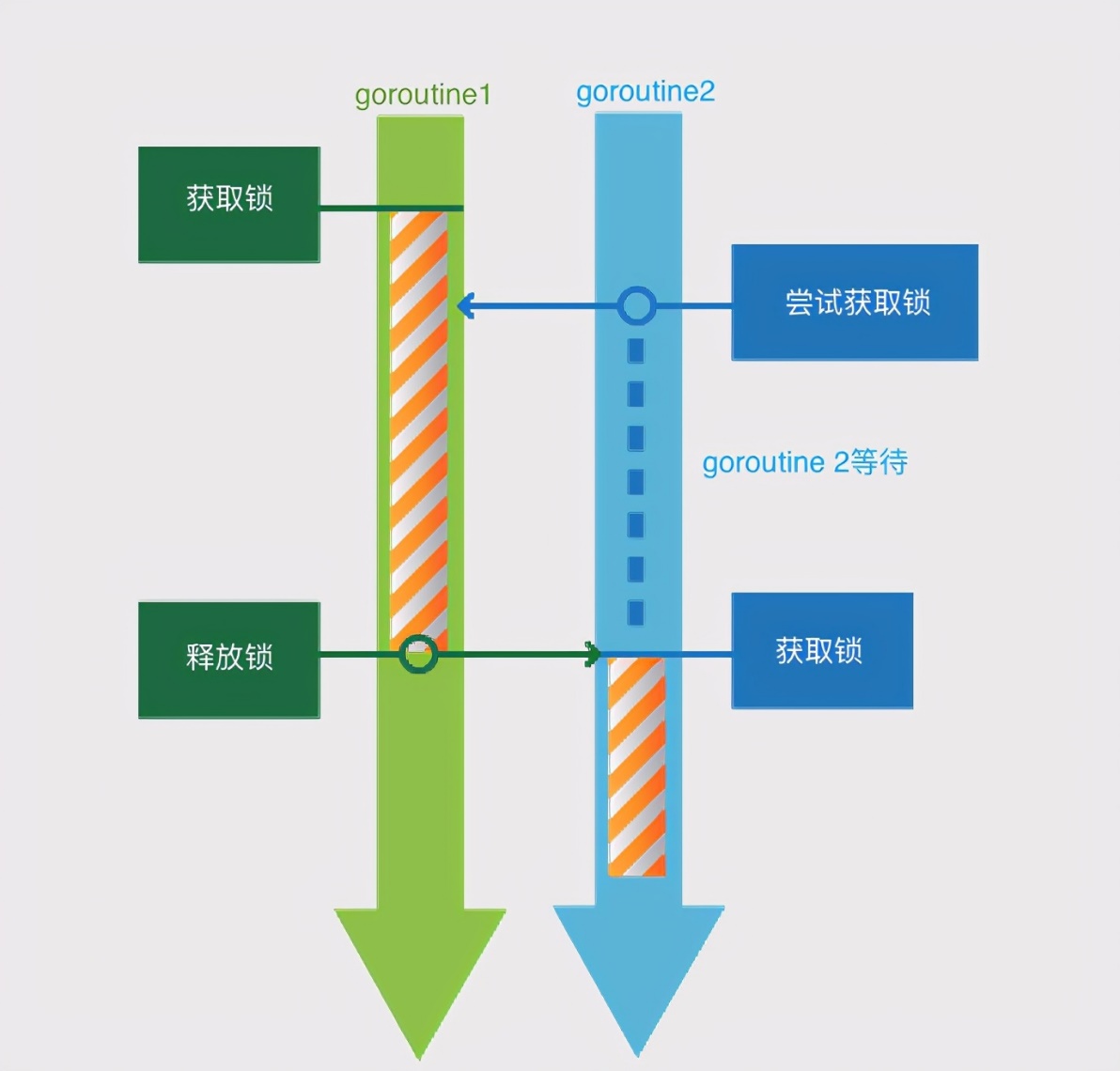


* channel是有方向的，可以单向，也可以双向。这里的单向指的是在同一个协程中只能读取或者只能写入，在另一个协程中相反。
* channel是有类型的，创建的时候需要声明，之后只能接收该类型的信息。
* channel中的数据可以通过阻塞和非阻塞的方式接收。
* channel可以有缓冲区，也可以没有缓冲区。

**WaitGroup,Mutex,RWMutex**

当我们想要确定所有的goroutine是否都执行完毕的时候，我们需要一个同步机制，这个时候我们可以使用WaitGroup来进行同步，它可以控制一组gorouine都执行完毕。

Mutex是互斥锁，同一个时间只有一个gorouine获取到锁才能执行，其它的需要等待。

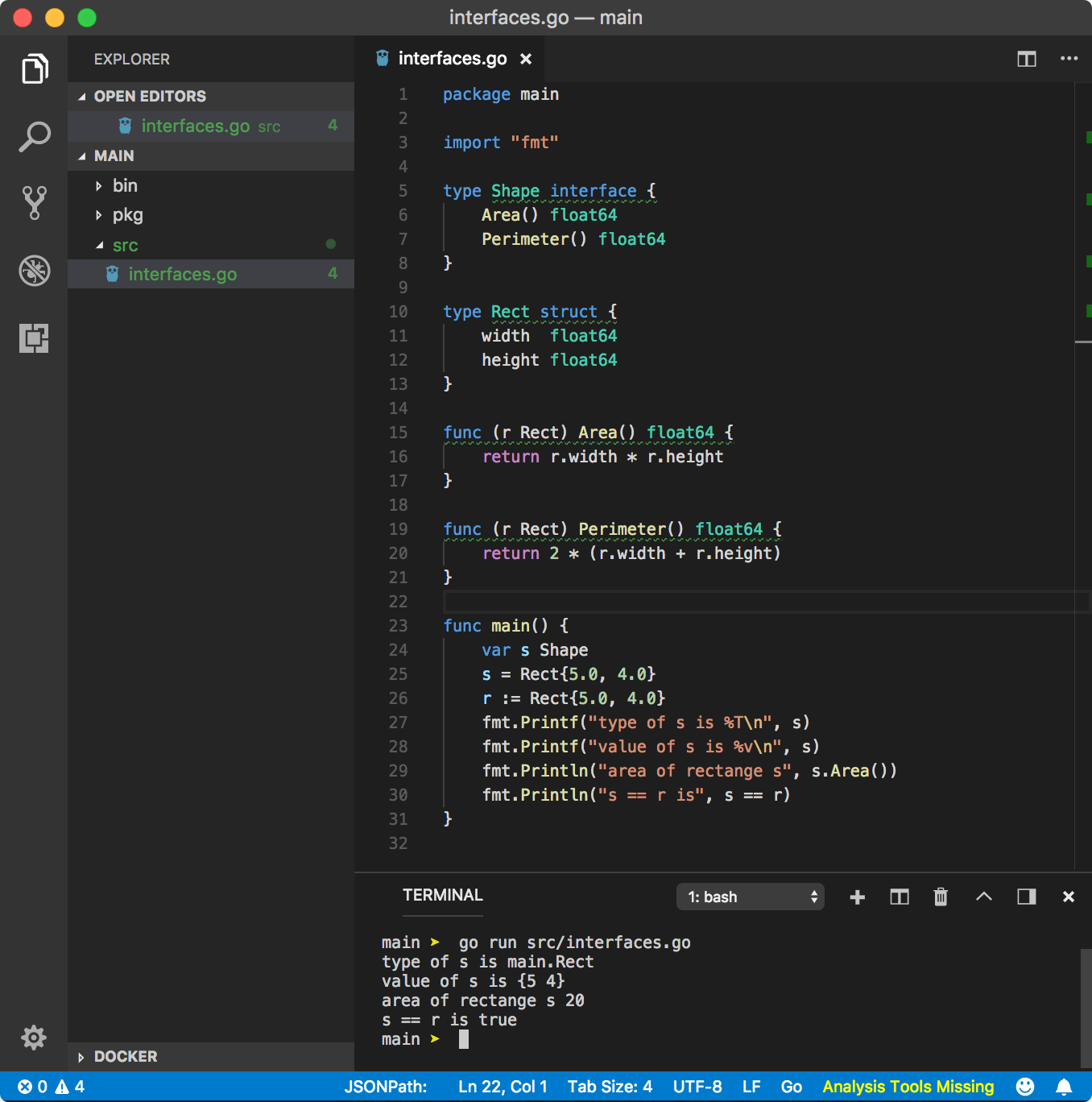


RWMutex是读写锁，顾名思义，当读的次数多的时候，我们可以加多个读锁，而写入操作的时候，只能加一个写锁。

锁的存在都是为了限制资源的竞争，防止资源被同时修改引起冲突。

**Interface**

golang语言中的接口是隐式实现的，也就是说它不需要显示声明，只要实现接口的方法，我们就说它实现了接口。



接口是golang语言中的一种类型，它可以成为变量的定义，函数的入参，函数的返回值。

结构体可以实现接口，结构体指针也可以实现接口，因为接口定义的时候没有指定类型。

golang中的接口不是任意类型，它是一种特殊类型。