* 并发（concurrency）：把任务在不同的时间点交给处理器进行处理。在同一时间点，任务并不会同时运行。
* 并行（parallelism）：把每一个任务分配给每一个处理器独立完成。在同一时间点，任务一定是同时运行。

如果希望让 goroutine 并行，必须使用多于一个逻辑处理器。当有多个逻辑处理器时，调度器会将 goroutine 平等分配到每个逻辑处理器上。这会让 goroutine 在不同的线程上运行。不过要想真的实现并行的效果，用户需要让自己的程序运行在有多个物理处理器的机器上。否则，哪怕 Go语言运行时使用多个线程，goroutine 依然会在同一个物理处理器上并发运行，达不到并行的效果。

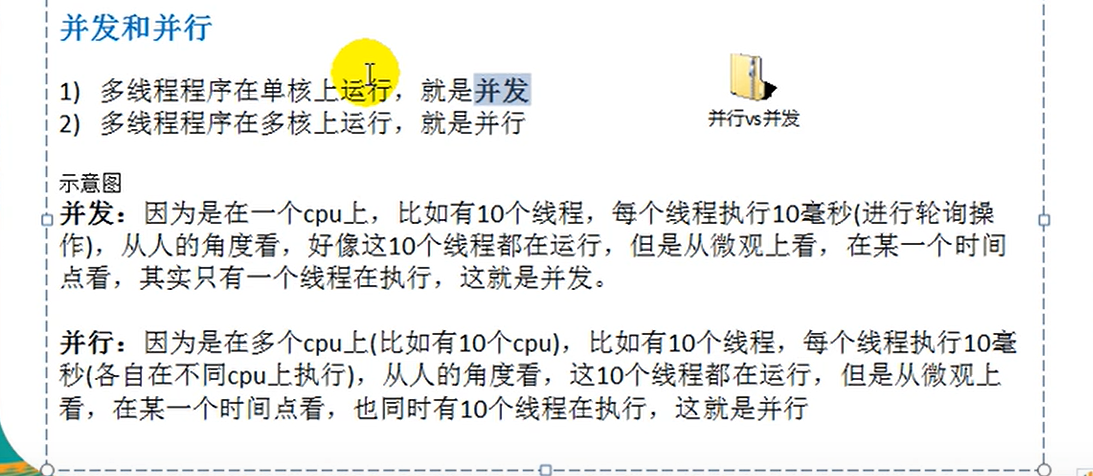
Go语言在 GOMAXPROCS 数量与任务数量相等时，可以做到并行执行，但一般情况下都是并发执行。

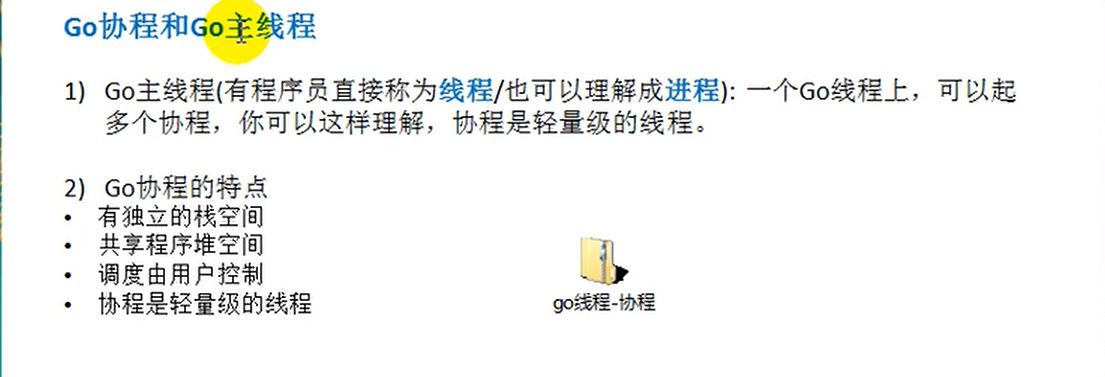
示例代码如下：

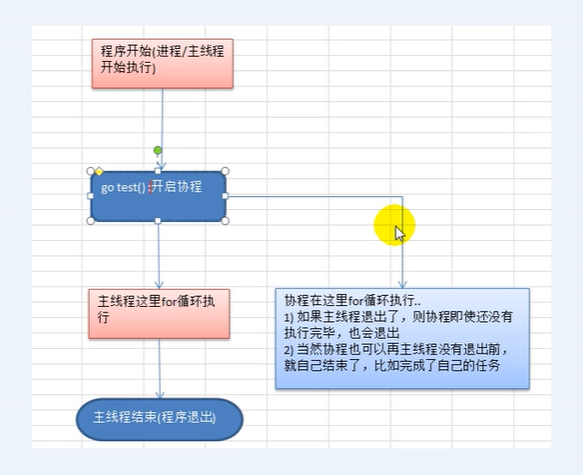
package main

1. import (
2. "fmt"
3. "runtime"
4. )
5. func main() {
6. cpuNum := runtime.NumCPU() //获得当前设备的cpu核心数
7. fmt.Println("cpu核心数:", cpuNum)
8. runtime.GOMAXPROCS(cpuNum) //设置需要用到的cpu数量
9. }



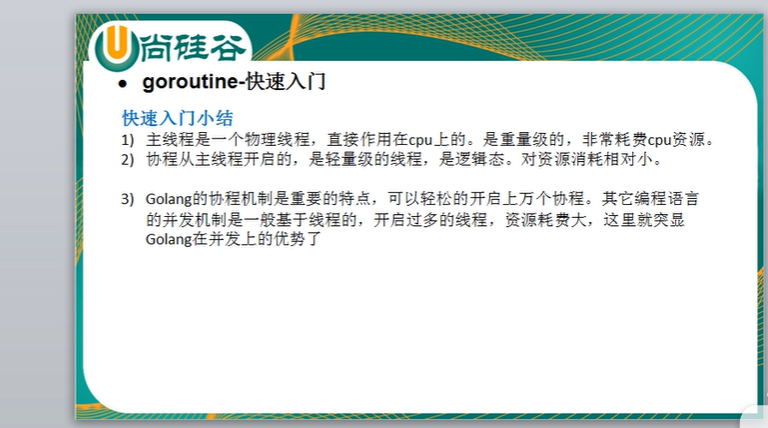






**多个协程可以同时读取一个空间，但是不能同时往一个空间写入数据；比如能同时阅读一本书但是不能同时写一本书；**





1. //go 关键字放在方法调用前新建一个 goroutine 并执行方法体
2. go GetThingDone(param1, param2);
3. //新建一个匿名方法并执行
4. go func(param1, param2) {
5. }(val1, val2)
6. //直接新建一个 goroutine 并在 goroutine 中执行代码块
7. go {
8. //do someting...
9. }