Go语言 等待组（sync.WaitGroup）

Go语言中除了可以使用通道（channel）和互斥锁进行两个并发程序间的同步外，还可以使用等待组进行多个任务的同步，等待组可以保证在并发环境中完成指定数量的任务

在 sync.WaitGroup（等待组）类型中，每个 sync.WaitGroup 值在内部维护着一个计数，此计数的初始默认值为零。

等待组有下面几个方法可用，如下表所示。

等待组的方法：

方法名	功能
(wg * WaitGroup) Add(delta int)	等待组的计数器 +1
(wg * WaitGroup) Done()	等待组的计数器 -1
(wg * WaitGroup) Wait()	当等待组计数器不等于 0 时阻塞直到变 0。

对于一个可寻址的 sync.WaitGroup 值 wg：

- 我们可以使用方法调用 wg.Add(delta) 来改变值 wg 维护的计数。
- 方法调用 wg.Done() 和 wg.Add(-1) 是完全等价的。
- 如果一个 wg.Add(delta) 或者 wg.Done() 调用将 wg 维护的计数更改成一个负数，一个恐慌将产生。
- 当一个协程调用了 wg.Wait() 时，
    - 如果此时 wg 维护的计数为零，则此 wg.Wait() 此操作为一个空操作（noop）；
    - 否则（计数为一个正整数），此协程将进入阻塞状态。当以后其它某个协程将此计数更改至 0 时（一般通过调用 wg.Done()），此协程将重新进入运行状态（即 wg.Wait() 将返回）。

等待组内部拥有一个计数器，计数器的值可以通过方法调用实现计数器的增加和减少。当我们添加了 N 个并发任务进行工作时，就将等待组的计数器值增加 N。每个任务完成时，这个值减 1。同时，在另外一个 goroutine 中等待这个等待组的计数器值为 0 时，表示所有任务已经完成。

示例：

```
package main

import (
    "fmt"
    "net/http"
    "sync"
)

func main() {
    // 声明一个等待组 对一组等待任务只需要一个等待组，而不需要每一个任务都使用一个等待组
    var wg sync.WaitGroup

    // 准备一系列的网站地址的字符串切片
    var urls = []string{
```

```
        "http://www.baidu.com/",
        "https://www.mi.com/",
        "https://pkg.go.dev/",
    }

    // 遍历这些地址
    for _, url := range urls {
        // 每一个任务开始时，将等待组增加1，也就是每一个任务加 1
        wg.Add(1)
        // 将一个匿名函数开启并发
        go func(url string) {
            // 使用defer，表示函数完成时将等待组值减1
            defer wg.Done()
            // 使用http访问提供的地址，Get() 函数会一直阻塞直到网站响应或者超时
            _, err := http.Get(url)
            // 访问完成后，打印地址和可能发生的错误
            fmt.Println(url, err)
            // 通过参数传递url地址
        }(url)
    }

    // 等待所有的任务完成，任务完成，Wait 就会停止阻塞
    wg.Wait()
    fmt.Println("over")
}
#结果
http://www.baidu.com/ <nil>
https://www.mi.com/ <nil>
https://pkg.go.dev/ <nil>
over
```