context包

并发控制：

1. waitgroup
2. channel
3. context ==> 上下文

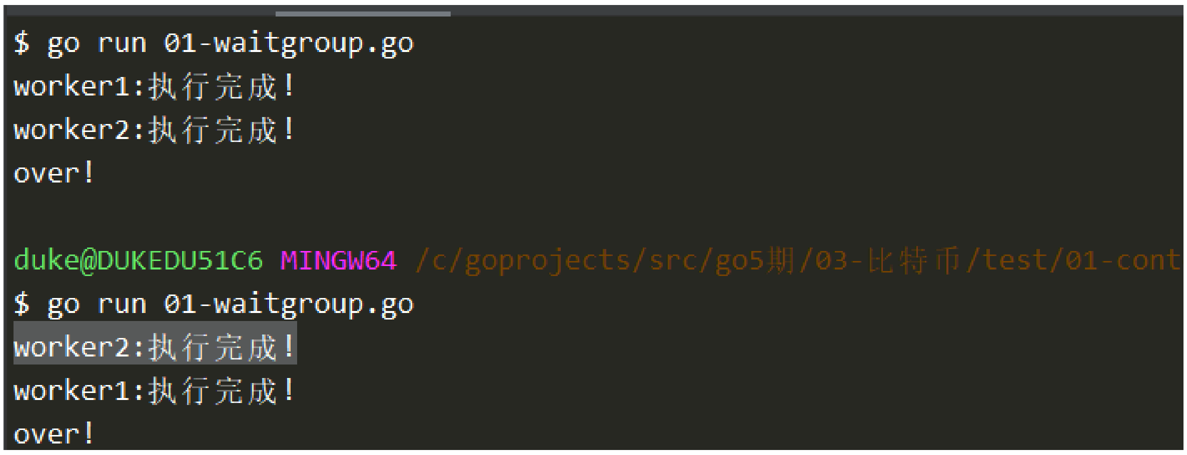
waitgroup

用途：控制多个goroutinue先后完成

方法：

1. Add() ==>添加监听的go + 1
2. Done() ==》表示go自己结束了 -1
3. Wait() == 0

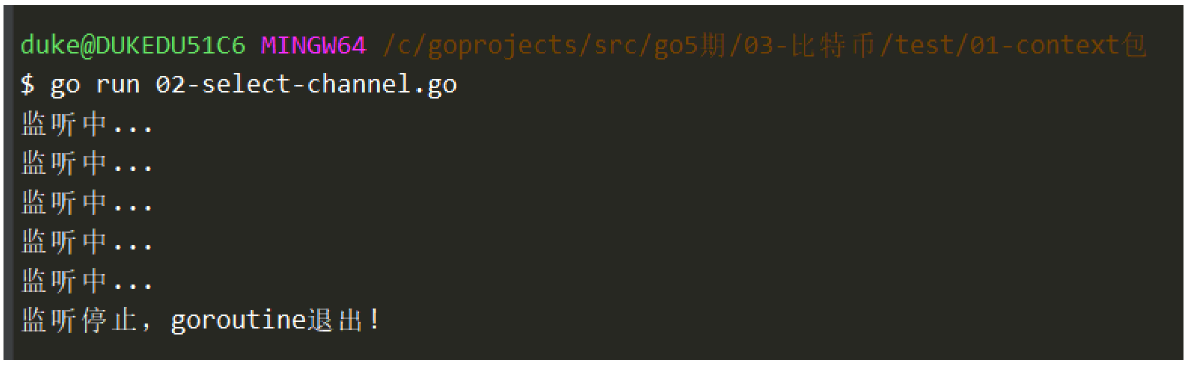
```
package main

import (
    "fmt"
    "sync"
    "time"
)

func main() {
    var wg sync.WaitGroup

    wg.Add(2)
    go watch("worker1", &wg)
    go watch("worker2", &wg)

    wg.Wait()
    fmt.Println("over!")
}

//使用多个go执行的函数
func watch(name string, wg *sync.WaitGroup) {
    time.Sleep(time.Second * 2)
    fmt.Println(name + ":执行完成!")
    wg.Done()
}
```



```
$ go run 01-waitgroup.go
worker1:执行完成!
worker2:执行完成!
over!

duke@DUKEDU51C6 MINGW64 /c/goprojects/src/go5期/03-比特币/test/01-cont
$ go run 01-waitgroup.go
worker2:执行完成!
worker1:执行完成!
over!
```

channel + select

```go
package main

import (
	"fmt"
	"time"
)

func main() {
	stopFlag := make(chan bool)

	go func() {
		for {
			select {
			case <-stopFlag:
				fmt.Println("监听停止, goroutine退出!")
			default:
				fmt.Println("监听中...")
				time.Sleep(1 * time.Second)
			}
		}
	}()

	time.Sleep(5 * time.Second)

	stopFlag <- true
}
```

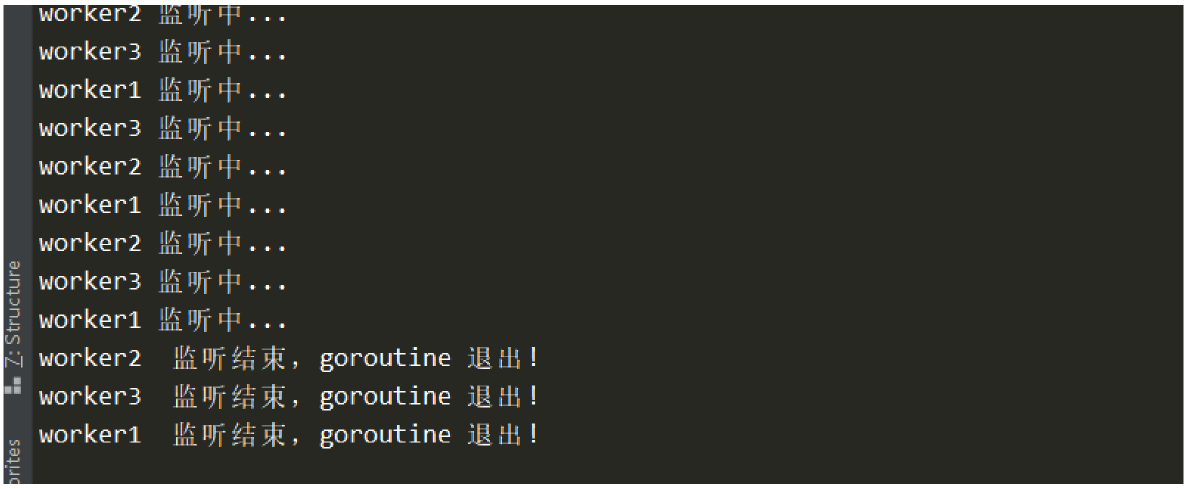


```
duke@DUKEDU51C6 MINGW64 /c/goprojects/src/go5期/03-比特币/test/01-context包
$ go run 02-select-channel.go
监听中...
监听中...
监听中...
监听中...
监听中...
监听停止, goroutine退出!
```

context

如果有很多goroutine都需要控制结束的话，使用上述两种方式很难管理，使用context可以有效解决。

```go
package main

import (
    "context"
    "fmt"
    "time"
)

func main() {

    //创建一个可以手动取消的contex函数
    //func WithCancel(parent Context) (ctx Context, cancel CancelFunc)
    //参数：根context，一般出入contex.Backgroud()
    //返回值1：子context，用于向内层传递
    //返回值2：cancel函数，回调函数，我们显示调用的时候，可以出发所有context返回
    cxt, cancel := context.WithCancel(context.Background())
    go watch("worker1", cxt)
    go watch("worker2", cxt)
    go watch("worker3", cxt)

    time.Sleep(5 * time.Second)
    cancel()
    time.Sleep(2 * time.Second)
}

//使用多个go执行的函数
func watch(name string, cxt context.Context) {
    fmt.Println("context watch called")
    for {
        select {
        case <-cxt.Done():
            fmt.Println(name, " 监听结束，goroutine 退出!")
            return
        default:
            fmt.Println(name + " 监听中...")
            time.Sleep(1 * time.Second)
        }
    }
}
```



```
worker2 监听中...
worker3 监听中...
worker1 监听中...
worker3 监听中...
worker2 监听中...
worker1 监听中...
worker2 监听中...
worker3 监听中...
worker1 监听中...
worker2  监听结束，goroutine 退出!
worker3  监听结束，goroutine 退出!
worker1  监听结束，goroutine 退出!
```