

# Go语言入门到精通

江洲老师云课堂

—— 主讲：江洲老师 ——

select

# 目录



PART 01  
**select作用**



PART 02  
**超时**



# select作用

1.select作用



## select作用

---

Go里面提供了一个关键字select，通过**select可以监听channel上的数据流动。**

select的用法与switch语言非常类似，**由select开始一个新的选择块，每个选择条件由case语句来描述。**

与switch语句相比，select有比较多的限制，其中最大的一条限制就是**每个case语句里必须是一个IO操作**，大致的结构如下：

```
select {  
    case <- chan1:  
        // 如果chan1成功读到数据，则进行该case处理语句  
    case chan2 <- 1:  
        // 如果成功向chan2写入数据，则进行该case处理语句  
    default:  
        // 如果上面都没有成功，则进入default处理流程  
}
```



## select作用

---

在一个select语句中，Go语言会按顺序从头至尾评估每一个发送和接收的语句。

- 1、如果其中的任意一语句可以继续执行(即没有被阻塞)，那么就从那些可以执行的语句中任意选择一条来使用。
- 2、如果没有任意一条语句可以执行(即所有的通道都被阻塞)，那么有两种可能的情况：
  - 如果给出了default语句，那么就会执行default语句，同时程序的执行会从select语句后的语句中恢复。
  - 如果没有default语句，那么select语句将被阻塞，直到至少有一个通信可以进行下去。



# 超时

1.超时





## 超时

有时候会出现goroutine阻塞的情况，那么我们如何避免整个程序进入阻塞的情况呢？  
我们可以利用select来设置超时，通过如下的方式实现：

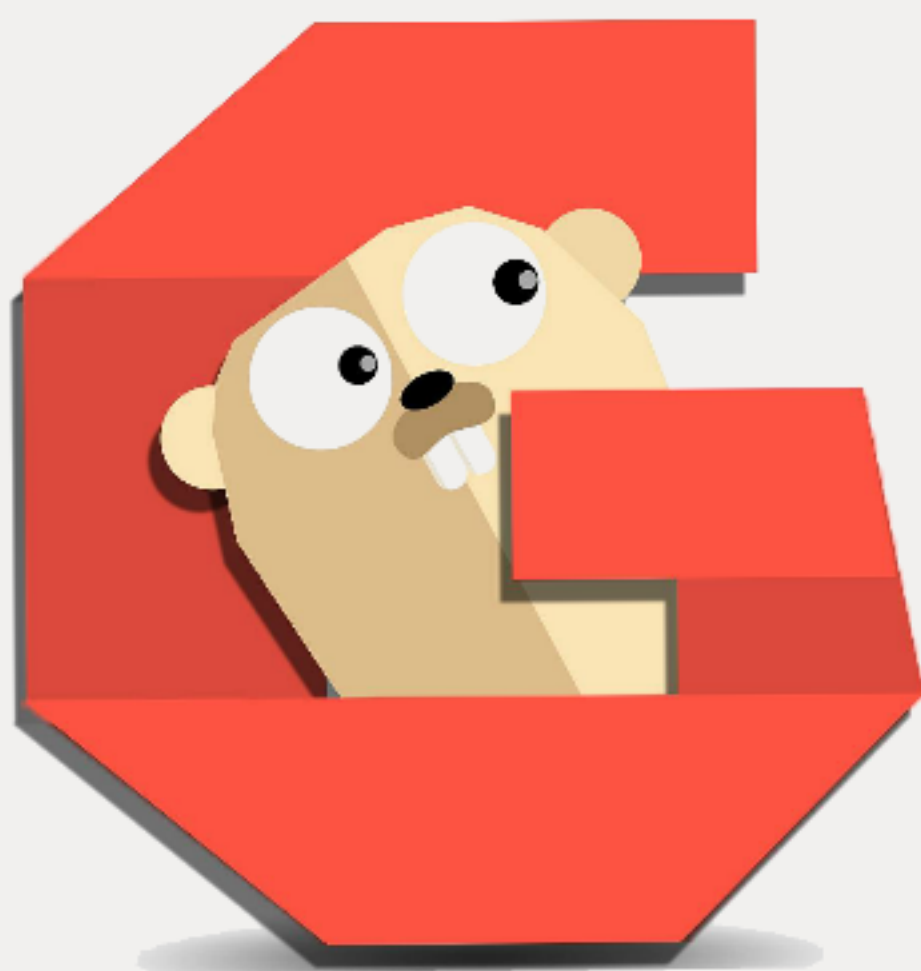
```
func main() {  
    c := make(chan int)  
    out := make(chan bool)  
    go func() {  
        for {  
            select {  
            case v := <-c:  
                fmt.Println(v)  
            case <-time.After(5 * time.Second):  
                fmt.Println("timeout")  
                out <- true  
                return  
            }  
        }  
    }()  
    //c <- 666           // 注释掉，引发 timeout  
    <-out  
}
```



## 实例演示







江洲老师云课堂

—— 主讲：江洲老师 ——

感谢您的聆听和观看