Go 并发元件

黎相敏

上海观源信息科技有限公司
上海市闵行区紫竹科技园 4 号楼 303B

12/01/2019

Focus

大纲

sync 包

基本元件

sync 包维护着用于同步底层内存访问的元件，包括

- WaitGroup
- Mutex 和 RWMutex
- Cond
- Once
- Pool

select 语句

SELECT 语句

- select 语句负责绑定通道到
  - 局部的类型或单个函数
  - 全局范围内多个系统组件
- 借助 select 语句安全地利用通道实现诸如取消、超时(代码片段 1)、等待和默认值等概念
- 空 select 语句 (select {}) 等价于一个死循环，会永远阻塞

代码片段 1: 基于 select 实现超时退出

```
1 var c <-chan int
2 select {
3 case <-c:
4 case <-time.After(1 * time.Second):
5     fmt.Println("Timed out.")
6 }
7
8 // 输出:
9 // Timed out.
```

CASE 子句的选择公平性

- 不同于switch语句，对于 select 语句
  - case子句不会线性地从上往下执行
  - 在没有任何满足条件的case子块时会一直阻塞
  - Go 运行时会随机均匀地选择非阻塞的case子句执行

case语句指明的所有通道读写操作会被同时检查是否已被满足，即

- 读操作的可读通道非空，或通道已关闭
- 写操作的可写通道非满

CASE 子句的选择公平性证明

```
1  c1 := make(chan interface{})
2  close(c1)
3  c2 := make(chan interface{})
4  close(c2)
5
6  var c1Count, c2Count int
7  for i := 1000; i >= 0; i-- {
8    select {
9    case <-c1:
10     c1Count++
11   case <-c2:
12     c2Count++
13   }
14 }
15 fmt.Printf("c1Count: %d\nc2Count: %d\n", c1Count, c2Count)
16
17 // 输出
18 // c1Count: 495
19 // c2Count: 506
```

DEFAULT 默认子句

- 用途: default子句负责执行其他case子句都不满足时的操作

```
start := time.Now()
var c1, c2 <-chan int

select {
case <-c1:
case <-c2:
default:
  fmt.Println("In default after", time.Since(start))
}

// 输出:
// In default after 1.482µs
```

- 通常default子句会与for-select循环结合使用。其中一个用例为：利用一个通道done作为终止循环的信号，而default子句负责执行终止之前的常规任务