Go语言入门到精通

江洲老师云课堂

主讲：江洲老师

互斥锁 (sync.Mutex)

目 录

死锁

1.锁的定义

2.死锁

什么是锁呢?

锁就是某个协程（线程）在访问某个资源时先锁住，防止其它协程的访问，等访问完毕解锁后其他协程再来加锁进行访问。

这和我们生活中加锁使用公共资源相似，例如：公共卫生间。

什么是死锁呢?

死锁是指两个或两个以上的进程在执行过程中，由于竞争资源或者由于彼此通信而造成的一种阻塞的现象，若无外力作用，它们都将无法推进下去。（此时称系统处于死锁状态或系统产生了死锁）

```
package main
import "fmt"
func main() {
    ch := make (chan int)
    ch <- 1                  // I'm blocked because there is no channel read yet.
    fmt.Println ("send")
    go func() {
        <-ch                 // I will never be called for the main routine is blocked!
        fmt.Println("received")
     }()
     fmt.Println("over")
}
```

互斥锁

1.互斥锁

互斥锁

每个资源都对应于一个可称为 "**互斥锁**" 的标记，这个标记用来保证在任意时刻，只能有一个协程（线程）访问该资源，其它的协程只能等待。

互斥锁是传统并发编程对共享资源进行访问控制的主要手段，它由标准库sync中的Mutex结构体类型表示。

sync.Mutex类型只有两个公开的指针方法，Lock和Unlock，Lock锁定当前的共享资源，Unlock进行解锁。

注意：对资源操作完成后，一定要解锁，否则会出现流程执行异常，死锁等问题。

通常借助defer，锁定后，立即使用defer语句保证互斥锁及时解锁。

```
var  mutex  sync.Mutex          // 定义互斥锁变量 mutex
func write (){
  mutex.Lock ( )
  defer  mutex.Unlock ( )
}
```

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