★ 首页 (/) / 网友博文 (/articles) / golang fatal error: all goroutines are asleep - deadlock!

## golang fatal error: all goroutines are asleep - deadlock!

```
≡ 2015-02-17 11:26 ♣ ghj1976
                                                            ● 阅读 3634 次 ♥ 1 人喜欢 严 0 条评论 (/articles/2410#commentForm) ☆ 收藏
channel默认上是阻塞的,也就是说,如果Channel满了,就阻塞写,如果Channel空了,就阻塞读。阻塞的含义就是一直等到轮到它为止。单有时候
我们会收到 fatal error: all goroutines are asleep - deadlock! 异常,这是如何呢?
代码例子:
package main
import "fmt"
func main() {
  channel := make(chan string, 2)
  fmt.Println("1")
  channel <- "h1"
  fmt.Println("2")
  channel <- "w2"
  fmt.Println("3")
  channel <- "c3" // 执行到这一步,直接报 error
  fmt.Println("...")
  msg1 := <-channel
  fmt.Println(msg1)
}
执行效果:
D:\mycodes\golang\src\test\testchan1>testchan1.exe
fatal error: all goroutines are asleep - deadlock!
goroutine 1 [chan send]:
 main.main()
         D:/mycodes/golang/src/test/testchan1/main.go:13 +0x364
D:\mycodes\golang\src\test\testchan1>
(http://images.cnitblog.com/blog/120296/201502/171125376424903.png)
参考:
http://stackoverflow.com/questions/26927479/go-language-fatal-error-all-goroutines-are-asleep-deadlock
(http://stackoverflow.com/questions/26927479/go-language-fatal-error-all-goroutines-are-asleep-deadlock)
fatal error: all goroutines are asleep - deadlock!
出错信息的意思是:
在main goroutine线,期望从管道中获得一个数据,而这个数据必须是其他goroutine线放入管道的
但是其他goroutine线都已经执行完了(all goroutines are asleep),那么就永远不会有数据放入管道。
所以,main goroutine线在等一个永远不会来的数据,那整个程序就永远等下去了。
这显然是没有结果的,所以这个程序就说"算了吧,不坚持了,我自己自杀掉,报一个错给代码作者,我被deadlock了"
这里是系统自动在除了主协程之外的协程都关闭后,做的检查,继而报出的错误, 证明思路如下, 在100秒内, 我们看不到异常, 100秒后,系统报
错。
package main
import (
  "fmt"
  "time"
func main() {
  channel := make(chan string, 2)
  go func() {
    fmt.Println("sleep 1")
    time.Sleep(100 * time.Second)
    fmt.Println("sleep 2")
```

```
fmt.Println("1")
channel <- "h1"
fmt.Println("2")
channel <- "w2"
fmt.Println("3")
channel <- "c3"
fmt.Println("...")
msg1 := <-channel
fmt.Println(msg1)
}
```

```
D:\mycodes\golang\src\test\testchan1>
D:\mycodes\golang\src\test\testchan1>
D:\mycodes\golang\src\test\testchan1>
D:\mycodes\golang\src\test\testchan1>
D:\mycodes\golang\src\test\testchan1>
D:\mycodes\golang\src\test\testchan1>testchan1.exe
1
2
3
sleep 1
```

(http://images.cnitblog.com/blog/120296/201502/171125385339231.png)

100秒后执行效果截图:

```
D:\mycodes\golang\src\test\testchan1>testchan1.exe

1

2

3

sleep 1

sleep 2

fatal error: all goroutines are asleep - deadlock!

goroutine 1 [chan send]:

main.main()

D:/mycodes/golang/src/test/testchan1/main.go:27 +0x375

D:\mycodes\golang\src\test\testchan1>
```

(http://images.cnitblog.com/blog/120296/201502/171125396586788.png)

如果避免上面异常抛出呢?这时候我们可以用 select来帮我们处理。

```
package main
import "fmt"
func main() {
   channel := make(chan string, 2)
   fmt.Println("1")
   channel <- "h1"
   fmt.Println("2")
   channel <- "w2"
   fmt.Println("3")
   select {
   case channel <- "c3":
     fmt.Println("ok")
   default:
     fmt.Println("channel is full !")
   fmt.Println("...")
   msg1 := <-channel
   fmt.Println(msg1)
}
执行效果:
```

```
\mycodes\golang\src\test\testchan1>testchan1.exe
channel is full !
h1
D:\mycodes\golang\src\test\testchan1>
```

(http://images.cnitblog.com/blog/120296/201502/171125406427386.png)

这时候,我们把第三个要写入的 chan 抛弃了。

上面的例子中是写的例子, 读的例子也一样, 下面的异常是 ws := <-channel 这一行抛出的。

channel := make(chan string, 2)

fmt.Println("begin") ws := <-channel fmt.Println(ws)

```
D:\mycodes\golang\src\test\testchan1>testchan1.exe
begin
fatal error: all goroutines are asleep - deadlock!
goroutine 1 [chan receive]:
main.main()
        D:/mycodes/golang/src/test/testchan1/main.go:9 +0x163
D:\mycodes\golang\src\test\testchan1>
```

(http://images.cnitblog.com/blog/120296/201502/171125415178486.png)

本文来自: 博客园 (/wr?u=http://www.cnblogs.com)

感谢作者: ghj1976

查看原文: golang fatal error: all goroutines are asleep - deadlock! (/wr?u=http%3a%2f%2fwww.cnblogs.com%2fghj1976%2fp%2f4295013.html)

♥ 1人喜欢

☆ 收藏

(http://www.jiathis.com/share?uid=1895190) •



- « (/articles/2409)上一篇: golang rpc 简单范例 (/articles/2409)
- » (/articles/2411)下一篇: Go语言\_并发篇 (/articles/2411)

文章点评: (您需要 登录 后才能评论 没有账号 (/user/register) ?)

编辑

我有话要说......

- 支持 Markdown 格式, \*\*粗体\*\*、~~删除线~~、`单行代码` 支持 @ 本站用户; 支持表情(输入:提示),见 Emoji cheat sheet (http://www.emoji-cheat-sheet.com/)

提交

最新主题 (/topics) | 最新资源 (/resources) |

粘贴代码的时候格式混乱 (/topics/2012)

can't load package: package internal: no buildable Go source files in (/topics/2011)

上海-大数据公司-GO语言开发 (/topics/2001)

Go最新资料汇总(八十) (/topics/2010)

【有偿】求Go语言大神帮忙一个小问题 (/topics/1977)

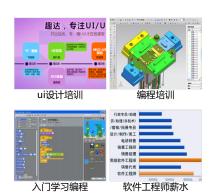
go接收到消息如何根据消息头判断接收 (/topics/1976)

使用go测试MySQL并发 不好使 (/topics/1975)

《The Go Programming Language》3.3节 习题 (/topics/2009)

beego1.6.1升级到1.7版本遇到的session问题 (/topics/2000)

golang能实现多进程吗(不是多线程,也不是协程) (/topics/1999)



开源项目 (/projects)

(/p/go-wechat-sdk)

Golang 开发的微信 SDK go-



(/p/leaps-golang)

Leaps (golang) (/p/leaps-



(/p/gabs)



Go 的 JSON 处理库 gabs (



(/p/graph) Go 语言的广义图形包 grapl



(/p/goraph)



实现图形数据结构和算法 gi



关于 (/wiki/about) | API (/api) | 贡献者 (/wiki/contributors) | 帮助推广 (/wiki) | 反馈 (/topics/node/16) | Github (http://github.com/studygolang) | 新浪微博 (http://weibo.com/studygolang) | 内嵌Wide (/wide/playground) | 免责声明 (/wiki/duty)

©2013-2016 studygolang.com 采用 Go语言 (http://golang.org) + MYSQL 构建 (http://www.mysql.com/) 当前在线: 30人 历史最高: 300人 运行时 间: 240h46m20.922220371s

网站编译信息 版本: V2.0.0/master-32141279e63198c1a1c2a3fbc086d5fac85adb0c, 时间: 2016-09-07 23:52:45.083794125 +0800 CST Go语言中文网,中国 Golang 社区,致力于构建完善的 Golang 中文社区,Go语言爱好者的学习家园。 京ICP备14030343号-1



(http://www.ucai.cn?fr=studygolang)



(http://click.aliyun.com/m/4526/)



(https://portal.qiniu.com/signup?code=3lfz4at7pxfma)