



首页	微博	讨论	文章	激励	活动	投票	团队会议	相册
----	----	----	----	----	----	----	------	----

## go语言规范指南

nickzydeng

2017年05月20日 19:52

浏览(554)

收藏(28)

评论(0)

分享

[| 导语](#) 此文是作者所在团队约定的编码规范

- 写完代码都必须格式化，保证代码优雅: gofmt goimports
- 编译前先执行代码静态分析: go vet pathxxx/ 可以检测出如下的常见错误：

```
//Print-format 打印格式错误
var str string
fmt.Printf("str:%d", str)
//Boolean 可疑布尔值
var i int
fmt.Println(i==0 && i==1)
//Range 闭包变量问题
words := []string{"f","b"}
for _, w := words {
    go func() {
        fmt.Println(w)
    }()
}
//位移问题
fmt.Println(i>>32)
//提前结束，后续代码不可达
return
```

- 竞态检测: go build -race (测试环境编译时加上 -race 选项，生产环境必须去掉，因为race 限制最多goroutine数量为8192个)
- package 名字:包名与目录保持一致，尽量有意义，简短，不和标准库冲突，全小写，不要有下划线
- 每行长度约定: 一行不要太长，超过请使用换行展示，尽量保持格式优雅；单个文件也不要太大，最好不要超过500行
- 错误处理机制:使用多返回值 error，不要使用 c 风格，返回错误码，错误 信息放在输入参数，特别是封装 cgo 的函数

### 关于作者

nickzydeng(邓之银)

SNG\SNG内容平台部基础后台开发...

### 作者文章

- Golang后台server配合织云实现热重启平滑升级
- 【Going项目】基于go语言的后台应用框架
- mnet2免密码登录mnet1
- windows开发机安装spark2.2 scala2.11 intelj
- Going服务性能分析报告

猜你喜欢

更多>>

- TDSQL-时态数据库T-TDSQL-理念与愿景
- 产品经理七年私藏工具大推荐
- 升级gcc版本为gcc-4.8.5，支持c++11
- TSF服务注册/配置中心Consul-Raft解析
- GLibc协程切换汇编理解

7, 多返回值最多返回三个, 超过三个请使用 struct

8. 请求路径携带标准库 context 上下文: 一个请求内的所有处理逻辑都必须带上 context, 并且以函数第一个参数传入, context 可以取消, 超时, 传值

9. 注意 slice 引用问题: 使用 copy 防止数据篡改, 防止无法 GC 而导致内存泄漏

10. new 新对象采用多行, 更加一目了然

```
p := &StructName{  
  
    field1: 1,  
  
    field2: "aaaa",  
}
```

11. 变量名采用驼峰法, 不要有下划线, 不要全部大写

12. 时刻提醒自己 goroutine 的生命周期, 注意同步 channel, 防止 goroutine 堵住, 释放不了, 出现内存泄漏

13. 注意关闭 channel 问题, 不能 close 两次, 不能向 closed 的 chan 写入数据

14. 不可在 goroutine 内部做耗时的纯逻辑计算工作, 如死循环, cgo 同步函数, cgo 系统调用

15. 自定义类型的 String 方法死循环问题, 不可在 String 方法内使用 fmt 打印自己

```
type MyInt int  
  
func (m MyInt) String() string {  
  
    return fmt.Sprint(m) //BUG:死循环  
}  
  
func (m MyInt) String() string {  
  
    return fmt.Sprint(int(m)) //这是安全的, 因为我们内部进行了类型转换  
}
```

16. 注释规范: 每个包, 可导出 struct, 函数, 常量, 变量, 都必须要有注释, 格式是紧挨着的上一行, 这样写:

```
// Package pkgname xxxx  
// FuncName xxxx  
// TODO xxxx  
// BUG xxxx
```

每一个对外服务的接口都要注释接口功能、调用方, 每个后端调用都要注释后台功能、负责人, 这样方便联调对接以及后续交接维护

17. 在逻辑处理中禁用 panic, 除非你知道你在做什么

18. 不要忽略 error，每个 error 都必须处理，实在不用处理，就打印 error。有 错误尽早返回  
不要这样写:

```
if err != nil {  
  
    // error handling  
  
} else {  
  
    // normal code  
  
}
```

而应该是:

```
if err != nil {  
  
    // error handling  
  
    return // or continue, etc.  
}  
  
// normal code
```

19. 常用的首字母缩写名词，使用全小写或者全大写，如 UIN URL HTTP ID IP OK

20. Receiver: : 用一两个字符，能够表示出类型，不要使用 me self this

21. 参数传递:

- 对于少量数据，不要传递指针
- 对于大量数据的 struct 可以考虑使用指针
- 传入参数是 map，slice，chan，interface，string 不要传递指针

22. 每个基础库都必须有实际可运行的例子，基础库的接口都要有单元测试用例

23. 尽量少用命名多返回值，如果要用，必须显示 return 结果，因为作用域问题容易出bug，  
例如：

```
func test1() (a int, e error) {  
    return 1, errors.New("test1")  
}  
  
func test2() (a int, e error) {  
    a, e = 10, nil  
    for {  
        aa, e := test1()    // 注意: 这里的 e 是新的变量  
        if aa == 1 || e != nil {  
            break  
        }  
    }  
    return // 这里返回的e不是预期错误，而是nil  
}
```

24. ctx是当前请求上下文，在请求退出时会cancel掉，在自己另起的goroutine里面不要使用同一个ctx请求后端
25. http服务在超时以及客户端主动断开连接的时候会有另外的goroutine写context，业务代码里面不能直接打印整个ctx，如 `fmt.Printf("%+v", ctx)`，否则会导致服务crash
26. 不要在for循环里面使用defer，defer只有在函数退出时才会执行
27. udp请求使用net.DialUDP时，内部会调用connect，只能接收来自对端的回包，这对于非原路径回包的场景下会有问题
28. 禁止在业务代码里面调用cgo
29. panic捕获只能到goroutine最顶层，每个自己启动的goroutine，必须在入口处就捕获panic，并打印出详细的堆栈信息
30. golang的内置类型slice，map，chan都是引用，初次使用前，都必须先用make分配好对象，不然会有空指针异常
31. 使用map时需要注意：map初次使用，必须用make初始化；map是引用，不用担心赋值内存拷贝；并发操作时，需要加锁；range遍历时顺序不确定，不可依赖；不能使用slice，map和func作为key
32. mysql scan过程中，不允许执行其他sql语句，否则会导致数据集错乱

如果觉得我的文章对您有用，请随意赞赏

赏

仅供内部学习与交流，未经授权切勿外传

分类：[开发](#) 标签：[go\(4\)](#) [规范\(4\)](#) [GoLang\(2\)](#)



本文专属二维码，扫一扫还能分享朋友圈

想要微信公众号推广本文章？[点击获取链接](#)

#### 相关阅读

- [【Going项目】基于go语言的后台应用框架](#)
- [go语言测试规范](#)
- [【成都业务组】go登录规范](#)
- [IAB规范-VAST3.0综合介绍](#)
- [Go高性能编程技巧](#)

我顶 (4)

收藏 (28)


分享到

转载 (1)

收录

评论 (0)

大家评论



切换到更多功能

发表评论