

Golang基础

15:04 19 Sep 2017

Wubo

内容

- 概述
- 语法
- channel
- goroutine
- 开发环境
- 常用命令
- 学习

概述

- 出身
- 特点
- 编程哲学

出身

- google出品

• 肯·汤普逊 (Ken Thompson) : 设计了B语言和C语言, 创建了Unix和Plan 9操作系统, 1983年图灵奖得主, Go语言的共同作者。

• 罗布·派克 (Rob Pike) : Unix小组的成员, 参与Plan 9和Inferno操作系统, 参与Limbo和Go语言的研发, 《Unix编程环境》作者之一
- 还有很多大牛

特点

- 高并发
- 高性能
- 安全
- 快编译, 无依赖
- 易移植
- 简单
- 基础库
- 开发效率高 (垃圾回收)

编程哲学

- KISS
- 提倡组合，反对面向对象
- 抽象定义，实现体现细节

语法

- 数据类型
- 关键字
- 表达式
- 新特性

数据类型-基础类型

- 整形: int8, int16, int, int32, int64
- 无符号整型: uint8, uint16, uint, uint32, uint64
- 浮点型: float32, float64
- 复数类型: complex64, complex128
- 字节类型: byte
- 布尔类型: bool
- 错误类型: error
- 字符类型: rune

数据类型-复合结构

- 数组: `[size]Type`
- 切片(slice): `[]Type`
- 映射(map): `map[Type]type`
- 指针: `*Type`
- 结构体: `Struct`
- 通道: `chan`
- 接口: `interface`

关键字-1

- var, const
- package, import
- func, return, defer
- go, select, chan
- type
- interface
- struct
- map

关键字-2

- range
- break、 case、 continue、 for、 fallthrough、 else、 if、 switch、 goto、 default

表达式

- $*, /, \%, <<, >>, \&, \&^$
- $+, -, |, ^$
- $==, !=, <, <=, >=$
- $\&\&$
- $||$

与C/C++语言不同点

- 数据类型不支持隐式转换
- 不支持指针运算操作
- switch分支执行结果直接退出switch语句(默认支持break)
- 没有extern, static, private, public, protected
- 没有继承,但是有新玩意interface
- C/C++头文件引用(include), golang是包引用(import)

defer

defer 延迟执行，当函数执行结束前执行一次

Once

Go语言提供了一个Once类型来保证全局的唯一性操作。Once的Do ()方法可以保证在全局范围内只调用指定的函数一次，而且所有其他goroutine在调用到此语句时，将会先被阻塞，直至全局唯一的Once.Do ()调用结束才继续

Once

```
1 package main
2
3 import (
4     "fmt"
5     "sync"
6 )
7
8 var count int
9 var once sync.Once
10
11 func allclosure() int {
12
13     onceDo := func() {
14         count = 100
15     }
16     once.Do(onceDo)
17     count := count + 1
18     return count
19 }
20
21 func onceHead() {
22     onceDo := func() {
23         fmt.Println("start")
24     }
25     var once sync.Once
26     for i := 0; i < 10; i++ {
27         once.Do(onceDo)
```



```
28         fmt.Println(i)
29     }
30 }
31
32 func main() {
33     onceHead()
34 }
```

panic与recover

```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func safedivide(a, b int) int {
8     defer func() {
9         if p := recover(); p != nil {
10             fmt.Println("occur error:", p)
11         }
12     }()
13
14     if a == 0 {
15         panic("invalid num zero")
16     }
17     fmt.Println("result:", b/a)
18     return b / a
19 }
20
21 func main() {
22     safedivide(1, 35)
23     safedivide(0, 10)
24 }
```

interface

```
1 // _接口_ 是方法特征的命名集合。
2
3 package main
4
5 import "fmt"
6 import "math"
7
8 // 这里是一个几何体的基本接口。
9 type geometry interface {
10     area() float64
11     perim() float64
12 }
13
14 // 在我们的例子中，我们将让 `square` 和 `circle` 实现
15 // 这个接口
16 type square struct {
17     width, height float64
18 }
19 type circle struct {
20     radius float64
21 }
22
23 // 要在 Go 中实现一个接口，我们只需要实现接口中的所有
24 // 方法。这里我们让 `square` 实现了 `geometry` 接口。
25 func (s square) area() float64 {
26     return s.width * s.height
27 }
```

```
28 func (s square) perim() float64 {
29     return 2*s.width + 2*s.height
30 }
31
32 // `circle` 的实现。
33 func (c circle) area() float64 {
34     return math.Pi * c.radius * c.radius
35 }
36 func (c circle) perim() float64 {
37     return 2 * math.Pi * c.radius
38 }
39
40 // 如果一个变量的是接口类型，那么我们可以调用这个被命名的
41 // 接口中的方法。这里有一个一通用的 `measure` 函数，利用这个
42 // 特性，它可以用在任何 `geometry` 上。
43 func measure(g geometry) {
44     fmt.Println(g)
45     fmt.Println(g.area())
46     fmt.Println(g.perim())
47 }
48
49 func main() {
50     s := square{width: 3, height: 4}
51     c := circle{radius: 5}
52
53     // 结构体类型 `circle` 和 `square` 都实现了 `geometry`
54     // 接口，所以我们可以使用它们的实例作为 `measure` 的参数。
55     measure(s)
```

```
56     measure(c)
```

```
57 }
```

channel

- 定义
- 特点
- 总结

定义

- channel为goroutine之间通信服务
- Do not communicate by sharing memory; instead, share memory by communicating.

特点

- FIFO
- 原子性
- 两种模式：同步模式 vs 队列模式
- 包容万象(interface, func ...)

总结

channel状态与操作之间关系

状态/操作	写操作	读操作	关闭	创建
nil状态	写阻塞	写阻塞	产生panic(close of nil channel)	-
同步写阻塞	写阻塞	成功读取数据	进入关闭状态，产生panic	-
同步读阻塞	成功写入数据	读阻塞	进入关闭状态	-
关闭状态	产生panic	立即返回(nil, false)	产生panic	-
队列写阻塞	写阻塞	成功读取队列中数据	进入关闭状态，成功写入队列的数据可读	-
队列读阻塞	成功写入数据	读阻塞	进入关闭状态	-
队列可读写	成功写入数据	成功读取数据	进入关闭状态，成功写入队列的数据可读	-

goroutine

- 定义
- 周期
- 应用

定义

goroutine是一种协程
实现同步编程，异步执行
特点是轻，支持高并发

周期

- 启动 go funcname
- 结束 执行结束或者主进程退出

应用

- channel与goroutine: 最佳搭档
- 防止goroutine leak
- 防止goroutine deadlock

开发环境

- 安装
- IDE

安装

- 下载解压到GOROOT位置
- 设置GOPATH

IDE

- VSCODE
- sublime text
- IntelliJ
- vim
- LiteIDE

推荐VSCODE

VSCODE

- gofmt
- autocomplete
- godef
- goimports

Go Tooling in Action (<https://www.youtube.com/watch?v=uBjoTxosSys>)

7 reasons why I switched to Visual Studio Code from Sublime Text (<https://www.youtube.com/watch?v=eg4jUYe1vpl>)

常用命令

- go build: 编译工具
- go clean: 清除对象文件
- go run: 编译执行Go程序
- go fmt: 格式化
- go get: 下载并安装依赖包
- go install: 编译并安装依赖包
- go test: 官方测试框架
- go doc: 查看文档
- go env: 打印Go环境变量
- go version: 打印Go版本

学习

- 实践
- 资源

实践

- 入门

[Go by example](https://gobyexample.com/) (https://gobyexample.com/)

[Go by example\(中文\)](http://go.yami.io/) (http://go.yami.io/)

[Tour of Go](https://tour.golang.org) (https://tour.golang.org)

[GolangTraining](https://github.com/GoesToEleven/GolangTraining) (https://github.com/GoesToEleven/GolangTraining)

[C VS go](http://hyperpolyglot.org/c) (http://hyperpolyglot.org/c)

- 进阶

[golang-web](https://github.com/GoesToEleven/golang-web) (https://github.com/GoesToEleven/golang-web)

[godoc](https://golang.org/doc/) (https://golang.org/doc/)

资源

Go packge (<https://golang.org/pkg/>)

talks (<https://github.com/golang/talks>)

awesome-go (<https://github.com/avelino/awesome-go>)

Go语言圣经（中文版） (<https://docs.hacknode.org/gopl-zh>)

Thank you

15:04 19 Sep 2017

Tags: [Golang](#) ([#ZgotmplZ](#))

Wubo

myself659@163.com (<mailto:myself659@163.com>)

<https://myself659.github.io> (<https://myself659.github.io>)

