内容概述

Go 语言基础	基本程序结构
	常用集合
	函数式编程
	面向对象编程
	错误处理
	模块化及依赖管理
进阶与实战	并发编程模式
	常见并发任务
	深入测试
	反射和 Unsafe
	常见架构模式的实现
	性能调优
	高可用性服务设计

Go 语言简介

软件开发的新挑战

1. 多核硬件架构
2. 超大规模分布式计算集群
3. Web 模式导致的前所未有的开发规模和更新速度

Go 的创始人

Rob Pike

Unix 的早期开发者

UTF-8 创始人

Ken Thompson

Unix 的创始人

C 语言创始人

1983 年获图灵奖

Robert Griesemer

Google V8 JS Engine 开发者

Hot Spot 开发者

Go 语言发展

Subject: Re: prog lang discussion
From: Rob ‘Commander' Pike
Date：Tue,Sep 25, 2007 at 3:12 PM
To: Robert Griesemer, Ken Thompson

i had a couple of thoughts on the drive home.

1.name

‘go'.you can invent reasons for this name but it has nice properties.
it's short,easy to type.tools:goc,gol,goa.if there's an interactive debugger/interpreter it could just be called‘go'.the suffix is .go
...

简单

C	Go	C++
37	25	84

高效

垃圾回收

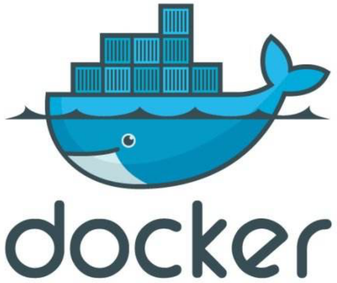
指针

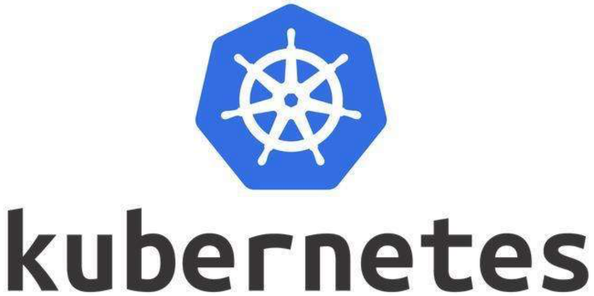
生产力

复合 vs 继承

极客时间

云计算语言





区块链语言

ethereum

Homestead Documentation

HYPERLEDGER

准备开始 Go 冒险之旅

下载安装 Go 语言

https://golang.org/doc/install

https://golang.google.cn/dl/

安装 IDE

Atom: https://atom.io + Package: go-plus

编写第一个 Go 程序

开发环境构建

GOPATH

1. 在 1.8 版本前必须设置这个环境变量

2. 1.8 版本后（含 1.8）如果没有设置使用默认值

   在 *Unix* 上默认为 *$HOME/go* , 在 *Windows* 上默认为 *%USERPROFILE%/go*

   在 *Mac* 上 *GOPATH* 可以通过修改 *~/.bash_profile* 来设置

基本程序结构

```
package main //包，表明代码所在的模块（包）

import "fmt" //引入代码依赖

//功能实现
func main() {
  fmt.Println("Hello World!")
}
```

应用程序入口

1. 必须是 main 包：`package main`
2. 必须是 main 方法：`func main()`
3. 文件名不一定是 `main.go`

退出返回值

与其他主要编程语言的差异

- Go 中 `main` 函数不支持任何返回值
- 通过 `os.Exit` 来返回状态

获取命令行参数

与其他主要编程语言的差异

- main 函数不支持传入参数

```
func main(arg []string)
```

- 在程序中直接通过 os.Args 获取命令行参数

变量与常量

The master has failed more times than the beginner has tried.

编写测试程序

1. 源码文件以 _test 结尾：xxx_test.go

2. 测试方法名以 Test 开头：func TestXXX(t *testing.T) {…}

实现 Fibonacci 数列

1, 1, 2, 3, 5, 8, 13, ...

变量赋值

与其他主要编程语言的差异

- 赋值可以进行自动类型推断
- 在一个赋值语句中可以对多个变量进行同时赋值

常量定义

与其他主要编程语言的差异

快速设置连续值

```
const (
  Monday = iota + 1
  Tuesday
  Wednesday
  Thursday
  Friday
  Saturday
  Sunday
)
```

```
const (
  Open = 1 << iota
  Close
  Pending
)
```

数据类型

基本数据类型

```
bool
string
int  int8  int16  int32  int64
uint uint8 uint16 uint32 uint64 uintptr
byte // alias for uint8
rune // alias for int32,represents a Unicode code point
float32 float64
complex64 complex128
```

类型转化

与其他主要编程语言的差异

1. Go 语言不允许隐式类型转换

2. 别名和原有类型也不能进行隐式类型转换

类型的预定义值

1. math.MaxInt64
2. math.MaxFloat64
3. math.MaxUint32

指针类型

与其他主要编程语言的差异

1. 不支持指针运算

2. string 是值类型，其默认的初始化值为空字符串，而不是 nil

极客时间

运算符

算术运算符

运算符	描述	实例
+	相加	A + B 输出结果 30
-	相减	A - B 输出结果 -10
*	相乘	A * B 输出结果 200
/	相除	B / A 输出结果 2
%	求余	B % A 输出结果 0
++	自增	A++ 输出结果 11
--	自减	A-- 输出结果 9

Go 语言没有前置的 ++，- -，（~~++a~~）

比较运算符

运算符	描述	实例
==	检查两个值是否相等，如果相等返回 True 否则返回 False。	(A == B) 为 False
!=	检查两个值是否不相等，如果不相等返回 True 否则返回 False。	(A != B) 为 True
>	检查左边值是否大于右边值，如果是返回 True 否则返回 False。	(A > B) 为 False
<	检查左边值是否小于右边值，如果是返回 True 否则返回 False。	(A < B) 为 True
>=	检查左边值是否大于等于右边值，如果是返回 True 否则返回 False。	(A >= B) 为 False
<=	检查左边值是否小于等于右边值，如果是返回 True 否则返回 False。	(A <= B) 为 True

用 == 比较数组

- 相同维数且含有相同个数元素的数组才可以比较
- 每个元素都相同的才相等

逻辑运算符

运算符	描述	实例
&&	逻辑 AND 运算符。 如果两边的操作数都是 True，则条件 True，否则为 False。	(A && B) 为 False
||	逻辑 OR 运算符。 如果两边的操作数有一个 True，则条件 True，否则为 False。	(A || B) 为 True
!	逻辑 NOT 运算符。 如果条件为 True，则逻辑 NOT 条件 False，否则为 True。	!(A && B) 为 True

位运算符

运算符	描述	实例
&	按位与运算符"&"是双目运算符。 其功能是参与运算的两数各对应的二进位相与。	(A & B) 结果为 12, 二进制为 0000 1100
|	按位或运算符"|"是双目运算符。 其功能是参与运算的两数各对应的二进位相或	(A | B) 结果为 61, 二进制为 0011 1101
^	按位异或运算符"^"是双目运算符。 其功能是参与运算的两数各对应的二进位相异或，当两对应的二进位相异时，结果	(A ^ B) 结果为 49, 二进制为 0011 0001
<<	左移运算符”<<"是双目运算符。左移 n 位就是乘以 2 的 n 次方。 其功能把"<<"左边的运算数的各二进位全部左移若干位，由"<<"右边的数指定移动的位数，高位丢弃，低位补	A << 2 结果为 240 ， 二进制为 1111 0000
>>	右移运算符”>>"是双目运算符。右移 n 位就是除以 2 的 n 次方。 其功能是把">>"左边的运算数的各二进位全部右移若干位，">>"右边的数指定移动的位数。	A >> 2 结果为 15 ， 二进制为 0000 1111

位运算符

与其他主要编程语言的差异

&^ 按位置零

```
1 &^ 0 --  1
1 &^ 1 --  0
0 &^ 1 --  0
0 &^ 0 --  0
```

编写结构化程序

循环

与其他主要编程语言的差异

Go 语言仅支持循环关键字 for

```
for ( j := 7; j <= 9; j++ )
```

代码示例

while 条件循环

while （n<5)

```
n := 0
for n < 5 {
  n++
  fmt.Println(n)
}
```

无限循环

while (true)

```
n := 0
for {
…
}
```

if 条件

```
if  condition {
    // code to be executed if condition is true
} else {
    // code to be executed if condition is false
}
```

```
if  condition-1 {
    // code to be executed if condition-1 is true
} else if condition-2 {
    // code to be executed if condition-2 is true
} else {
    // code to be executed if both condition1 and condition2 are false
}
```

if 条件

与其他主要编程语言的差异

1. condition 表达式结果必须为布尔值
2. 支持变量赋值：

```
if  var declaration;  condition {
    // code to be executed if condition is true
}
```

switch 条件

```
switch os := runtime.GOOS; os {
case "darwin":
  fmt.Println("OS X.”)
    //break
  case "linux":
    fmt.Println("Linux.")
  default:
    // freebsd, openbsd,
    // plan9, windows...
    fmt.Printf("%s.", os)
}
```

```
switch {
    case 0 <= Num && Num <= 3:
        fmt.Printf("0-3")
    case 4 <= Num && Num <= 6:
        fmt.Printf("4-6")
    case 7 <= Num && Num <= 9:
        fmt.Printf("7-9")
}
```

switch 条件

与其他主要编程语言的差异

1. 条件表达式不限制为常量或者整数；
2. 单个 case 中，可以出现多个结果选项, 使用逗号分隔；
3. 与 C 语言等规则相反，Go 语言不需要用break来明确退出一个 case；
4. 可以不设定 switch 之后的条件表达式，在此种情况下，整个 switch 结构与多个 if…else… 的逻辑作用等同