# lind-go

对go语言的学习

# 基础知识

Go（又称 Golang）是 Google 的 Robert Griesemer，Rob Pike 及 Ken Thompson 开发的一种计算机编程语言语言。  
**设计初衷**  
Go语言是谷歌推出的一种的编程语言，可以在不损失应用程序性能的情况下降低代码的复杂性。谷歌首席软件工程师罗布派克(Rob Pike)说：我们之所以开发Go，是因为过去10多年间软件开发的难度令人沮丧。派克表示，和今天的C++或C一样，Go是一种系统语言。他解释道，"使用它可以进行快速开发，同时它还是一个真正的编译语言，我们之所以现在将其开源，原因是我们认为它已经非常有用和强大。"

* 计算机硬件技术更新频繁，性能提高很快。目前主流的编程语言发展明显落后于硬件，不能合理利用多核多CPU的优势提升软件系统性能。
* 软件系统复杂度越来越高，维护成本越来越高，目前缺乏一个足够简洁高效的编程语言。
* 企业运行维护很多c/c++的项目，c/c++程序运行速度虽然很快，但是编译速度确很慢，同时还存在内存泄漏的一系列的困扰需要解决。

**应用领域**

![fig:](data:application/octet-stream;base64,)

**数据类型**

* int :有符号的整数类型:具体占几个字节要看操作系统的分配:不过至少分配给32位。
* uint:非负整数类型:具体占几个字节要看操作系统的分配:不过至少分配给32位。
* int8:有符号的整数类型:占8位bit:1个字节。范围从负的2的8次方到正的2的8次方减1。
* int16:有符号的整数类型:占16位bit:2个字节。范围从负的2的16次方到正的2的16次方减1。
* int32:有符号的整数类型:占32位bit:4个字节。范围从负的2的32次方到正的2的32次方减1。
* int64:有符号的整数类型:占64位bit:8个字节。范围从负的2的64次方到正的2的64次方减1。
* uint8:无符号的正整数类型:占8位:从0到2的9次方减1.也就是0到255.
* uint16:无符号的正整数类型:占16位:从0到2的8次方减1.
* uint32:无符号的正整数类型:占32位:从0到2的32次方减1.
* uint64:无符号的正整数类型:占64位:从0到2的64次方减1.
* uintptr:无符号的储存指针位置的类型。也就是所谓的地址类型。
* rune :等于int32:这里是经常指文字符。
* byte:等于uint8:这里专门指字节符
* string:字符串:通常是一个切片类型:数组内部使用rune
* float32:浮点型:包括正负小数:IEEE-754 32位的集合
* float64:浮点型:包括正负小数:IEEE-754 64位的集合
* complex64:复数:实部和虚部是float32
* complex128:复数:实部和虚部都是float64
* error:错误类型,真实的类型是一个接口。
* bool:布尔类型

**基础组件分为以下几种**

* 引用类型
  + slice
  + interface
  + chan
  + map
* 非引用类型
  + array
  + func
  + struct

**声明包和引用包**

package main  
  
import (  
"fmt"  
"lind-go/common"  
//自定义的本项目的包  
\_ "lind-go/common"  
)

**赋值符号**

var a  
b :=

其中var 这种方式不论是局部还是全局变量都可以使用，但是后者也就是:=只有局部变量可以使用。也就是只有函数内部才能使用。  
并且，var后面的变量后面的类型是可以省略的，省略后，go会在编译过程中自动判断。所以如果不省略就是长这样 var a int 。  
**数组的初始化**

// 初始化的方式1  
a := [6]string{}  
// 初始化的方式2  
var a [6]string  
  
a[0] = "0"  
a[1] = "1"  
a[2] = "2"  
a[3] = "3"  
a[4] = "4"  
a[5] = "5"

# 常量、接口和结构体

Go语言中没有“类”的概念:也不支持像继承这种面向对象的概念。但是Go 语言的结构体与“类”都是复合结构体:而且Go 语言中结构体的组合方式比面向对象具有更高的扩展性和灵活性。  
**结构体的定义和初始化**

结构体定义:  
type identifier struct {  
field1 type1  
field2 type2  
}  
  
结构体实例创建  
s1 := new(Student) //第一种方式  
s2 := Student{“james”, 35} //第二种方式  
s3 := &Student { //第三种方式  
Name: “LeBron”,  
Age: 36,  
}

**common文件夹定义常量**

当前目录的common子目录下建立Info.go文件:内容如下:

package common  
  
const (  
 Name = "lind"  
 Title = "go语言"  
)

**接口定义**

// 接口定义  
type Usb interface {  
 Start()  
 Stop()  
}

**结构体实现接口**

// Mp3 将接口两个方法都实现了  
type Mp3 struct {  
 Name string  
}  
  
func (c \*Mp3) Start() {  
 fmt.Println(c.Name, "start working")  
}  
func (c \*Mp3) Stop() {  
 fmt.Println(c.Name, "stop work")  
}

\*\*定义一个使用接口的结构:它的方法参数是接口Usb的实例对象

// 定义使用者的结构体  
type Computer struct {  
 Name string  
}  
  
// Working方法接收Usb接口对象:然后调用它的两个方法  
func (c \*Computer) Working(u Usb) {  
 u.Start()  
 u.Stop()  
}

\*\*在main方法中使用它:程序启动时执行

func main() {  
 fmt.Println("auth:%s", common.Name)  
 mp3 := Mp3{"mp3"}  
 computer := Computer{"computer"}  
 computer.Working(&mp3)  
}