**序**

第一次与大家见面，如果看到有问题的地方，欢迎指正。同时也欢迎联系我，或者留言评论，发布论坛。不让错误的东西存在时间太长。

写程序过一段时间后发现有些内容还是容易忘记。有时会记录一些，有时便会忘记记录，过段时间再遇到相同的问题会很难受。要是基础问题，那就是丢人，脸疼了。因此本书是针对基础的工具书。

工作过小几年的朋友应该都清楚，再次遇到面试官。他们依然是要求基础要过硬，其它的都好说。基础不够硬，会让人家很为难的。老艾也是经常去网上各种搜索，又是看各种版本的书，去扫除脑袋里问号。有很多书词汇过于专业，咬文嚼字的过于多。有的更是非程序人员翻译的国外书，那看起来酸爽。

其中《GO语言圣经》是翻译的不错的基础类书籍。专业词翻译的也都非常好。Go现在的好书越来越多，而且大多自己创作，可谓是匠心著作。希望我也能是你们中的一员吧。

本书本着由浅入深的思路。所以新手一般看章节的前半部分即可，其它的回头再看。本书从定义，常规，特性，性能，内存，运行时太，以及一些Go隐形问题，以及建议所构成。

**Go简书**

创始人[Robert Griesemer](http://research.google.com/pubs/author96.html)，[Rob Pike](http://genius.cat-v.org/rob-pike/)和[Ken Thompson](http://genius.cat-v.org/ken-thompson/)三位大牛。他们就职于Google公司，于2007年9月开始设计和实现，然后于2009年的11月对外正式发布。

GO是一个开源的编程语言。有时会被喜欢的它的人称“21世纪的C”。相比于C和C++它易于构建简单、可靠和高效的软件。

Go的编译速度很快。不用在这点担心它。

Go的可移植性很高。不用担心在什么平台下写的。

Go的文化“简单哲学”。

Go的生态已成熟，标准库和第三方的库越来越多。

Go具有CSP并行特性，非常适合服务端编程。

Go有类C的语句，但它的来源先祖不仅仅是C。

Go是纯粹的函数式语言。

Go有GC内存垃圾自动回收。GC运行微秒级。

Go的构建工具go tool越来越完整。

Go少了面向对象类语言很多部件，像宏，异常，继承，运算符重载，构造析构函数等等它都没有要求在语言里。

Go原生支持Unicode。可以处理任何字符。

我们从Go1.11+ 开始，过去的就不多说了。平台我们选择amd64类unix系统。

**本书结构**

第一章是为了对go有兴趣的人而作，即使你没有太多其它语言基础，也可以轻松的运行第一个程序，从而敲响golang的大门。

目录

[第一章 新手入门 3](#_Toc58867154)

[1.1 Hello, world 3](#_Toc58867155)

[1.2 \*unix环境 5](#_Toc58867156)

[1.2.1 官方安装地址 5](#_Toc58867157)

[1.2.2 安装步骤 6](#_Toc58867158)

[1.2.3 运行hello, world! 9](#_Toc58867159)

[1.3 命令行参数 10](#_Toc58867160)

[1.4 类C语言 12](#_Toc58867161)

[1.5 新手建言 14](#_Toc58867162)

[第二章 程序结构基础 14](#_Toc58867163)

[2.1 关键字和内置词 14](#_Toc58867164)

[2.2 包与go源码文件·简 15](#_Toc58867165)

[2.3 命名 16](#_Toc58867166)

[2.4 声明 17](#_Toc58867167)

[2.5 变量 18](#_Toc58867168)

# 第一章 新手入门

## 1.1 Hello, world

永恒的“hello, world”。现在各种程序语言都喜欢用 hello, world作为敲门砖。我们也看看Go的代码。

**CODE 1-1** ：

/\*\* 注释

\* Hello world.

\* module:github.com/aixgl/ch01/code1.1 module 是项目初始化的名称

\* source: https://github.com/aixgl/gobook/tree/master/basic.magic/ch01/code1.1

\*/

package main

import "fmt"

func main() {

fmt.Println("Hello,world!")

}

**编译运行**

如果你还没有安装go环境可以只看输出，下一节我们讲述安装环境。这里对执行的命令有个概念即可。

* **go run：**

Go run 可以对一个或者多个go后缀的文件进行编译, 不会生成执行文件。测试单个文件代码时用着不错。

$ go run helloworld.go

*输出*：

$ Hello,world!

* **go build：**

Go build 编译项目为一个可运行文件，生成一个可执行文件。

$ go build helloworld.go

$ ./helloworld

*输出*：

$ Hello,world!

只运行下面的命令是编译整个项目。可执行文件名是初始化的项目名。它们的运行结果是一样的。

$go build

至于编译运行命令详细介绍可以查工具链一章。

**程序介绍**

接下来我们讨论下这段小程序。

程序的文件必须是以.go为后缀。

* **package main**

Go是用包组织的，类似java包或者其它语言的库或者模块概念。Package就是开头定义包main就是包名。这里的main有些特殊，而是go工程入口包，同时也不能被引用。package详细内容在第二章第二节中再进行讲述。

* **import**

第二行import “fmt” 是引入标准库的fmt包。Import紧挨着package，它们之间不能写go的其它语句。Go的其它程序只能写在它的后面。Import要精确，不需要包的不用导入否则会编译报错。

* **func main**

定义主函数main，所有的go的项目入口都是这个函数。可以带参数，可以不带。func 是定义函数的关键字。

函数内的fmt.Println是引用fmt包的Println函数是按行打印字符串，后面自动跟回车符，它可以接收多个参数，这样写fmt.Println(“hello,”, “world”) 也是可以的。

在go语句结尾分号“;”是不用写了。同行写多个语句必须用分号分割。

程序块一般是用{}一对花括弧为界，作用域于此也是息息相关。

## 1.2 \*unix环境

### 1.2.1 官方安装地址

<https://golang.org/dl/>

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

地址1 经常需要高科技上网，地址2正常能上网即可打开



图1-1

### 1.2.2 安装步骤

1 下载：

用wget命令下载，可以从官网地址上选择一个对应系统的版本，右键复制链接，跟在wget后面即可。下面示例的命令可直接用于运行。

例：$ wget https://golang.google.cn/dl/go1.15.5.linux-amd64.tar.gz



图1-2

2 安装到指定目录

源码安装到指定目录。下面例子安装目录为 /usr/local/go

例：$ tar -C /usr/local -xzf go1.15.5.linux-amd64.tar.gz



图1-3

3 添加环境变量

**GOROOT添加**

* 配置文件修改

$ vi /etc/profile

此命令打开的是全局配置文件。将光标移动到文件最末尾（快捷键：shift+g），添加下面的变量（换行插入的命令是o）。

export GOROOT=/usr/local/go

export PATH=$PATH:$GOROOT/bin

* 保存并退出

按： ESC键

输入： :wq

* 导入

$ source /etc/profile

此命令是导入变量以及环境变量

* 判断是否安装成功，输入命令显示如图2-4类似，则OK。

$ go version



图1-4

**GOPATH添加**

此变量一般添加在开发者用户下，比如多人在同一台服务器上作业。又有各自的工作账号以及工作目录。

1 添加开发者用户

如果有用户里此步骤可以省略

例：$ useradd boy

添加用户

$ passwd 123456

设置用户密码

$ su boy

切换到此用户下

2 配置GOPATH

$ vi ~/.bash\_profile

在export PATH 前添加如下内容

export GOPATH=$HOME/go

export PATH=$PATH:$GOPATH/bin

export GOPROXY=https://goproxy.cn #代理

export GO111MODULE=on

保存并退出

导入： $ source ~/.bash\_profile

测试： $ echo $GOPATH

### 1.2.3 运行hello, world!

在boy用户下创建一个go项目文件夹hello，在这个工程下添加一个hello.go 的文件夹。将第一章的“hello，world”代码粘贴到这个文件里。Boy用户你可以替换成你自己的用户。

执行前先确认是否在项目文件下。

$ pwd

输出：

/home/boy/hello

**go run**

$ go run hello.go

输出：

Hello，world!

**go build**

*初始化项目*

若是第一次使用它，先用go mod 初始化我们的项目。下面的命令常用格式，并不是严格要求格式必须这样。命令的斜线部分是可以自定义的。

$ go mod init *github.com/xxx/hello*

github.com: 远程仓库地址

xxx: 你在github仓库的用户名

hello: 项目名

*执行*

$ go build && ./hello

同样输出：

Hello，world!

还会生成一个 hello 的可执行文件下次再运行 可以直接是用  
 $ ./hello

*注*：

以下命令都是有效的

$ go mod init github.com/xxx/path\_a/path\_b/hello  
 $ go mod init hello

## 1.3 命令行参数

这一节我们再举个小示例来解释，加深下go程序的运行，顺便解决部分人脑袋中的问号。如：命令行参数是个什么东东？ go的命令行参数是怎么解析的？

**CODE 1-2** ：

package main

/\*\*

\* Command Args.

\* module:github.com/aixgl/ch01/code1.2

\* source: https://github.com/aixgl/gobook/tree/master/basic.magic/ch01/code1.2

\*/

import (

"fmt"

"os"

)

func main() {

for i, arg := range os.Args {

fmt.Println("arg", i, "=", arg)

}

}

**编译运行 ：**

Init: 未初始化的先初始化。以后我们会省略不再提醒此步骤。

$ go mod init  *github.com/aixgl/ch02/code1.2*

Build: 生成可执行文件

$ go build

$ ./code1.2 721 boy mycode

上命令的 721, boy, mycode就是可执行文件code1.2的三个命令行参数

输出：



图1-5

由此可以看出命令行的参数是通过os.Args传递go程序。

**源码解析：**

* Import

fmt 标准库包专职格式化打印，这里只用了Fprintln按行打印输出，每行自动追加换行符。此函数可以支持任意多个参数，通常称为变长参数函数。它参数类型支持的很多，一些结构性复合变量也可以直接传个它且能被打印出来。如：fmt.Println(os.Args[1:])。

os 标准库包系统包，这里只用了Args包变量，类型是切片（可以理解为动态数组，但不一样）。用os.Args[0:]可以完全代替os.Args 且结果是一样的。

* Main函数代码

for range 是go循环遍历的一种语法格式，每次循环将键和值分别赋值给变量i和arg，fmt.Println接收它们后执行输出。这2个变量只能在for语句程序员块使用，作用域外也没定义的话，在for大括号外使用会报编译错误。

:= 是一种简易赋值方式，这里不详细介绍，在后续章节会专门讲解。

字符串 是用英文双引号括起来。

**扩展：**

通过以上结果以及分析，我们看到os.Args将执行文件名放在了第一个元素里了，若是我们只要参数应该怎么办，我们稍微改动下程序即可实现。

伪代码：

// 直接改动下os.Args，利用切片的特性

for i, arg := range os.Args[1:] {

fmt.Println("arg", i, "=", arg)

}

带有背景颜色的部分就是修改的部分

## 1.4 类C语言

C语言是最早的高级语言。汇编语言通常称为B类语言。机器语言01我们可以叫它A类吧。

很多编程语言底层要么全部或者部分都是C或者C++语言写的，尤其是脚本语言，像php，python，以及嵌入式语言lua。所以它们的语法部分都类C

它的近亲比如说c++, java, c#。它们的语法同样类C。

Go早期版本部分底层也是C 加 汇编写的，在1.5版本以前。很多关键字，语法等同样保留C的，它也同样是类C语言。并被很多喜欢它的人称为新一代的C语言。伴随它的还有go汇编以及自举（自己编译自己）。

**CODE OF GO:**

//赋值

a = 2

//if语句

If a== true {

//code to do something.

}

//for语句

for var i =0; i< 10; i++ {

}

...

看到上面的code估计你同样会有原来如此，嗯...的感觉

**对比**：

1、**简易**。 相较于C它更简单，有垃圾回收（GC），相同的实现用更少的代码且更稳定可靠易于维护。

2、**并行和异步**。 Go 语言的 Goroutine 和 Channel 这两个可以堪称为神器。使用同样简单，就像用一个普通函数和变量且高性能，这是具有开创性的。

3、**Go语言生态完善迅速**。Go语言的标准库库中有绝大多数常用的库，足够开发使用。第三方库与框架更是发展迅速。

4、**强悍**。相比于java,python,c#等。保证高性能的同时，同样适合比较多的领域。工具类开发如docker等，web领域，游戏等。

## 1.5 新手建言

待定

# 第二章 程序结构基础

## 2.1 关键字和内置词

**关键字**

25个

不能用于自定义名字

不能命名成变量， 自定义类型

break default func interface select

case defer go map struct

chan else goto package switch

const fallthrough if range type

continue for import return var

**预定义**

37个以上

与关键字不同

可以定义中重新使用它们

内建常量: true false iota nil

内建类型: int int8 int16 int32 int64

uint uint8 uint16 uint32 uint64 uintptr

float32 float64 complex128 complex64

bool byte rune string error

内建函数: make len cap new append copy close delete

complex real imag

panic recover

**注**

自定义的变量，函数，类型等尽量不用关键字和内置词做名称。

## 2.2 包与go源码文件·简

这里我们对包做个简单介绍，有助于阅读，详细的介绍见十三章。

**包简述**

包系统目的都是为了简化大型程序的设计和维护工作，模块化，封装性，共享以及重用。

每个包对应一个目录。

定义的关键字：package。声明文件首行如下

package flag

每个包还通过控制包内名字的可见性和是否导出来实现封装特性。依据名字的首字母大小写导出，大写的外部可见，小写的不可见。

修改了源码文件，必须从头构建该包以及依赖包。

Go闪电编译速度。

导入使用import

import (

"fmt"

"github.com/go-sql-driver/mysql"

)

**Go源码文件**

后缀是go

文件名命名一般是小写，多个词用下划线连接（推荐习惯）。

例：

源码文件：study\_args.go

package main // 首行有效代码

import ( // 导入fmt和 os包

"fmt"

"os"

)

func main() {

for i, arg := range os.Args {

fmt.Println("arg", i, "=", arg)

}

}

## 2.3 命名

Go命名也通C语言类似，主要包含

函数名

变量名

常量名

类型名

**命名规范**

名称必须以一个字母（Unicode字母）或者下划线开头。

名称其它位置可以是数字字母或者下划线。

区分大小写，Head 和 head 是不同的。

名称不能使用关键字命名，也不用预定义的内置词。

名称尽量简短，名字的长度没有逻辑限制，理论上长短皆可。推荐：名称的作用域比较大，和业务逻辑的相关名称，尽量使用有意义的英文单词连接。

官方推荐 **驼峰式** 命名。如escapePackage。

## 2.4 声明

**声明语句**

变量声明 var

常量声明 const

类型声明 type

函数声明 func

**例**：

package main

/\*\*

\* 命名.

\* module:github.com/aixgl/ch02/code2.1

\* source: https://github.com/aixgl/gobook/tree/master/basic.magic/ch02/code2.1

\*/

import "fmt"

type myInt int32 //定义类型

const PI = 3.14 //定义常量

func main() { //定义函数

var r = 5.0 //定义变量

var cArea = PI \* r \* r

fmt.Println("circle R:=%f, C:=%f", r, cArea)

}

Output:

circle R:=%f, C:=%f 5 78.5

## 2.5 变量

**语法通用格式**：

var 变量名字 类型 = 表达式

类型 和 =表达式 我们分为2个部分，可以省略任意其中之一。

var 变量名字 类型 √

var 变量名字 = 表达式 √

var 变量名字 ❌

**默认值为零值:**

简单类型

var s string 零值=”” 字符串

var s int/float 零值=0 整形/浮点型

var s bool 零值=false 布尔值

接口引用类型的零值nil(包括slice、指针、map、chan和函数)

var s []byte/[]string/… 零值=nil 切片

var s map[string]string 零值=nil map

var s \*int 零值=nil 指针

var s chan int 零值=nil chan

var f func() 零值=nil 函数变量

**同行多个变量声明:**

可同类型或者不同类型皆可

var i, j, k int // int, int, int

var b, f, s = true, 3.14, "one" // bool, float64, string

### 2.5.1 简短变量

快速声明且初始化局部变量

作用范围：函数内

符号“:=”。 同一个作用域同一个变量只能使用一次该符号。

通用表达式：

变量名字 := 表达式

我们来看看具体的例子：

一个整数变量

i := 100

一个浮点数

f := 100.0

一个切片

s := []int{1,2,5,2,1}

同一行多变量

i, j := 1, 10

=(赋值) 和 := 的区别很大

i, j = j, i // 交换i 和 j 的值

### 2.5.2 指针

定义：指针的值是另一个变量的地址。

指针可以直接读或更新对应变量的值，而不需要知道该变量的名字

通用定义格式

var 变量名字 \*类型 //类型：go中常用或者自定义的类型

变量名字 := &其它变量

与指针相关的2个符号 \*（指针符号）和 &（取地址符号）

\*在部分情况下是可省略的，声明部分时它肯定是不能省略的

取指针的值 用 \*变量名

取指针的地址 用 变量名

DEMO:

x := 1

p := &x

fmt.Println(p, x, \*p) //output: *0xc0000160a8* 1 1

\*p = 10

fmt.Println(p, x, \*p) // output: *0xc0000160a8* 10, 10

\*p++

fmt.Println(x, p, p==&x) //output: 11 11 true

p为指针 变量p保存的x的地址p==&x为true，直接打印p为一个16进制内存地址的值。\*p取的是x的值，所以\*p == x为true，且修改\*p 可以改变x的值。