# **Go 语言基础和数值布尔类型**

## **实验简介**

这一节中，我们详细介绍Go语言的一些基础知识，以及数值和布尔类型，下一节中我们会介绍字符串类型。

## **一、实验说明**

本课程所有源代码，可以在XfceTerminal中通过以下方式克隆到实验环境:

$ git clone http://git.shiyanlou.com/shiyanlou/Golang\_Programming

## **二. 环境变量和常用命令**

### **1. Go语言环境变量**

Go语言开发环境依赖一些操作系统环境变量，常用的环境变量如下：

* $GOROOT 表示 Go 在你的电脑上的安装位置，它的值一般都是 $HOME/go，当然，你也可以安装在别的地方。
* $GOARCH 表示目标机器的处理器架构，它的值可以是 386，amd64 或 arm。
* $GOOS 表示目标机器的操作系统，它的值可以是 darwin，freebsd，linux 或 windows
* $GOBIN 表示编译器和链接器的安装位置，默认是 $GOROOT/bin，如果你使用的是 Go 1.0.3 及以后的版本，一般情况下你可以将它的值设置为空，Go 将会使用前面提到的默认值。
* $GOPATH 表示工作路径，允许包含多个目录。当有多个目录时，请注意分隔符，多个目录的时候Windows是分号，Linux系统是冒号，当有多个GOPATH时，默认会将go get命令的内容放在第一个目录下。$GOPATH 目录约定有三个子目录：
  + src 存放源代码（比如：.go .c .h .s等）
  + pkg 编译后生成的文件（比如：.a）
  + bin 编译后生成的可执行文件（为了方便，可以把此目录加入到 \$PATH 变量中，如果有多个gopath，那么使用${GOPATH//://bin:}/bin添加所有的bin目录）很多Go命令都依赖于此变量，例如go get命令会将获取到的包放到GOPATH中。

### **2. Go语言的命令行工具**

Go语言自带有一套完整的命令操作工具，你可以通过在shell中输入go来查看：

$ go

Go is a tool for managing Go source code.

Usage:

go command [arguments]

The commands are:

build compile packages and dependencies

clean remove object files

env print Go environment information

fix run go tool fix on packages

fmt run gofmt on package sources

get download and install packages and dependencies

install compile and install packages and dependencies

list list packages

run compile and run Go program

test test packages

tool run specified go tool

version print Go version

vet run go tool vet on packages

Use "go help [command]" for more information about a command.

Additional help topics:

c calling between Go and C

filetype file types

gopath GOPATH environment variable

importpath import path syntax

packages description of package lists

testflag description of testing flags

testfunc description of testing functions

Use "go help [topic]" for more information about that topic.

我们简单介绍下常用的命令。

go build 主要用于测试编译。在包的编译过程中，若有必要，会同时编译与之相关联的包。

* + 如果是普通包，当你执行go build之后，它不会产生任何文件。如果你需要在\$GOPATH/pkg下生成相应的文件，那就得执行go install了
  + 如果是main包，当你执行go build之后，它就会在当前目录下生成一个可执行文件。如果你需要在\$GOPATH/bin下生成相应的文件，需要执行go install。
  + go build会忽略目录下以“\_”或“.”开头的go文件。

go fmt 有过C/C++经验的读者会知道,一些人经常为代码采取K&R风格还是ANSI风格而争论不休。在go中，代码则有标准的风格。由于之前已经有的一些习惯或其它的原因我们常将代码写成ANSI风格或者其它更合适自己的格式，这将为人们在阅读别人的代码时添加不必要的负担，所以Go语言强制了代码格式（比如左大括号必须放在行尾），不按照此格式的代码将不能编译通过，为了减少浪费在排版上的时间，go工具集中提供了一个go fmt命令 它可以帮你格式化你写好的代码文件，使你写代码的时候不需要关心格式，你只需要在写完之后执行go fmt <文件名>.go，你的代码就被修改成了标准格式。

go install 这个命令在内部实际上分成了两步操作：第一步是生成结果文件(可执行文件或者.a包)，第二步会把编译好的结果移到\$GOPATH/pkg或者\$GOPATH/bin。

go test 执行这个命令，会自动读取源码目录下面名为\*\_test.go的文件，生成并运行测试用的可执行文件。输出的信息类似

ok archive/tar 0.011s

FAIL archive/zip 0.022s

ok compress/gzip 0.033s

...

默认的情况下，不需要任何的参数，它会自动把你源码包下面所有test文件测试完毕，当然你也可以带上参数，详情请参考go help testflag。

* go run 编译并运行Go语言源代码，这对于我们快速运行测试代码非常方便。

Go语言还有其他一些命令参数，如go env， godoc，详细信息可以通过go help进行查看。

## **三. Go语言基础**

### **1. 标示符和关键字**

Go语言的标示符（变量名，函数名）是一个非空的字母或数字串，其中第一个字符必须是字母，该字符也不能是关键字的名字。Go语言一共有25个关键字，如下：

break default func interface selectcase defer go map struct

chan else goto package switch

const fallthrough if range type

continue for import return var

Go语言同样预定义了许多标示符，可以很方便的使用，如下：

* 类型：

bool byte complex64 complex128 error float32 float64int int8 int16 int32 int64 rune stringuint uint8 uint16 uint32 uint64 uintptr

* 常量

true false iota

* 零值

nil

* 函数

append cap close complex copy delete imag len

make new panic print println real recover

空标示符"\_"是一个占位符，它用于在赋值操作的时候将某个值赋值给空标示符号，从而达到丢弃该值的目的。空标示符不是一个新的变量，因此将它用于:=操作符号的时候，必须同时为至少另一个值赋值。下面有几个例子：

count, err = fmt.Println(x) // 获取打印的字节数以及相应的error值

count, \_ = fmt.Println(x) // 获取打印的字节数，并且丢弃error值

### **2. 常量和变量**

常量使用关键字const声明，变量可以使用关键字var声明，也可以通过使用快捷变量声明语法:=。Go语言可以自动推断出所声明变量的类型。对于没有显式初始化的变量，Go语言总是将零值赋值给该变量。在Go语言中，声明变量的时候类型名总是在变量名的后面。下面有几个例子：

const limit = 512 // 常量，其类型兼容任何数字const top uint16 = 1421 // 常量，类型：uint16

last := 1.5 // 变量，推断类型 flat64var a int // 变量，值为0，类型 intvar debug = false // 变量，推断类型 bool

当需要设置多个常量的时候，不必重复使用const关键字，可以使用以下语法(var声明同样可以使用):

const (

Cyan = 0

Black = 1

White = 2

)

## **四. 数值类型和布尔类型**

Go语言提供了大量的内置数据类型，每一个数值类型都不同，意味着不能在不同类型之间进行二进制数值运算或者比较操作。无类型的数值常量可以兼容内置的任何类型的数值，在不同类型数值之间进行运算或者比较操作，需要进行类型转换。类型转换采用type(value)的方式，只要合法就能转换成功，即使会导致数据精度丢失。

### **1. 整型**

Go语言提供了11中整型，如下列表所示

| **类型** | **说明** |
| --- | --- |
| byte | 等同于uint8 |
| int | 依赖于不同平台下的实现，可以是int32或者int64 |
| int8 | [-128, 127] |
| int16 | [-32768, 32767] |
| int32 | [-2147483648, 2147483647] |
| int64 | [-9223372036854775808, 9223372036854775807] |
| rune | 等同于uint32 |
| uint | 依赖于不同平台下的实现，可以是uint32或者uint64 |
| uint8 | [0, 255] |
| uint16 | [0, 65535] |
| uint32 | [0, 4294967295] |
| uint64 | [0, 18446744073709551615] |
| uintptr | 一个可以恰好容纳指针值的无符号整型（对32位平台是uint32, 对64位平台是uint64） |

在C语言中我们可以通过sizeof操作符查看类型的字节长度，在Go语言中可以通过unsafe.Sizeof函数进行，使用vim创建源文件type\_length.go，输入以下代码：

package main

import (

"fmt"

"unsafe"

)

func main() {

a := 12

fmt.Println("length of a: ", unsafe.Sizeof(a))

var b int = 12

fmt.Println("length of b(int): ", unsafe.Sizeof(b))

var c int8 = 12

fmt.Println("length of c(int8): ", unsafe.Sizeof(c))

var d int16 = 12

fmt.Println("length of d(int16): ", unsafe.Sizeof(d))

var e int32 = 12

fmt.Println("length of e(int32): ", unsafe.Sizeof(e))

var f int64 = 12

fmt.Println("length of f(int64): ", unsafe.Sizeof(f))

}

以上代码中，首先声明了目前的源文件属于main包，然后导入了fmt和unsafe包，fmt包用于格式化字符串，unsafe包含了用于获取Go语言类型信息的方法。然后在main函数中，我们分别声明了几种类型的整型变量，并通过unsafe.Sizeof方法获取该类型的字节长度。最后我们通过以下方法运行type\_length.go，同时打印出了输出：

$ go run type\_length.go length of a: 8length of b(int): 8length of c(int8): 1length of d(int16): 2length of e(int32): 4length of f(int64): 8

### **2. 浮点类型**

Go语言提供了两种浮点类型和两种复数类型, 具体如下：

| **类型** | **说明** |
| --- | --- |
| float32 | ±3.402 823 466 385 288 598 117 041 834 845 169 254 40x10<sup>38</sup> 计算精度大概是小数点后7个十进制数 |
| float64 | ±1.797 693 134 862 315 708 145 274 237 317 043 567 981x10<sup>38</sup> 计算精度大概是小数点后15个十进制数 |
| complex32 | 复数，实部和虚部都是float32 |
| complex64 | 复数，实部和虚部都是float64 |

### **3. 布尔类型**

Go语言提供了内置的布尔值true和false。Go语言支持标准的逻辑和比较操作，这些操作的结果都是布尔值。值得注意的地方是可以通过!b的方式反转变量b的真假。

## **作业**

请使用上述讲的数值类型做一些基本算术运算，并思考结果。