# **Go 语言介绍**

## **实验简介**

本节实验介绍Go语言，并编写了两个例子程序。

## **一、实验说明**

### **1. 环境登录**

无需密码自动登录，系统用户名shiyanlou，该用户具备sudo的权限，可以执行安装软件等管理员操作。

### **2. 环境介绍**

本实验环境采用Ubuntu Linux桌面环境，实验中会用到桌面上的程序：

1. Xfce终端: Linux命令行终端，打开后会进入Bash环境，可以使用Linux命令
2. Firefox及Opera：浏览器，可以用在需要前端界面的课程里，只需要打开环境里写的HTML/JS页面即可
3. gvim：非常好用的Vim编辑器，最简单的用法可以参考课程[Vim编辑器](http://www.shiyanlou.com/courses/2" \t "https://www.shiyanlou.com/courses/11/labs/49/_blank)
4. 其他编辑器：如果Vim不熟悉可以使用gedit或brackets，其中brackets比较适合开发前端代码。

### **3. 环境使用**

使用GVim编辑器输入实验所需的代码，然后使用XfceTerminal命令行环境进行编译运行，查看运行结果，运行后可以截图并分享自己的实验成果，实验楼提供的截图是后台截图，无法作弊，可以真实有效证明您已经完成了实验。

实验报告可以在个人主页中查看，其中含有每次实验的截图及笔记，以及每次实验的有效学习时间（指的是在实验桌面内操作的时间，如果没有操作，系统会记录为发呆时间）。这些都是您学习的真实性证明。

注：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

### **4. 参考文档**

本课程的文档部分参考了以下文档。

* [1]: [Go 语言官方文档](http://golang.org/doc/" \t "https://www.shiyanlou.com/courses/11/labs/49/_blank)
* [2]: [Go 入门指南](https://github.com/Unknwon/the-way-to-go_ZH_CN" \t "https://www.shiyanlou.com/courses/11/labs/49/_blank)
* [3]: [Go web编程](https://github.com/astaxie/build-web-application-with-golang/blob/master/LICENSE.md" \t "https://www.shiyanlou.com/courses/11/labs/49/_blank)

## **二. Go 语言介绍**

### **1. Go语言起源**

Go 语言起源 2007 年，并于 2009 年正式对外发布。它从 2009 年 9 月 21 日开始作为谷歌公司 20% 兼职项目，即相关员工利用 20% 的空余时间来参与 Go 语言的研发工作。该项目的三位领导者均是著名的计算机工程师：Robert Griesemer，参与开发 Java HotSpot 虚拟机；Rob Pike，Go 语言项目总负责人，贝尔实验室 Unix 团队成员，参与的项目包括 Plan 9，Inferno 操作系统和 Limbo 编程语言；Ken Thompson，贝尔实验室 Unix 团队成员，C 语言、Unix 和 Plan 9 的创始人之一，与 Rob Pike 共同开发了 UTF-8 字符集规范。2009 年 11 月 10 日，开发团队将 Go 语言项目以 BSD-style 授权（完全开源）正式公布了 Linux 和 Mac OS X 平台上的版本。Hector Chu 于同年 11 月 22 日公布了 Windows 版本。

作为一个开源项目，Go 语言借助开源社区的有生力量达到快速地发展，并吸引更多的开发者来使用并改善它。自该开源项目发布以来，超过 200 名非谷歌员工的贡献者对 Go 语言核心部分提交了超过 1000 个修改建议。在过去的 18 个月里，又有 150 开发者贡献了新的核心代码。这俨然形成了世界上最大的开源团队，并使该项目跻身 Ohloh 前 2% 的行列。大约在 2011 年 4 月 10 日，谷歌开始抽调员工进入全职开发 Go 语言项目。开源化的语言显然能够让更多的开发者参与其中并加速它的发展速度。在 Go 语言在 2010 年 1 月 8 日被 [Tiobe](http://www.tiobe.com/" \t "https://www.shiyanlou.com/courses/11/labs/49/_blank)（闻名于它的编程语言流行程度排名）宣布为“2009 年年度语言”后，引起各界很大的反响。目前 Go 语言在这项排名中的最高记录是在 2010 年 2 月创下的第13名，流行程度 1.778%。

### **2. Go语言的特性**

Go 语言将静态语言的安全性和高效性与动态语言的易开发性进行有机结合，达到完美平衡，从而使编程变得更加有乐趣，而不是在艰难抉择中痛苦前行。因此，Go 语言是一门类型安全和内存安全的编程语言。虽然 Go 语言中仍有指针的存在，但并不允许进行指针运算。

Go 语言对网络通信、并发和并行编程有着极佳的支持，这能更好地利用大量的多核计算机。设计者通过 goroutine 这种轻量级线程的概念来实现这个目标，然后通过 channel 来实现各个 goroutine 之间的通信。这个特性是 Go 语言最强有力的部分，不仅支持了日益重要的多核与多处理器计算机，也弥补了现存编程语言在这方面所存在的不足。

Go 语言中另一个非常重要的特性就是它的构建速度（编译和链接到机器代码的速度），一般情况下构建一个程序的时间只需要数百毫秒到几秒。这不仅极大地提升了开发者的生产力，同时也使得软件开发过程中的代码测试环节更加紧凑，而不必浪费大量的时间在等待程序的构建上。

由于内存问题（通常称为内存泄漏）长期以来一直伴随着 C++ 的开发者们，Go 语言的设计者们认为内存管理不应该是开发人员所需要考虑的问题。因此尽管 Go 语言像其它静态语言一样执行本地代码，但它依旧运行在某种意义上的虚拟机，以此来实现高效快速的垃圾回收。

有许多活跃的开源项目都是基于Go语言进行开发，目前十分流行的[Docker](https://github.com/docker/docker" \t "https://www.shiyanlou.com/courses/11/labs/49/_blank)也是基于Go语言进行开发。在正式开始学习Go之前，让我们先尝试几个例子，直观感受下Go语言。

## **三. 示例一 Hello World**

### **1. 源代码**

第一个例子当然是HelloWorld了，使用[VIM](http://www.shiyanlou.com/courses/2" \t "https://www.shiyanlou.com/courses/11/labs/49/_blank)编辑源代码：

$ gvim hello.go

输入以下代码保存

package main

import (

"fmt"

"os"

)

func main() {

target := "World"

if len(os.Args) > 1 { /\* os.Args是一个参数切片 \*/

target = os.Args[1]

}

fmt.Println("Hello", target)

}

### **2. 说明**

Go语言使用C++风格的注释：//表示单行注释，到行尾结束，/\*...\*/表示多行注释。所有的Go语言代码只能放置于一个包中，每一个Go程序都必须包含一个main包以及一个main()函数，main()函数作为整个程序的入口。我们的第一个Go程序源代码中，第一行package main 表明当前源文件属于main包，import后面是导入的包，只有需要用到的包才需要导入，导入的包没有使用，则会导致编译失败。Go语言语句是使用分号进行分隔的，但是在一般情况下我们不用手动添加，编译器会自动完成这些工作，除非需要在一行中写多条语句。

Go语言的函数和方法都以关键字func 进行定义。以上代码中的main函数的第一行使用了:=操作符，该符号声明并初始化了一个字符串变量。Go语言虽然是一门静态强类型的语言，但是在使用:=操作符时，Go语言会根据符号右边的值推导出符号左边变量的类型。这儿的例子中，'World'是一个字符串，所以Go语言将变量target创建为一个字符串类型，并且进行赋值。

代码中os.Args是一个字符串切片(slice)，切片内容是传递给Go程序的参数。切片是一个可以动态增长的数组，可以通过len()内置函数计算切片的长度，通过slice[n]的方式访问切片中的第n个元素，而slice[n:]则返回从第n个元素到最后一个元素的切片，像极了Python中的切片。

### **3. 编译运行**

下面该对helloworld程序进行编译运行了。随着Go语言发布的Go命令行工具可以很方便的构建Go程序，包括编译连接Go源代码，管理Go 软件包等。如果需要快速执行Go程序，使用以下命令：

$ go run hello.go

Hello World

go run hello.go命令直接让Go源文件运行，不会生成任何可执行文件，这可以快速查看源代码的执行效果。一般情况下，我们将Go源代码编译链接成可执行文件，以方便随时运行。可以使用以下命令进行：

$ go build hello.go

$ ./hello Aiden

Hello Aiden

以上的命令中，go build hello.go 编译和链接hello.go代码，并且生成了可执行文件hello。接着我们运行hello程序，并传递了一个参数给它，可以看到程序正常运行并且如期输出了结果。

## **四. 示例二 goroutine和channel**

Go语言能之所以这么流行，goroutine和channel是两个很重要的因素，因为这两者让并发(并行)变的如此简单，具体请看如下分组求和的例子。

### **1. 源码**

使用[VIM](http://www.shiyanlou.com/courses/2" \t "https://www.shiyanlou.com/courses/11/labs/49/_blank)编写源代码：

$ gvim sum\_by\_group.go

输入以下源代码:

package main

import "fmt"

func sum(a []int, result chan int) {

sum := 0

for \_, v := range a {

sum += v

}

result <- sum

}

func main() {

a := []int{2, 3, 5, 6, 10, -5, 1, 0}

result := make(chan int)

go sum(a[:len(a)/2], result)

go sum(a[len(a)/2:], result)

x, y := <-result, <-result

fmt.Println(x, y, x+y)

}

### **2. 说明**

goroutine是Go语言并行设计的核心。goroutine是一种比线程更轻量的实现，十几个goroutine可能在底层就是几个线程。要使用goroutine只需要简单的在需要执行的函数前添加go关键字即可。当执行goroutine时候，go语言立即返回，接着执行剩余的代码，goroutine不阻塞主线程。channel就像一个管道，但是可以双向传输数据，通过它我们可以接收和发送数据，假如result 是一个channel那么：result <- value 是将数据发送到result, 而key <- result就是从result中接收一个数据，就如以上的代码所示，值得注意的地方是channel只能通过Go语言内建的函数make(chan type)创建，其中type指明了该channel能传递的数据类型。

下面我们来说说以上的代码。在main函数中，我们声明了一个int类型的切片a，然后通过内置函数make创建了一个能接收和发送int类型的channel。然后通过关键字go执行了两个goroutine，这两个goroutine的功能是分别计算切片a前半部分和后半部分的和。在这里main函数碰到go关键字，派发goroutine执行相应的函数后，立即返回执行剩余的代码，不会等待goroutine的返回。sum函数中，计算切片的和，然后将结果发送到channel。接下来main函数，从channel中获取结果，在这里，main函数会阻塞至直到能从channel result中接收到数据，最后我们打印出了结果。

### **3. 运行**

运行结果如下:

$ go run sum\_by\_group.go16 6 22

目前为止我们通过两个示例初步了解了Go语言，下一节开始我们正式开始学习Go语言。

## **作业**

请了解Go语言与C语言、Java语言和Python语言的差别。