{.title}

学习 Go 语言

{.title} !—

!—

前言

《学习 Go 语言（第二版）》是一本 Go 语言的快速入门。 本书的第一版翻译自 Miek Gieben 开始于 2010 年的开源书籍 “Learning Go[learning_go]”。 第二版将脱离严格翻译原英文版的形式，重新编排内容，并撰写新的内容。 由于 Go 仍然是一门年轻的语言，还在快速的迭代与发展的过程中。 所以本书的描述会努力与 Go 语言的最新发行版本的功能保持一致。 这本书的目标是为这个新的、革命性的语言提供一个快速指南。 所以本书将假设你已经在系统中安装

了 Go，但读者仍然可以在附录中找到流行操作系统的安装指南。 本书采用开源创作的方式，所有素材可以在 GitHub 上的项目 中获得。 书中内容使用 Attribution-NonCommercial-ShareAlike 3.0 Unported License 授权。 书中的所有示例代码使用 Apache License version 2.0 授权。

致谢

感谢所有为英文原版 Learning Go 和《学习 Go 语言》第一版作出贡献的朋友们。 无论如何得感谢 Miek Gieben 在 Learning Go 中构建起来的良好的内容框架， 和他使用 Go 语言开发的 markdown 处理工具 mmark [mmark]。没有他的分享与贡献，本书的所有内容都无法成型。

读者

这本书的目标读者是那些有基本的编程概念,并且已经对解某些主流编程语言有所了解。 主流编程语言包括但并不限于 C[c]，C++[cpp]，Java[java]，PHP[php]，Python[python] 等等。 请注意，这不是一本教你如何编程的书。本书的主要目标是教你如何使用 Go 来在现代工业环境下工作。 学习某样新语言的最佳的方式就是通过编写程序来探索它。 因此每章都包含了若干练习、答案和答案解析来让你熟悉这个语言。 每个练习都有独立的编号。在练习编号后面的圆括号中标明了该题的难度。 1. 易 2. 中 3. 难 并且为了帮助索引，每个题目都有一个简短的标题。例如： > Q1. （中） map 函数 编号 Q1 的这个问题是一道中等难度的关于 map() 函数的练习。相关答案会在练习的下一页提供。答案的顺序和练习一致并使用类似的编号方式，以 A 开头并附带序号。例如： > A1. map 函数 答案只是提供一种思路和参考，不是绝对的标准，也不是问题的最佳解。对于那些没有答案的练习，它们将用星号标记出来。 在书中的代码使用了下面的约定： * 代码、关键字和注释使用代码块显示； * 代码中对某行代码的说明使用 Go 行注释符号 `//` 在同一行后标识； * 较长的注释使用 `/* ... */` 或使用行注释符号配合数字，例如 `// 1` 标识后，在随后的内容中详细解释。 * 行号在右边显示； * shell 的例子使用 $ 作为输入符； * 用户在 shell 输入内容的例子用黑体显示，系统反馈用斜体显示； * 强调的段落会使用缩进，并在左边使用符号  标识。

主要内容

简介

本章对 Go 语言本身进行了介绍，同时对相关学习文档进行了罗列。让读者对于他们将要学习的这门语言有一个大致的概念。 [语言基础]
本章将讨论语言中可用的基本类型、变量和控制结构 [函数与数据]
函数是 Go 程序中的重要基本部件，通过函数可以对数据进行操作。同时本章也会对 Go 的垃圾自动回收和内存分配进行简要介绍 [包与模块]
通过包可以整合功能相关的函数与数据，而模块可以更加方便的管理包。同时如何编写文档和进行单元测试也会是本章的重要内容 [接口]
Go 语言的接口与传统意义的面相对象语言的接口大相径庭，而接口的设计是这个语言的核心思想之一 [并发]
通过引入关键字 `go` ，Go 语言使并发程序的编写变得异常轻松。而在这个语言中的并发单元是一种轻线程 `goroutine`。而引入 `channel` 来在 `goroutine` 之间通讯也使得传统的并发编程任务变得轻松起来 [通讯]
本章会对标准库中的文件读写、网络通讯等常用库进行简单介绍，并示范如何从头编写一个完整的 Go 程序

{mainmatter}

第一章 简介

{.epigraph} > Go 是面向对象的语言吗？是也不是。
**** Go 开发团队在 FAQ 中如是说到

{{introduction/index.md}}

1. 历史发展

2. 语言特性

清晰并且简洁
Go 努力保持小并且优美，你可以在短短几行代码里做许多事情； 并行

Go 让函数很容易成为非常轻量的线程。这些轻量线程在 Go 中被叫做 goroutines； Channel goroutines 之间的通讯由 channel 完成； 快速 相比其他编译语言，Go 的编译速度很快，而执行速度也毫不逊色。其设计目标是跟 C 一样快。对一般应用程序，编译时间做到可以用秒计算； 安全
当转换一个类型到另一个类型的时候需要显式的转换并遵循严格的规则。Go 有垃圾收集，所以在 Go 中无须像 C 语言那样调用 free() 函数释放内存，语言会处理这一切； 格式化标准
Go 程序可以被格式化为程序员希望的（几乎）任何形式，但是官方格式是存在的。标准也非常简单 `gofmt 的输出就是官方认可的格式`； 类型后置
类型在变量名的后面，像这样 `var a int`，来代替 C 中的 `int a`； UTF-8
任何地方都是 UTF-8 的，包括字符串以及程序代码。你可以在代码中使用 Φ = Φ + 1； 开源
Go 的许可证是完全开源的，参阅 Go 发布的源码中的 LICENSE 文件； 开心
用 Go 写程序会非常开心！:-)

3. 可用文档

Go 已经有大量的文档。例如 A Tour of Go[go_tour] 和 Effective Go[effective_go]。官方网站的文档[godoc]也是绝佳的起点 。虽然学习 Go 语言并不一定要阅读这些文档，但是这些第一手的免费资源强有着其他文档不可匹敌的权威性，所以仍然强烈建议加以阅读。当前 Go 语言通过叫做 godoc[godoc_app] 的标准程序提供其文档。 例如，如果你想了解内建函数相关信息，可以像这样获取： `text $ godoc buildin` 关于如何安装和使用 `godoc` 命令可以参考附录中的内容。

4. 学习方式

第二章 语言基础

{.epigraph} > 我对此十分感兴趣，并且希望做点什么。 **** Ken Thompson 为 Go 添加复数支持的原因

{{basics/index.md}}

1. Hello world

在 Go 指南[go_tour]中，用一个传统的方式展现了 Go：让它打印 “Hello World” （Ken Thompson 和 Dennis Ritchie 在 20 世纪 70 年代，发布 C 语言的时候开创了这个先河）。我们不认为能找到更好的其他方法，所以就用这个来展示我们的第一个 Go 语言例子：Go 的 “Hello World”。

<{{../src/basics/helloworld.go}}
```
1 所有的 Go 代码文件必须以 package 包名 开头。对于独立运行的程序，包名必须是 main；
2 通过 import 将 fmt 包引入到当前程序中。那些不是命名为 main 的包都作为库使用。末尾以 // 开头的内容是单行注释；
3 包裹于 /* 和 */ 之间的内容是多行注释，这中注释在类 C 语言中经常使用；
4 Go 程序在执行的时候，首先调用的函数是 main 包中的 main() 函数。这是从 C 中借鉴而来的规则；
5 这里声明了变量 文字。由于 Go 完全支持 Unicode，所以可以使用任何可输出的 Unicode 字符作为包、变量、函数等等的名字；
6 这里调用了来自于 fmt 包的函数 Println(...) 输出字符串到终端。字符串由双引号（”）包裹。同样由于 Go 对 Unicode 的
```

良好支持，所以这里无需任何编码处理。 这个例子纯粹为了用尽可能少的代码来展示一个完整的 Go 程序。 所以为了展示 Go 对 Unicode 的支持，我们使用了中文变量名。 然而而在实际开发中，不应当使用这样的命名方式。

{{inc/bib.xml}} {backmatter}