PEARSON

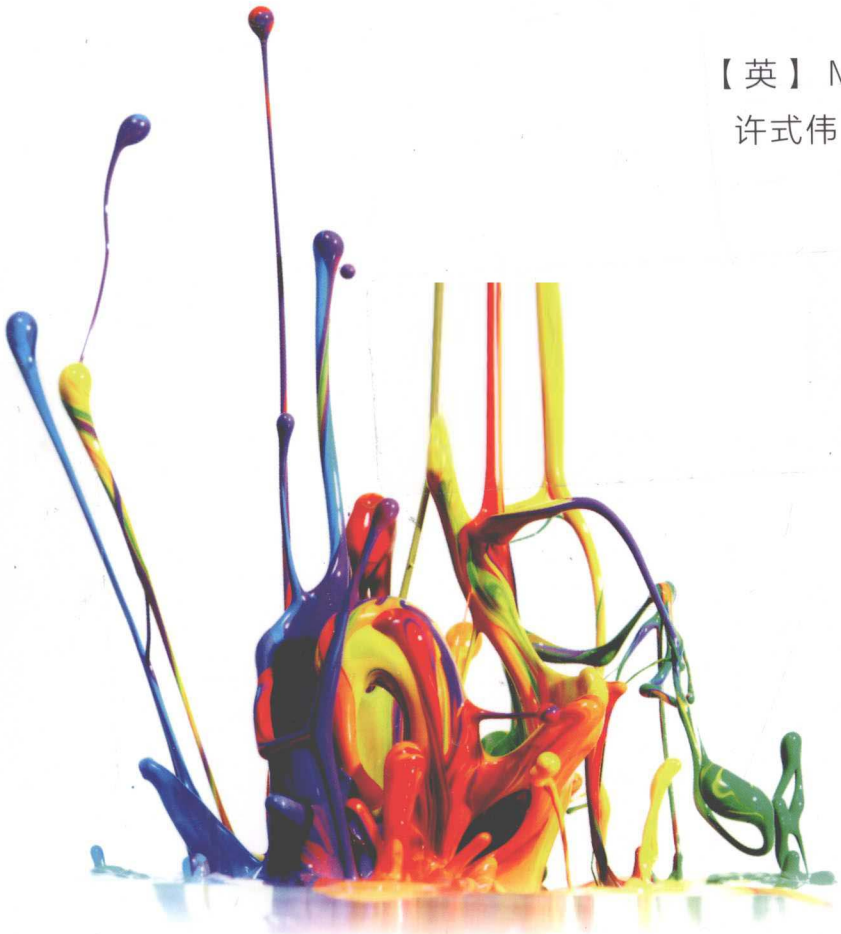
·新·锐·编·程·语·言·集·萃·

Programming in Go

Creating Applications for the 21st Century

Go语言程序设计

【英】Mark Summerfield 著
许式伟 吕桂华 徐立 何李石 译



人民邮电出版社
POSTS & TELECOM PRESS

·新·锐·编·程·语·言·集·萃·

Programming in Go

Creating Applications for the 21st Century

Go语言程序设计

【英】Mark Summerfield 著

许式伟 吕桂华 徐立 何李石 译

安徽大学图书馆 藏书 www.lib.ahu.edu.cn

人民邮电出版社

北京

图书在版编目（CIP）数据

Go语言程序设计 / (英) 萨默菲尔德(Summerfield,M.) 著 ; 许式伟等译. -- 北京 : 人民邮电出版社, 2013.8
（新锐编程语言集萃）
书名原文: Programming in Go
ISBN 978-7-115-31790-2

Ⅰ. ①G… Ⅱ. ①萨… ②许… Ⅲ. ①程序语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2013)第099941号

内 容 提 要

本书既是一本实用的 Go 语言教程，又是一本权威的 Go 语言参考手册。书中从如何获取和安装 Go 语言环境，以及如何建立和运行 Go 程序开始，逐步介绍了 Go 语言的语法、特性以及一些标准库，内置数据类型、语句和控制结构，然后讲解了如何在 Go 语言中进行面向对象编程，Go 语言的并发特性，如何导入和使用标准库包、自定义包及第三方软件包，提供了评价 Go 语言、以 Go 语言思考以及用 Go 语言编写高性能软件所需的所有知识。

本书的目的是通过使用语言本身提供的所有特性以及 Go 语言标准库中一些最常用的包，向读者介绍如何进行地道的 Go 语言编程。本书自始至终完全从实践的角度出发，每一章提供多个生动的代码示例和专门设计的动手实验，帮助读者快速掌握开发技能。本书适合对 Go 语言感兴趣的各个层次的 Go 语言程序员阅读和参考。

◆ 著　[英] Mark Summerfield
译　许式伟 吕桂华 徐　立 何李石
责任编辑　杨海玲
责任印制　程彦红　杨林杰

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
邮编　100061　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷

◆ 开本：800×1000　1/16
印张：23.25
字数：523 千字　2013 年 8 月第 1 版
印数：1 – 3 500 册　2013 年 8 月北京第 1 次印刷

著作权合同登记号　图字：01-2012-6496 号

定价：69.00 元

读者服务热线：(010)67132692　印装质量热线：(010)67129223
反盗版热线：(010)67171154
广告经营许可证：京崇工商广字第 0021 号

译者序

关注过我的人可能都知道，我在新浪微博、《Go 语言编程》一书中都非常高调地下了一个论断：Go 语言将超过 C、Java，成为未来十年最流行的语言。

为什么我可以如此坚定地相信，选择 Go 语言不会有错，并且相信 Go 语言会成为未来 10 年最流行的语言？除了 Go 语言的并发编程模型深得我心外，Go 语言的各种语法特性显得那么深思熟虑、卓绝不凡，其对软件系统架构的领悟，让我深觉无法望其项背，处处带给我惊喜。

Go 语言给我的第一个惊喜是大道至简的设计哲学。

Go 语言是非常简约的语言。简约的意思是少而精。少就是指数级的多。Go 语言极力追求语言特性的最小化，如果某个语法特性只是少写几行代码，但对解决实际问题的难度不会产生本质的影响，那么这样的语法特性就不会被加入。Go 语言更关心的是如何解决程序员开发上的心智负担。如何减少代码出错的机会，如何更容易写出高品质的代码，是 Go 设计时极度关心的问题。

Go 语言追求显式表达。任何封装都是有漏洞的，最佳的表达方式就是用最直白的表达方式，所以也有人称 Go 语言为“所写即所得”的语言。

Go 语言也是非常追求自然（nature）的语言。Go 不只是提供极少的语言特性，并极力追求语言特性最自然的表达，也就是这些语法特性被设计成恰如多少人期望的那样，尽量避免惊异。事实上，Go 语言的语法特性上的争议是非常少的。这些也让 Go 语言的入门门槛变得非常低。

Go 语言给我的第二个惊喜是最对胃口的并行支持。

我对服务端开发的探索，始于 Erlang 语言，并且认为 Erlang 风格并发模型的精髓是轻量级进程模型。然而，Erlang 除了语言本身不容易被程序员接受外，其基于进程邮箱做消息传递的并发编程模型也小有瑕疵。我曾经在 C++中实现了一个名为 CERL 的网络库，刚开始在 C++中完全模仿 Erlang 风格的并发编程手法，然而在我拿 CERL 库做云存储服务的实践中，发现了该编程模型的问题所在并做了相应的调整，这就是后来的 CERL 2.0 版本。有意思的是，CERL 2.0 与 Go 语言的并行编程思路不谋而合。某种程度上来说，这种默契也是我创办七牛时，Go 语言甚至语法特性都还没有完全稳定，我们技术选型就坚决地采纳了 Go 语言的重要原因。

Go 语言给我的第三个惊喜是接口。

Go 语言的接口，并非是你在 Java 和 C#中看到的接口，尽管看起来有点像。Go 语言的接口是非侵入式的接口，具体表现在实现一个接口不需要显式地进行声明。不过，让我意外的不是 Go 的非侵入式接口。非侵入式接口只是我接受 Go 语言的基础。在接口（或契约）的表达上，

我一直认为 Java 和 C#这些主流的静态类型语言都走错了方向。C++的模板尽管机制复杂，但是走在了正确的方向上。C++0x（后来的 C++11）呼声很高的 concept 提案被否，着实让不少人伤了心。但 Go 语言的接口远不是非侵入式接口那么简单，它是 Go 语言类型系统的纲，这表现在以下几个方面。

（1）只要某个类型实现了接口要的方法，那么我们就说该类型实现了此接口。该类型的对象可赋值给该接口。

（2）作为 1 的推论，任何类型（包括基础类型如 `bool`、`int`、`string` 等）的对象都可以赋值给空接口 `interface{}`。

（3） 支持接口查询。如果你曾经是 Windows 程序员，你会发现 COM 思想在 Go 语言中通过接口优雅呈现。并且 Go 语言吸收了其中最精华的部分，而 COM 中对象生命周期管理的负担，却因为 Go 语言基于 gc 方式的内存管理而不复存在。

Go 语言给我的第四个意外惊喜是极度简化但完备的面向对象编程（OOP）方法。

Go 语言废弃大量的 OOP 特性，如继承、构造/析构函数、虚函数、函数重载、默认参数等；简化的符号访问权限控制，将隐藏的 `this` 指针改为显式定义的 `receiver` 对象。Go 语言让我看到了 OOP 编程核心价值原来如此简单——只是多数人都无法看透。

Go 语言带给我的第五个惊喜是它的错误处理规范。

Go 语言引入了内置的错误（error）类型以及 `defer` 关键字来编写异常安全代码，让人拍案叫绝。下面这个例子，我在多个场合都提过：

```
f, err := os.Open(file)
if err != nil {
     ... // 错误处理
     return
}
defer f.Close()
... // 处理文件数据
```

Go 语言带给我的第六个惊喜是它功能的内聚。

一个最典型的案例是 Go 语言的组合功能。对于多数语言来说，组合只是形成复合类型的基本手段，这一点只要想想 C 语言的 `struct` 就清楚了。但 Go 语言引入了匿名组合的概念，它让其他语言原本需要引入继承这样的新概念来完成事情，统一到了组合这样的一个基础上。

在 C++中，你需要这样定义一个派生类：

```
class Foo : public Base {
    ...
};
```

试读结束：需要全本请在线购买： www.ertongbook.com

在 Go 语言中你只要：

```
type Foo struct {
    Base
    ...
}
```

更有甚者，Go 语言的匿名组合允许组合一个指针：

```
type Foo struct {
    *Base
    ...
}
```

这个功能可以实现 C++中一个无比晦涩难懂的特性，叫“虚拟继承”。但同样的问题换成从组合角度来表达，直达问题的本质，清晰易懂。

Go 语言带给我的第七个惊喜是消除了堆与栈的边界。

在 Go 语言之前，程序员清楚地知道哪些变量在栈上，哪些变量在堆上。堆与栈是基于现代计算机系统的基础工作模型上形成的概念，Go 语言屏蔽了变量定义在堆上还是栈上这样的物理结构，相当于封装了一个新的计算机工作模型。这一点看似与 Go 语言显式表达的设计哲学不太一致，但我个人认为这是一项了不起的工作，而且与 Go 语言的显式表达并不矛盾。Go 语言强调的是对开发者的程序逻辑（语义）的显式表达，而非对计算机硬件结构的显示表达。对计算机硬件结构的高度抽象，将更有助于 Go 语言适应未来计算机硬件发展的变化。

Go 语言带给我的第八个惊喜是 Go 语言对 C 语言的支持。

可以这么说，Go 语言是除了 Objective-C、C++这两门以兼容 C 为基础目标的语言外的所有语言中，对 C 语言支持最友善的一个。什么语言可以直接嵌入 C 代码？只有 Go。什么语言可以无缝调用 C 函数？只有 Go。对 C 语言的完美支持，是 Go 快速崛起的关键支撑。还有比 C 语言更让人觊觎的社区财富吗？那是一个取之不尽的金矿。

总而言之，Go 语言是一门非常具有变革性的语言。尽管 40 年（从 1970 年 C 语言诞生开始算起）来出现的语言非常之多，各有各的特色，让人眼花缭乱。但是我个人固执地认为，谈得上突破了 C 语言思想，将编程理念提高到一个新高度的，仅有 Go 语言而已。

Go 语言很简单，但是具备极强的表现力。从目前的状态来说，Go 语言主要关注服务器领域的开发，但这不会是 Go 语言的完整使命。

我们说 Go 语言适合服务端开发，仅仅是因为它的标准库支持方面，目前是向服务端开发倾斜：.

- 网络库（包括 socket、http、rpc 等）；
- 编码库（包括 json、xml、gob 等）；
- 加密库（各种加密算法、摘要算法，极其全面）；

- Web（包括 template、html 支持）。

而作为桌面开发的常规组件：GDI 和 UI 系统与事件处理，基本没有涉及。

尽管 Go 还很年轻，Go 语言 1.0 版本在 2012 年 3 月底发布，到现在才近 1 年，然而 Go 语言已经得到了非常普遍的认同。在国外，有人甚至提出“Go 语言将制霸云计算领域”。在国内，几乎所有你听到过名字的大公司（腾讯、阿里巴巴、京东、360、网易、新浪、金山、豆瓣等），都有团队对 Go 语言做服务端开发进行了小范围的实践。这是不能不说是一个奇迹。

Go 语言是一门前途非常光明的语言，很少有语言在如此年轻的时候就得到如此热捧。

但因为年轻，导致了 Go 语言的书籍哪怕在全球都非常稀少。这本书由知名技术作家 Mark Summerfield 撰写，它会让你了解 Go 语言，按 Go 语言的方式思考，以及使用 Go 语言来编写高性能软件。一直以来，Summerfield 的教学方式都是深入实践的。每一章节都提供了多个活生生的代码示例，它们都是经过精心设计的用于鼓励读者动手实验并且能够帮助读者快速掌握如何开发的。

许式伟

2013 年 6 月

引 言

本书介绍如何使用 Go 语言的语言特性以及标准库中的常用包来进行地道的 Go 语言编程。同时，本书也设计成在学会 Go 语言后依然有用的参考资料。为了实现这两个目标，这本书覆盖面非常广，尽量保证每一章只涵盖一个主题，各章之间会进行内容上的交叉引用。

从语言的设计精神来说，Go 语言与 C 语言非常相似，是一门精小而高效的语言，它有便利的底层设施，如指针。不过 Go 语言还提供了许多只在高级或者非常高级的语言中才有的特性，如 Unicode 字符串、强大的内置数据结构、鸭子类型、垃圾收集和高层次的并发支持，使用通信而非常规的共享数据和锁方式。另外，Go 语言还提供了一个庞大且覆盖面全的标准库。

虽然所有的 Go 语言特性或者编程范式都会以完整可运行的示例来详细讲解，但是本书还是假设读者有主流编程语言的经验，比如 C、C++、Java、Python 或其他类似的语言。

要学好任何一门语言，使用它进行编程都是必经之路。为此，本书采用完全面向实战的方式，鼓励读者亲自去练习书中的例子，尝试着去解决练习题中给出的问题，自己去写程序，以获得宝贵的实践经验。正如我以前写的书一样，本书中所引用的代码片段都是“活代码”。也就是说，这些代码自动提取自 `.go` 源文件，并直接嵌入到提供给出版商的 PDF 文件中，故此不会有剪切和粘贴错误，可以直接运行。只要有可能，本书都会提供小而全的程序或者包来作为贴近实际应用场景的例子。本书的例子、练习和解决方案都可以从 `www.qtrac.eu/gobook.html` 这个网址获得。

本书的主要目的是传授 Go 语言本身，虽然我们使用了 Go 语言标准库中的许多包，但不会试图全都涉及。这并不是问题，因为本书向读者提供了足够的 Go 语言知识来使用任何标准库中的包或者是任何第三方 Go 语言的包，当然还能够创建自己的包。

为什么是 Go

Go 语言始于 2007 年，当时只是 Google 内部的一个项目，其最初设计者是 Robert Griesemer、Unix 泰斗 Rob Pike 和 Ken Thompson。2009 年 11 月 10 日，Go 语言以一个自由的开源许可方式公开亮相。Go 语言由其原始设计者加上 Russ Cox、Andrew Gerrand、Ian Lance Taylor 以及其他许多人在内的一个 Google 团队开发。Go 语言采取一种开放的开发模式，吸引了许多来自世界各地的开发者为这门语言的发展贡献力量。其中有些开发者获得了非常好的声望，因此他们也获得了与 Google 员工一样的代码提交权限。此外，Go Dashboard 这个网站（`godashboard.appspot.com/project`）也提供了许多第三方的 Go 语言包。

Go 语言是近 15 年来出现的最令人兴奋的新主流语言。它是第一个直接面向 21 世纪计算机

和开发者的语言。

Go 语言被设计为可高效地伸缩以便构建非常大的应用，并可在普通计算机上用几秒钟即完成编译。快如闪电的编译速度可能在一定程度上是因为语言的语法很容易解析，但更主要是因为它的依赖管理。如果文件 app.go 依赖于文件 pkg1.go，而 pkg1.go 又依赖于 pkg2.go，在传统的编译型语言中，app.go 需要依赖于 pkg1.go 和 pkg2.go 目标文件。但在 Go 语言中，一切 pkg2.go 导出的内容都被缓存在 pkg1.go 的目标文件中，所以 pkg1.go 的目标文件足够独立构建 app.go。对于只有三个源文件的程序来说，这看不出什么优劣，但对于有着大量依赖关系的大型应用程序来说，这样做可以获得巨大的编译速度提升。

由于 Go 语言程序的构建是如此之快，因此它也适用一些本来应该使用脚本语言的场景（见“Go 语言版 Shebang 脚本”，参见 1.2 节）。此外，Go 语言可用于构建基于 Google App Engine 的 Web 应用程序。

Go 语言使用了一种非常干净且易于理解的语法，避免了像老的语言如 C++（发布于 1983 年）或 Java（发布于 1995 年）一样的复杂和冗长。Go 语言是一种强静态类型的语言，这在有些程序员看来是构建大型应用程序的必备特性。然而，使用 Go 语言进行编程并不需要像使用别的静态语言那样打太多的字，这要归功于 Go 语言简短的“声明并初始化”的变量声明语法（由于编译器会推断类型，因此并不需要显式地写明），以及它对鸭子类型强大而便捷的支持。

像 C 和 C++这样的语言，当涉及内存管理时需要程序员非常谨慎地面对，特别是对于并发程序，要跟踪它们的内存分配简直犹如噩梦，而这些本来可以交给计算机去做。近年来，C++在这方面用各种“智能”指针进行了很大的改善，但在线程库方面还一直在追赶 Java。通过使用垃圾收集器，Java 减轻了程序员管理内存的负担。虽然 C++语言现在有一个标准的线程库，但是 C 语言还只能使用第三方线程库。然而，在 C、C++或 Java 中编写并发程序仍然需要相当地谨慎，以确保在恰当的时间正确地锁定和解锁资源。

Go 编译器和运行时系统会处理这些繁琐的跟踪问题。对于内存管理而言，Go 语言提供了一个垃圾收集器，因此无需使用智能指针或者手动释放内存。Go 语言的并发机制基于计算机科学家 C. A. R. Hoare 提出的 CSP（Communicating Sequential Processes）模型构建，这意味着许多并发的 Go 语言程序不需要加任何锁[①]。此外，Go 语言引入 goroutine ——一种非常轻量级的进程，可以一次性大量创建，并可跨处理器和处理器核心自动进行负载平衡，以提供比老的基于线程的语言更细粒度的并发。事实上，因为 Go 语言的并发支持使用起来如此简单和自然，移植单线程程序到 Go 时经常会发现转为并发模型的机会大增，从而可以更充分地利用计算机资源。

Go 语言是一门务实的语言，与语言的纯净度相比，它更关注语言效率以及为程序员带来的便捷性。例如，Go 语言的内置类型和用户自定义的类型是不一样的，因为前者可以高度优化，后者却不能。Go 语言也提供了两个基本的内置集合类型：切片（slice，它的实际用途是为了提供变长功能的数组）和映射（map，也叫键值字典或散列表）。这些集合类型非常高效，并且在大多数情况下都能非常好地满足需求。当然，Go 语言也支持指针（它是一个完全编译型的语言，

① 指不需要用户主动加锁，而不是指从内部实现来说没有锁。——译者注

因此在性能方面没有虚拟机挡路），所以它可以轻松创建复杂的自定义类型，如平衡二叉树。

虽然 C 语言仅支持过程式编程，而 Java 则强制要求程序员按照面向对象的方式来编程，但 Go 语言允许程序员使用最合适的编程范式来解决问题。Go 语言可以被用做一个纯粹的过程式编程语言，但对面向对象编程也支持得很好。不过，我们也会在后文看到，Go 语言面向对象编程的方式与 C++、Java 或 Python 非常不同，它更容易使用且在形式上更加灵活。

就像 C 语言一样，Go 语言也不支持泛型（用 C++的话来说就是模板）。然而，Go 语言所提供的别的功能特性消除了对泛型支持的依赖。Go 并不使用预处理器或者包含文件（这也是为什么它编译得如此之快的另一个原因），因此也无需像 C 和 C++那样复制函数签名。同时，因为没有使用预处理器，程序的语义就无法在 Go 语言程序员背后悄悄变化，但这种情况在 C 和 C++下使用`#define`时一不小心就会发生。

可以说，C++、Objective-C 和 Java 都试图成为更好的 C 语言（后者是间接地成为了更好的 C++语言）。尽管 Go 语言干净而轻盈的语法容易让人联想到 Python，Go 语言的切片和映射也非常类似于 Python 的列表和字典，但 Go 语言也可以被认为试图成为一个更好的 C。然而，与任何其他语言相比，Go 语言从语言本质上都更接近于 C 语言，并可以被认为保留了 C 语言的所有精华的同时试图消除 C 语言中的缺陷，同时加入了许多强大而有用的独有特性。

Go 语言最初被构思为一门可充分利用分布式系统以及多核联网计算机优势且适用于开发大型项目的编译速度很快的系统级语言。现在，Go 语言的触角已经远远超出了原定的范畴，它正被用做一个具有高度生产力的通用编程语言。使用 Go 语言开发和维护系统都让人感觉是一种享受。

本书的结构

第 1 章开始讲解如何建立和运行 Go 程序。这一章通过 5 个简短的示例简要介绍了 Go 语言的语法与特性，以及一些标准库。每个例子都介绍了一些不同的特性。这一章主要是为了让读者尝试一下 Go 语言，以此让读者感受一下学习 Go 语言需要学习的大致内容是什么。（这一章章还讲解了如何获取和安装 Go 语言环境。）

第 2 章至第 7 章更深入地讲解了 Go 语言的方方面面。其中有三章专门讲解了 Go 语言的内置数据类型：第 2 章涵盖了标识符、布尔值和数值类型，第 3 章涵盖了字符串，第 4 章涵盖了 Go 语言内置的集合类型。

第 5 章描述并讲解了 Go 语言的语句和控制结构，还解释了如何创建和使用自定义的函数，最后展示了如何使用 Go 语言创建一个过程式的非并发程序。

第 6 章展示了如何在 Go 语言中进行面向对象编程。本章的内容包括可用于聚合和嵌入（委托）其他类型的结构体，可作为一个抽象类型的接口，以及如何在某些情况下产生类似继承的效果。由于 Go 语言中进行面向对象编程的方式可能与大多数读者的经验不同，这一章会给出几个完整的例子并详细讲解，以确保读者完全理解 Go 语言的面向对象编程方式。

第 7 掌讲解了 Go 语言的并发特性，与面向对象编程一章相比，这一章给出了更多实例，

以确保读者对这些新的 Go 语言特性有透彻的了解。

第 8 章展示了如何读取和写入自定义的二进制文件、Go 二进制（gob）文件、文本、JSON 以及 XML 文件。（读取和写入文本文件的知识在第 1 章和后续几章中都有所涉及，因为这些知识可以更易于提供一些有价值的示例和练习。）

本书的最后一章是第 9 章。这一章先展示了如何导入和使用标准库包、自定义包以及第三方软件包。它还展示如何对自定义的包进行文档的自动提取、单元测试和性能基准测试。这一章的最后一节对 Go 编译器（gc）提供的工具集以及 Go 语言的标准库做了简要的概述。

Go 语言虽然小巧，但它同时也是一门功能丰富和强大表达能力（在语法结构、概念和编程习惯方面）的语言。本书的例子都符合良好的 Go 语言编程范式[①]。当然，这种做法也意味着有些概念出现时不会被当场解释。但我们希望读者相信，所有的概念都会在本书中进行解释（当然，没有当场解释的内容都会以交叉引用的形式给出相应讲解的位置）。

Go 是一门迷人的语言，使用起来感觉非常好。学习 Go 语法和编程习惯并不会很难，但它的确引入了一些新颖的、对许多读者来说可能不那么熟悉的概念。这本书试图给读者概念上的突破，尤其是在面向对象的 Go 语言编程和并发 Go 语言编程方面。如果只阅读那些定义良好却非常简要的文档，读者可能需要花费数周甚至数月的时间才能真正理解相关的知识。

① 这里有一个例外：前面几章中，即使通道只被当做单向通道使用，我们也总是将通道声明为双向的。从第 7 章开始，通道只被声明为只有某一特殊的方向，这样，这里所说的 Go 语言风格用法也就讲得通了。

试读结束：需要全本请在线购买： www.ertongbook.com

致　谢

我写每一本技术书时都得到过来自他人的帮助与建议，本书也不例外。

我想特别感谢两个之前没有 Go 语言编程经验的程序员朋友——asmin Blanchette 和 Trenton Schulz。他们两个曾多年为我的书贡献诸多。他们对本书的反馈也让本书能更符合程序员初学 Go 语言时的需求。

来自 Go 语言核心开发者 Nigel Tao 的反馈也让本书受益良多。虽然我并未完全采纳他的所有建议，但是他的反馈总是能够提点我，进而给代码以及书的内容带来极大的改进。

此外，我得到过其他许多人的帮助，包括 Go 语言初学者 David Boddie。他提供了一些有价值的反馈。同时，Go 语言的开发者 Ian Lance Taylor 特别是 Russ Cox 为我解决了很多代码以及概念上的问题，他们提供的清晰准确的解释对本书的精确性有极大的贡献。

在撰写本书时，我在 `golang-nuts` 这个邮件列表里提了许多问题，每次提问总能从众多回邮件者那里收到深思熟虑且实用的回复。同时，Safari 上的本书初稿读者也给了我许多反馈，从而让本书中的一些讲解清晰了很多。

意大利的软件公司 `www.develer.com` 以 Giovanni Bajo 个人的名义，给我提供免费的 Mercurial 代码库托管服务，让我在写作的漫长过程中能够静心思考。谢谢 Lorenzo Mancini 为我设置整个环境然后帮我打理它。同时，我也非常感谢 Anton Bowers 以及 Ben Thompson，自 2011 年初起，我的网站 `www.qtrac.eu` 就托管在他们的网络服务器上。

谢谢 Russel Winder 在他的博客 `www.russel.org.uk` 上讨论软件专利的事情，附件 B 中有许多思想是从他那里来的。

然后，我要一如既往地感谢 lout 排版系统的作者 Jeff Kingston，我所有的书以及许多其他写作项目都是用这个系统排版而成的。

特别感谢我的责任编辑 Debra Willians Cauley，是他将本书成功带给出版社，同时也在本书的写作过程中提供了支持与实际帮助。

同时也感谢出版经理 Anna Popick，他再次将书的出版过程管理得如此好，也感谢校对人员 Audrey Doyle 的出色工作。

与以往一样，我还要感谢我的妻子 Andrea，谢谢她的爱与支持。

版权声明

Authorized translation from the English language edition, entitled: *Programming in Go*, 978-0-321-77463-7 by Mark Summerfield, published by Pearson Education, Inc., publishing as Addison-Wesley Professional, Copyright © 2012 Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

CHINESE SIMPLIFIED language edition published by PEARSON EDUCATION ASIA LTD. and POSTS & TELECOM PRESS Copyright © 2013.

本书中文简体字版由 Pearson Education Asia Ltd.授权人民邮电出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

本书封面贴有 Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

版权所有，侵权必究。

目 录

试读结束：需要全本请在线购买： www.ertongbook.com