HZ BOOKS
华章教育

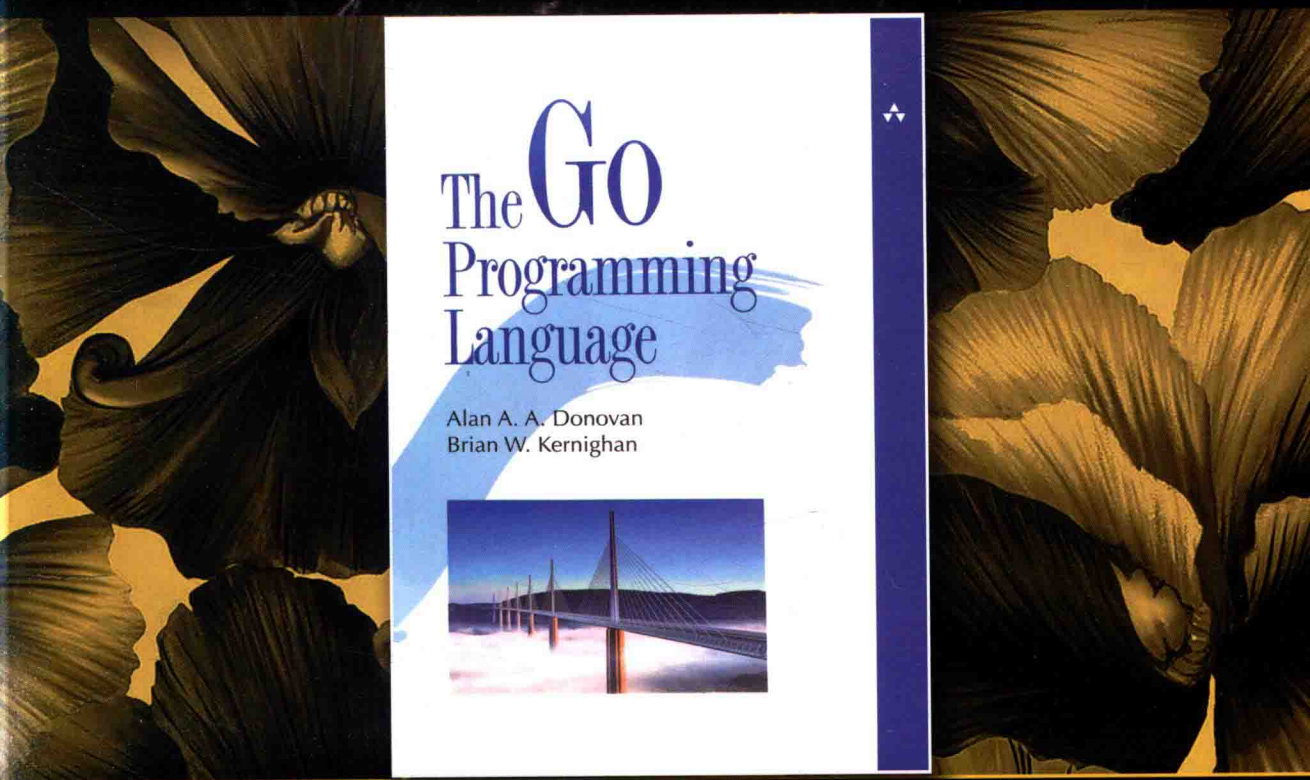
计 算 机 科 学 丛 书

Pearson

Go程序设计语言

[美] 艾伦 A. A. 多诺万（Alan A. A. Donovan） 谷歌公司
布莱恩 W. 柯尼汉（Brian W. Kernighan） 普林斯顿大学 著
李道兵 高博 庞向才 金鑫鑫 林齐斌 译

The Go Programming Language



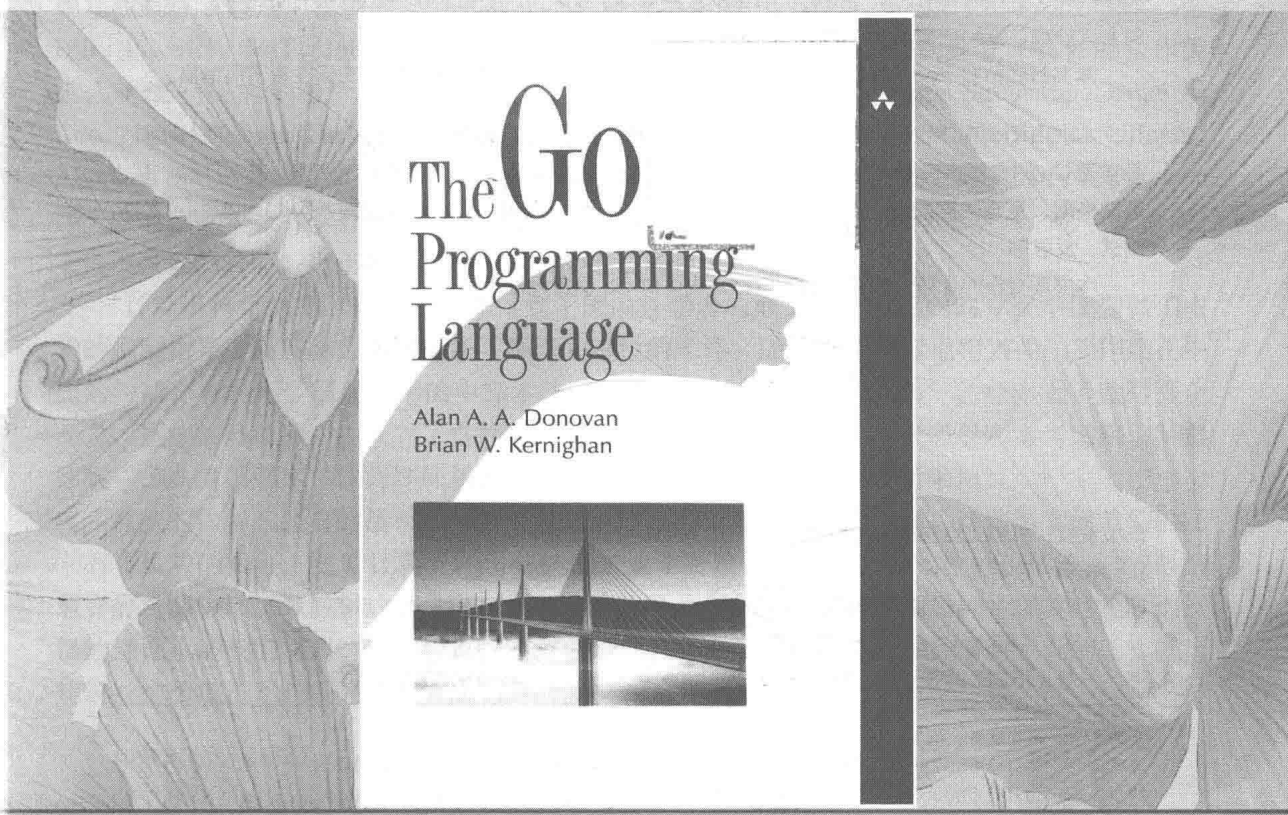
机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

Go程序设计语言

艾伦 A. A. 多诺万（Alan A. A. Donovan）
谷歌公司
[美] 著
布莱恩 W. 柯尼汉（Brian W. Kernighan）
普林斯顿大学
李道兵 高博 庞向才 金鑫鑫 林齐斌 译

The Go Programming Language



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Go 程序设计语言 /（美）艾伦 A. A. 多诺万（Alan A. A. Donovan），（美）布莱恩 W. 柯尼汉（Brian W. Kernighan）著；李道兵等译 . —北京：机械工业出版社，2017.1
（计算机科学丛书）
书名原文：The Go Programming Language

ISBN 978-7-111-55842-2

I. G… II. ① 艾… ② 布… ③ 李… III. C 语言 - 程序设计 IV. TP312.8

中国版本图书馆 CIP 数据核字（2017）第 011442 号

本书版权登记号：图字：01-2015-7914

本书由《C 程序设计语言》的作者 Kernighan 和谷歌公司 Go 团队主管 Alan Donovan 联袂撰写，是学习 Go 语言程序设计的权威指南。本书共 13 章，主要内容包括：Go 的基础知识、基本结构、基本数据类型、复合数据类型、函数、方法、接口、goroutine、通道、共享变量的并发性、包、go 工具、测试、反射等。

本书适合作为计算机相关专业的教材，也可供 Go 语言爱好者阅读。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码 100037）
责任编辑：谢晓芳　　责任校对：董纪丽
印　　刷：北京市荣盛彩色印刷有限公司　　版　　次：2017 年 4 月第 1 版第 1 次印刷
开　　本：185mm × 260mm　1/16　　印　　张：18.75
书　　号：ISBN 978-7-111-55842-2　　定　　价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的优势，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brian W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力相助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

华章科技图书出版中心

译者序

The Go Programming Language

很高兴这次应高博的邀请，与高博及上海七牛信息技术有限公司的几位同事一起来完成本书的翻译。

Go 语言在 2009 年发布，当年就被选为 TIOBE 年度语言，并在若干年后的今天，再度当选为 TIOBE 年度语言。这有力证明了 Go 语言在工业界和开发者社区的良好口碑，以及与时俱进的生命力。本书简体中文版的出版，可谓恰逢其时！

2012 年，Go 语言发布 1.0 版本后，推广速度更是突飞猛进，比如最好的容器软件 Docker 就是用 Go 语言写成的，ETCD、Kubernetes 这类有望构建新一代软件架构的基础软件也是基于 Go 语言的。除此之外，数据库领域有 TiDB 和 InfluxDB，消息系统有 NSQ，缓存系统有 GroupCache。可以看到，几乎在基础架构软件的每一个领域，都涌现了由 Go 语言编写的新软件，这些软件已经取得或者正在取得越来越高的市场占有率。除了作为基础架构软件的语言之外，Go 语言作为服务器端通用语言的机会也越来越多，这从 Beego、Gorilla 等 Go 语言 Web 框架的热门程度也可以看出一些端倪。

在基础架构软件这个层面，最早只有 C 语言，后来又有了 C++ 语言。在性能不受影响的情况下，C++ 语言让我们得以驾驭规模更大、更复杂的项目，比如 MySQL、MongoDB 这类数据库软件就是用 C++ 语言写的。尽管 C++ 语言功能强大，但是它并没有太好地解决代码的易用性和健壮性互相平衡的问题，所以我们接下来看到了很多基于 Java 语言的基础架构软件的出现，例如整个 Hadoop 生态。在这之后，随着高并发需求的逐步增强，不少针对高并发设计的语言流行起来，如 Erlang（代表作 RabbitMQ）、Scala（代表作 Apache Spark），还有最近由 Mozilla 基金会推出的 Rust，以及本书的主角 Go 语言。从目前的状态来看，Go 语言取得的成就远高于其他三种语言，尽管未来究竟哪种语言会成为新的基础架构语言还不可知，但高并发肯定会是一个必备的特性。

Go 语言的作者是 Robert Griesemer、Rob Pike 和 Ken Thompson，与我年龄相仿的程序员对 Ken Thompson 应该不会陌生，他在 UNIX 和 C 语言开发中的巨大贡献让他的名字被大量的程序员所熟知。也正因为如此，在 Go 语言中，我们看到了大量 C 的痕迹和 UNIX 的设计哲学。

本书的作者之一 Brian Kernighan 也是著名的经典 C 语言手册《C 程序设计语言》[⊖]的作者，程序员们甚至将《C 程序设计语言》亲切地称为“K&R C”。而本书的名字也暗示了全书的品质将再现经典。我们也不应该忽视，Kernighan 和 Go 语言作者之一 Rob Pike 是另一本经典作品《程序设计实践》的作者。这些极其珍贵的历史经验和始终在第一线实践的宝贵经验，结合作者多年的教学和写作凝成的文笔，更是为本书的经典品质铸就了坚实的基础。我们在翻译本书中充分体会到：通过对一个又一个语言特性深入浅出的介绍，对设计取舍和具体实例的全面分析，以及与其他语言的综合对比，本书揭示了 Go 语言背后的设计思想。本书能够让新手一开始就走在正确的道路上，让老手能够更精准地把握语言的设计意

⊖ 本书中文版由机械工业出版社引进并出版，ISBN：978-7-111-12806-0。

图，确实是 Go 语言的一本经典之作！

这本书是“七牛人”为推广 Go 语言贡献的第三本书，2012 年上海七牛信息技术有限公司的创始人许式伟和吕桂华合著了国内的第一本 Go 语言书籍《 Go 语言编程》，2013 年又集合该公司的力量翻译了本书，还组织了以 Go 语言为主题的 ECUG 年度大会。“七牛云”从 Go 语言中获益良多，我们也很乐于为 Go 语言以及 Go 社区的发展贡献一份我们自己的力量。

祝大家开卷有益！

李道兵
上海七牛信息技术有限公司首席架构师

前　言

The Go Programming Language

“Go 是一种开源的程序设计语言，它意在使得人们能够方便地构建简单、可靠、高效的软件。”（来自 Go 官网 golang.org）

Go 在 2007 年 9 月形成构想，并于 2009 年 11 月发布，其发明人是 Robert Griesemer、Rob Pike 和 Ken Thompson，这几位都任职于 Google。该语言及其配套工具集使得编译和执行既富有表达力又高效，而且使得程序员能够轻松写出可靠、健壮的程序。

Go 和 C 从表面上看起来相似，而且和 C 一样，它也是专业程序员使用的一种工具，兼有事半功倍之效。但是 Go 远不止是 C 的一种升级版本。基于多种其他语言，它取其精华，去其糟粕。它实现并发功能的设施是全新的、高效的，实现数据抽象和面向对象的途径是极其灵活的。它还实现了自动化的内存管理，或称为垃圾回收。

Go 特别适用于构建基础设施类软件（如网络服务器），以及程序员使用的工具和系统等。但它的的确确是一种通用语言，而且在诸多领域（如图像处理、移动应用和机器学习）中都能发现它的身影。它在很多场合下用于替换无类型的脚本语言，这是由于它兼顾了表达力和安全性：Go 程序通常比动态语言程序运行速度要快，由于意料之外的类型错误而导致崩溃的情形更是少得多。

Go 是个开源项目，所以其编译器、库和工具的源代码是人人皆可免费取得的。来自全世界的社区都在积极地向这个项目贡献代码。Go 的运行环境包括类 UNIX 系统——Linux、FreeBSD、OpenBSD 和 Mac OS X，还有 Plan 9 和 Microsoft Windows。只要在其中一个环境中写了一个程序，那么基本上不加修改它就可以运行在其他环境中。

本书旨在帮助读者立刻开始使用 Go，以及熟练掌握这门语言，并充分地利用 Go 的语言特性和标准库来撰写清晰的、符合习惯用法的、高效的程序。

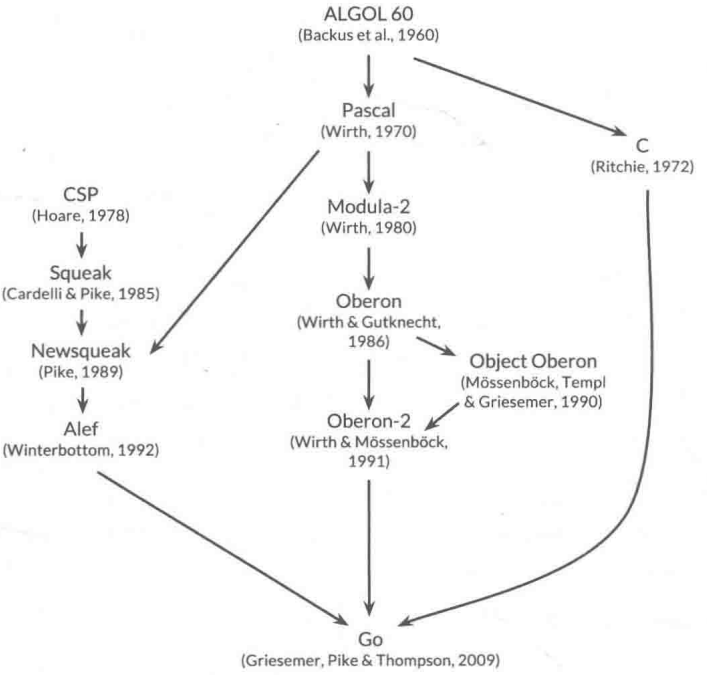
Go 的起源

和生物学物种一样，成功的语言会繁衍后代，这些后代语言会从它们的祖先那里汲取各种优点；有时候，语言间的“混血”会产生异常强大的力量；在一些罕见情况下，某个重大的语言特性也可能凭空出现而并无先例。通过考察语言间的影响，我们可以学得不少知识，比如语言为什么会变成这个样子，以及它适合用于哪些环境，等等。

下图展示了更早出现的程序设计语言对 Go 产生的最重要影响。

Go 有时会称为“类 C 语言”或“21 世纪的 C”。从 C 中，Go 继承了表达式语法、控制流语句、基本数据类型、按值调用的形参传递和指针，但比这些更重要的是，继承了 C 所强调的要点：程序要编译成高效的机器码，并自然地与所处的操作系统提供的抽象机制相配合。

可是，Go 的家谱中还有其他祖先。产生主要影响的是由 Niklaus Wirth 设计的、以 Pascal 为发端的一个语言支流。Modula-2 启发了包概念。Oberon 消除了模块接口文件和模块实现文件之间的差异。Oberon-2 影响了包、导入和声明的语法，并提供了方法声明的语法。



Go 的另一支世系祖先——它使得 Go 相对于当下的程序设计语言显得卓然不群，是在贝尔实验室开发的一系列名不见经传的研究用语言。这些语言都受到了通信顺序进程（Communicating Sequential Process，CSP）的启发，CSP 由 Tony Hoare 于 1978 年在发表的关于并发性基础的开创性论文中提出。在 CSP 中，程序就是一组无共享状态进程的并行组合，进程间的通信和同步采用通道完成。不过，Hoare 提出的 CSP 是一种形式语言，仅用于描述并发性的基本概念，并不是一种用来撰写可执行程序的程序设计语言。

Rob Pike 等人开始动手做一些实验，尝试把 CSP 实现为真正的语言。第一种这样的语言称为 Squeak（“和鼠类沟通的语言”）[⊖]，它是一种用于处理鼠标和键盘事件的语言，其中具有静态创建的通道。紧接着它的是 Newsqueak，它具有类 C 的语句和表达式语法，以及类 Pascal 的类型记法。它是一种纯粹的函数式语言，具有垃圾回收功能，同样也以管理键盘、鼠标和窗口事件为目标。通道变成了“一等”值（first-class value），它可以动态创建并用变量存储。

Plan 9 操作系统将这些思想都纳入一种称为 Alef 的语言中。Alef 尝试将 Newsqueak 改造成一种可用的系统级程序设计语言，但垃圾回收功能的缺失使得它在处理并发性时捉襟见肘。

Go 中的其他结构也会不时显示出某些并非来自祖先的基因。例如，`iota` 多多少少有点 APL 的影子，而嵌套函数的词法作用域则来自 Scheme（以及由之而来的大部分语言）。在 Go 语言中，也可以发现全新的变异。Go 中新颖的 slice 不仅为动态数组提供了高效的随机访问功能，还允许旧式链表的复杂共享机制。另外，`defer` 语句也是 Go 中新引入的。

Go 项目

所有的程序设计语言都反映了其发明者的程序设计哲理，其中相当大的一部分是对于此

⊖ 该单词直译为（老鼠的）吱吱叫声，是为隐喻和双关。——译者注

前语言已知缺点的应对措施。Go 这个项目也诞生于挫败感，这种挫败感来源于 Google 的若干复杂性激增的软件系统。（而且这个问题绝非 Google 所独有的。）

“复杂性是以乘积方式增长的。” Rob Pike 如是说。为了修复某个问题，一点点地将系统的某个部分变得更加复杂，这不可避免地也给其他部分增加了复杂性。在不断要求增加系统功能、选项和配置，以及快速发布的压力之下，简单性往往被忽视了（尽管长期来看，简单性才是好软件的不二法门）。

要实现简单性，就要求在项目的一开始就浓缩思想的本质，并在项目的整个生命周期制定更具体的准则，以分辨出哪些变化是好的，哪些是坏的或致命的。只要足够努力，好的变化就既可以实现目的，又能够不损害 Fred Brooks 所谓软件设计上的“概念完整性”。坏的变化就做不到这一点，而致命的变化则会牺牲“简单性”去换得浅薄的“方便性”。[⊖]但是，只有通过设计上的简单性，系统才能在增长过程中保持稳定、安全和自洽。

Go 项目不仅包括该语言本身及其工具和标准库，还有决不能忽视的一点，就是它保持极端简单性的行为文化。在高级语言中，Go 出现得较晚，因而有一定后发优势，它的基础部分实现得不错：有垃圾回收、包系统、一等公民函数、词法作用域、系统调用接口，还有默认用 UTF-8 编码的不可变字符串。但相对来说，它的语言特性不多，而且不太会增加新特性了。比如，它没有隐式数值类型强制转换，没有构造或析构函数，没有运算符重载，没有形参默认值，没有继承，没有泛型，没有异常，没有宏，没有函数注解，没有线程局部存储。这门语言成熟而且稳定，并且保证兼容更早版本：在旧版本的 Go 语言中写的程序，可以在新版本的编译器和标准库下编译与运行。

Go 的类型系统足可以使程序员避免在动态语言中会无意犯下的绝大多数错误，但相对而言，它在带类型的语言中又算是类型系统比较简单的。其实现方法有时候会导致类型框架林立却彼此孤立的“无类型”程序设计风格，并且 Go 程序员在类型方面不会像 C++ 或 Haskell 程序员那样走极端——反复表达类型安全性以证明语言是基于类型的。但在实际工作中，Go 却能为程序员提供只有强类型的系统才能实现的安全性和运行时性能，而不让程序员承担其复杂性。

Go 提倡充分利用当代计算机系统设计，尤其强调局部性的重要意义。其内置数据类型和大多数库数据结构都经过仔细设计，力求以自然方式工作，而不要求显式的初始化或隐式的构造函数。这么一来，隐藏在代码中的内存分配和内存写入就大大减少了。Go 中的聚合类型（结构体和数组）都以直接方式持有其元素，与使用间接字段的语言相比，它需要更少的存储空间以及更少的分配操作和指针间接寻址操作。正如前面提到的那样，由于现代计算机都是并行工作的，因此 Go 具有基于 CSP 的并行特性。Go 还提供了变长栈来运行其轻量级线程，或称为 goroutine。这个栈初始时非常小，所以创建一个 goroutine 的成本极低，创建 100 万个也完全可以接受。

Go 标准库常常称作“自带电池的语言”，它提供了清晰的构件，以及用于 I/O、文本处理、图形、加密、网络、分布式应用的 API，而且对许多标准文件格式和协议都提供了支持。Go 的库和工具充分地尊重惯例，避免了配置和解释，从而简化了程序逻辑，提高了多种多样的 Go 程序之间的相似性，使得它更容易学习和掌握。采用 go 工具构建的项目，仅使用文件和标识符的名字（在极少情况下使用特殊注释），就可以推断出一个项目使用的所有

⊖ 见《人月神话》。——译者注

库、可执行文件、测试、性能基准、示例、平台相关变体，以及文档。Go 的源代码中就包含了构建的规格说明。

本书结构

我们假定你已用一两种其他语言编过程序，可能是像 C、C++ 或 Java 那样的编译型语言，也可能是像 Python、Ruby 或 JavaScript 那样的解释型语言，所以本书不会像针对一个零基础的初学者那样事无巨细地讲述所有内容。表面上的语法大体雷同，变量、常量、表达式、控制流和函数也一样。

第 1 章是关于 Go 的基础结构的综述，通过十几个完成日常任务（包括读写文件、格式化文本、创建图像，以及在 Internet 客户端和服务器之间通信）的程序来介绍这门语言。

第 2 章讲述 Go 程序的组成元素——声明、变量、新类型、包和文件，以及作用域。第 3 章讨论数值、布尔量、字符串、常量，还解释如何处理 Unicode。第 4 章描述复合类型，即使用简单类型构造的类型，形式有数组、map、结构体，还有 slice（Go 中动态列表的实现）。第 5 章概述函数，并讨论错误处理、宕机（`panic`）和恢复（`recover`），以及 `defer` 语句。

可以看出，第 1 ～ 5 章是基础性的，其内容是任何主流命令式语言都有的。Go 的语法和风格可能与其他语言有所不同，但大多数程序员都能很快掌握这些内容。余下的章节重点讨论 Go 语言中与惯常做法有一定区别的内容，包括方法、接口、并发、包、测试和反射。

Go 以一种不同寻常的方式来诠释面向对象程序设计。它没有类继承，甚至没有类。较复杂的对象行为是通过较简单的对象组合（而非继承）完成的。方法可以关联到任何用户定义的类型，而不一定是结构体。具体类型和抽象类型（即*接口*）之间的关系是隐式的，所以一个具体类型可能会实现该类型设计者没有意识到其存在的接口。第 6 章讲述方法，第 7 章讲述接口。

第 8 章介绍 Go 的并发性处理途径，它基于 CSP 思想，采用 goroutine 和通道实现。第 9 章则讨论并发性中基于共享变量的一些传统话题。

第 10 章讨论包，也就是组织库的机制。该章也说明如何高效地利用 `go` 工具，仅仅这个工具，就提供了编译、测试、性能基准测试、程序格式化、文档，以及完成许多其他任务的功能。

第 11 章讨论测试，在这里 Go 采取了显著的轻量级途径，避免了重重抽象的框架，转而使用简单的库和工具。测试库提供了一个基础，在其之上根据需要可以构建更复杂的抽象。

第 12 章讨论反射，即程序在执行期间考察自身表示方式的能力。反射是一种强大的工具，不过要慎重使用它，该章通过演示如何用它来实现某些重要的 Go 库，解释了如何统筹兼顾。第 13 章解释低级程序设计的细节（它运用 `unsafe` 包来绕过 Go 的类型系统），以及什么时候适合这样做。

每章都配以一定数量的练习，可以用来测试你对 Go 的理解，或者探索对书中示例的扩展和变形。

除了最简单的示例代码以外，书中所有的示例代码都可以从 `gopl.io` 网站的公开 Git 仓库下载。每个示例以其包的导入路径开头和命名，从而能够方便地使用 `go get` 命令获取、构建和安装。你需要选取一个目录作为你的 Go 工作空间，并使 `GOPATH` 环境变量指向它。在必要时，`go` 工具会创建该目录。例如：

```
$ export GOPATH=$HOME/gobook            # choose workspace directory
$ go get gopl.io/ch1/helloworld         # fetch, build, install
$ $GOPATH/bin/helloworld                # run
Hello, 世界
```

要运行这些例子，至少需要使用 1.5 版本的 Go 语言。

```
$ go version
go version go1.5 linux/amd64
```

如果你的计算机上的 go 工具版本太旧或者缺失，请按 https://golang.org/doc/install 上的步骤操作。

更多信息来源

关于 Go 的更多信息，最好的来源就是 Go 的官方网站：https://golang.org。其中列出了文档供读者访问，包括 Go 程序设计语言规范、标准包等。其中还列出 Go 语言教程，指导如何撰写 Go 程序，以及如何撰写好的 Go 程序，还有大量在线文本和视频资源，这些都是本书的主要补充资源。位于 blog.golang.org 的 Go 博客发布的是关于 Go 的最好文章，内容涉及该语言当下的状态、未来的计划、会议方面的报告，还有 Go 相关的大量话题的深度解读。

Go 官网在线访问最有用的一个方面（这也是纸质书的一个令人遗憾的限制），就是提供了从描述 Go 程序的网页上直接运行的能力。这种功能由位于 play.golang.org 的 Go 训练场（Playground）提供，也可以嵌入其他页面，比如 golang.org 的首页，或者由 godoc 工具提供的文档页面。

训练场为读者对简短的程序执行简单的实验提供了方便，有助于读者检验自己对语法、语义和库包的理解，并且它在很多方面取代了其他语言中的读取 – 求值 – 输出循环（Read-Eval-Print Loop，REPL）。它的永久 URL 对于共享 Go 代码段、报告 bug 或提出建议都很有用。

在训练场的基础之上，位于 tour.golang.org 的 Go Tour 就是一系列简短的交互式课程（内容是 Go 语言的基础思想和结构），也是学习整门语言的系统资源。

训练场和 Go Tour 的主要缺点在于它只允许导入标准库，并且很多库特性（比如网络库）都出于可操作性或安全原因限制使用。而要编译和运行每个程序，都要求 Internet 连接。所以，欲进行更详尽的实验，需要在本机上运行 Go 程序。幸运的是，下载过程相当简单，从 golang.org 获取 Go 的安装版本并开始撰写和运行你自己的 Go 程序，用不了几分钟。

由于 Go 是个开源项目，因此你可以从 https://golang.org/pkg 上在线读取标准库中的任何类型或函数的代码，每个供下载的版本都同样包含这些代码。请使用这些代码来弄明白某些程序的运行原理、回答关于程序细节的问题，也可以用它们来学一学专家是如何写出一流的 Go 代码的。

致谢

Go 团队的核心成员 Rob Pike 和 Russ Cox 仔细通读了初稿数次，他们从遣词造句到整体结构都对本书提出了重要的建议。在准备本书的日语版时，柴田芳树所做的贡献大大超过了他负担的义务，他的火眼金睛发现了英语版中的上下文不一致性，以及代码中的错误。非常感谢 Brian Goetz、Corey Kosak、Arnold Robbins、Josh Bleecher Snyder 以及 Peter Weinberger 对全书初稿进行彻底的审查并提出批判性的建议。

感谢 Sameer Ajmani、Ittai Balaban、David Crawshaw、Billy Donohue、Jonathan Feinberg、Andrew Gerrand、Robert Griesemer、John Linderman、Minux Ma、Bryan Mills、Bala Natarajan、Cosmos Nicolaou、Paul Staniforth、Nigel Tao 以及 Howard Trickey 提供的诸多有用建议。也感谢 David Brailsford 和 Raph Levien 的排版建议。

Addison-Wesley 出版社的编辑 Greg Doench 策划了本书，而且一直不断地给予帮助。Addison-Wesley 的制作团队——John Fuller、Dayna Isley、Julie Nahil、Chuti Prasertsith 以及 Barbara Wood——非常杰出，给予作者大量的支持。

Alan Donovan 想要感谢 Google 的 Sameer Ajmani、Chris Demetriou、Walt Drummond 以及 Reid Tatge 让他腾出时间来写作这本书，还要感谢 Stephen Donovan 的建议和及时的鼓励。最重要的是，感谢他的妻子 Leila Kazemi 无限的热情和长期的支持，谅解了他在家庭生活中的疏忽。

Brian Kernighan 对他的朋友和同事深表谢意，他们对 Kernighan 花费了很长时间以通俗易懂的语言写作本书表现出了极大的耐心和理解。尤其是他的妻子 Meg，她为 Kernighan 的写作以及太多的其他事务提供了不懈的支持。

目　录

The Go Programming Language

出版者的话
译者序
前言

第 1 章　入门 …… 1

1.1 hello，world …… 1
1.2 命令行参数 …… 3
1.3 找出重复行 …… 6
1.4 GIF 动画 …… 10
1.5 获取一个 URL …… 12
1.6 并发获取多个 URL …… 13
1.7 一个 Web 服务器 …… 14
1.8 其他内容 …… 17

第 2 章　程序结构 …… 20

2.1 名称 …… 20
2.2 声明 …… 21
2.3 变量 …… 22
2.3.1 短变量声明 …… 22
2.3.2 指针 …… 23
2.3.3 new 函数 …… 25
2.3.4 变量的生命周期 …… 26
2.4 赋值 …… 27
2.4.1 多重赋值 …… 27
2.4.2 可赋值性 …… 28
2.5 类型声明 …… 29
2.6 包和文件 …… 30
2.6.1 导入 …… 31
2.6.2 包初始化 …… 33
2.7 作用域 …… 34

第 3 章　基本数据 …… 38

3.1 整数 …… 38
3.2 浮点数 …… 42
3.3 复数 …… 45
3.4 布尔值 …… 47
3.5 字符串 …… 47
3.5.1 字符串字面量 …… 49
3.5.2 Unicode …… 49
3.5.3 UTF-8 …… 50
3.5.4 字符串和字节 slice …… 53
3.5.5 字符串和数字的相互转换 …… 56
3.6 常量 …… 56
3.6.1 常量生成器 iota …… 57
3.6.2 无类型常量 …… 59

第 4 章　复合数据类型 …… 61

4.1 数组 …… 61
4.2 slice …… 63
4.2.1 append 函数 …… 66
4.2.2 slice 就地修改 …… 69
4.3 map …… 71
4.4 结构体 …… 76
4.4.1 结构体字面量 …… 78
4.4.2 结构体比较 …… 80
4.4.3 结构体嵌套和匿名成员 …… 80
4.5 JSON …… 82
4.6 文本和 HTML 模板 …… 87

第 5 章　函数 …… 92

5.1 函数声明 …… 92
5.2 递归 …… 93
5.3 多返回值 …… 96
5.4 错误 …… 98
5.4.1 错误处理策略 …… 99
5.4.2 文件结束标识 …… 101
5.5 函数变量 …… 102
5.6 匿名函数 …… 104
5.7 变长函数 …… 110
5.8 延迟函数调用 …… 111

5.9 宕机 …… 115
5.10 恢复 …… 118

第 6 章 方法 …… 120

6.1 方法声明 …… 120
6.2 指针接收者的方法 …… 122
6.3 通过结构体内嵌组成类型 …… 124
6.4 方法变量与表达式 …… 127
6.5 示例：位向量 …… 128
6.6 封装 …… 130

第 7 章 接口 …… 133

7.1 接口即约定 …… 133
7.2 接口类型 …… 135
7.3 实现接口 …… 136
7.4 使用 flag.Value 来解析参数 …… 139
7.5 接口值 …… 141
7.6 使用 sort.Interface 来排序 …… 144
7.7 http.Handler 接口 …… 148
7.8 error 接口 …… 152
7.9 示例：表达式求值器 …… 154
7.10 类型断言 …… 160
7.11 使用类型断言来识别错误 …… 161
7.12 通过接口类型断言来查询特性 …… 162
7.13 类型分支 …… 164
7.14 示例：基于标记的 XML 解析 …… 166
7.15 一些建议 …… 168

第 8 章 goroutine 和通道 …… 170

8.1 goroutine …… 170
8.2 示例：并发时钟服务器 …… 171
8.3 示例：并发回声服务器 …… 174
8.4 通道 …… 176
8.4.1 无缓冲通道 …… 177
8.4.2 管道 …… 178
8.4.3 单向通道类型 …… 180
8.4.4 缓冲通道 …… 181
8.5 并行循环 …… 183
8.6 示例：并发的 Web 爬虫 …… 187
8.7 使用 select 多路复用 …… 190
8.8 示例：并发目录遍历 …… 192
8.9 取消 …… 195
8.10 示例：聊天服务器 …… 198

第 9 章 使用共享变量实现并发 …… 201

9.1 竞态 …… 201
9.2 互斥锁：sync.Mutex …… 205
9.3 读写互斥锁：sync.RWMutex …… 208
9.4 内存同步 …… 208
9.5 延迟初始化：sync.Once …… 210
9.6 竞态检测器 …… 212
9.7 示例：并发非阻塞缓存 …… 212
9.8 goroutine 与线程 …… 218
9.8.1 可增长的栈 …… 219
9.8.2 goroutine 调度 …… 219
9.8.3 GOMAXPROCS …… 219
9.8.4 goroutine 没有标识 …… 220

第 10 章 包和 go 工具 …… 221

10.1 引言 …… 221
10.2 导入路径 …… 221
10.3 包的声明 …… 222
10.4 导入声明 …… 223
10.5 空导入 …… 223
10.6 包及其命名 …… 225
10.7 go 工具 …… 226
10.7.1 工作空间的组织 …… 227
10.7.2 包的下载 …… 228
10.7.3 包的构建 …… 229
10.7.4 包的文档化 …… 231
10.7.5 内部包 …… 232
10.7.6 包的查询 …… 233

第 11 章 测试 …… 235

11.1 go test 工具 …… 235
11.2 Test 函数 …… 236
11.2.1 随机测试 …… 239
11.2.2 测试命令 …… 240
11.2.3 白盒测试 …… 242
11.2.4 外部测试包 …… 245

11.2.5 编写有效测试……246
11.2.6 避免脆弱的测试……247
11.3 覆盖率……248
11.4 Benchmark 函数……250
11.5 性能剖析……252
11.6 Example 函数……254

第 12 章 反射……256

12.1 为什么使用反射……256
12.2 reflect.Type 和 reflect.Value……257
12.3 Display：一个递归的值显示器……259
12.4 示例：编码 S 表达式……263
12.5 使用 reflect.Value 来设置值……266
12.6 示例：解码 S 表达式……268
12.7 访问结构体字段标签……271
12.8 显示类型的方法……273
12.9 注意事项……274

第 13 章 低级编程……276

13.1 unsafe.Sizeof、Alignof 和 Offsetof……276
13.2 unsafe.Pointer……278
13.3 示例：深度相等……280
13.4 使用 cgo 调用 C 代码……282
13.5 关于安全的注意事项……286

第 1 章
The Go Programming Language

入　门

本章是对于 Go 语言基本组件的一些说明。希望本章所提供的足够信息和示例，能够使您尽可能快地做一些有用的东西。本书所有的例子都是针对现实世界的任务的。本章将带您尝试体验用 Go 语言来编写各种程序：从简单的文件、图片处理到并发的客户端和服务器的互联网应用开发。虽然在一章里不能把所有东西讲清楚，但是以这类应用作为学习一门语言的开始是一种高效的方式。

学习新语言比较自然的方式，是使用新语言写一些你已经可以用其他语言实现的程序。我们试图说明和解释如何用好 Go 语言，当你写自己的代码的时候，本章的代码可以作为参考。

1.1 hello，world

我们依然从永恒的“hello，world”例子开始，它出现在 1978 年出版的《The C Programming Language》这本书的开头。C 对 Go 的影响非常直接，我们用“hello，world”来说明一些主要的思路：

gopl.io/ch1/helloworld

```
package main

import "fmt"

func main() {
    fmt.Println("Hello, 世界")
}
```

Go 是编译型的语言。Go 的工具链将程序的源文件转变成机器相关的原生二进制指令。这些工具可以通过单一的 go 命令配合其子命令进行使用。最简单的子命令是 run，它将一个或多个以 .go 为后缀的源文件进行编译、链接，然后运行生成的可执行文件（本书中我们使用 $ 符号作为命令提示符）：

```
$ go run helloworld.go
```

不出意料地，这将输出：

```
Hello, 世界
```

Go 原生地支持 Unicode，所以它可以处理所有国家的语言。

如果这个程序不是一次性的实验，那么编译输出成一个可复用的程序比较好。这通过 go build 来实现：

```
$ go build helloworld.go
```

这条命令生成了一个叫作 helloworld 的二进制程序，它可以不用进行任何其他处理，随时执行：

```
$ ./helloworld
Hello, 世界
```

我们给每一个重要的例子都加了一个标签，提示你可以从本书在 gopl.io 的源码库获取代码：

gopl.io/ch1/helloworld

如果执行 go get gopl.io/ch1/helloworld，它将会把源代码取到相应的目录。这将在 2.6 节和 10.7 节进行更多的讨论。

现在我们来说说该程序本身。Go 代码是使用包来组织的，包类似于其他语言中的库和模块。一个包由一个或多个 .go 源文件组成，放在一个文件夹中，该文件夹的名字描述了包的作用。每一个源文件的开始都用 package 声明，例子里面是 package main，指明了这个文件属于哪个包。后面跟着它导入的其他包的列表，然后是存储在文件中的程序声明。

Go 的标准库中有 100 多个包用来完成输入、输出、排序、文本处理等常规任务。例如，fmt 包中的函数用来格式化输出和扫描输入。Println 是 fmt 中一个基本的输出函数，它输出一个或多个用空格分隔的值，结尾使用一个换行符，这样看起来这些值是单行输出。

名为 main 的包比较特殊，它用来定义一个独立的可执行程序，而不是库。在 main 包中，函数 main 也是特殊的，不管在什么程序中，main 做什么事情，它总是程序开始执行的地方。当然，main 通常调用其他包中的函数来做更多的事情，比如 fmt.Println。

我们需要告诉编译器源文件需要哪些包，用 package 声明后面的 import 来导入这些包。“hello，world”程序仅使用了一个来自于其他包的函数，而大多数程序可能导入更多的包。

你必须精确地导入需要的包。在缺失导入或存在不需要的包的情况下，编译会失败，这种严格的要求可以防止程序演化中引用不需要的包。

import 声明必须跟在 package 声明之后。import 导入声明后面，是组成程序的函数、变量、常量、类型（以 func、var、const、type 开头）声明。大部分情况下，声明的顺序是没有关系的。示例中的程序足够短，因为它只声明了一个函数，这个函数又仅仅调用了一个其他的函数。为了节省空间，在处理示例的时候，我们有时不展示 package 和 import 声明，但是它们存在于源文件中，并且编译时必不可少。

一个函数的声明由 func 关键字、函数名、参数列表（main 函数为空）、返回值列表（可以为空）、放在大括号内的函数体组成，函数体定义函数是用来做什么的，这将在第 5 章详细介绍。

Go 不需要在语句或声明后面使用分号结尾，除非有多个语句或声明出现在同一行。事实上，跟在特定符号后面的换行符被转换为分号，在什么地方进行换行会影响对 Go 代码的解析。例如，“{”符号必须和关键字 func 在同一行，不能独自成行，并且在 x+y 这个表达式中，换行符可以在 + 操作符的后面，但是不能在 + 操作符的前面。

Go 对于代码的格式化要求非常严格。gofmt 工具将代码以标准格式重写，go 工具的 fmt 子命令使用 gofmt 工具来格式化指定包里的所有文件或者当前文件夹中的文件（默认情况下）。本书中包含的所有 Go 源代码文件都使用 gofmt 运行过，你应该养成对自己的代码使用 gofmt 工具的习惯。定制一个标准的格式，可以省去大量无关紧要的辩论，更重要的是，如果允许随心所欲的格式，各种自动化的源代码转换工具将不可用。

许多文本编辑器可以配置为每次在保存文件时自动运行 gofmt，因此源文件总可以保持正确的形式。此外，一个相关的工具 goimports 可以按需管理导入声明的插入和移除。它不是标准发布版的一部分，可以通过执行下面的命令获取到：