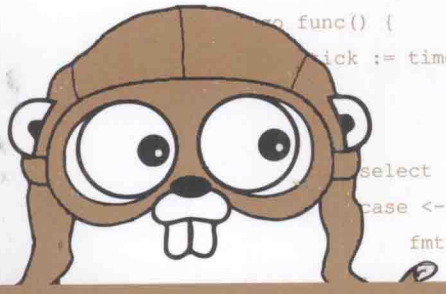
Broadview
www.broadview.com.cn

```
func main() {
    go func() {
        for {
            select {
            case <-time.After(time.Second * 5):
                fmt.Println("timeout ...")
                os.Exit(0)
            }
        }
    }()

    go func() {
        tick := time.Tick(time.Second)

        select {
        case <-tick:
            fmt.Println(time.Now())
```



Go 语言学习笔记

雨痕 / 著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Go语言学习笔记

雨痕 / 著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

作为时下流行的一种系统编程语言，Go 简单易学，性能很好，且支持各类主流平台。已有大量项目采用 Go 编写，这其中就包括 Docker 等明星作品，其开发和执行效率早已被证明。

本书经四年多逐步完善，内容覆盖了语言、运行时、性能优化、工具链等各层面知识。且内容经大量读者反馈和校对，没有明显的缺陷和错误。上卷细致解析了语言规范相关细节，便于读者深入理解语言相关功能的使用方法和注意事项。下卷则对运行时源码做出深度剖析，引导读者透彻了解语言功能背后的支持环境和运行体系，诸如内存分配、垃圾回收和并发调度等。

本书不适合编程初学者入门，可供有实际编程经验或正在使用 Go 工作的人群参考。

图书在版编目（CIP）数据

Go 语言学习笔记 / 雨痕著. —北京：电子工业出版社，2016.7
ISBN 978-7-121-29160-9

Ⅰ. ①G… Ⅱ. ①雨… Ⅲ. ①程序语言－程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2016)第 141787 号

策划编辑：许　艳
责任编辑：许　艳
印　　刷：北京中新伟业印刷有限公司
装　　订：三河市良远印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980　1/16　　印张：29.25　　字数：552 千字
版　　次：2016 年 7 月第 1 版
印　　次：2016 年 7 月第 1 次印刷
定　　价：89.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819　faq@phei.com.cn。

本书的版本历程

2012-01-11	开始学习 Go。
2012-01-15	第一版，基于 R60。
2012-03-29	升级到 1.0。
2012-06-15	升级到 1.0.2。
2013-03-26	升级到 1.1。
2013-12-12	第二版，基于 1.2。
2014-05-22	第三版，基于 1.3。
2014-12-20	第四版，基于 1.4。
2015-06-15	第五版，基于 1.5。
2015-11-01	全新《学习笔记 . 第五版》。
2015-12-09	下卷《源码剖析》截稿，基于 1.5.1。
2016-04-01	上卷《语言详解》截稿，基于 1.6。

前言

前两天忙里偷闲将第五版《Go 学习笔记》上下册合并，预备交给出版社编辑。不经意扫了一眼更新记录，才发觉四年光阴恍然而过。不知从何时起，岁月流逝的速度越来越快，抓不得，留不住。

我很擅长坚持，不知是因为笨，还是性情迟钝的缘故。在给编辑写作者简介时，我努力回忆自己最近二十年的经历，好像除了些纷扰的人和事外，就是一段段在不同技术圈子里日夜探索的记忆，历久弥新。

现在带了些学生，每每交流时，总偷偷庆幸自己是个先行者，没有互联网的“黑暗时代”反而造就了踏实的基础，远不是现今乱花迷眼的境况。看着他们对于具体实现“懵懂无知”的表现，我对于写书这事就愈发虔诚，生怕误了别人的光阴和热情。似乎《学习笔记》这个名字才是最好的诠释，立不得案头，权作闲书，稍能观感一二即可。

因喜爱 C，故对 Go 关注得很早。观望良久，终究受不住诱惑，一头栽了进去。边学边记，于是有了最早的《学习笔记》。只因错漏过多，发到某论坛着实没砸出什么水花来。此后，对于宣传也淡了心思，再不愿出去，只自己默默更新，或发到微博，给一些熟识尚惦记这事的人打个招呼。

某日，一编辑发来消息，询问我是否出版，才恍然知道这书原也是可印的，好像自己从没想过。犹豫再三，且将几本笔记从 GitHub 下架。只可惜，因某些理念不同，最终未能如愿，这一拖就是许多时日。

去年受老谢的邀请，前往上海参加 Gopher China 大会。期间多次被问及何时能有实体书出版，熄了许久的心思方又活过来。年中，重新写了书稿，年底几乎又重来一遍，心底对于出书总有些忐忑。直到圣诞节，才放了下册出来。幸好，并没有人出来指责我粗制滥造，方得心安。

我儿小乖还太小，于是猴年我一人回老家过年。也许是在外面太久，对搬进城里的老家全然陌生，每日里除了陪父母吃饭外，其他时间都用来写上册书稿。偶尔透过窗看见远处的山影，才找回些幼时记忆。书写得意外顺利，即便网络不算通畅也未能影响到我。回京路上，我彻底定了主意，准备交付出版。

节后忙于培训一事，书稿校对稍稍拖后了些。边按章节调整，边请群里的伙伴们帮忙审校，所幸赶在截止日期前完成。样稿交到编辑手里，虽尚有些收尾工作，但总算能放轻松些。这于我是个解脱，困于此的心思总算少了一大半。

依惯例，需在此感谢很多人。其中自然少不了对我多加鼓励的家中太上领导和惦记良久的网络众位大仙们。当然，最需感谢的是群里帮忙校对的小伙伴们，有溺水的鱼、大内总管、starchou、老虎、日下、小 E、春婶、奋斗娃等等。

读者定位

本书并不适合用作编程初学者入门，因内容和文体都太过简练了些。我厚脸推荐给有实际经验或正用 Go 工作的人群，可于路途中当闲书翻看几页。

联系方式

鉴于能力有限，书中难免错漏。如您看到任何问题，请与我联系，以便更正。谢谢！

- 微博：weibo.com/qyuhen
- 邮件：qyuhen@hotmail.com
- 社区：qyuhen.bearychat.com

雨 痕

二〇一六年春

目录

上卷　语言详解

上卷 语言详解

基于 Go 1.6

第 1 章 概述

对我而言，Go 是一门很有意思的编程语言。虽算不得优雅，但也不浅薄。自 C 一脉相承，又吸收了些时髦的东西。最重要的是，它依旧简单。我喜欢简单，平日里也是竭尽所能将复杂的东西简单化。

有关 Go 的宣传语已经太多，优点或缺点都能罗列出长长的清单。我甚至不知道该如何做才能堆砌出本章内容，所以一直拖到全书的最后才去完成。

1.1 特征

我并不想，似乎也没资格为 Go 增添什么新的荣誉。在此仅从这几年的使用经验说说个人看法，一如书名《学习笔记》那样。

语法简单

抛开语法样式不谈，单就类型和规则而言，Go 与 C99、C11 相似之处颇多，这也是我能接受它被冠以“NextC”名号的重要原因。

即便我是个坚定的 C 拥趸，也不得不承认，它处于简单和复杂的两极。C 简单到你每写下一行代码，都能在脑中想象出编译后的模样，指令如何执行，内存如何分配，等等。而 C 的复杂在于，它有太多隐晦而不着边际的规则，着实让人头疼。相比较而言，Go 从零开始，没有历史包袱，在汲取众多经验教训后，可从头规划一个规则严谨、条理简单的世界。

人们习惯拿关键字和控制语句的数量来作为 Go 简单的例证，我倒觉着这并不合适。诚

然，更少的语言规则有助于入门学习，这无可厚非。但更重要的在于，语言规则严谨，没有歧义，更没什么黑魔法变异用法。任何人写出的代码都基本一致，这才是简单的本质。放弃部分“灵活”和“自由”，换来更好的维护性，我觉得是值得的。

将“++”、“--”从运算符降级为语句，保留指针，但默认阻止指针运算。初时的不习惯，并不能掩盖它们带来的长期的好处。还有，将切片和字典作为内置类型，从运行时的层面进行优化，这也算是一种“简单”。

并发模型

时至今日，并发编程已成为程序员的基本技能，在各个技术社区都能看到诸多与之相关的讨论主题。究竟哪种方式是最佳并发编程体验，或许会一直争论下去。但 Go 却一反常态做了件极大胆的事，从根子上将一切都并发化，运行时用 Goroutine 运行所有的一切，包括 main.main 入口函数。

可以说，Goroutine 是 Go 最显著的特征。它用类协程的方式来处理并发单元，却又在运行时层面做了更深度的优化处理。这使得语法上的并发编程变得极为容易，无须处理回调，无须关注执行绪切换，仅一个关键字，简单而自然。

搭配 channel，实现 CSP 模型。将并发单元间的数据耦合拆解开来，各司其职，这对所有纠结于内存共享、锁粒度的开发人员都是一个可期盼的解脱。若说有所不足，那就是应该有个更大的计划，将通信从进程内拓展到进程外，实现真正意义上的分布式。

内存分配

将一切并发化固然是好，但带来的问题同样很多。如何实现高并发下的内存分配和管理就是个难题。好在 Go 选择了 tcmalloc，它本就是为并发而设计的高性能内存分配组件。

可以说，内存分配器是运行时三大组件里变化最少的部分。刨去因配合垃圾回收器而修改的内容，内存分配器完整保留了 tcmalloc 的原始架构。使用 cache 为当前执行线程提供无锁分配，多个 central 在不同线程间平衡内存单元复用。在更高层次里，heap 则管理着大块内存，用以切分成不同等级的复用内存块。快速分配和二级内存平衡机制，让内存分配器能优秀地完成高压力下的内存管理任务。

在最近几个版本中，编译器优化卓有成效。它会竭力将对象分配在栈上，以降低垃圾回

收压力，减少管理消耗，提升执行性能。可以说，除偶尔因性能问题而被迫采用对象池和自主内存管理外，我们基本无须参与内存管理操作。

垃圾回收

垃圾回收一直是个难题。早年间，Java 就因垃圾回收低效被嘲笑了许久，后来 Sun 连续收纳了好多人和技术才发展到今天。可即便如此，在 Hadoop 等大内存应用场景下，垃圾回收依旧捉襟见肘、步履维艰。

相比 Java，Go 面临的困难要更多。因指针的存在，所以回收内存不能做收缩处理。幸好，指针运算被阻止，否则要做到精确回收都难。

每次升级，垃圾回收器必然是核心组件里修改最多的部分。从并发清理，到降低 STW 时间，直到 Go 的 1.5 版本实现并发标记，逐步引入三色标记和写屏障等等，都是为了能让垃圾回收在不影响用户逻辑的情况下更好地工作。尽管有了努力，当前版本的垃圾回收算法也只能说堪用，离好用尚有不少距离。可对一个从 R60 一路跟踪源码走过来的程序员而言，我目睹了 Go Team 为此所付出的全部努力。我在此表达敬意，以及对未来某个飞跃的预期。

静态链接

Go 刚发布时，静态链接被当作优点宣传。只须编译后的一个可执行文件，无须附加任何东西就能部署。这似乎很不错，只是后来风气变了。连着几个版本，编译器都在完善动态库 buildmode 功能，场面一时变得有些尴尬。

暂不说未完工的 buildmode 模式，静态编译的好处显而易见。将运行时、依赖库直接打包到可执行文件内部，简化了部署和发布操作，无须事先安装运行环境和下载诸多第三方库。这种简单方式对于编写系统软件有着极大好处，因为库依赖一直都是个麻烦。事实上，我们也能看到越来越多的工具采用 Go 开发，其中恐怕就有此等原因。

标准库

学习编程语言，早已不是学一点语法规则那么简单。现在更习惯称作选择 Ecosystem（生态圈），而这其中标准库的作用和分量尤为明显。

功能完善、质量可靠的标准库为编程语言提供了充足动力。在不借助第三方扩展的情况下，就可完成大部分基础功能开发，这大大降低了学习和使用成本。最关键的是，标准库有升级和修复保障，还能从运行时获得深层次优化的便利，这是第三方库所不具备的。

Go 标准库虽称不得完全覆盖，但也算极为丰富。其中值得称道的是 net/http，仅须简单几条语句就能实现一个高性能 Web Server，这从来都是宣传的亮点。更何况大批基于此的优秀第三方 Framework 更是将 Go 推到 Web/Microservice 开发标准之一的位置。

当然，优秀第三方资源也是语言生态圈的重要组成部分。近年来崛起的几门语言中，Go 算是独树一帜，大批优秀作品频繁涌现，这也给我们学习 Go 提供了很好的参照。

工具链

完整工具链对于日常开发极为重要。Go 在此做得相当不错，无论是编译、格式化、错误检查、帮助文档，还是第三方包下载、更新都有对应工具。其功能未必完善，但起码算得上简单易用。

内置完整测试框架，其中包括单元测试、性能测试、代码覆盖率、数据竞争，以及用来调优的 pprof，这些都是保障代码能正确而稳定运行的必备利器。

除此之外，还可通过环境变量输出运行时监控信息，尤其是垃圾回收和并发调度跟踪，可进一步帮助我们改进算法，获得更佳的运行期表现。

遗憾的是，发展 6 年的 Go 依然缺少一个真正意义上的调试器，对此我个人颇有怨念。另外，依赖包管理也是社区争论的焦点之一。

1.2 简介

本节简单预览一下语言功能，有个相对完整的印象更利于学习后续知识。

> 可依照第 12 章安装配置编译环境。

本书相关内容和示例运行环境：