Mihalis Tsoukalos 著

# 精通Go语言

（影印版）

Mastering Go

东南大学出版社
SOUTHEAST UNIVERSITY PRESS

Packt

# 精通 Go 语言（影印版）
# Mastering Go

Mihalis Tsoukalos 著

南京　东南大学出版社

**图书在版编目(CIP)数据**

精通 Go 语言:英文/(希)米哈里斯·图卡洛斯(Mihalis Tsoukalos)著. —影印本. —南京:东南大学出版社,2019.5

书名原文:Mastering Go

ISBN 978-7-5641-8322-6

Ⅰ.①精… Ⅱ.①米… Ⅲ.①程序语言-程序设计-英文 Ⅳ.①TP312

中国版本图书馆 CIP 数据核字(2019)第 046189 号

图字:10-2018-497 号

© 2018 by PACKT Publishing Ltd.

Reprint of the English Edition, jointly published by PACKT Publishing Ltd and Southeast University Press, 2019. Authorized reprint of the original English edition, 2018 PACKT Publishing Ltd, the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 PACKT Publishing Ltd 出版 2018。

英文影印版由东南大学出版社出版 2019。此影印版的出版和销售得到出版权和销售权的所有者 —— PACKT Publishing Ltd 的许可。

版权所有,未得书面许可,本书的任何部分和全部不得以任何形式重制。

精通 Go 语言(影印版)

出版发行:东南大学出版社
地　　址:南京四牌楼 2 号　　邮编:210096
出 版 人:江建中
网　　址:http://www.seupress.com
电子邮件:press@seupress.com
印　　刷:常州市武进第三印刷有限公司
开　　本:787 毫米×980 毫米　16 开本
印　　张:37.75
字　　数:739 千字
版　　次:2019 年 5 月第 1 版
印　　次:2019 年 5 月第 1 次印刷
书　　号:ISBN 978-7-5641-8322-6
定　　价:118.00 元

本社图书若有印装质量问题,请直接与营销部联系。电话(传真):025-83791830

Mapt

mapt.io

Mapt is an online digital library that gives you full access to over 5,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

## Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Mapt is fully searchable
- Copy and paste, print, and bookmark content

## PacktPub.com

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

# Contributors

## About the author

**Mihalis Tsoukalos** is a technical author, a Unix administrator, a developer, and a mathematician, who enjoys learning new things. He has written more than 250 technical articles for many publications, including *Sys Admin*, *MacTech*, *Linux User and Developer*, *Usenix ;login:*, *Linux Format*, and *Linux Journal*.

Mihalis is also the author of *Go Systems Programming*, by *Packt Publishing, 2017* and the technical editor for *MongoDB in Action, Second Edition*, by *Manning*. Mihalis' research interests include databases, operating systems, and statistics. You can reach him at `http://www.mtsoukalos.eu/` and `@mactsouk`. He is also a photographer (`http://www.highiso.net/`).

*I would like to thank the people at Packt Publishing for helping me write this book, including Frank Pohlmann and Gary Schwartz, my technical reviewer, Mat Ryer, Radhika Atitkar, for her encouragement and trust, and Kishor Rit, for answering all my questions and encouraging me during the whole process.*

*For all people everywhere: You will never change your life until you change something you do daily!*

试读结束，需要全本PDF请购买 www.ertongbook.com

## About the reviewer

**Mat Ryer** has been programming computers since he was 6 years old. He would build games and programs, first in BASIC on a ZX Spectrum and then in AmigaBASIC and AMOS on Commodore Amiga with his father. Many hours were spent on manually copying the code from the Amiga Format magazine and tweaking variables or moving GOTO statements around to see what might happen. The same spirit of exploration and obsession with programming led Mat to starting work with a local agency in Mansfield, England, when he was 18, where he started to build websites and other online services.

After several years of working with various technologies and industries in London and around the world, Mat noticed a new systems language called Go that Google was pioneering. Since it addressed very pertinent and relevant modern technical challenges, Mat started using it to solve problems while the language was still in the beta stage. He has used it ever since. Mat contributes to open-source projects and founded Go packages, including Testify, Moq, Silk, and Is, as well as a macOS developer tool called BitBar.

In 2018, Mat co-founded Machine Box and still spends a lot of time speaking at conferences, writing about Go on his blog, and is an active member of the Go community.

## Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit `authors.packtpub.com` and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

# Table of Contents

**Preface** 1

**Chapter 1: Go and the Operating System** 7

**The structure of the book** 8

**The history of Go** 8

**Why learn Go?** 9

**Go advantages** 9

Is Go perfect? 11

What is a preprocessor? 11

The godoc utility 12

**Compiling Go code** 13

**Executing Go code** 14

**Two Go rules** 14

You either use a Go package or do not include it 15

There is only one way to format curly braces 16

**Downloading Go packages** 17

**Unix stdin, stdout, and stderr** 19

**About printing output** 19

**Using standard output** 21

**Getting user input** 23

**About := and =** 23

Reading from standard input 24

Working with command-line arguments 26

**About error output** 28

**Writing to log files** 30

Logging levels 31

Logging facilities 31

Log servers 31

A Go program that sends information to log files 32

About log.Fatal() 35

About log.Panic() 36

**Error handling in Go** 38

The error data type 38

Error handling 40

**Additional resources** 43

**Exercises** 44

**Summary** 44

**Chapter 2: Understanding Go Internals** 45

**The Go compiler** 46
**Garbage Collection** 47
The Tricolor algorithm 50
More about the operation of the Go Garbage Collector 53
Unsafe code 55
About the unsafe package 57
Another example of the unsafe package 57
**Calling C code from Go** 59
Calling C code from Go using the same file 59
Calling C code from Go using separate files 60
The C code 61
The Go code 62
Mixing Go and C code 63
**Calling Go functions from C code** 64
The Go package 64
The C code 66
**The defer keyword** 67
**Panic and Recover** 69
Using the panic function on its own 71
**Two handy Unix utilities** 72
The strace tool 73
The dtrace tool 74
**Your Go environment** 76
**The Go Assembler** 78
**Node Trees** 79
**Learning more about go build** 85
**General Go coding advices** 86
**Additional Resources** 86
**Exercises** 87
**Summary** 87
**Chapter 3: Working with Basic Go Data Types** 89
**Go loops** 90
The for loop 90
The while loop 91
The range keyword 91
Examples of Go for loops 91
**Go arrays** 93
Multi-dimensional arrays 94
The shortcomings of Go arrays 97
**Go slices** 97
Performing basic operations on slices 98
Slices are being expanded automatically 100
Byte slices 102

The copy() function 102
Multidimensional slices 105
Another example of slices 105
Sorting slices using sort.slice() 108
**Go maps** 110
Storing to a nil map 112
When you should use a map? 113
**Go constants** 113
The constant generator iota 115
**Go pointers** 118
**Dealing with times and dates** 121
Working with times 123
Parsing times 123
Working with dates 125
Parsing dates 125
Changing date and time formats 127
**Additional resources** 129
**Exercises** 129
**Summary** 129
**Chapter 4: The Uses of Composite Types** 131
**About composite types** 132
**Structures** 132
Pointers to structures 135
Using the new keyword 137
**Tuples** 137
**Regular expressions and pattern matching** 139
Now for some theory 140
A simple example 140
A more advanced example 143
Matching IPv4 addresses 146
**Strings** 151
What is a rune? 154
The Unicode package 156
The strings package 157
**The switch statement** 161
**Calculating Pi with great accuracy** 165
**Developing a key/value store in Go** 168
**Additional resources** 173
**Exercises** 174
**Summary** 174
**Chapter 5: Enhancing Go Code with Data Structures** 175
**About graphs and nodes** 176

**Algorithm complexity** 176
**Binary trees in Go** 177
Implementing a binary tree in Go 178
Advantages of binary trees 180
**Hash tables in Go** 181
Implementing a hash table in Go 182
Implementing the lookup functionality 185
Advantages of hash tables 186
**Linked lists in Go** 186
Implementing a linked list in Go 187
Advantages of linked lists 191
**Doubly linked lists in Go** 191
Implementing a doubly linked list in Go 193
Advantages of doubly linked lists 196
**Queues in Go** 196
Implementing a queue in Go 197
**Stacks in Go** 200
Implementing a stack in Go 200
**The container package** 203
Using container/heap 204
Using container/list 207
Using container/ring 209
**Generating random numbers** 211
Generating random strings 214
**Additional Resources** 217
**Exercises** 217
**Summary** 218

**Chapter 6: What You Might Not Know About Go Packages** 219
**About Go packages** 220
**About Go functions** 220
Anonymous functions 221
Functions that return multiple values 221
The return values of a function can be named! 223
Functions with pointer parameters 225
Functions that return pointers 226
Functions that return other functions 228
Functions that accept other functions as parameters 229
**Developing your own Go packages** 231
Compiling a Go package 233
Private variables and functions 233
The init() function 233
**Reading the Go code of a standard Go package** 236
Exploring the code of the net/url package 236

Looking at the Go code of the log/syslog package 238
**Creating good Go packages** 239
**The syscall package** 241
Finding out how fmt.Println() really works 244
**Text and HTML templates** 246
Generating text output 247
Constructing HTML output 249
Basic SQLite3 commands 257
**Additional resources** 257
**Exercises** 258
**Summary** 258

**Chapter 7: Reflection and Interfaces for All Seasons** 259
**Type methods** 259
**Go interfaces** 262
**About type assertion** 263
**Developing your own interfaces** 265
Using a Go interface 266
Using switch with interface and data types 268
**Reflection** 270
A simple Reflection example 271
A more advanced reflection example 273
The three disadvantages of reflection 276
**Object-oriented programming in Go!** 277
**Additional resources** 281
**Exercises** 281
**Summary** 282

**Chapter 8: Telling a Unix System What to Do** 283
**About Unix processes** 284
**The flag package** 284
**The io.Reader and io.Writer interfaces** 290
Buffered and unbuffered file input and output 290
**The bufio package** 290
**Reading text files** 291
Reading a text file line by line 291
Reading a text file word by word 293
Reading a text file character by character 295
Reading from /dev/random 297
**Reading the amount of data you want from a file** 299
**Why are we using binary format?** 301
**Reading CSV files** 302
**Writing to a file** 305
**Loading and saving data on disk** 308

试读结束，需要全本PDF请购买 www.ertongbook.com

**The strings package revisited** 312
**About the bytes package** 314
**File permissions** 316
**Handling Unix signals** 317
Handling two signals 318
Handling all signals 320
**Programming Unix pipes in Go** 323
Implementing the cat(1) utility in Go 323
**Traversing directory trees** 325
**Using eBPF from Go** 328
**About syscall.PtraceRegs** 329
**Tracing system calls** 331
**User ID and group ID** 336
**Additional resources** 337
**Exercises** 338
**Summary** 339
**Chapter 9: Go Concurrency – Goroutines, Channels, and Pipelines** 341
**About processes, threads, and goroutines** 342
The Go scheduler 343
Concurrency and parallelism 343
**Goroutines** 344
Creating a goroutine 344
Creating multiple goroutines 346
**Waiting for your goroutines to finish** 348
What if the number of Add() and Done() calls do not agree? 350
**Channels** 352
Writing to a channel 352
Reading from a channel 354
Channels as function parameters 356
**Pipelines** 357
**Additional resources** 361
**Exercises** 361
**Summary** 362
**Chapter 10: Go Concurrency – Advanced Topics** 363
**The Go scheduler revisited** 364
The GOMAXPROCS environment variable 366
**The select keyword** 367
**Timing out a goroutine** 370
Timing out a goroutine – take 1 370
Timing out a goroutine – take 2 372
**Go channels revisited** 375
Signal channels 376

Buffered channels 376
Nil channels 379
Channel of channels 380
Specifying the order of execution for your goroutines 383
**Shared memory and shared variables** 386
The sync.Mutex type 387
What happens if you forget to unlock a mutex? 389
The sync.RWMutex type 391
Sharing memory using goroutines 395
**Catching race conditions** 397
**The context package** 403
An advanced example of the context package 407
Worker pools 412
**Additional resources** 417
**Exercises** 418
**Summary** 419
**Chapter 11: Code Testing, Optimization, and Profiling** 421
**The Go version used in this chapter** 422
Comparing Go version 1.10 with Go version 1.9 422
**Installing a beta or RC version of Go** 423
**About optimization** 425
**Optimizing Go code** 425
**Profiling Go code** 426
The net/http/pprof standard Go package 427
A simple profiling example 427
A convenient external package for profiling 435
The web interface of the Go profiler 437
A profiling example that uses the web interface 437
A quick introduction to Graphviz 440
**The go tool trace utility** 441
**Testing Go code** 447
Writing tests for existing Go code 448
**Benchmarking Go code** 452
**A simple benchmarking example** 452
A wrong benchmark function 458
**Benchmarking buffered writing** 459
**Finding unreachable Go code** 464
**Cross-compilation** 465
**Creating example functions** 467
**Generating documentation** 469
**Additional resources** 475
**Exercises** 476
**Summary** 477

**Chapter 12: The Foundations of Network Programming in Go** 479
**About net/http, net, and http.RoundTripper** 480
The http.Response type 480
The http.Request type 481
The http.Transport type 482
**About TCP/IP** 483
**About IPv4 and IPv6** 484
**The nc(1) command-line utility** 484
**Reading the configuration of network interfaces** 485
**Performing DNS lookups** 490
Getting the NS records of a domain 492
Getting the MX records of a domain 494
**Creating a web server in Go** 496
Profiling an HTTP server 499
Creating a website in Go 504
**HTTP tracing** 514
Testing HTTP handlers 517
**Creating a web client in Go** 520
Making your Go web client more advanced 522
**Timing out HTTP connections** 526
More information about SetDeadline 528
Setting the timeout period on the server side 529
Yet another way to time out! 531
**Wireshark and tshark tools** 533
**Additional resources** 533
**Exercises** 534
**Summary** 535
**Chapter 13: Network Programming – Building Servers and Clients** 537
**The net standard Go package** 538
**A TCP client** 538
A slightly different version of the TCP client 540
**A TCP server** 542
A slightly different version of the TCP server 544
**A UDP client** 547
**Developing a UDP server** 549
**A concurrent TCP server** 551
A handy concurrent TCP server 556
**Remote Procedure Call (RPC)** 562
The RPC client 563
The RPC server 564
**Doing low-level network programming** 566
Grabbing raw ICMP network data 569

Where to go next? 574
Additional resources 574
Exercises 575
Summary 576
Other Books You May Enjoy 577
Index 581

# Preface

The book you are reading right now is called *Mastering Go* and is all about helping you become a better Go developer!

I tried to include the right amount of theory and hands on practice, but only you, the reader, can tell if I succeeded or not! Additionally, all presented examples are self-contained, which means that they can be used on their own or as templates for creating more complex applications.

Please try to do the exercises located at the end of each chapter and do not hesitate to contact me with ways to make any future editions of this book even better!

## Who this book is for

This book is for amateur and intermediate Go programmers who want to take their Go knowledge to the next level as well as for experienced developers in other programming languages who want to learn Go without learning again how a `for` loop works.

Some of the information found in this book can be also found in my other book, *Go Systems Programming* by *Packt Publishing*. The main difference between these two books is that *Go Systems Programming* is about developing system tools using the capabilities of Go, whereas *Mastering Go* is about explaining the capabilities and the internals of Go in order to become a better Go developer. Both books can be used as a reference after reading them for the first or the second time.

## What this book covers

`Chapter 1`, *Go and the Operating System*, begins by talking about the history of Go and the advantages of Go before describing the `godoc` utility and explaining how you can compile and execute Go programs. After that, it talks about printing the output and getting user input, working with the command-line arguments of a program, and using log files. The last topic of the first chapter is error handling, which plays a key role in Go.

`Chapter 2`, *Understanding Go Internals*, discusses the Go garbage collector and the way it operates. Then it talks about unsafe code and the unsafe package, how to call C code from a Go program, and how to call Go code from a C program. After that, it showcases the use of the `defer` keyword and presents the `strace(1)` and `dtrace(1)` utilities. In the remaining sections of the chapter, you will learn how to find information about your Go environment and the use of the Go assembler.

`Chapter 3`, *Working with Basic Go Data Types*, talks about the data types offered by Go, which includes arrays, slices, and maps as well as Go pointers, constants, loops, and working with dates and times. You would not want to miss this chapter!

`Chapter 4`, *The Uses of Composite Types*, begins by teaching you about Go structures and the `struct` keyword before discussing tuples, strings, runes, byte slices, and string literals. The rest of the chapter talks about regular expressions and pattern matching, the switch statement, the strings package, the `math/big` package, and about developing a key-value store in Go.

`Chapter 5`, *Enhancing Go Code with Data Structures*, is about developing your own data structures when the structures offered by Go do not fit a particular problem. This includes developing binary trees, linked lists, hash tables, stacks, and queues and learning about their advantages. This chapter also showcases the use of the structures found in the container standard Go package. The last topic of this chapter is random number generation.

`Chapter 6`, *What You Might Not Know About Go Packages*, is all about packages and functions, which also includes the use of the `init()` function, the `syscall` standard Go package, and the `text/template` and `html/template` packages. This chapter will definitely make you a better Go developer!

`Chapter 7`, *Reflection and Interfaces for All Seasons*, discusses three advanced Go concepts: reflection, interfaces, and type methods. The last part of the chapter is about object oriented programming in Go!

`Chapter 8`, *Telling a Unix System What to Do*, is about systems programming in Go, which includes subjects such as the `flag` package for working with command-line arguments, handling Unix signals, file input and output, the bytes package, and the `io.Reader` and `io.Writer` interfaces. As I told you before, if you are really into systems programming in Go, then getting *Go Systems Programming* after reading *Mastering Go* is highly recommended!

试读结束，需要全本PDF请购买 www.ertongbook.com