Preface

Hey, welcome to *Concurrency in Go*! I'm delighted that you've picked up this book and excited to join you in exploring the topic of concurrency in Go over the next six chapters!

Go is a wonderful language. When it was first announced and birthed into the world, I remember exploring it with great interest: it was terse, compiled incredibly fast, performed well, supported duck typing, and—to my delight—I found working with its concurrency primitives to be intuitive. The first time I used the `go` keyword to create a goroutine (something we'll cover, I promise!) I got this silly grin on my face. I had worked with concurrency in several languages, but I had never worked in a language that made concurrency so easy (which is not to say they don't exist; I just hadn't used any). I had found my way to Go.

Over the years I moved from writing personal scripts in Go, to personal projects, until I found myself working on a many-hundreds-of-thousands-of-lines project professionally. Along the way the community was growing with the language, and we were collectively discovering best practices for working with concurrency in Go. A few people gave talks on patterns they had discovered. But there still weren't many comprehensive guides on how to wield concurrency in Go in the community.

It was with this in mind that I set out to write this book. I wanted the community to have access to high-quality and comprehensive information about concurrency in Go: how to use it, best practices and patterns for incorporating it into your systems, and how it all works under the covers. I have done my best to strike a balance between these concerns.

I hope this book proves useful!

Who Should Read This Book

This book is meant for developers who have some experience with Go; I make no attempt to explain the basic syntax of the language. Knowledge of how concurrency is presented in other languages is useful, but not necessary.

By the end of this book we will have discussed the entire stack of Go concurrency concerns: common concurrency pitfalls, motivation behind the design of Go's concurrency, the basic syntax of Go's concurrency primitives, common concurrency patterns, patterns of patterns, and various tooling that will help you along the way.

Because of the breadth of topics we'll cover, this book will be useful to various cross-sections of people. The next section will help you navigate this book depending on what needs you have.

Navigating This Book

When I read technical books, I usually hop around to the areas that pique my interest. Or, if I'm trying to ramp up on a new technology for work, I frantically skim for the bits that are immediately relevant to my work. Whatever your use case is, here's a roadmap for the book with the hopes that it help guide you to where you need to be!

Chapter 1, An Introduction to Concurrency
: This chapter will give you a broad historical perspective on why concurrency is an important concept, and also discuss some of the fundamental problems that make concurrency difficult to get correct. It also briefly touches on how Go helps ease some of this burden.

  If you have a working knowledge of concurrency or just want to get to the technical aspects of how to use Go's concurrency primitives, it's safe to skip this chapter.

Chapter 2, Modeling Your Code: Communicating Sequential Processes
: This chapter deals with some of the motivational factors that contributed to Go's design. This will help give you some context for conversations with others in the Go community and help to frame your understanding of why things work the way they do in the language.

Chapter 3, Go's Concurrency Building Blocks
: Here we'll start to dig into the syntax of Go's concurrency primitives. We'll also cover the `sync` package, which is responsible for handling Go's memory access synchronization. If you haven't used concurrency within Go before and are looking to hop right in, this is the place to start.

Interspersed with the basics of writing concurrent code in Go are comparisons of concepts to other languages and concurrency models. Strictly speaking, it's not necessary to understand these things, but these concepts help you to achieve a complete understanding on concurrency in Go.

Chapter 4, Concurrency Patterns in Go
: In this chapter, we begin to look at how Go's concurrency primitives are composed together to form useful patterns. These patterns will both help us solve problems and avoid issues that can come up when combining concurrency primitives.

  If you've already been writing some concurrent code in Go, this chapter should still prove useful.

Chapter 5, Concurrency at Scale
: In this chapter, we take the patterns we have learned and compose these into larger patterns commonly employed in larger programs, services, and distributed systems.

Chapter 6, Goroutines and the Go Runtime
: This chapter describes how the Go runtime handles scheduling goroutines. This is for those of you who want to understand the internals of Go's runtime.

Appendix
: The appendix simply enumerates various tools and commands that can help make writing and debugging concurrent programs easier.

Online Resources

Go has a very active and passionate community! For those newer to Go, take heart, it will be easy to find friendly, helpful people to guide you along on your path to Go. Here are a few of my favorite community-oriented resources for reading, getting help, and interacting with your fellow gophers:

- *https://golang.org/*
- *https://golang.org/play*
- *https://go.googlesource.com/go*
- *https://groups.google.com/group/golang-nuts*
- *https://github.com/golang/go/wiki*

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This icon signifies a tip, suggestion, or general note.

This icon indicates a warning or caution.

Using Code Examples

All of the code contained in this book can be found on the landing page for the book, *http://katherine.cox-buday.com/concurrency-in-go*. It is released under the MIT license and may be used under those terms.

O'Reilly Safari

Safari (formerly Safari Books Online) is a membership-based training and reference platform for enterprise, government, educators, and individuals.

Members have access to thousands of books, training videos, Learning Paths, interactive tutorials, and curated playlists from over 250 publishers, including O'Reilly Media, Harvard Business Review, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Adobe, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, and Course Technology, among others.

For more information, please visit *http://oreilly.com/safari*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/concurrency-in-go*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

Writing a book is a daunting and challenging task. What you have before you would not have been possible without a team of people supporting me, reviewing things, writing tools, and answering questions. I am deeply grateful to everyone who helped, and they have my sincerest thanks. We did this together!

One swallow does not a summer make...
—Proverb

- Alan Donovan, who helped with the original proposal and also helped set me on my way.
- Andrew Wilkins, who I had the great fortune of working with at Canonical. His insight, professionalism, and intelligence influenced this book, and his reviews made it better.
- Ara Pulido, who helped me see this book through a new gopher's eyes.
- Dawn Schanafelt, my editor, who played a large part in making this book read as clearly as possible. I especially appreciate her (and O'Reilly's) patience while life placed a few difficulties on my path while writing this book.
- Francesc Campoy, who helped ensure I always kept newer gophers in mind.
- Ivan Daniluk, whose attention to detail and interest in concurrency helped ensure this is a comprehensive and useful book.
- Yasushi Shoji, who wrote `org-asciidoc`, a tool that I used to export AsciiDoc artifacts from Org mode. He didn't know he was helping to write a book, but he was always very responsive to bug reports and questions!
- The maintainers of Go: thank you for your dedication.
- The maintainers of Org mode, the GNU Emacs mode with which this book is written. My entire life is in org; seriously, thanks all.
- The maintainers of GNU Emacs, the text editor I wrote this book in. I cannot think of a tool that has served as more of a lever in my life.
- The St. Louis public libraries where most of this book was written.