

Hands-On System Programming with Go

Build modern and concurrent applications for Unix and Linux systems using Golang

Packt >

www.packt.com

Alex Guerrieri

Hands-On System Programming with Go

Build modern and concurrent applications for Unix and Linux systems using Golang

Alex Guerrieri

Packt>

BIRMINGHAM - MUMBAI

Hands-On System Programming with Go

Copyright © 2019 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Commissioning Editor: Richa Tripathi
Acquisition Editor: Shriram Shekhar
Content Development Editor: Tiksha Sarang
Senior Editor: Afshaan Khan
Technical Editor: Sabaah Navlekar
Copy Editor: Safis Editing
Language Support Editor: Storm Mann
Project Coordinator: Prajakta Naik
Proofreader: Safis Editing
Indexer: Rekha Nair
Production Designer: Shraddha Falebhai

First published: July 2019

Production reference: 1040719

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-78980-407-2

www.packtpub.com

To my family, who laid the foundation of the path I am on today

– Alex Guerrieri



Packt.com

Subscribe to our online digital library for full access to over 7,000 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
- Fully searchable for easy access to vital information
- Copy and paste, print, and bookmark content

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.packt.com and, as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customer-care@packtpub.com for more details.

At www.packt.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

Alex Guerrieri is a software developer who specializes in backend and distributed systems. Go has been his favorite tool for the job since first using it in 2013. He holds a degree in computer science engineering and has been working in the field for more than 6 years. Originally from Italy, where he completed his studies and started his career as a full-stack developer, he now lives in Spain, where he previously worked in two start-ups—source{d} and Cabify. He is now working for three companies—as a software crafter for BBVA, one of the biggest Spanish banks; as a software architect for Security First, London, a company focusing on digital and physical security; and as a cofounder of DauMau, the company behind Vidsey, software that generates videos in a procedural manner.

To all the people in my life who supported me and put up with me throughout this endeavor of writing this book, thank you.

About the reviewers

Corey Scott is a principal software engineer currently living in Melbourne, Australia. He has been programming professionally since 2000, with the last 5 years spent building large-scale distributed services in Go.

A blogger on a variety of software-related topics, he is passionate about designing and building quality software. He believes that software engineering is a craft that should be honed, debated, and continuously improved. He takes a pragmatic, non-zealous approach to coding, and is always up for a good debate about software engineering, continuous delivery, testing, or clean coding.

Janani Selvaraj is currently working as a data analytics consultant for Gaddiel Technologies, Trichy, where she focuses on providing data analytics solutions for start-up companies. Her previous experience includes training and research development in relation to data analytics and machine learning.

She has a PhD in environmental management and has more than 5 years' research experience with regard to statistical modeling. She is also proficient in a number of programming languages, including R, Python, and Go.

She reviewed a book entitled *Go Machine Learning Projects*, and also coauthored a book entitled *Machine Learning Using Go*, published by *Packt Publishing*.

Arun Muralidharan is a software developer with over 9 years' experience as a systems developer. Distributed system design, architecture, event systems, scalability, performance, and programming languages are some of the aspects of a product that interest him the most. Professionally, he spends most of his time coding in C++, Python, and C (and perhaps Go in the near future). Away from his job, he also develops software in Go and Rust.

I would like to take this opportunity to thank my family, who have provided me with unconditional support over the years.

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
<hr/>	
Section 1: An Introduction to System Programming and Go	
<hr/>	
Chapter 1: An Introduction to System Programming	9
Technical requirements	9
Beginning with system programming	10
Software for software	10
Languages and system evolution	11
System programming and software engineering	11
Application programming interfaces	11
Types of APIs	12
Operating systems	12
Libraries and frameworks	12
Remote APIs	12
Web APIs	13
Understanding the protection ring	13
Architectural differences	13
Kernel space and user space	14
Diving into system calls	14
Services provided	14
Process control	15
File management	15
Device management	15
Information maintenance	15
Communication	16
The difference between operating systems	16
Understanding the POSIX standard	16
POSIX standards and features	17
POSIX.1 – core services	17
POSIX.1b and POSIX.1c – real-time and thread extensions	17
POSIX.2 – shell and utilities	18
OS adherence	18
Linux and macOS	18
Windows	19
Summary	19
Questions	20
Chapter 2: Unix OS Components	21
Technical requirements	21

Memory management	21
Techniques of management	22
Virtual memory	22
Understanding files and filesystems	23
Operating systems and filesystems	23
Linux	23
macOS	24
Windows	24
Files and hard and soft links	24
Unix filesystem	25
Root and inodes	25
Directory structure	26
Navigation and interaction	26
Mounting and unmounting	27
Processes	27
Process properties	28
Process life cycle	28
Foreground and background	29
Killing a job	29
Users, groups, and permissions	29
Users and groups	30
Owner, group, and others	30
Read, write, and execute	31
Changing permission	32
Process communications	32
Exit codes	32
Signals	33
Pipes	34
Sockets	35
Summary	35
Questions	36
Chapter 3: An Overview of Go	37
Technical requirements	37
Language features	38
History of Go	38
Strengths and weaknesses	38
Namespace	39
Imports and exporting symbols	40
Type system	42
Basic types	42
Composite types	42
Custom-defined types	43
Variables and functions	44
Handling variables	44
Declaration	45

Operations	46
Casting	48
Scope	49
Constants	49
Functions and methods	50
Values and pointers	53
Understanding flow control	55
Condition	55
Looping	56
Exploring built-in functions	57
Defer, panic, and recover	58
Concurrency model	59
Understanding channels and goroutines	59
Understanding memory management	61
Stack and heap	61
The history of GC in Go	63
Building and compiling programs	63
Install	63
Build	64
Run	65
Summary	65
Questions	66
<hr/> Section 2: Advanced File I/O Operations <hr/>	
Chapter 4: Working with the Filesystem	68
Technical requirements	68
Handling paths	68
Working directory	69
Getting and setting the working directory	70
Path manipulation	70
Reading from files	73
Reader interface	74
The file structure	74
Using buffers	75
Peeking content	76
Closer and seeker	78
Writing to file	79
Writer interface	80
Buffers and format	81
Efficient writing	83
File modes	84
Other operations	84
Create	85
Truncate	85
Delete	86

Move	86
Copy	86
Stats	87
Changing properties	87
Third-party packages	87
Virtual filesystems	88
Filesystem events	88
Summary	89
Questions	90
Chapter 5: Handling Streams	91
Technical requirements	91
Streams	91
Input and readers	92
The bytes reader	92
The strings reader	93
Defining a reader	93
Output and writers	94
The bytes writer	94
The string writer	96
Defining a writer	97
Built-in utilities	99
Copying from one stream to another	99
Connected readers and writers	100
Extending readers	101
Writers and decorators	104
Summary	104
Questions	105
Chapter 6: Building Pseudo-Terminals	106
Technical requirements	106
Understanding pseudo-terminals	106
Beginning with teletypes	107
Pseudo teletypes	107
Creating a basic PTY	108
Input management	108
Selector	109
Command execution	109
Some refactor	111
Improving the PTY	112
Multiline input	112
Providing color support to the pseudo-terminal	114
Suggesting commands	116
Extensible commands	117
Commands with status	120
Volatile status	120

Persistent status	122
Upgrading the Stack command	123
Summary	125
Questions	125
Section 3: Understanding Process Communication	
<hr/>	
Chapter 7: Handling Processes and Daemons	127
Technical requirements	127
Understanding processes	127
Current process	128
Standard input	128
User and group ID	128
Working directory	130
Child processes	131
Accessing child properties	132
Standard input	133
Beginning with daemons	134
Operating system support	134
Daemons in action	135
Services	137
Creating a service	137
Third-party packages	143
Summary	144
Questions	144
Chapter 8: Exit Codes, Signals, and Pipes	145
Technical requirements	145
Using exit codes	145
Sending exit codes	146
Exit codes in bash	146
The exit value bit size	147
Exit and deferred functions	147
Panics and exit codes	147
Exit codes and goroutines	148
Reading child process exit codes	149
Handling signals	150
Handling incoming signals	150
The signal package	150
Graceful shutdowns	151
Exit cleanup and resource release	152
Configuration reload	154
Sending signals to other processes	157
Connecting streams	158
Pipes	158
Anonymous pipes	158
Standard input and output pipes	160

Summary	162
Questions	162
Chapter 9: Network Programming	163
Technical requirements	163
Communicating via networks	164
OSI model	164
Layer 1 – Physical layer	164
Layer 2 – Data link layer	165
Layer 3 – Network layer	165
Layer 4 – Transport layer	165
Layer 5 – Session layer	166
Layer 6 – Presentation layer	166
Layer 7 – Application layer	166
TCP/IP – Internet protocol suite	166
Layer 1 – Link layer	166
Layer 2 – Internet layer	167
Layer 3 – Transport layer	167
Layer 4 – Application layer	167
Understanding socket programming	167
Network package	167
TCP connections	168
UDP connections	172
Encoding and checksum	174
Web servers in Go	176
Web server	177
HTTP protocol	177
HTTP/2 and Go	179
Using the standard package	180
Making a HTTP request	180
Creating a simple server	182
Serving filesystem	183
Navigating through routes and methods	183
Multipart request and files	184
HTTPS	186
Third-party packages	187
gorilla/mux	187
gin-gonic/gin	188
Other functionalities	190
HTTP/2 Pusher	190
WebSockets protocol	191
Beginning with the template engine	191
Syntax and basic usage	191
Creating, parsing, and executing templates	192
Conditions and loops	193
Template functions	195
RPC servers	196

Defining a service	196
Creating the server	199
Creating the client	200
Summary	202
Questions	202
Chapter 10: Data Encoding Using Go	203
Technical requirements	203
Understanding text-based encoding	204
CSV	204
Decoding values	204
Encoding values	205
Custom options	207
JSON	207
Field tags	208
Decoder	209
Encoder	209
Marshaler and unmarshaler	210
Interfaces	211
Generating structs	213
JSON schemas	213
XML	214
Structure	214
Document Type Definition	215
Decoding and encoding	216
Field tags	217
Marshaler and unmarshaler	218
Generating structs	219
YAML	219
Structure	220
Decoding and encoding	220
Learning about binary encoding	222
BSON	222
Encoding	223
Decoding	223
gob	223
Interfaces	224
Encoding	224
Decoding	225
Interfaces	226
Proto	227
Structure	227
Code generation	228
Encoding	229
Decoding	230
gRPC protocol	230
Summary	232
Questions	233

Section 4: Deep Dive into Concurrency

Chapter 11: Dealing with Channels and Goroutines	235
Technical requirements	235
Understanding goroutines	235
Comparing threads and goroutines	236
Threads	236
Goroutines	236
New goroutine	237
Multiple goroutines	239
Argument evaluation	239
Synchronization	240
Exploring channels	241
Properties and operations	241
Capacity and size	241
Blocking operations	242
Closing channels	244
One-way channels	245
Waiting receiver	246
Special values	247
nil channels	247
Closed channels	248
Managing multiple operations	249
Default clause	250
Timers and tickers	251
Timers	251
AfterFunc	253
Tickers	253
Combining channels and goroutines	254
Rate limiter	254
Workers	257
Pool of workers	259
Semaphores	262
Summary	264
Questions	265
Chapter 12: Synchronization with sync and atomic	266
Technical requirements	266
Synchronization primitives	266
Concurrent access and lockers	267
Mutex	267
RWMutex	270
Write starvation	271
Locking gotchas	273
Synchronizing goroutines	274
Singleton in Go	276
Once and Reset	277

Resource recycling	278
Slices recycling issues	280
Conditions	281
Synchronized maps	284
Semaphores	287
Atomic operations	287
Integer operations	288
clicker	288
Thread-safe floats	289
Thread-safe Boolean	291
Pointer operations	291
Value	292
Under the hood	292
Summary	293
Questions	294
Chapter 13: Coordination Using Context	295
Technical requirements	295
Understanding context	295
The interface	296
Default contexts	297
Background	297
TODO	297
Cancellation, timeout, and deadline	298
Cancellation	298
Deadline	299
Timeout	299
Keys and values	300
Context in the standard library	302
HTTP requests	302
Passing scoped values	302
Request cancellation	303
HTTP server	305
Shutdown	305
Passing values	306
TCP dialing	308
Cancelling a connection	308
Database operations	310
Experimental packages	310
Context in your application	311
Things to avoid	311
Wrong types as keys	311
Passing parameters	312
Optional arguments	313
Globals	313
Building a service with Context	314
Main interface and usage	314

Exit and entry points	315
Exclude list	316
Handling directories	317
Checking file names and contents	317
Summary	319
Questions	319
Chapter 14: Implementing Concurrency Patterns	320
Technical requirements	320
Beginning with generators	321
Avoiding leaks	323
Sequencing with pipelines	324
Muxing and demuxing	328
Fan-out	328
Fan-in	330
Producers and consumers	331
Multiple producers (N * 1)	332
Multiple consumers (1 * M)	333
Multiple consumers and producers (N*M)	333
Other patterns	335
Error groups	335
Leaky bucket	338
Sequencing	341
Summary	344
Questions	344
Section 5: A Guide to Using Reflection and CGO	
<hr/>	
Chapter 15: Using Reflection	346
Technical requirements	346
What's reflection?	346
Type assertions	347
Interface assertion	348
Understanding basic mechanics	349
Value and Type methods	349
Kind	350
Value to interface	351
Manipulating values	351
Changing values	351
Creating new values	353
Handling complex types	353
Data structures	353
Changing fields	354
Using tags	355
Maps and slices	357
Maps	357

Slices	359
Functions	360
Analyzing a function	361
Invoking a function	362
Channels	363
Creating channels	363
Sending, receiving, and closing	363
Select statement	364
Reflecting on reflection	365
Performance cost	365
Usage in the standard library	366
Using reflection in a package	367
Property files	368
Using the package	371
Summary	372
Questions	372
Chapter 16: Using CGO	373
Technical requirements	373
Introduction to CGO	373
Calling C code from Go	374
Calling Go code from C	375
The C and Go type systems	376
Strings and byte slices	377
Integers	380
Float types	382
Unsafe conversions	382
Editing a byte slice directly	383
Numbers	384
Working with slices	385
Working with structs	386
Structures in Go	387
Manual padding	388
Structures in C	389
Unpacked structures	389
Packed structures	390
CGO recommendations	393
Compilation and speed	393
Performance	393
Dependency from C	394
Summary	394
Questions	395
Assessments	396
Other Books You May Enjoy	410

Preface

This book will provide good, in-depth explanations of various interesting Go concepts. It begins with Unix and system programming, which will help you understand what components the Unix operating system has to offer, from the kernel API to the filesystem, and allow you to familiarize yourself with the basic concepts of system programming.

Next, it moves on to cover the application of I/O operations, focusing on the filesystem, files, and streams in the Unix operating system. It covers many topics, including reading from and writing to files, among other I/O operations.

This book also shows how various processes communicate with one another. It explains how to use Unix pipe-based communication in Go, how to handle signals inside an application, and how to use a network to communicate effectively. Also, it shows how to encode data to improve communication speed.

The book will, toward the end, help you to understand the most modern feature of Go—concurrency. It will introduce you to the tools the language has, along with sync and channels, and how and when to use each one.

Who this book is for

This book is for developers who want to learn system programming with Go. Although no prior knowledge of Unix and Linux system programming is necessary, some intermediate knowledge of Go will help you to understand the concepts covered in the book.

What this book covers

Chapter 1, *An Introduction to System Programming*, introduces you to Go and system programming and provides some basic concepts and an overview of Unix and its resources, including the kernel API. It also defines many concepts that are used throughout the rest of the book.

Chapter 2, *Unix OS Components*, focuses on the Unix operating system and the components that you will interact with—files and the filesystem, processes, users and permissions, threads, and others. It also explains the various memory management techniques of the operating system, and how Unix handles resident and virtual memory.

Chapter 3, *An Overview of Go*, takes a look at Go, starting with some history of the language and then explaining, one by one, all its basic concepts, starting with namespaces and the type system, variables, and flow control, and finishing with built-in functions and the concurrency model, while also offering an explanation of how Go interacts and manages its memory.

Chapter 4, *Working with the Filesystem*, helps you to understand how the Unix filesystem works and how to master the Go standard library to handle file path operations, file reading, and file writing.

Chapter 5, *Handling Streams*, helps you to learn about the interfaces for the input and output streams that Go uses to abstract data flows. It explains how they work and how to combine them and best use them without leaking information.

Chapter 6, *Building Pseudo-Terminals*, helps you understand how a pseudo-terminal application works and how to create one. The result will be an interactive application that uses standard streams just as the command line does.

Chapter 7, *Handling Processes and Daemons*, provides an explanation of what processes are and how to handle them in Go, how to start child processes from a Go application, and how to create a command-line application that will stay in the background (a daemon) and interact with it.

Chapter 8, *Exit Codes, Signals, and Pipes*, discusses Unix inter-process communication. It explains how to use exit codes effectively. It shows you how signals are handled by default inside an application, and how to manage them with some patterns for effective signal handling. Furthermore, it explains how to connect the output and input of different processes using pipes.

Chapter 9, *Network Programming*, explains how to use a network to make processes communicate. It explains how network communication protocols work. It initially focuses on low-level socket communication, such as TCP and UDP, before moving on to web server development using the well-known HTTP protocol. Finally, it shows how to use the Go template engine.

Chapter 10, *Data Encoding Using Go*, explains how to leverage the Go standard library to encode complex data structures in order to facilitate process communications. It analyzes both text-based protocols, such as XML and JSON, and binary-based protocols, such as GOB.

Chapter 11, *Dealing with Channels and Goroutines*, explains the basics of concurrency and channels and some general rules that prevent the creation of deadlocks and resource-leaking inside an application.

Chapter 12, *Synchronization with sync and atomic*, discusses the synchronization packages of the `sync` and `sync/atomic` standard libraries, and how they can be used instead of channels to achieve concurrency easily. It also focuses on avoiding the leaking of resources and on recycling resources.

Chapter 13, *Coordination Using Context*, discusses `Context`, a relatively new package introduced in Go that offers a simple way of handling asynchronous operations effectively.

Chapter 14, *Implementing Concurrency Patterns*, uses the tools from the previous three chapters and demonstrates how to use and combine them to communicate effectively. It focuses on the most common patterns used in Go for concurrency.

Chapter 15, *Using Reflection*, explains what reflection is and whether you should use it. It shows where it's used in the standard library and guides you in creating a practical example. It also shows how to avoid using reflection where there is no need to.

Chapter 16, *Using CGO*, explains how CGO works and why and when you should use it. It explains how to use C code inside a Go application, and vice versa.

To get the most out of this book

Some basic knowledge of Go is required to try the examples and to build modern applications.

Each chapter includes a set of questions that will help you to gauge your understanding of the chapter. The answers to these questions are provided in the *Assessments* section of the book. These questions will prove very beneficial for you, as they will help you revisit each chapter at a glance.

Apart from this, each chapter provides you with instructions on how to run the code files, while the GitHub repository of the book provides the requisite details.

Download the example code files

You can download the example code files for this book from your account at www.packt.com. If you purchased this book elsewhere, you can visit www.packt.com/support and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at www.packt.com.
2. Select the **SUPPORT** tab.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at <https://github.com/PacktPublishing/Hands-On-System-Programming-with-Go>. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at <https://github.com/PacktPublishing/>. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: https://static.packt-cdn.com/downloads/9781789804072_ColorImages.pdf.

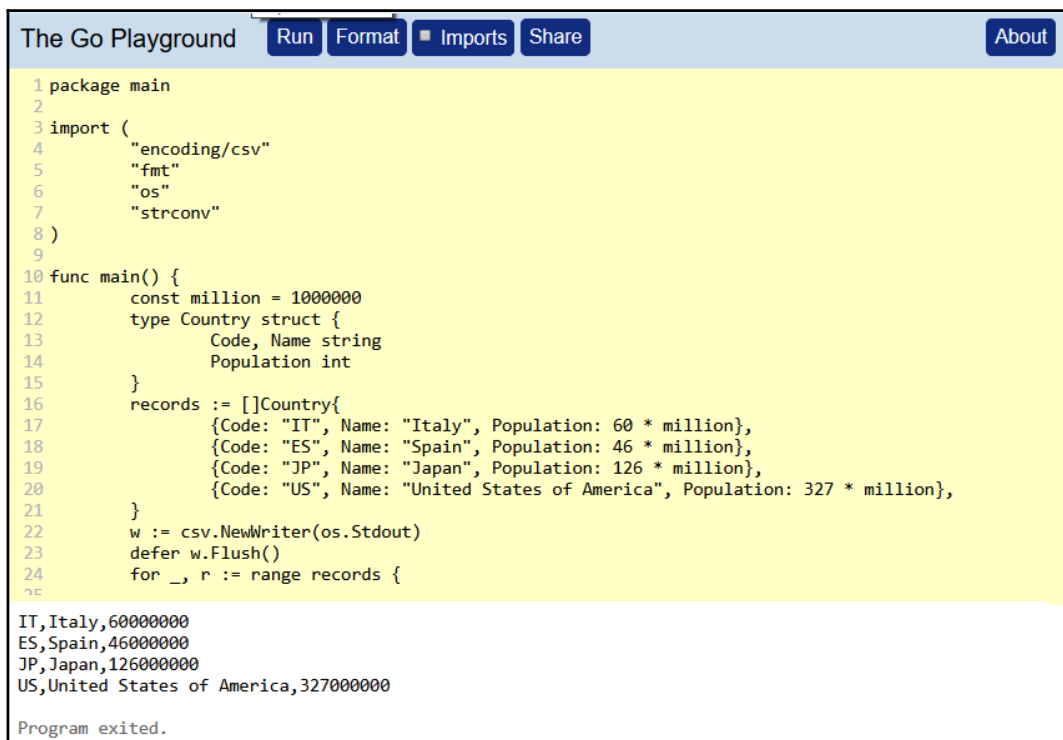
Code in Action

Visit the following link to check out videos of the code being run: <http://bit.ly/2ZWgJb5>.

Playground examples

In the course of the book you will find many snippets of code followed by a link to <https://play.golang.org>, a service that allows you to run Go applications with some limitations. You can read more about it at <https://blog.golang.org/playground>.

In order to see the full source code of such examples, you need to visit the Playground link. Once on the website, you can press the **Run** button to execute the application. The bottom part of the page will show the output. The following is an example of the code running in the Go Playground:



The screenshot shows the Go Playground interface. At the top, there are buttons for 'Run', 'Format', 'Imports', 'Share', and 'About'. The main area contains Go code for a program that prints CSV data for four countries. The code is as follows:

```
1 package main
2
3 import (
4     "encoding/csv"
5     "fmt"
6     "os"
7     "strconv"
8 )
9
10 func main() {
11     const million = 1000000
12     type Country struct {
13         Code, Name string
14         Population int
15     }
16     records := []Country{
17         {Code: "IT", Name: "Italy", Population: 60 * million},
18         {Code: "ES", Name: "Spain", Population: 46 * million},
19         {Code: "JP", Name: "Japan", Population: 126 * million},
20         {Code: "US", Name: "United States of America", Population: 327 * million},
21     }
22     w := csv.NewWriter(os.Stdout)
23     defer w.Flush()
24     for _, r := range records {
25
```

The output of the program is displayed below the code:

```
IT,Italy,60000000
ES,Spain,46000000
JP,Japan,126000000
US,United States of America,327000000
```

Program exited.

If you want, you have the possibility of experimenting by adding and editing more code to the examples, and then running them.

Conventions used

There are a number of text conventions used throughout this book.

CodeInText: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "This type of service includes `load`, which adds a program to memory and prepares for its execution before passing control to the program itself, or `execute`, which runs an executable file in the context of a pre-existing process."

A block of code is set as follows:

```
<meta name="go-import" content="package-name vcs repository-url">
```

Any command-line input or output is written as follows:

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
```

Bold: Indicates a new term, an important word, or words that you see on screen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "In the meantime, systems started to get distributed, and applications started to get shipped in containers, orchestrated by other system software, such as **Kubernetes**."



Warnings or important notes appear like this.



Tips and tricks appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at customercare@packtpub.com.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit www.packt.com/submit-errata, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at copyright@packt.com with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit authors.packtpub.com.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit packt.com.