CSE 124 AND CSE 224: GO PROGRAMMING FUNDAMENTALS

George Porter
April 3, 2025

UC San Diego

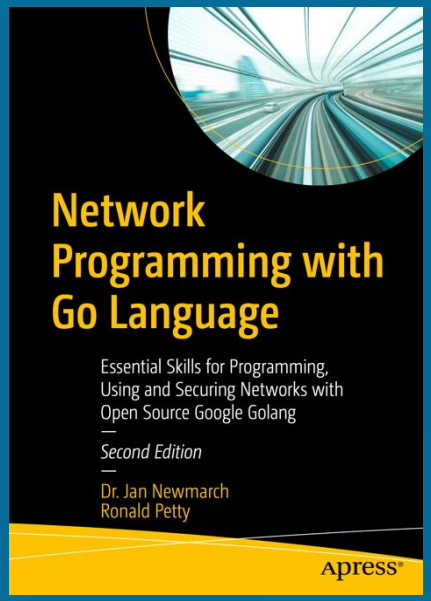
ATTRIBUTION

UC San Diego

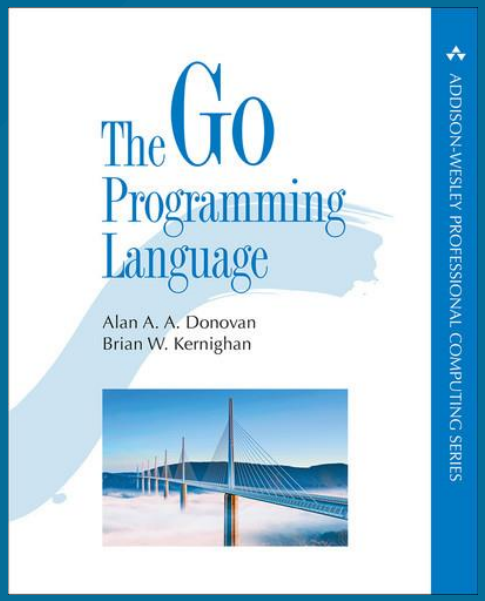
QUICK ADVICE FOR LAB 1

- How do you read bytes of data from a file? Write data to a file?
  - The whole file? One record at a time? One part of a record at a time?
- How do you represent a key-value record type in Go? [struct type]
- How do you maintain some kind of list/array of those records? [slice]
- There is a sort package for built-in types (e.g. ints). But what about custom record types?
  - **Hint: check out sort's Slice() method and bytes package**

REFERENCE MATERIAL



Chapter 2



Chapters 2-5

PROGRAMMING SKILLS FOR THIS CLASS

- We'll be using the "Go" language
  - golang.org
  - Designed at Google in 2007
  - Goals: improve programming productivity in an era of multicore, networked machines, and large codebases
  - Kernighan (of 'C' fame) co-created
- Why?
  - Simple, readable, no explicit memory management needed (similar to Python)
  - High-performance networking
  - Concurrency/parallelism
  - Static typing, strong typing, and efficient runtime
  - Industry-quality and deployed at massive scale

GO OVERVIEW

- Statically typed language
  - Trust me, you don’t want to discover type errors or syntax errors in production
- Compiled, not interpreted
  - Fast compilation
  - Standard code formatting tools (`gofmt`)
- Simple syntax
  - Cross between Python and C, basically
  - Avoids lots of C++ complexity (inheritance, overloading)

WHY GO FOR WEB/DATACENTER/BACKEND COMPUTING?

- 2017 ACM SIGPLAN paper: “Energy Efficiency across Programming Languages” (Pereira et al.)
- Comparison of energy usage across synthetic compute-orientied benchmarks (Binary-trees, chameneos-redux, fannkuch-redux, k-nucleotide, n-body, regex-redux, thread-ring, etc…)
- Not representative, but useful to understand compiler/language behavior (mostly ignoring IO, user interfaces, ease of writing code, tooling support, security, bugs per line of code, etc.)

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84

Table 4 from Pereira et al.

INSTALLING GO (HTTPS://GO.DEV)

GO

Why Go ▾ Learn Docs ▾ Packages Community ▾

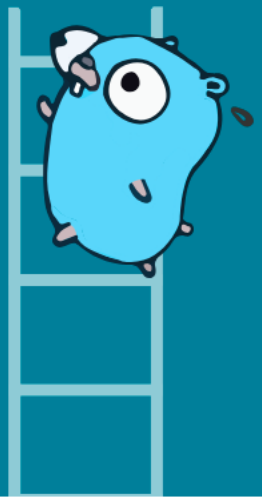
Build simple, secure, scalable systems with Go

- An open-source programming language supported by Google
- Easy to learn and great for teams
- Built-in concurrency and a robust standard library
- Large ecosystem of partners, communities, and tools

Get Started | Download

Download packages for Windows 64-bit, macOS, Linux, and more

The go command by default downloads and authenticates modules using the Go module mirror and Go checksum database run by Google. Learn more.



Companies using Go

Organizations in every industry use Go to power their software and services View all stories

Google | PayPal | American Express | mercado libre | bitly | Capital One

WRITING CODE

- No special requirements
- vim, emacs, neovim, notepad, nano, …
- VSCode
- Jetbrains “Goland” (https://www.jetbrains.com/go/
  - Normally $100/year, but free for students/professors
- Please don’t pay for anything!
  - Just not necessary given all the free options out there

RESOURCES

- https://go.dev/doc/tutorial/getting-started
- Generally, spend some time on go.dev
  - Especially “Selected tutorials” including:
    - “Tour of Go”
    - “Go by example”
  - Today’s lesson roughly drawn from these resources

TODAY'S OUTLINE

We have a LOT to cover in a short period of time, so let's Go!

- Lesson 1: Hello World
- Lesson 2: Variables and types
- Lesson 3: Functions, errors, multiple returns, conditionals, and loops
- Lesson 4: Structs, arrays, slices, and maps
- Lesson 5: File IO and 'defer'
- Lesson 6: Higher-ordered functions

UC San Diego