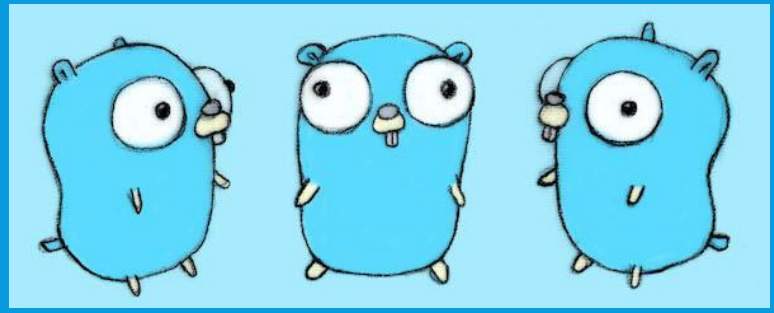
THE GO PROGRAMMING LANGUAGE

Colin J. Mills

Kyle A. Kreutzer



PLEASE NOTE

All material covered here is available at this github link:

https://github.com/c-jm/att_go_slideshow

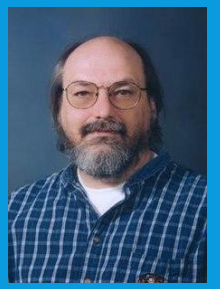
If you would like to follow along please boot up a virtual machine on the school computers and clone the repo. There will be an `install.sh` file that you can run to get all the necessary components.

INTRODUCTION TO GO

- Go is a compiled statically typed language that was created by Google for having a modern, open sourced language for their infrastructure.
- Go's main design goals were to modernize and provide a stable foundation for systems programming. Some features include:
  - **Garbage Collection**
  - **Package System**
  - **Good Standard Library**
  - **Memory Safety**
  - **Concurrency Primitives**

GO'S HISTORY

- Go was started in 2007 by Google, to help introduce a better systems language for their infrastructure.
- Go is interesting due to the software engineering team behind it as well. Its primary developers were:
  - **Robert Griesemer**
  - **Ken Thompson (Of Unix and Bell Labs Fame)**
  - **Rob Pike (Of Plang and Bell Labs Fame)**



MAIN FEATURES WE WILL BE DISCUSSING TODAY

Using the below examples, we will be able to illustrate why Go is a powerful language for modern development.

- **Concurrency within the Go ecosystem.**
- **Packages / Interfaces within Go**
- **Modern System Level Programming**

A NOTE ABOUT THE BUILD SYSTEM

- When designing Go one of the main design considerations was to make the language easy to use. In many cases, *convention* was chosen over *configuration.*
- Working with the language was reduced to a few main tools within the compiler itself. These include:
  - ***go build:*** **Builds the program and produces an executable**
  - ***go run:*** **Builds the executable**
  - ***go fmt:*** **Formats any go source file you give it to the go formatting standard.**
- This gives developers more time to focus on the problem at hand rather than "religious" issues in their style.

BUT FIRST THE BASICS!

THE CANONICAL HELLO WORLD EXAMPLE

```
package main

import "fmt"

func main() {
	fmt.Println("Hello World")
}
```

LET'S BREAK THIS DOWN A BIT

- Every Go program needs to start with a package declaration. Packages provide a way to create reusable libraries that other go programs can call and use within their own scope. All executable programs must start in package main.
- The next noticeable thing, is the import “fmt”. “fmt” is a package that provides common formatting routines analogous to C such as Printf and friends.
- Every Go program starts at main.
- Finally, we reach into the fmt package and call the Println function which just prints a line.

BASIC DATA TYPES

- bool: A basic boolean type which represents true or false.
- (u)int8-(u)int64: All signed and unsigned integer types.
- byte: Byte is an alias for uint8, or char in C
- rune: An alias for int32, typically used for representing Unicode code points.
- float32, float64: Floating point values ranging in bit size.

DECLARING VARIABLES

- In Go there are two ways to declare variables:

- Type Variable Declaration:
- var i int = 10

- Type Inference Variable Declaration
- Similar to the var keyword in C# or auto in C++, Go offers type inference:
- i := 10

OTHER BASICS (LIVE DEMO)

- Functions
- Loops
- Pointers
- Resource Management (defer)

THE REAL MEAT AND POTATOES

- Concurrency
- Receivers
- Interfaces

Concurrency And Go Routines

CONCURRENCY VS. PARALLELISM

Before we begin it is important to note that there is a difference between *concurrency* and *parallelism.*

- **Concurrency:** Refers to a program's ability to start and run tasks in overlapping time periods.
- **Parallelism:** Refers to the program's ability to run tasks at *the same time.* *(*Multiple cores)

GO'S IMPLEMENTATION OF CONCURRENCY

- Go prides itself on providing the programmer with an easy to use interface for performing concurrent tasks.
- This API is exposed through what Go calls go routines. The magic of the implementation is the way they are exposed to the language. To make a function a go routine you just put `go` in front.
- Lets look at an example.

GO'S IMPLEMENTATION OF OOP AND INTERFACES

- Gos implementation of object oriented programming is a bit different then the classical version. Instead of directly having structs which can have methods within them, Go uses the convention of implementing functions with a receiver.

- Interfaces are unique in Go. They can be used as types that are passed to functions, and as long as a struct implements *all* methods in that interface, it can be passed as that interface.

- Lets take a look at an example.

Overall Conclusions

- Go provides many different facilities for writing modern software. It's problem spaces span from low level systems to web applications and everything in between.
- It makes thinking about systems level problems, easier due to a great standard library and a convenient simple specification.
- With the support it has received from Google it has been growing in popularity and used in many areas.

ACTIVITY: LETS USE GO TO WRITE A (VERY) SIMPLE SOFTWARE RENDERER! :)

A COUPLE OF REFERENCES

Mott, J. (n.d.). Games With Go. Retrieved March 20, 2018, from https://gameswithgo.org/

Cox, R. (n.d.). Retrieved March 20, 2018, from https://tour.golang.org/welcome/1

A. (2013, October 20). Rob Pike - 'Concurrency Is Not Parallelism'. Retrieved March 20, 2018, from https://www.youtube.com/watch?v=cN_DpYBzKso

R. (2012, June 20). A Tour of Go. Retrieved March 20, 2018, from https://www.youtube.com/watch?v=ytEkHepK08c