COS418 Precept 1 - Programming in Go

9/07/22

Welcome to the first COS418 precept in Fall 2022!

Because all the assignments in this class use Go, there will be an emphasis on programming in Go in the first few weeks of precepts. In general, topics covered in precepts will be centered around the assignments.

The best way to learn Go is to work through examples and exercises. The golang tour (https://tour.golang.org/list) is a great resource for this and will help you significantly with assignment 1.1 due next week on 9/16.

Environment Setup

Teaching support is only provided for work done using the cycles or courselab servers. The tests are known to work under Go 1.13. Thus later version 1 releases should also work. Learn more about semantic versioning here (https://semver.org/). Please follow instructions for requesting access to the servers if you have not already done so: https://github.com/cos418atPrinceton/assignments_template/blob/master/setup.md

Tools

There are many commonly used tools in the Go ecosystem. The three most useful starting out are: Go fmt and Go vet, which are built-ins, and Golint, which is similar to the splint tool you used in COS217.

- Go fmt: https://golang.org/cmd/gofmt/
- Go vet: https://golang.org/cmd/vet/
- Golint: https://github.com/golang/lint

Editors

For those of you in touch with your systems side (this *is* Distributed Systems, after all), there are quite a few resources for Go development in both emacs and vim.

- Emacs: https://github.com/dominikh/go-mode.el, additional information available http://dominik.honnef.co/posts/2013/03/emacs-go-1/

- Vim: https://github.com/fatih/vim-go, additional resources http://farazdagi.com/blog/2015/vim-as-golang-ide/

As many Princeton COS students have become attached to Sublime, here are the two indispensible Sublime packages for Go development: GoSublime and Sublime-Build. And -- learning from the ancient emacs-vi holy war -- it would be inviting trouble to offer Sublime information without likewise dispensing the must-have Atom plugin: Go-Plus.

- GoSublime: https://github.com/DisposaBoy/GoSublime
- Sublime-Build: https://github.com/golang/sublime-build
- GoPlus: https://atom.io/packages/go-plus
- GoPlus walkthrough and additional info: https://rominirani.com/setup-go-development-environment-with-atom-editor-a87a12366fcf#.v49dtbadi

Goland is also a good IDE to try. You can use it for free as a student. When you input your @princeton.edu email and your graduation date, you should get at least one year free.

- Goland: https://www.jetbrains.com/go/

Coding Style

All of the code you turn in for this course should have good style. Make sure that your code has proper indentation, descriptive comments, and a comment header at the beginning of each file, which includes your name, userid, and a description of the file.

It is recommended to use the standard tools gofmt and go vet. You can also use the Go Checkstyle tool for advice on how to improve your code's style. It would also be advisable to produce code that complies with Golint where possible.

- Go Checkstyle: https://github.com/qiniu/checkstyle
- Golint: https://github.com/golang/lint

How do I git?

Please read this Git Tutorial (https://git-scm.com/docs/gittutorial).

The basic git workflow in the shell (assuming you already have a repo set up):

- git pull

- do some work
- git status (shows what has changed)
- git add *all files you want to commit*
- git commit -m "brief message on your update"
- git push

All programming assignments require Git for submission.

We are using Github for distributing and collecting your assignments. At the time of seeing this, you should have already joined the cos418atPrinceton organization on Github and forked your private repository. Your Github page should have a link. Normally, you only need to clone the repository once, and you will have everything you need for all the assignments in this class.

- cos418atPrinceton organization: https://github.com/orgs/cos418atPrinceton

Exercises

Easy

1. Print the first 10 squared numbers.
2. Print the first 10 fibonacci numbers.
3. Fizz-buzz: replace multiples of 3 with *Fizz* and multiples of 5 with *Buzz*, and multiples of both with *FizzBuzz*. Print the first 10 numbers in this sequence.
4. Write a function that reverses a slice.
5. Write a function that returns the number of unique items in a slice.

Medium

6. Implement a binary tree in which each node contains a number. Then write a function that sums all the numbers in the tree.
7. Write a function that launches *n* goroutines to square all entries in a slice in parallel, where *n* is provided by the caller. Your function should block until all goroutines terminate.

A little harder

8. Given an *n* by *n* matrix, print all the entries in spiral order. Now do it in both directions (clockwise and anti-clockwise).
9. Implement merge-sort using goroutines. If the size of the input slice is *n*, how many goroutines are launched in total?