Proposal: Go Exercise

JI YOON OK, jyo1031@gmail.com
BENJAMIN GEE, bmgee27@gmail.com
CHRISTINE SAM, xtinesam@gmail.com

---

1 INTRODUCTION

For our project, we chose to create a program that highlights a particular feature of a lesser used programming language. Specifically, we will be using Go, which is a relatively esoteric language we haven't been exposed to, to implement an exercise interval timer. Through this program, we aim to highlight the particular features of Go that support concurrency.

2 STRENGTHS OF GO

Go is a general purpose object-oriented programming language that was aimed to be suitable for modern large scale systems [1]. Some of its major features include concise syntax, expressive type system, concurrency, garbage collection, fast compilation of code, and efficient execution [1]. Of particular interest to our project, is Go's support for concurrency.

Go supports concurrency through two main language features: goroutines and channels. A goroutine is a thread that is managed by the GO runtime. [2]. Goroutines are described as being cheaper to create and lighter than normal operating system threads because stacks are small, segmented and sized on demand [1]. Like many threading models, goroutines run on the same address space and therefore must be synchronized [2]. This, however, is easily accomplished by the use of channels. Channels are what connects goroutines and allow for communication and synchronization between them. Sends and receives will block goroutines until both the sender and receiver are ready, which eliminates the use of mutexes [3]. Channels can be either unbuffered (by default) or buffered [2]. With goroutines and channels combined, concurrent programs are easy and efficient to write and are highly scalable.

3 PROGRAM TO BUILD

Using Go, we can take advantage of goroutines and channels for concurrency to create a exercise interval timer program. The goal of our program is to exploit and showcase the power and ease of creating a concurrent program using goroutines and channels.

As a starter program, we aim to use goroutines to show how easy it is to start up a concurrent program. With our example of an exercise timer, we can use goroutines to model separate interval timers, each starting its own independent timer for each goroutine executed. This will simulate for example, a runner who may have a total timer running in the background, but has an individual timer

start per lap to keep track of average times per lap. Furthermore, with our multiple interval timers running simultaneously, we can now use channels to synchronize and allow the concurrent events to communicate with each other. An application of channels to our program could be to pass in time differentials between lap timers and have it output to the screen whether or not the current lap was faster or slower than the previous. This would involve computing the difference, creating channel, sending in the value and have one of the goroutines receive the difference and output it to the console.

4 SKETCH/PLAN OF HOW TO ACHIEVE MILESTONES

4.1 80% Research Report

In order to write up a background research report for this milestone, we will spend appropriate time into the following tasks: 1. researching about language we chose (Go) 2. familiarizing ourselves with the language and concurrency features 3. compiling our research results into a report. Research topics will include, but are not limited to language specification, other applications written in different languages and how they exploit concurrency, and similar programs implemented using channels and goroutines. To familiarize ourselves with the language, we will read blog posts, tutorials, and actually play with the language. In our report, we will emphasize the value and importance of using channels and goroutines in the program we plan on implementing.

4.2 90% Research Report

By this milestone, we will be familiar with Go, and will have enough knowledge to create a simple application that uses concurrency by following a tutorial. First, we will create and document plan on how to achieve the 100% level by breaking up our goal into core goals. Then we can further expand on these core goals into a fuller, extended goals. Lastly, we will create a simple application that demonstrates the technical skills needed for completion of project, and document the implementation performed.

4.3 100% Research Report

In order to achieve the 100% mark, we will first create a poster explaining the importance and neat features of our project, and then divide up core goals and start implementing features associated with our program. Then, once the core goals have been implemented, we can start diving up full goals laid out for the project plan and implement these features. Once program functionalities are in place, we can create project description that illustrates what we had accomplished in completing the project.

5 STARTING DOCUMENTS

To get us started with the project, we have looked for a few webpages that will help familiarize us with Go.

As a general overview to the Go language, we will use the following, which consists of many slides and videos that introduce the language:

- https://github.com/golang/go/wiki/GoTalks

As a general guide to Go syntax and language specifications we will use the following:

- https://golangbot.com/learn-golang-series/

As an intro to concurrency using goroutines and channels, we will use the following:

- https://www.golang-book.com/books/intro/10

REFERENCES

[1] Rob Pike. 2010. Another Go at Language Design [pdf slides]. Retrieved from GoTalks: https://github.com/golang/go/wiki/GoTalks
[2] Matt Aimonetti. 2016. Chapter 8 Concurrency. Retrieved from Go Bootcamp: http://www.golangbootcamp.com/book/concurrency
[3] Mark McGranaghan. 2013. Go by Example: Channels. Retrieved from Go by Example: https://gobyexample.com/channels