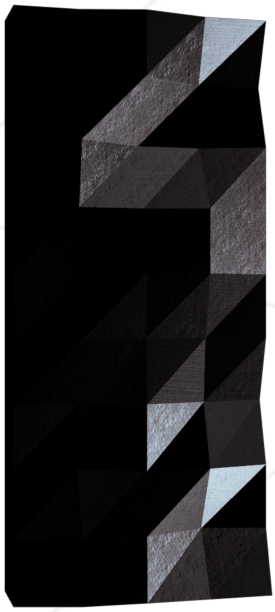

Frozen

Let it GO

Giacomo Guiulfo
giacomo@42.us.org

Summary:
Go go
Go Johnny go!

Contents

Chapter I

Foreword

A snowman is an anthropomorphic snow sculpture often built by children in regions with sufficient snowfall. In many places, typical snowmen consist of three large snowballs of different sizes with some additional accoutrements for facial and other features. Due to the sculptability of snow, there is also a wide variety of other styles. Common accessories include branches for arms and a rudimentary smiley face, with a carrot standing in for a nose. Human clothing, such as a hat or scarf, may be included. Low-cost and availability are the common issues, since snowmen are usually abandoned to the elements once completed.

Snow becomes suitable for packing when it approaches its melting point and becomes moist and compact. Making a snowman of powdered snow is difficult since it will not stick to itself, and if the temperature of packing snow drops, it will form an unusable denser form of powdered snow called crust. Thus, a good time to build a snowman may be the next warm afternoon directly following a snowfall with a sufficient amount of snow. Using more compact snow allows for the construction of a large snowball by simply rolling it until it grows to the desired size.

Figure 1.1 Picture of a snowman.

Reports indicate that this project is extremely easier to do if you listen to the following songs in an infinite loop:

- Wake Me Up Before You Go Go
- Go Your Own Way
- I'll Go Crazy If I Don't Go Crazy Tonight
- Go Go Go
- My Heart Will Go On
- The Show Must Go On
- Don't Let the Sun Go Down on Me
- I'll never Let You Go
- Go Zone
- Go Down
- Let Her Go
- Are You Gonna Go My Way
- Let it Go (Rock Version)

To make things easier and even more fun, I took the liberty to make a playlist for all of you. Yes, I know. You're welcome.

Chapter II

Introduction

In this rush, you have the chance to (re)discover the amazing world of IRC. Even better, you have to opportunity to do it in a new language, Go.

According to the definition of Wikipedia, an Internet Relay Chat (IRC) is an application layer protocol that facilitates communication in the form of text. The chat process works on a client/server networking model. IRC clients are computer programs that a user can install on their system. These clients communicate with chat servers to transfer messages to other clients. IRC is mainly designed for group communication in discussion forums, called channels, but also allows one-on-one communication via private messages as well as chat and data transfer, including file sharing. You can read more here.

Now let's talk about Go. First of all, take a look at its beautiful mascot:

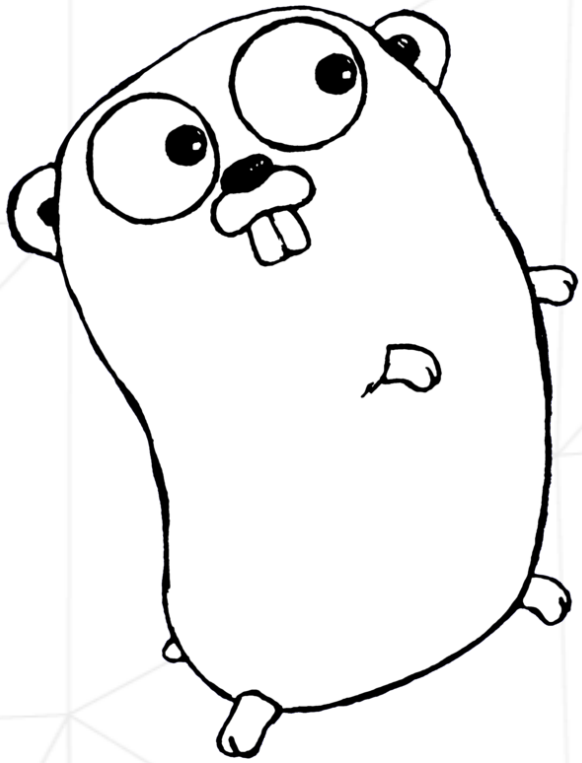

Figure 2.1 Golang's mascot, the Gopher.

And that's pretty much is all you need to know about Go, seriously. However, you should probably check out these resources and then come back:

- A Tour of Go
- Learn Go in Y Minutes
- Go's Official Documentation

Chapter III

Goals

There are two main goals for this project: discover Go and enrich your concurrent programming skills.

Go is a fascinating programming language, really. You may even find it similar to C, your favourite language. In fact, Go is heavily influenced by C because its creator, Ken Thompson, created the Unix operating system together with Dennis Ritchie, the creator of C. However, it incorporates a lot of new concepts that you will soon start to enjoy. Some of them are type inference, garbage collection, interfaces, and anonymous functions, just to name a few. Two of Go's built-in facilities will be your new best friends for this project: goroutines and channels.

Concurrent computing is a form of computing in which computations are performed in an interleaved fashion. It must not be confused with parallel computing, in which computations are executed at the same time. In this project, you will create an IRC server suitable for chatting. IRC servers allow the connection of multiple clients at different times or simultaneously. Due to its nature, an IRC server is a great example of a concurrent program, and recreating one in a language very suitable for concurrency is the perfect opportunity to take your skills to the next level.

When in doubt of how an IRC server works, refer to the RFC standard

Chapter IV

General instructions

- This project will only be corrected by actual human beings. You are therefore free to organize and name your files as you wish, although you need to respect some requirements listed below.
- Each member of your group must be fully aware of the works of the project. Should you choose to split the workload, make sure you understand what each member has done. During the defense, you'll be asked questions, and the final grade will be based on the worst explanations.
- It goes without saying, but gathering the group is your responsibility. You've got all the means to get in contact with your teammates: phone, email, carrier pigeon, spiritism, etc. So don't bother spouting up excuses. Life isn't always fair, that's just the way it is.
- However, if you've really tried everything and one of your teammates remains unreachable: do the project anyway, and we'll try and see what we can do about it during defense. Even if the group leader is missing, you still have access to the submission directory.
- Only the Go standard library is allowed for this project. The use of any third-party library is strictly forbidden.
- You have to handle errors in a sensitive manner. In no way can your program quit in an unexpected manner (runtime error, panic, etc). If you are unsure, this rush is not for you.
- You'll have to submit at the root of your folder, a file called author containing the logins of your team members, each one followed by a newline.

```
$> cat -e author
xlogin$
ylogin$
$>
```

- You can ask your questions on the forum and on the slack channel #frozen.

Chapter V

Mandatory part

Now the good stuff:

- Write an IRC server using only the Go language.
- The server should use concurrency and goroutines to handle multiple clients.
- Users must be able to sign up on your server through a client with a unique username and password. They should be able to log in with those credentials later on.
- Users must also have a unique but modifiable nickname. Passwords have to be modifiable too. However, usernames cannot change once they are created.
- Your server must support standard messaging:
  - Users should be able to join different channels.
  - If a user joins a channel that doesn't exist, that channel must be created.
  - Channel moderators can kick other users from a channel.
  - Users can also communicate to other users through private messaging.

You have to be able to use any IRC client to connect and test the functionality of your server.

It is NOT required for your server to support cross-server communication.

Chapter VI

Bonus part

Once you're done with the mandatory part and sure that it is 100% functional, consider doing the following bonuses:

- Support file sharing on your server.
- Provide an IRC client that is able to connect to any IRC server in the world. Obviously, it must be able to connect to yours too.
- Make your server conform to the IRC RFC standard. Evidently, this means that cross-server communication must be supported.
- A beautiful GUI for your IRC client.

You can add some bonuses of your own creation if you like, but most of the bonus points will be reserved for the above bonuses.

Chapter VII

Turn-in and peer-evaluation

Turn your work in using your `GiT` repository, as usual. Only work present on your repository will be graded in defense.