Getting Started with Go

CMPSC443 Fall 2017 – Professor McDaniel
November 14, 2017

Installing go: I highly recommend you install Go and program all of the assignments on a virtual machine. If you don't have access to VMWare, please email vallalla@engr.psu.edu and tell him that you are a 443 student of Professor McDaniel and what platform you would like to run on (Windows or Mac). Follow his instructions and obtain the most recent Ubuntu distribution (17.10 as of 11/17) and install that as a VM. Once you have your VM installed, you need to install the go language, which is very straight forward—just enter the following command inside of xterm or other terminal:

```
sudo apt install golang-go
```

Setup: The first thing you need to understand the philosophy of the go language. A central part of that philosophy is the notion of the workspace which contains all of your code. It take a little while to wrap your head around it, but read the following and it will help sort this out. (https://golang.org/doc/code.html#Workspaces)

Part of this philosophy is that all of your code is kept in the workspace and the go language exploits this. The driver of the environmental setup is the **GOPATH** environment variable. By default, this is "**~/go**", and all of your files will be under that. You can move the by resetting **GOPATH** to something else.

Each program you are going to build is placed in a subdirectory under the "**GOPATH/src**" subdirectory. This is where you are going to place all of the code for that program (unless you develop some packages that will be used for other programs too). For starters, just put everything in the same directory.

Creating code: Stealing the example from the link above, create a subdirectory under source called hello. E.g.,

```
mkdir ~/go/src/hello
```

Now create a file "**hello.go**" (all go code uses the go extension) in that directory, as follows:

```
package main
import "fmt"
func main() {
        fmt.Printf("hello, world\n")
}
```

Building and executing: Using the go language. The key to go is that it has a weird interaction with the compiler, in that it has a tool that does everything from compiling to installing the program within the "**~/go/bin**" directory. Walking through this a bit, lets start with the basic

building of the program we just made. Run the following commands.

```
cd ~/go/src/hello
go build
```

Now look in the directory and you will see that you have a new program in that directory, "hello". Run it just like any other program, and it will look something like this:

```
mcdaniel@ubuntu:~/go/src/hello$ ./hello
hello, world
```

Note that the program is named for the directory, NOT the go file. Try the following commands to see what this means.

```
rm hello
mv hello.go xhello.go
go build
ls -lt
```

See that the program is still "**hello**"? There you go.