GoLang With Robin

By Craig Walker May 14, 2023

Go,batexpe,robin

As far as robin is concerned, if you want to edit robin “go” code, do the following:

```
git clone https://gitlab.inria.fr/batsim/batexpe.git
cd batexpe
<change code>
export GO111MODULE=on          ← this supposedly turns off GOPATH mode, or something like that.  It is needed.
go mod init batexpe
make
<your robin will be in ./bin/>

you don’t have to use make
you could just do:
go run ./cmd/robin/robin.go (generate <args…> | path/to/yaml.yaml)
everytime you want to run robin


# to download deps beforehand
go mod init batexpe
go mod vendor        (download and tidy also work for downloading dependencies but vendor makes a folder called vendor with all the stuff in there)
go build -mod=vendor …
```

Caveats: fmt.Print,fmt.Println,fmt.Printf don’t always work, especially when you get an error. Though some others have reported this to be because of timing(main() exiting before it had a chance to print to stdout), I worked to address this (channel, time.sleep) and it didn’t make any difference. I think they were talking about go-functions, like calling “go <some function>” within a go script.

fyi: a package is a folder of .go files, a module is a folder of packages

Robin files

- **robin.go**
  - takes care of starting robin, providing usage and getting the arguments, and calling the right function(generate yaml, vs run yaml).
- **expe.go**
  - takes care of getting things out to a yaml or in from a yaml
- **exec.go**
  - takes care of execution (executeOne and executeBatsimAndSched)
- **batsim.go**
  - takes care of making a temp command file with the batsim command in it, then giving it a couple more arguments, then running it and parsing the output of it (the output is basically the command that it took in), so as to make sure the batsim command will work (I guess) and so the arguments are properly formed (I guess). It pulls the arguments out of the output and sends them back to exec.go to run along with the scheduler.

Go syntax - Work In Progress

variables:

set them using name:=value

declare with var <name> <type>[=value]