Introduction à Go – TD0

Franklin ”Snaipe” Mathieu

2023

1 Installation

Cette formation utilise Go 1.21.3 pour ses travaux dirigés, et assume un environnement Linux x86_64.

Connectez-vous sur votre environnement de développement, ouvrez un terminal, et lancez ces commandes:

```
$ curl -LO https://go.dev/dl/go1.21.3.linux-amd64.tar.gz
_ % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    75  100    75    0     0    328      0 --:--:-- --:--:-- --:--:--   330
100 63.5M  100 63.5M    0     0  17.1M      0  0:00:03  0:00:03 --:--:-- 18.6M
$ sudo tar -C /usr/local -xzf go1.21.3.linux-amd64.tar.gz
$ rm -f go1.21.3.linux-amd64.tar.gz
```

Go est maintenant installé sous `/usr/local/go`.

Pour ajouter la commande `go` dans le PATH:

```
$ echo 'export PATH="$PATH:/usr/local/go/bin"' >> ~/.profile && source ~/.profile
```

Vous pouvez maintenant confirmer que Go est bien installé et fonctionnel:

```
$ go version
go version go1.21.3 linux/amd64
```

2 Utiliser Go

2.1 go mod init

Dans cette section, nous allons couvrir l’utilisation de la plupart des outils standard de Go sur un simple projet ”Hello, World!”

```
$ mkdir hello
$ cd hello
$ go mod init example.com/hello
go: creating new go.mod: module example.com/hello
```

`go mod init` initialise le projet local avec une définition de module dans un fichier go.mod:

```
$ ls
go.mod
$ cat go.mod
module example.com/hello

go 1.21.3
```

Le fichier `go.mod` contient le nom du module, sa version de Go minimale supportée, ainsi que tout autre dépendance sur des modules externes (non-visibles ici).

2.2 go build

Écrivons un programme pour dire bonjour; créez un fichier nommé `hello.go`, avec ce contenu:

```
package main

import "fmt"

func main() {
    fmt.Println("Bonjour tout le monde!" )
}
```

Vous pouvez compiler et lancer ce programme:

```
$ go build
$ ls
go.mod  hello  hello.go
$ ./hello
Bonjour tout le monde!
```

2.3 go run

`go run` est une version plus courte de `go build && ./hello`:

```
$ go run hello.go
Bonjour tout le monde!

$ go run hello.go arg1 arg2 ...
Bonjour tout le monde!
```

2.4 go fmt

`go fmt` est le formatteur de code standard de Go.

hello.go n'est pas correctement formatté; formattons-le avec `go fmt`:

```
$ go fmt
hello.go
$ cat hello.go
package main

import "fmt"

func main() {
        fmt.Println("Bonjour tout le monde!")
}
```

2.5 go doc

Vous pouvez consulter la documentation d'un symbole spécifique à tout moment avec la commande `go doc`:

```
$ go doc fmt.Println
package fmt // import "fmt"

func Println(a ...any) (n int, err error)
    Println formats using the default formats for its operands and writes to
    standard output. Spaces are always added between operands and a newline
    is appended. It returns the number of bytes written and any write error
    encountered.
```

La documentation est aussi consultable en ligne: `https://pkg.go.dev/fmt#Println`