Introduction au langage GO

SDI 2021-2022

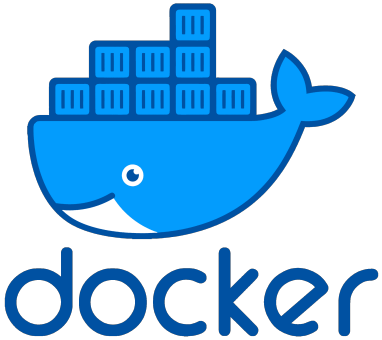
G. Riondet - N. Abdennadher

Présentation GO

- Créé en 2009 par Google à partir du concept de
  - Robert Griesemer
  - Rob Pike
  - Ken Thompson
- Basé sur le langage C
- Inspiré du
  - C
  - Pascal

- But
  - Faciliter et accélérer la programmation à grande échelle

Exemples de programmes basés sur GO



Syntaxe de base

- Ecriture d'un programme classique : Hello world !
- Ecriture d'une fonction
- Initialisation de variable
- Equivalent boucle while
- Gestion de la concurrence
- Librairies principales

Ecriture d'un programme classique : Hello world !

```go
package main

import (
    "fmt"
)

func main() {
    fmt.Printf("Hello, world\n")
}
```

Ecriture d'une fonction

```
1  func foo(a int, b int) int {
2      return a + b
3  }
```

Initialisation de variable

```go
func foo(a int, b int) int {
    c := a + b
    return c
}
```

```go
func foo(a int, b int) int {
    var c int = a + b
    return c
}
```

Equivalent boucle while

- Pas de mot clé `while` en GO

```go
for {
    //Do something
}

for a < b {
    //Do something
}

for i := 0; i < 10; i++{
    //Do something
}

var array []int = [3]int{1,2,3}

for index, value := range array{
    //Do something
}
```

Concurrence en GO

- Goroutine
  - Si le mot clé `go` est présent avant l'appel d'une fonction, alors pas d'attente de retour de la fonction (appel asynchrone)

```go
package main

import (
    "fmt"
)

func foo(threadID int){
    fmt.Printf("Hello from thread %d", threadID)
}

func main() {
    go foo(1)
    go foo(2)
    go foo(3)
    fmt.Printf("Hello, world\n")
}
```

Librairies principales

- fmt
  - Print en console
- time
  - Gestion du temps
- net
  - Création et gestion de serveur tcp
- io
  - Interaction avec la machine : lecture de fichiers,…
- math
  - Divers fonctions mathématiques (min, max, sqrt, log,…)

Compilation et exécution

- 0 tolérance sur les variables non utilisées

```
1  go run yourprogram.go
2
3          OR
4
5  go build yourprogram.go
6  ./yourprogram
```

Exercice

- Etape 1 : Réaliser deux programme GO : `server.go` et `client.go`
  - Le server envoie un message toutes les deux secondes
  - Le client reçoit les messages et les print
  - Version synchrone et locale
- Etape 2 : Transformer en asynchrone
  - Traitement du message après réception adapté pour maximiser le temps d'écoute des nouveaux messages
  - Version asynchrone et locale
- Etape 3 : Déployer le serveur et le client sur 2 VMs sur AWS
  - Faire fonctionner le tout en utilisant les IP publics des VMs
  - Version asynchrone et cloud

Suite du cours

- RDV dans 15 min en A406-A404 pour les labos