Les Pièges du Go 3

Michel Casabianca
casa@sweetohm.net

Le langage de programmation Go a la réputation d'être simple à apprendre. Cependant, il recèle quelques pièges qui peuvent être difficiles à détecter pour un novice. Cette série d'articles propose d'en désamorcer quelques uns.

Supposons que nous voulions lancer des goroutines pour afficher les entiers du *0* à *9*, nous pourrions écrire (code sur le Playground):

```
package main

import "sync"

const max = 10

func main() {
        wg := sync.WaitGroup{}
        wg.Add(max)
        for i := 0; i < max; i++ {
                go func() {
                        println(i)
                        wg.Done()
                }()
        }
        wg.Wait()
}
```

Mais si nous lançons ce programme, nous obtenons :

```
$ go run broken.go
10
10
10
10
10
10
10
10
10
10
```

Ce n'est pas ce à quoi nous nous attendions ! Pourquoi ce code ne fonctionne-t-il pas ? Comment le corriger ?

Explication

Ce code ne fonctionne pas parce que les goroutines mettent un certain temps à démarrer et lorsqu'elles le sont, la boucle dans laquelle elles tournent est déjà arrivée à sa valeur maximale de *10*. Donc toutes le goroutines affichent cette valeur. Il peut arriver que quelques goroutines affichent une valeur différente, mais ce code ne fonctionne pas pour autant.

La meilleure façon de corriger ce code est de passer la valeur de *i* à la fonction lancée par la goroutine (code sur le Playground):

```
package main

import "sync"

const max = 10

func main() {
        wg := sync.WaitGroup{}
        wg.Add(max)
        for i := 0; i < max; i++ {
                go func(i int) {
                        println(i)
                        wg.Done()
                }(i)
        }
        wg.Wait()
}
```

Ce code fonctionne car la valeur de *i* est passée par valeur a la goroutine et est donc recopiée pour la lancer :

```
$ go run fixed.go
9
0
1
2
3
4
5
6
7
8
```

On notera cependant que ce code n'affiche pas toujours les valeurs de *i* dans un ordre croissant... car le certain temps que mettent les goroutines à se lancer est incertain.

Conclusion

Vous devriez passer tous les paramètres nécessaires à une fonction lancée dans une goroutine et ne pas compter sur le contexte.

Enjoy!