Go Booby Traps 3

Michel Casabianca
casa@sweetohm.net

Go programming language is easy to learn, but there are some tricky traps. This article series is trying to show these booby traps so that you avoid them.

Let's say we want to start goroutines to print integers from *0* to *9*. We could write (code on the 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()
}
```

But if we run this program we print:

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

This is not what we expected! Why is this code broken? How to fix it?

Explanation

This code doesn't work because goroutines take some time to start and when they run, the loop has

already reached its maximum value of *10*. Thus all goroutines print this value. Sometimes goroutines print other values, but this code doesn't work anyway.

The best way to fix this code is to pass value for *i* to the goroutine function (code on the 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()
}
```

This code works because value for *i* is passed by value to the goroutine and is copied to start the goroutine:

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

You may notice that this code doesn't print values in ascending order... this is because goroutines take some time to start and this time is not exactly the same for all goroutines.

Conclusion

You should pass all parameters to functions that run in goroutines and don't rely on context.

Enjoy!