Go Booby Traps 1

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 consider this code (code on the Playground):

```
package main

func modify(array [3]string) {
    array[0] = "modified!"
}

func main() {
    var array = [3]string{"unchanged", "foo", "bar"}
    modify(array)
    println(array[0])
}
```

In this example, we send an array by value to a function that modifies it. As we sent it by value, original value is not modified. Thus when we run it, we have:

```
$ go run array.go
unchanged
```

Now let's replace our array with a slice (code on the Playground):

```
package main

func modify(slice []string) {
        slice[0] = "modified!"
}

func main() {
        var slice = []string{"unchanged", "foo", "bar"}
        modify(slice)
        println(slice[0])
}
```

When we run this example:

```
$ go run slice.go
modified!
```

This might sound strange as we sent the slice by value, original should not have been modified!

Take five minutes to try to figure out why this happens and then have a look at the explanation below.

Explanation

When we send a slice by value, everything happens as if we had send it by reference. Slices behave as if we passed an array by reference (code on the Playground):

```
package main

func modify(array *[3]string) {
        array[0] = "modified!"
}

func main() {
        var array = [3]string{"unchanged", "foo", "bar"}
        modify(&array)
        println(array[0])
}
```

Which behaves as slices:

```
$ go run reference.go
modified!
```

The explanation is quite simple: slices are structures that contain a reference to an array, as we can see in source code for slices:

```
type slice struct {
        array unsafe.Pointer
        len   int
        cap   int
}
```

Thus, when you pass a slice by value, the structure is copied, and the array is the same, and when you modify the array, the original slice points to the same array, and it is also modified.

Conclusion

Beware, when you pass a slice by value, you pass it by reference instead.

Enjoy!