今日总结 2019-03-25

每日一学

问题一：

```
go mod 命令
download    download modules to local cache (下载依赖的module到本地cache))
edit        edit go.mod from tools or scripts (编辑go.mod文件)
graph       print module requirement graph (打印模块依赖图))
init        initialize new module in current directory (在当前文件夹下初始化一个新的module, 创建go.mod文件))
tidy        add missing and remove unused modules (增加丢失的module，去掉未用的module)
vendor      make vendored copy of dependencies (将依赖复制到vendor下)
verify      verify dependencies have expected content (校验依赖)
why         explain why packages or modules are needed (解释为什么需要依赖)
```

讨论结果：

1. 后续实战项目都会使用 module 来管理依赖，所以大家可以抽时间试试，实际使用下上面的命令，有问题可以评论交流。

面试题

问题一：

以下代码是否能正常编译？如果不能，哪里有问题？

```
package main

func main() {
    Sum(1, 2)
    len("Hello")
}
func Sum(a, b int) int {
    return a + b
}
```

讨论结果：

1. 不能正常编译；
2. Sum自定义的函数，虽然有返回值，但是可以不用变量接收；
3. len() 内置函数必须要有变量接收，不然编译报错；
4. 球主答案：
    - Go 语言规范规定：append cap complex imag len make new real unsafe.Alignof unsafe.Offsetof unsafe.Sizeof 以上这些 builtin 函数必须接收返回值，也就是不能用于表达式语句；而且函数没有此限制；

- 咱们思考为什么会有此限制？以下是我的思考： 我们知道，函数有两种作用，1）通过输入、处理、得到输出（返回值）；2）函数的副作用（比如输出到控制台）。 有些函数，既有输出，也有副作用；有些却只有其中之一。而 builtin 中那些函数，只有输出，没有副作用，唯一的作用是返回值，因此，Go 不允许不接收返回值。而自定义函数，Go 没有也没必要检查是否只有副作用，因此允许不接收返回值。

问题二：

以下代码有问题吗？为什么？

```go
type student struct {
    Name string
    Age  int
}

func parseStudent() {
    m := make(map[string]*student)
    stus := []student{
        {Name: "zhou", Age: 24},
        {Name: "li", Age: 23},
        {Name: "wang", Age: 22},
    }
    for _, stu := range stus {
        m[stu.Name] = &stu
    }
}
```

讨论结果：

1. 有问题，m中的三个元素的值都是stus最后一个元素的值，将map打印出来的结果：`map[wang:0xc000048400 zhou:0xc000048400 li:0xc000048400]`；
2. stu地址不变，即所取地址不变，随着迭代，地址中的值为最后一个值，地址总是指向这个值。所以代码是有问题的；
3. 期待球主明天的讲解 TODO……

今日链接

- GCTT | Go 语言中的两种 slice 表达式
- 之前也有很多小伙伴问过我要资料，看到星球里有人问。我也分享一波，希望对大家有所帮助！如下:
    1. https://golang.org/doc/effective_go.html
    2. GitHub - Unknwon/the-way-to-go_ZH_CN: 《The Way to ...
    3. GitHub - qyuhen/book: 学习笔记
    4. GitHub - EDDYCJY/blog: 煎鱼的博客，啊。
    5. GitHub - chai2010/advanced-go-programming-book: 《G... (入门后再看)
    6. GitHub - changkun/go-under-the-hood: Go 源码研究 (1.... (入门后再看)
    7. GitHub - gopherchina/conference

自取，顺便安利。谢谢