[Go语言第一课 1](#_Toc21183)

[Go语言简介 1](#_Toc29242)

[Go语言安装与设置 1](#_Toc24769)

[Go语言开发环境 1](#_Toc31035)

# Go语言第一课

## Go语言简介

### 特点

1. 静态类型，编译型开源语言
2. 脚本话语法，多种编程范式(面向过程与面向对象)
3. 原生，给力的并发编程支持

### 优势

1. 脚本汉化语法
2. 静态类型+编译型，程序运行速度有保障
3. 原生的支持并发编程(降低开发，维护成本)
4. 程序可以更好的执行

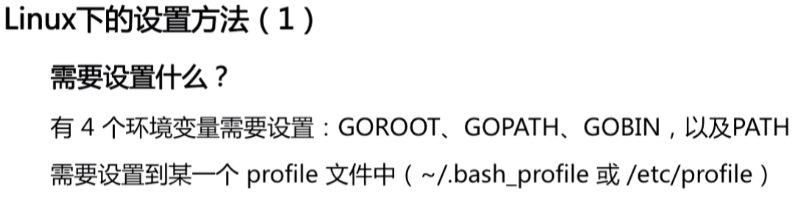
### 劣势

1. 语法糖并没有Python和Ruby那么多
2. 目前的程序运行速度还不及C
3. 第三方函数库暂时不像绝对主流的编程语言那么多

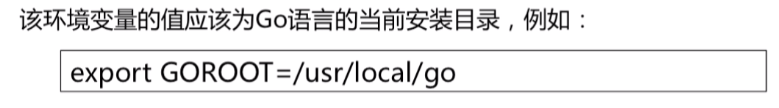
## Go语言安装与设置

跟java安装差不多

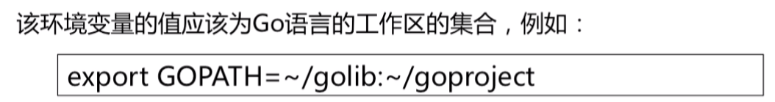
### Linux下的设置方法



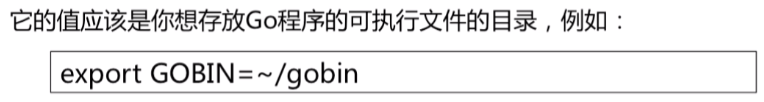
### GOROOT



### GOPATH



### GOBIN



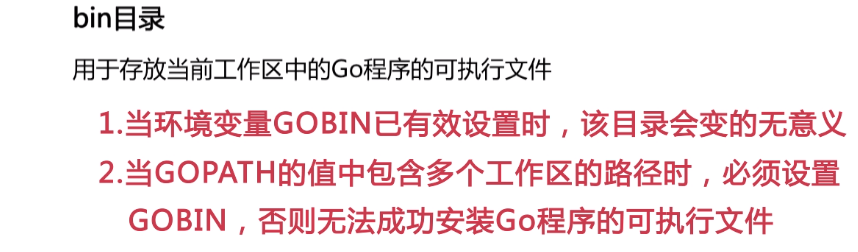
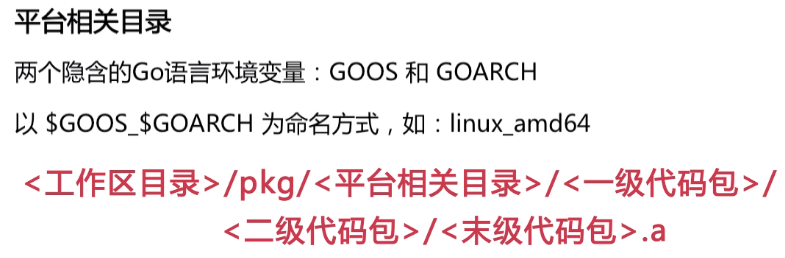
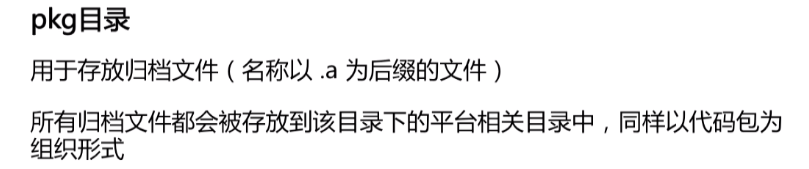
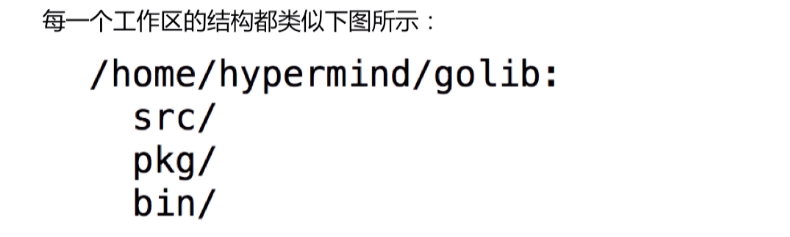
## Go语言开发环境

### 工作区和GOPATH

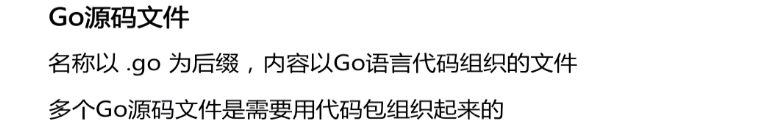
工作区是方式Go源码文件的目录

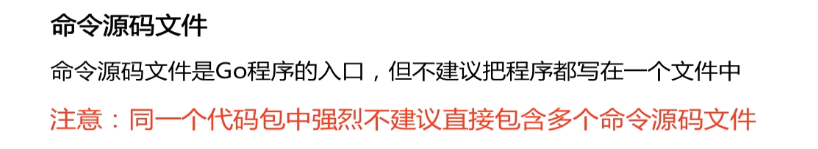
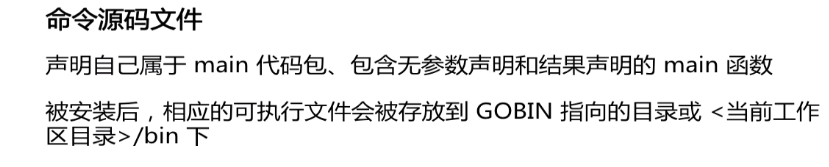
一般情况下,Go源码文件都需要存放到工作区中

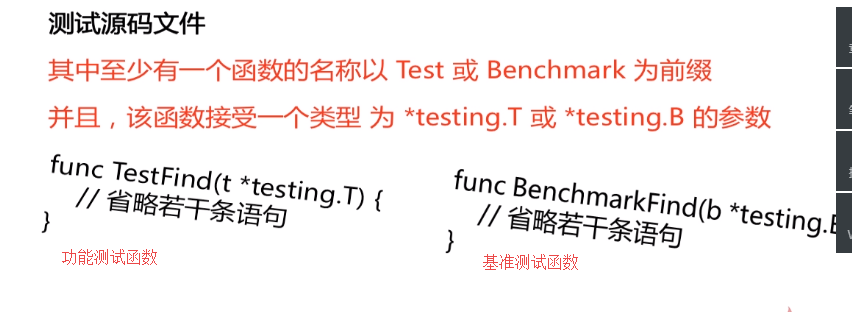
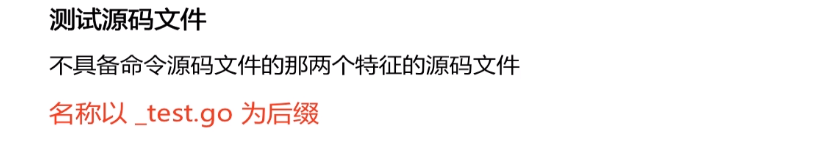
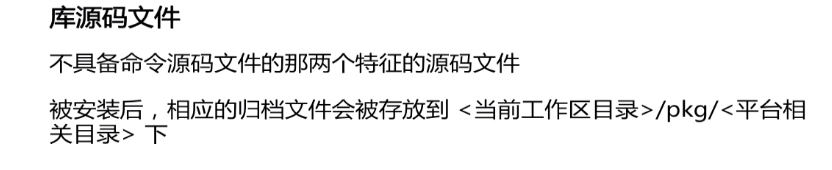
但是对于命令源码文件来说，这都是必须



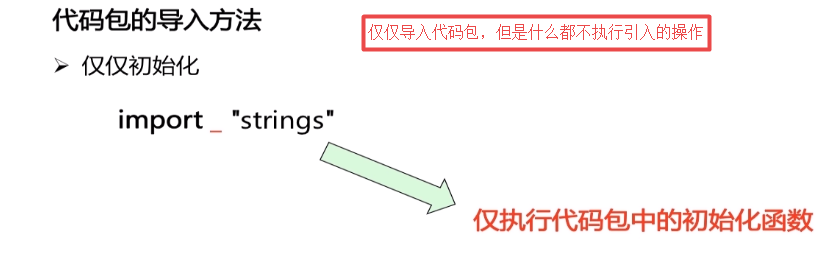
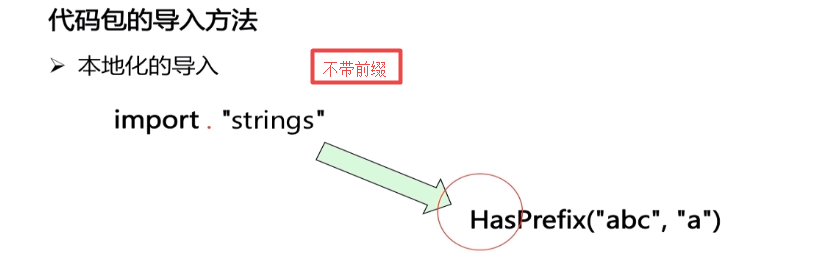
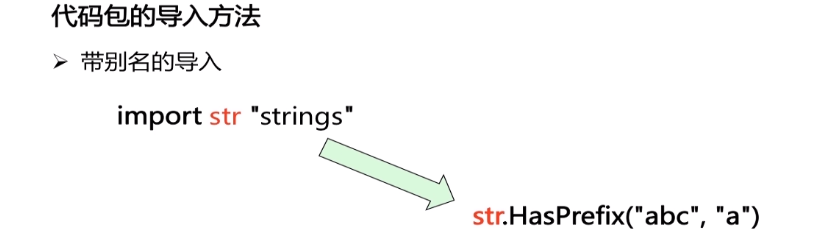
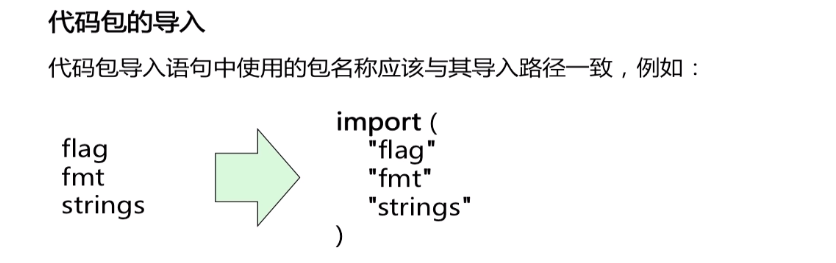
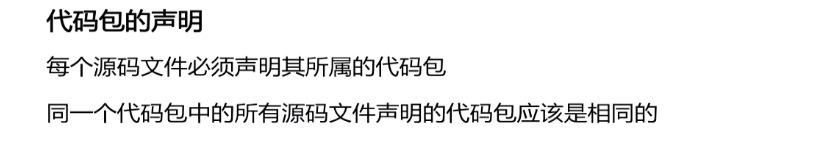
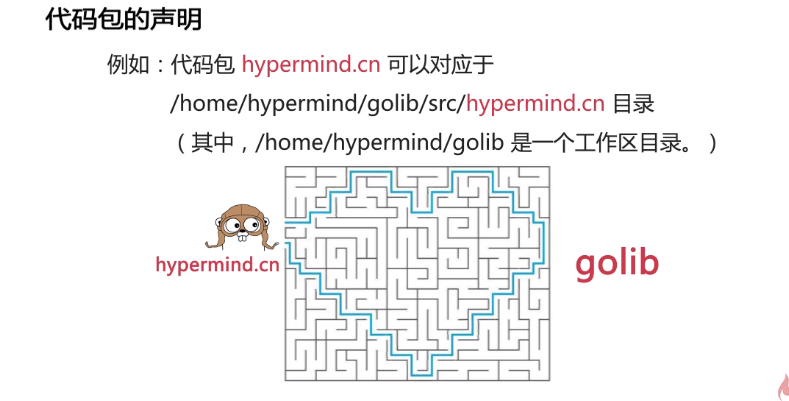
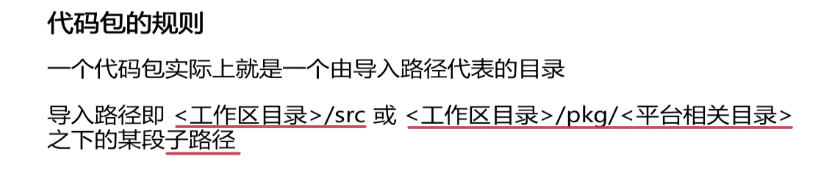
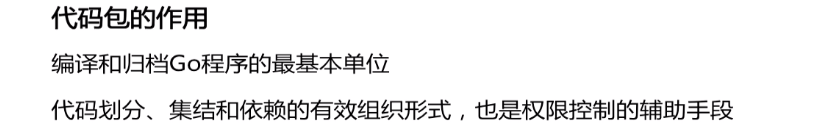
### 源码文件的分类和含义

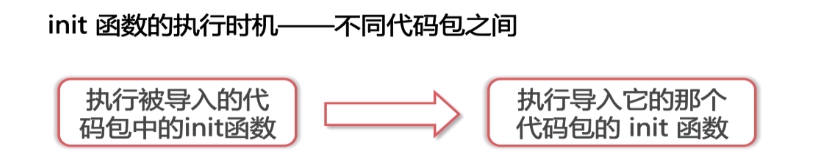
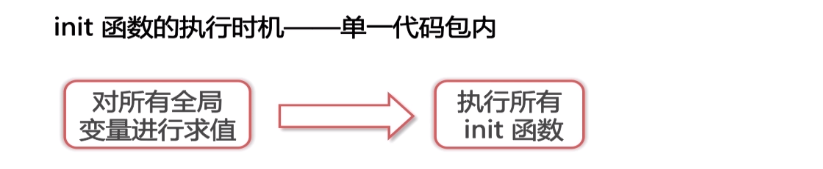
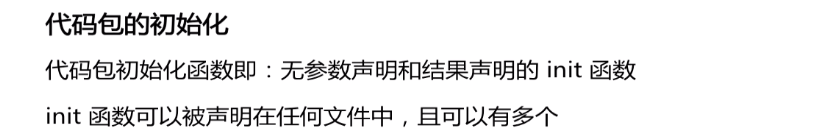




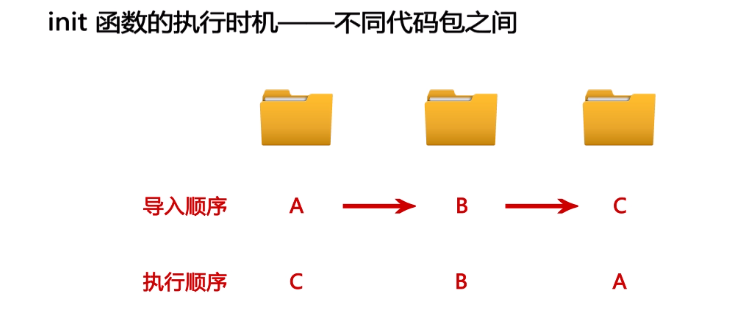


### 代码包相关知识

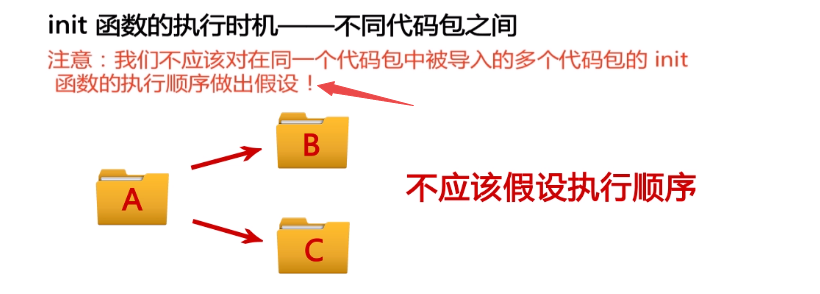




导入顺序

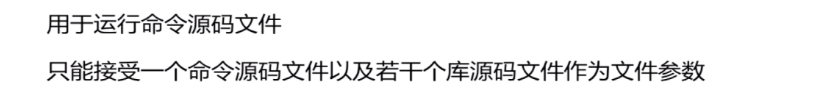


每一个包只会被执行调用一次，不管依赖是怎么走的



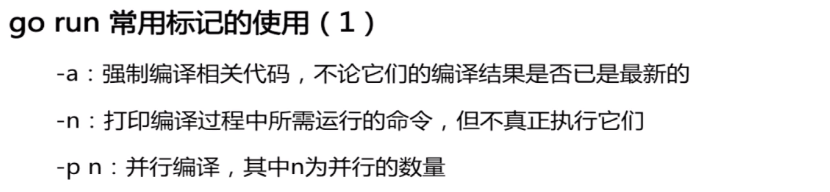
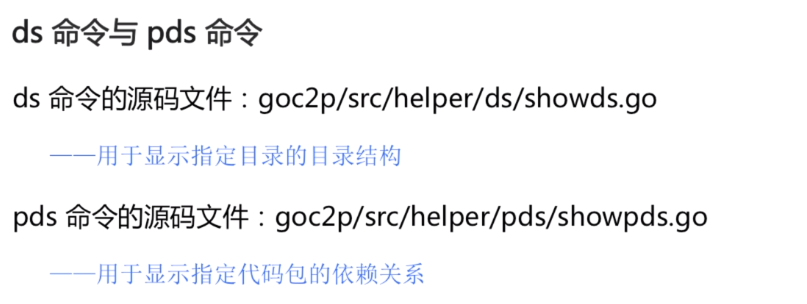
## 命令基础

### Go run

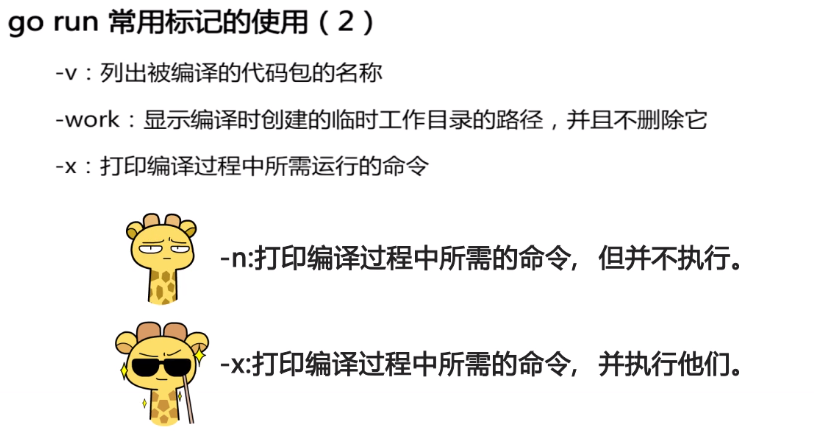
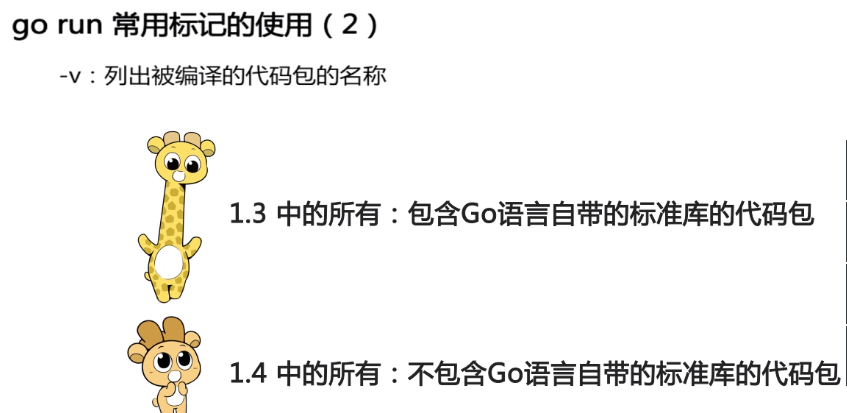


### ds命令与pds命令

是属于linux命令，这个效果可以通过运行showds.go来模拟运行

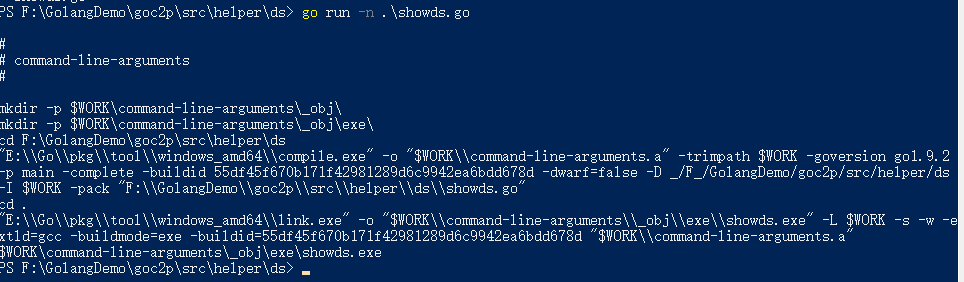


可以使用 go run -av

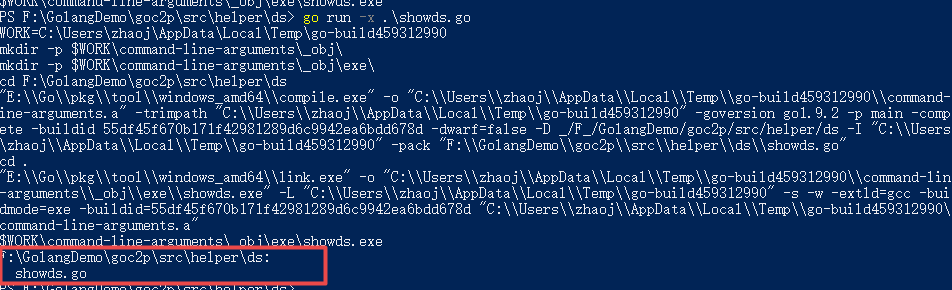


例子演示:

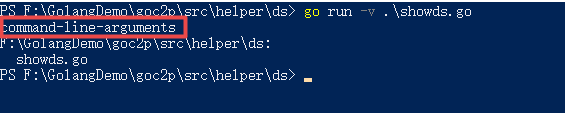
Go run -n ：只会打印将会执行的内容，但是不会执行程序，所以不会输出结果



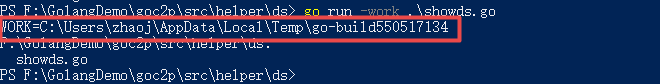
Go run -x ：会打印将会执行的内容，会执行程序，所以会输出结果



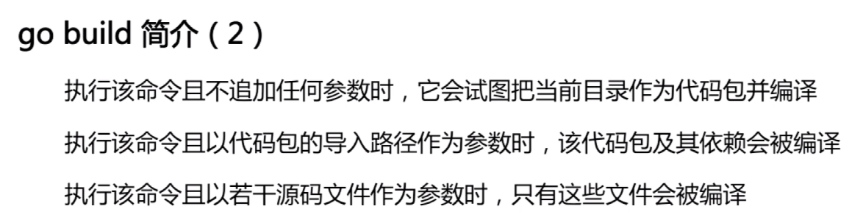
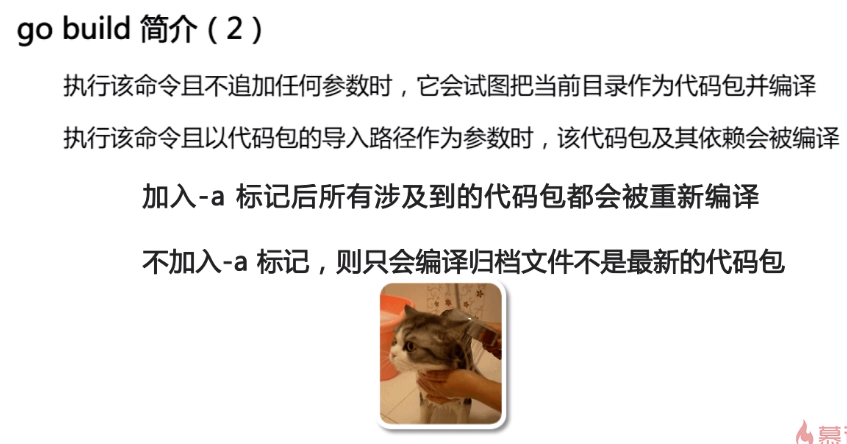
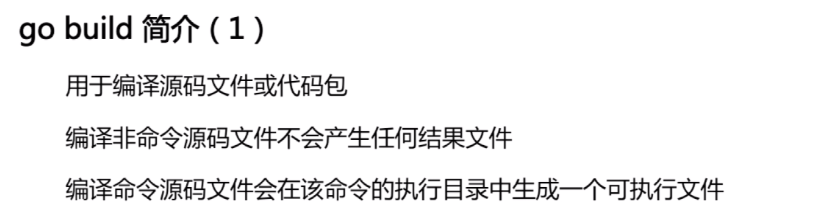
Go run -v:会打印临时运行的内容



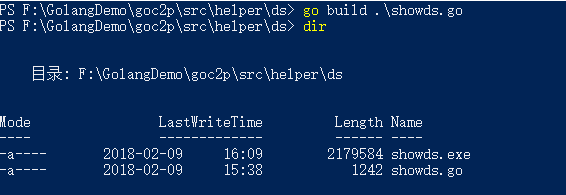
Go run -work ：打印临时执行的工作路径，并不会删除这个临时文件路径,便于查询路径的编译过程



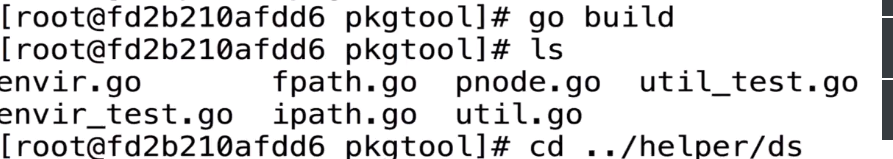
### Go build 和 Go install

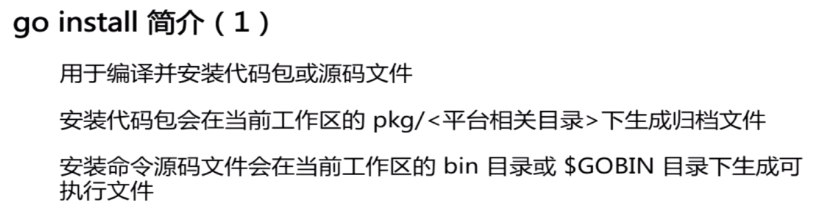


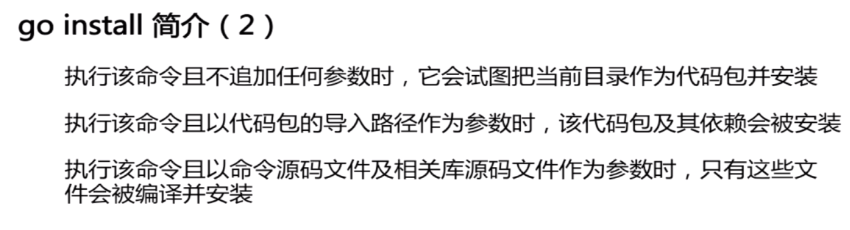
例子实例:



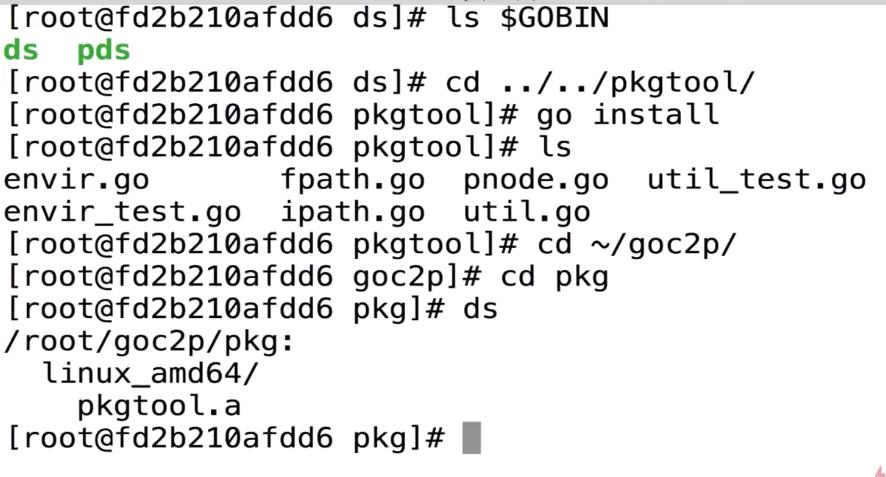
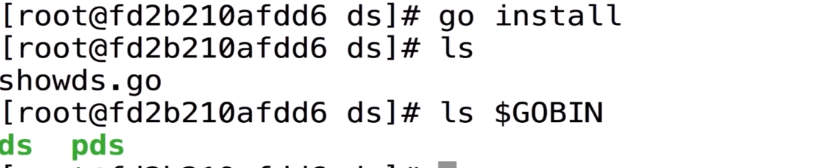
Go build如果不包含main方法就不执行编译



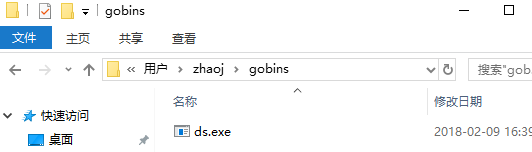




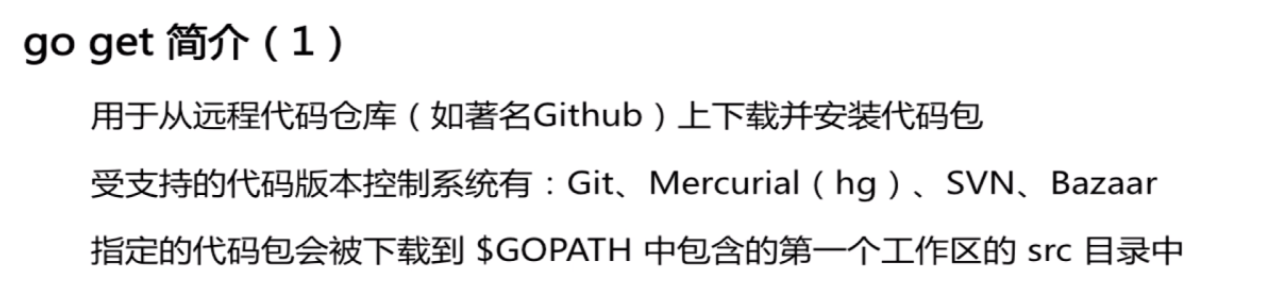
例子实例:



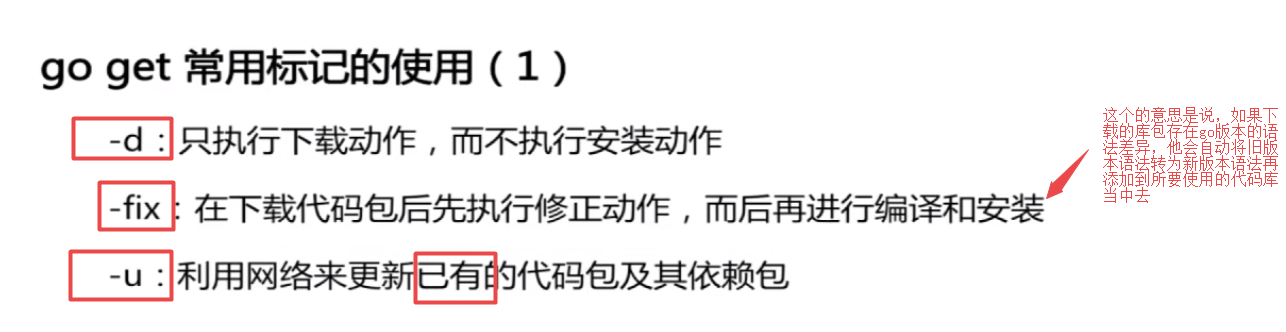
gobin需要自己定义



### Go get



Go get命令会将下载下来的库的内容添加到gopath路径下的第一个路径的src文件夹中，会自动忽略后面的安装添加，所以go get安装内容记得修改gopath路径.



-u 会自动更新本地代码包，会将当前的代码包与网络的代码包进行比对

例子实例:

-d 会下载到src，但不会输出到pkg文件夹中

