今日总结 2019-03-20

每日一学

问题：

```
chan int
chan <- int
<- chan int
```

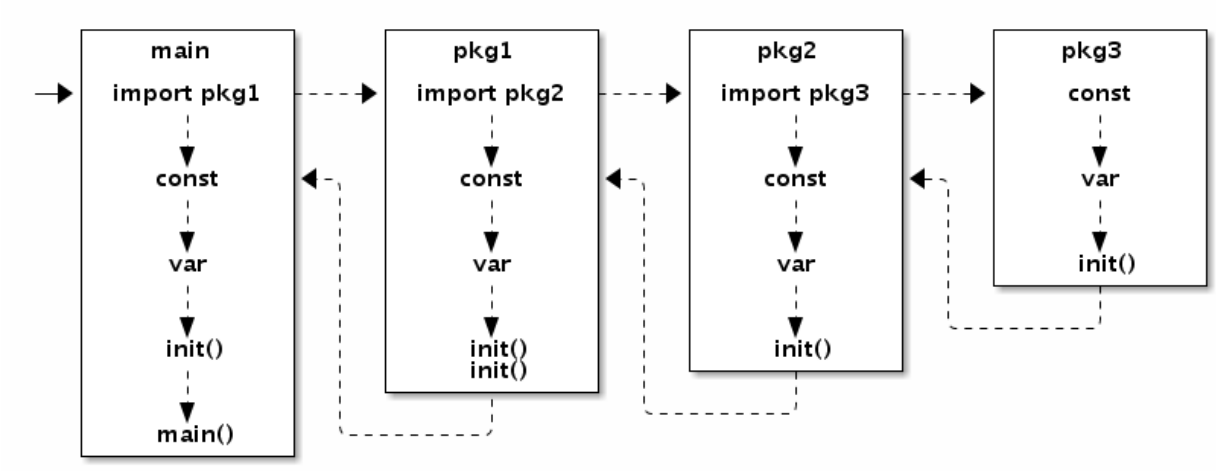
以上三者有什么区别？下面两种用于什么场景？它们之间能相互赋值吗？

讨论结果：

1. `chan int`可读可写(双向), `chan <- int`只写(单向), `<- chan int`只读(单向);
2. 双向chan可以转成单向chan，单向chan不能够转成双向chan;
3. 应用场景 ......

知识点学习

1. 使用数组和切片时，len 函数返回的是 int 类型，如果没有其他限制，理论上最大长度不能超过 int，否则溢出。所以，32 位机器上，最长不能超过 1<<31 - 1，64 位机器就是 1<<63 - 1。当然，一般场景不会用到这么大数组。
2. 空结构不占“空间”，所以，经常会有：chan struct{} 这种定义。
3. Go 包初始化流程：



常见坑

问题：如下代码输出什么： 应该如何改进?

```
func main() {
    for i := 0; i < 3; i++ {
        defer func(){ println(i) } ()
    }
}
```

讨论结果：

1. 输出：3 3 3；
2. `defer func(){ println(i) } ()` 应改为： `defer func(i int){ println(i) } (i)`；
3. 原理： i 的内存地址一直是一样的，在匿名函数三次打印之前，已经被for循环赋值为3，所以输出 3 3 3；
4. 类似坑：

```
func main(){
    for i:=0;i<10;i++{
        go func(){
            fmt.println{i}
        }()
    }
}

//  正确应为：
func main(){
    for i:=0;i<10;i++{
        go func(i int){
            fmt.println{i}
        }(i)
    }
}
```

面试题

问题一：

以下代码是否有问题？为什么?

```
package main

var a string
var done bool

func setup() {
    a = "hello, world"
```

```
    done = true
}
func main() {
    go setup()
    for !done {
    }
    print(a)
}
```

讨论结果：

1. 以上代码能够打印出 “hello world”
2. go的并发是非抢占的，其他协程想运行，首先要有协程放弃运行；
3. 当指定CPU数量为 1 时(main函数的第一行加上`runtime.GOMAXPROCS(1)`)，会一直for循环下去，否则done的值依然会被 go程 改变；

问题二：

如何控制并发执行的 Goroutine 的最大数目?

讨论结果：

1. demo01：

```
package main

import (
    "fmt"
    "time"
)

var ch chan int

func test(i int) {
    fmt.Println(i)
    time.Sleep(1 * 1e9)
    <-ch
}

func main() {
    ch = make(chan int, 10)
    for i:=0; i<1000; i++ {
        ch<-i
        go test(i)
    }
}
```

2. demo02:

```
type pool struct {
    maxNum int // 最大Goroutine 数目
    taskChan chan *Task // 接收并传递任务的通道
}
func (pool)work(){
    for range taskChan {
        Task() // 这里执行任务
    }
}
func (pool)run(){
    for i:=0;i<pool.maxNum;i++{
        go pool.work()  // 这里只启动maxNum个go程
    }
}
```

同学问的问题

问题一：

去除大数据文件的重复行

解决思路：

1. 求出每行数据的hash，存入map的key中；每得到一行数据的hash，利用map判断该key是否有值；有则过滤，无则添加到map中。