**日志模块计数**

1. **简述**

**应用场景：当日志数量太多时，检查自己log会很乱，增加模块计数功能，新修改的模块使用日志时采用修改后的接口**

**功能：为方便查找自己的log，设置专属模块名，为检查自己的log是否丢失，增加计数功能。最终产生的log会携带模块名和计数**

**例如：**

**原始：**

**INFO [08-07|17:13:47] info1 ans=1**

**WARN [08-07|17:13:47] info2 ans=2**

**INFO [08-07|17:13:47] info3 ans=3**

**WARN [08-07|17:13:47] info4 ans=4**

**现在：**

**INFO [08-07|17:13:47] [刘德华][0]:info1 ans=1**

**WARN [08-07|17:13:47] [周杰伦][0]:info2 ans=2**

**INFO [08-07|17:13:47] [刘德华][1]:info3 ans=3**

**WARN [08-07|17:13:47] [周杰伦][1]:info4 ans=4**

1. **模块设计**

## 2.1原始log机制

函数描述：打印log

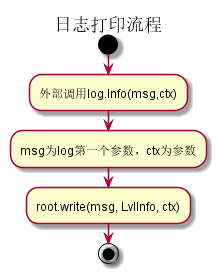
函数信息：

函数名：func Info(msg string, ctx ...interface{}) {}

函数输入：msg 与ctx是参数列表（key，value）

函数输出：打印log

详细设计：



## 2.2 字符串拼接函数

函数描述：拼接模块名+Num+Msg

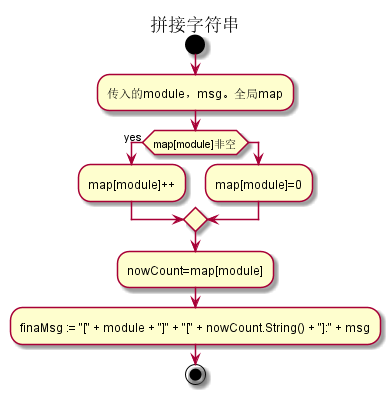
函数信息：

函数名：func TransCount(module string, msg string) string {}

函数输入：模块名,msg

函数输出：[模块名][Num]:msg

详细设计：



2.3 改进后的log机制

函数描述：增加模块名称和模块计数功能

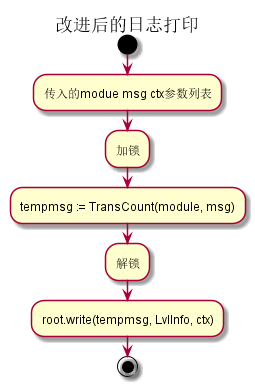
函数信息：

函数名：func INFO(module string, msg string, ctx ...interface{}) {}

函数输入：module模块名 msg第一个参数 ctx参数列表

函数输出：打印日志

详细设计:



## 2.1 可能存在问题：

**使用全局变量map[string]\*big.Int:**

目前做法：为了避免改动log底层，在原来log.Info前面重写了一个函数提前做拼接字符串的工作，为了方便计数且原函数并不是依赖于某个结构体，所以暂时用全局的map表示

存在问题：全局变量的安全性问题

**使用互斥锁：**

目前做法：因为每次调用log模块都要频繁写入（++）和读取（拼接字符串）,所以，需要对map加锁

存在问题：加锁对性能的影响程度？是否可以用

1. **单元测试**