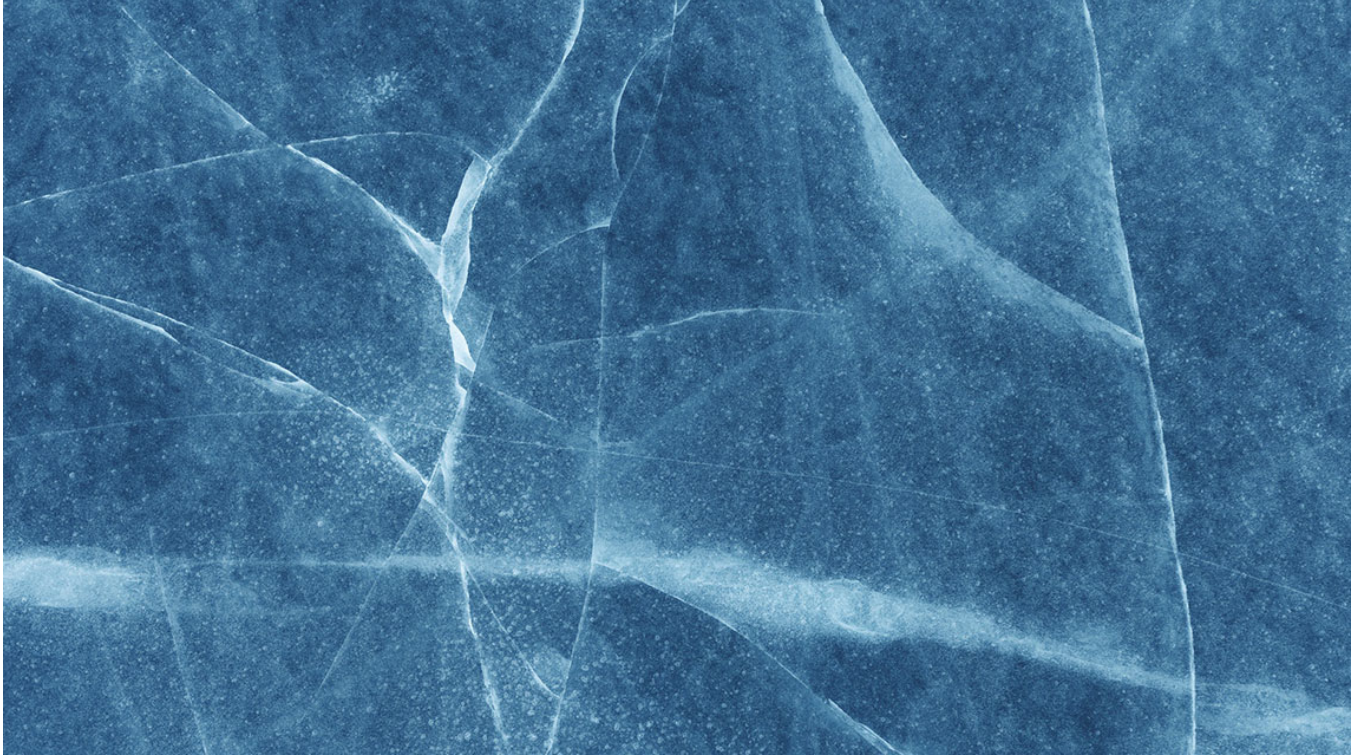


02 | 命令源码文件

2018-08-13 郝林



02 | 命令源码文件

朗读人：黄洲君
08'45" | 8.02M

0:00 / 8:45

【Go 语言代码较多，建议配合文章收听音频。】

我们已经知道，环境变量 `GOPATH` 指向的是一个或多个工作区，而每个工作区中都会有以代码包为基本组织形式的源码文件。

这里的源码文件又分为三种，即：命令源码文件、库源码文件和测试源码文件，它们都有着不同的用途和编写规则。

一旦开始学习用编程语言编写程序，我们就一定希望在编码的过程中及时地得到反馈，只有这样才能知道对错。实际上，我们的有效学习和进步都是通过不断地接受反馈和执行修正实现的。


对于 Go 语言，你在学习阶段一定会经常编写可以直接运行的程序。直至今日，我如果要做一些实验的话，依然会使用这种方法。这样肯定会涉及命令源码文件的编写，而且，命令源码文件可以很方便地用 `go run` 命令启动。

那么，我今天的问题就是：命令源码文件的用途是什么，怎样编写它？

这里，我给出你一个参考的回答：命令源码文件是程序的运行入口，是每个可独立运行的程序必须拥有的。我们可以通过构建或安装生成与其对应的可执行文件，后者一般会与该命令源码文件的直接父目录同名。

如果一个源码文件声明属于`main`包，并且包含一个无参数声明且无结果声明的`main`函数，那么就是命令源码文件。就像这样：

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     fmt.Println("Hello, world!")
7 }
```

 复制代码

如果你把这段代码存成 `demo1.go` 文件，那么运行`go run demo1.go`命令后就会在屏幕（标准输出）中看到`Hello, world!`

当需要模块化编程时，我们往往会将代码拆分到多个文件，甚至拆分到不同的代码包中。但无论怎样，对于一个独立的程序来说，命令源码文件永远只会也只能有一个。如果有与命令源码文件同包的源码文件，那么它们也应该声明属于`main`包。

问题解析

命令源码文件如此重要，以至于它毫无疑问地成为了我们学习 Go 语言的第一助手。不过，只会打印`Hello, world`是远远不够的，咱们千万不要成为“Hello, world”党。既然决定学习 Go 语言，你就应该从每一个知识点深入下去。

无论是 Linux 还是 Windows，如果你用过命令行（command line）的话，肯定就会知道几乎所有命令（command）都是可以接收参数（argument）的。通过构建或安装命令源码文件生成的可执行文件就可以被视为“命令”，既然是命令，那么就应该具备接收参数的能力。

下面，我就带你深入了解一下与命令参数的接收和解析有关的一系列问题。

知识精讲

1. 命令源码文件怎样接收参数

先看一段不完整的代码：

```
1 package main
2
```

 复制代码

pdf 由 我爱学 it (www.52studyit.com) 收集并免费发布

```
3 import (  
4     // 需在此处添加代码。[1]  
5     "fmt"  
6 )  
7  
8 var name string  
9  
10 func init() {  
11     // 需在此处添加代码。[2]  
12 }  
13  
14 func main() {  
15     // 需在此处添加代码。[3]  
16     fmt.Printf("Hello, %s!\n", name)  
17 }  
18
```

如果我想让你帮我在注释处添加相应的代码，并让程序实现“根据运行程序时给定的参数问候某人”的功能，你打算怎样做？


如果你知道做法，请现在就动手实现它。如果不知道也不要着急，咱们一起来搞定。

首先，Go 语言标准库中有一个代码包专门用于接收和解析命令参数。这个代码包的名字叫 `flag`。

我之前说过，如果想要在代码中使用某个包中的程序实体，那么应该先导入这个包。因此，我们需要在[1]处添加代码“`flag`”。注意，这里应该在代码包导入路径的前后加上英文半角的引号。如此一来，上述代码导入了`flag`和`fmt`这两个包。

其次，人名肯定是由字符串代表的。所以我们要在[2]处添加调用`flag`包的`StringVar`函数的代码。就像这样：

```
1 flag.StringVar(&name, "name", "everyone", "The greeting object.")
```

 复制代码

函数`flag.StringVar`接受 4 个参数。第 1 个参数是用于存储该命令参数的值的地址，具体到这里就是在前面声明的变量`name`的地址了，由表达式`&name`表示。


第 2 个参数是为了指定该命令参数的名称，这里是`name`。第 3 个参数是为了指定在未追加该命令参数时的默认值，这里是`everyone`。

至于第 4 个函数参数，即是该命令参数的简短说明了，这在打印命令说明时会用到。

顺便说一下，还有一个与`flag.StringVar`函数类似的函数，叫`flag.String`。这两个函数的区别是，后者会直接返回一个已经分配好的用于存储命令参数值的地址。如果使用它的话，我


们就需要把

```
1 var name string
```

 复制代码

改为

```
1 var name = flag.String("name", "everyone", "The greeting object.")
```

 复制代码

所以，如果我们使用`flag.String`函数就需要改动原有的代码。这样并不符合上述问题的要求。

再说最后一个填空。我们需要在[3]处添加代码`flag.Parse()`。函数`flag.Parse`用于真正解析命令参数，并把它们的值赋给相应的变量。


对该函数的调用必须在所有命令参数存储载体的声明（这里是对变量`name`的声明）和设置（这里是在[2]处对`flag.StringVar`函数的调用）之后，并且在读取任何命令参数值之前进行。

正因为如此，我们最好把`flag.Parse()`放在`main`函数的函数体的第一行。

2. 怎样在运行命令源码文件的时候传入参数，又怎样查看参数的使用说明


如果我们把上述代码存成名为 `demo2.go` 的文件，那么运行如下命令就可以为参数`name`传值：

```
1 go run demo2.go -name="Robert"
2
```

 复制代码


运行后，打印到标准输出（`stdout`）的内容会是：

```
1 Hello, Robert!
```

 复制代码


另外，如果想查看该命令源码文件的参数说明，可以这样做：

```
1 $ go run demo2.go --help
```

 复制代码


其中的`$`表示我们是在命令提示符后运行`go run`命令的。运行后输出的内容会类似：

```
1 Usage of /var/folders/ts/7lg_tl_x2gd_k1lm5g_48c7w0000gn/T/go-build155438482/b001/exe/demo2
2   -name string
3       The greeting object. (default "everyone")
4 exit status 2
```

 复制代码

你可能不明白下面这段输出代码的意思。


```
1 /var/folders/ts/7lg_tl_x2gd_k1lm5g_48c7w0000gn/T/go-build155438482/b001/exe/demo2
```

 复制代码

这其实是`go run`命令构建上述命令源码文件时临时生成的可执行文件的完整路径。


如果我们先构建这个命令源码文件再运行生成的可执行文件，像这样：

```
1 $ go build demo2.go
2 $ ./demo2 --help
```

 复制代码

那么输出就会是

```
1 Usage of ./demo2:
2   -name string
3       The greeting object. (default "everyone")
```

 复制代码

3. 怎样自定义命令源码文件的参数使用说明


这有很多种方式，最简单的一种方式就是对变量`flag.Usage`重新赋值。`flag.Usage`的类型是`func()`，即一种无参数声明且无结果声明的函数类型。

`flag.Usage`变量在声明时就已经被赋值了，所以我们才能够在运行命令`go run demo2.go --help`时看到正确的结果。

注意，对`flag.Usage`的赋值必须在调用`flag.Parse`函数之前。

现在，我们把 `demo2.go` 另存为 `demo3.go`，然后在`main`函数体的开始处加入如下代码。

```
1 flag.Usage = func() {
2   fmt.Fprintf(os.Stderr, "Usage of %s:\n", "question")
3   flag.PrintDefaults()
4 }
```

 复制代码

那么当运行

```
1 $ go run demo3.go --help
```

 复制代码

后，就会看到


```
1 Usage of question:
2   -name string
3     The greeting object. (default "everyone")
4 exit status 2
```

[复制代码](#)

现在再深入一层，我们在调用flag包中的一些函数（比如StringVar、Parse等等）的时候，实际上是在调用flag.CommandLine变量的对应方法。

flag.CommandLine相当于默认情况下的命令参数容器。所以，通过对flag.CommandLine重新赋值，我们可以更深层次地定制当前命令源码文件的参数使用说明。

现在我们把main函数体中的那条对flag.Usage变量的赋值语句注销掉，然后在init函数体的开始处添加如下代码：

```
1 flag.CommandLine = flag.NewFlagSet("", flag.ExitOnError)
2 flag.CommandLine.Usage = func() {
3     fmt.Fprintf(os.Stderr, "Usage of %s:\n", "question")
4     flag.PrintDefaults()
5 }
```

[复制代码](#)

再运行命令go run demo3.go --help后，其输出会与上一次的输出的一致。不过后面这种定制的方法更加灵活。比如，当我们把为flag.CommandLine赋值的那条语句改为

```
1 flag.CommandLine = flag.NewFlagSet("", flag.PanicOnError)
```

[复制代码](#)

后，再运行go run demo3.go --help命令就会产生另一种输出效果。这是由于我们在这里传给flag.NewFlagSet函数的第二个参数值是flag.PanicOnError。

flag.PanicOnError和flag.ExitOnError都是预定义在flag包中的常量。

flag.ExitOnError的含义是，告诉命令参数容器，当命令后跟--help或者参数设置的不正确的时候，在打印命令参数使用说明后以状态码2结束当前程序。

状态码2代表用户错误地使用了命令，而flag.PanicOnError与之的区别是在最后抛出“运行时恐慌（panic）”。

上述两种情况都会在我们调用flag.Parse函数时被触发。顺便提一句，“运行时恐慌”是Go程序错误处理方面的概念。关于它的抛出和恢复方法，我在本专栏的后续部分中会讲到。

下面再进一步，我们索性不用全局的flag.CommandLine变量，转而自己创建一个私有的命令参数容器。我们在函数外再添加一个变量声明：

然后，我们把对`flag.StringVar`的调用替换为对`cmdLine.StringVar`调用，再把`flag.Parse()`替换为`cmdLine.Parse(os.Args[1:])`。

其中的`os.Args[1:]`指的就是我们给定的那些命令参数。这样做就完全脱离了`flag.CommandLine`。`*flag.FlagSet`类型的变量`cmdLine`拥有很多有意思的方法。你可以去探索一下。我就不在这里一一讲述了。

这样做的好处依然是更灵活地定制命令参数容器。但更重要的是，你的定制完全不会影响到那个全局变量`flag.CommandLine`。

总结

恭喜你！你现在已经走出了 Go 语言编程的第一步。你可以用 Go 编写命令，并可以让它们像众多操作系统命令那样被使用，甚至可以把它们嵌入到各种脚本中。

虽然我为你讲解了命令源码文件的基本编写方法，并且也谈到了为了让它接受参数而需要做的各种准备工作，但这并不是全部。

别担心，我在后面会经常提到它的。另外，如果你想详细了解`flag`包的用法，可以到[这个网址](#)查看文档。或者直接使用`godoc`命令在本地启动一个 Go 语言文档服务器。怎样使用`godoc`命令？你可以参看[这里](#)。

思考题

我们已经见识过为命令源码文件传入字符串类型的参数值的方法，那还可以传入别的吗？这就是今天我留下的思考题。

1. 默认情况下，我们可以让命令源码文件接受哪些类型的参数值？
2. 我们可以把自定义的数据类型作为参数值的类型吗？如果可以，怎样做？

你可以通过查阅文档获得第一个问题的答案。记住，快速查看和理解文档是一项必备的技能。

至于第二个问题，你回答起来可能会有些困难，因为这涉及了另一个问题：“怎样声明自己的数据类型？”这个问题我在专栏的后续部分中也会讲到。如果是这样，我希望你记下它和这里说的另一问题，并在能解决后者之后再回来回答前者。

[戳此查看 Go 语言专栏文章配套详细代码。](#)

GO语言核心36讲

3个月带你通关 GO 语言

郝林

《Go 并发编程实战》作者
GoHackers 技术社群发起人
前轻松筹大数据负责人



版权归极客邦科技所有，未经许可不得转载

写留言

精选留言



咖啡色的羊驼

看完本文，记住的两点：

1. 源码文件分为三种：命令、库、测试。
2. 编写命令源码文件的关键包：flag。

回答下问题：

1. 命令源码文件支持的参数：

int(int|int64|uint|uint64),

float(float|float64)

string,

bool,

duration(时间),

var(自定义)

2. 关键就是使用flag.var()，关键在于需要实现flag包的Value接口。

2018-08-13



Dragoonium

我试着把参数增加到两个，然后试试运行结果

```
func init() {
```

```
    flag.StringVar(&name, "name1", "ladies", "The greeting object 1")
```

```
    flag.StringVar(&name, "name2", "gentlemen", "The greeting object 2")
```

```
}
```

pdf 由 我爱学it (www.52studyit.com) 收集并免费发布

46

23


```
# go run test.go
```

Hello gentlemen!

和想像的一样，name2的默认值覆盖了name1的默认值

```
# go run test.go -name1=Robert
```

Hello Robert!

和想像的略有不同，只指定了name1，没有指定name2，输出了name1的指定值，name2的默认值没有生效

```
# go run test.go -name2=Jose
```

Hello Jose!

没毛病

```
# go run test.go -name1=Robert -name2=Jose
```

Hello Jose!

没毛病

```
# go run test.go -name2=Jose -name1=Robert
```

Hello Robert!

这有点奇怪了，输出的值是以参数的先后顺序为准的，而不是以flag.StringVar函数的顺序为准的

2018-08-13



阿杜

👍 8

喜欢这种重实践和编码的风格，便于上手

2018-08-13



Abirdcfly

👍 7

解答一下Dragoonium同学的疑惑，在flag包的文档里第一个example里就有你提到的这种情况，注释已经说明白了。

我不太精确的翻译一下：

两个flag分享一个变量时可以用来做命令参数缩写，由于初始化顺序未定义，所以要确保两者使用同一默认值，并且他们必须在init()函数中设置

2018-08-14



云学

👍 4

init在main之前执行，go程序的执行顺序是否可以讲下

2018-08-13



Tron

👍 3

go语言ide还是推荐goland

2018-09-05



吉祥

👍 3



undefined: os 怎么回事

2018-08-13

| 作者回复

你好，请到GitHub上下载完整的源码文件。

2018-08-15



wjq310

👍 2

demo3之后，要import os

2018-08-15



梦里追逐

👍 2

咱们用的都是哪个IDE？

2018-08-15

| 作者回复

你好，我用的是goland，但是代码不会依赖于IDE的，只会依赖于Go语言本身。免费的编辑器推荐vs code。

2018-08-15



成都福哥

👍 2

用自定义的cmdLine的时候，usage函数里的flag.PrintDefaults()应该相应的变成cmdLine.PrintDefaults()吧。

2018-08-15



芒果

👍 2

关于变量以标准输入为准的问题，我个人认为init中的定义只是定义了解析规则，真正执行解析是flag.Parse()时开始，因此以标准输出为准。想想我们自己写的时候会怎么实现，先获得输入如：-name1=a，然后解析为key=name1和value=a，然后走一个if，else判断，如果key匹配则对其赋值。所以就很好解释了。个人感觉自己的理解还是比较靠谱的，虽然没有研究源码。欢迎大神们交流

2018-08-14



松烽

👍 2

自定义参数，还可以自己通过字符串转对象的方式实现

2018-08-14



飞吧蛐蛐

👍 1

问题1：通过flag库的提示，或者看flag包的用法，参数支持Bool/Duration/Float64/Int/Int64/Uint/Uint64，也支持Float32，猜测考虑到精度问题，flag没有支持float32。

问题2：参数值的类型可以是自定义的数据类型，使用实现flag包里的Value接口，然后使用flag.Var()实现。（flag源码里有提示，Value is the interface to the dynamic value stored in a flag）

PS：看一下flag包的用法和源码就能解答这两个问题了。

2018-10-12



丸子说

👍 1

从flag.stringvar/flag.string到flag.commandline再到私有cmdline命令参数容器，循序渐进，由浅到深。

pdf由 我爱学it (www.52studyit.com) 收集并免费发布

2018-08-14



世风十三

👍 1

flag.Usage 那部分有个 os 变量是 undefined 啊？

2018-08-13



茶底

👍 1

1.string int bool

2.使用flag.Var().重点在于实现Value接口。

2018-08-13



五流诗人二手黑客

👍 1

之前练习go时候写了一个将预定格式xml转换成sql的方法，感觉这个语言写起来很有趣

2018-08-13



alan

👍 1

感谢老师。感觉编码细节有些偏多了哈，希望多一些总结性和主观的内容。

2018-08-13



objcoding

👍 1

命令源码对应的包也是通过go install唯一能够生成可执行文件的包

2018-08-13



々雪虎々_卍

👍 0

学习语言最重要的是动手实践，稍微修改了一下代码做了一个实验，发现方式2和方式三可以结合使用会比较方便，而且不用导入 os 模块：

```
package main
```

```
import (  
    "flag"  
    "fmt"  
)
```

```
var name string
```

```
func init() {  
    flag.CommandLine = flag.NewFlagSet("question 2-2", flag.ExitOnError)  
    flag.StringVar(&name, "name", "everyone", "The greeting object.")  
}
```

```
func main() {  
    flag.Parse()  
    fmt.Printf("Hello, %s!\n", name)  
}
```

2018-09-27



Geek_091a67

👍 0

为什么会有这样的错误啊？

cannot use &name (type **string) as type *string in argument to flag.StringVar

2018-09-18



属鱼

0

给VS Code的用户一个脚本，用于安装VS Code的Go扩展所需要的。

VS code的go插件需要的扩展组件有的需要翻墙，安装不了，

Linux

https://github.com/cloudfstrife/note/blob/develop/golang/install_vs_code_golang_ext_tools.sh

windows

https://github.com/cloudfstrife/note/blob/develop/golang/install_vs_code_golang_ext_tools.ps1

这个是powershell脚本，在win7 powershell 5测试通过。

执行powershell 需要 修改策略，详见：

<http://www.cnblogs.com/Jim-william/p/5492507.html>

注意shell中的说明，GOPATH目前只支持一个目录。

2018-09-12



jv

0

按照demo2.go写了程序，但输出结果不是文章介绍的，而是一直是 hello , everyone ! , 是不是我运行的命令有问题？ - name= "Robert" , -和name间有个空格，其他的没有，我试了很多遍，结果还是不对

2018-09-11



manky

0

新建的文件需要放在GOPATH目录下吗，如果不需要，那什么时候需要新建项目在GOPATH目录下，期待作者的解答。

2018-09-06



Neo

0

感谢老师，要是代码有语法高亮就更好了

2018-09-01



郁柏

0

能否分享一个vscode +go的搭建和使用教程

2018-08-30

作者回复

网上到处都是啊。

2018-09-01



郁柏

0

go get 下载源码包是否需要翻墙。我下载一个源码包请求超时。

Fetching <https://golang.org/x/tools/imports?go-get=1>

https fetch failed: Get https://golang.org/x/tools/imports?go-get=1: dial tcp 216.239.37.1:443: i/o timeout
package golang.org/x/tools/imports: unrecognized import path "golang.org/x/tools/imports" (https fetch: Get https://golang.org/x/tools/imports?go-get=1: dial tcp 216.239.37.1:443: i/o timeout)
2018-08-30

作者回复

可以手动下载到对应位置。

2018-09-01



郁柏

0

执行 go get -v github.com/sqs/goreturns 失败。

请求连接超时。 请问怎么办。

2018-08-30

作者回复

可以去github上手动下载，然后存储到指定位置。最好有git clone命令下载。

2018-09-01



mayunian

0

老师，今天试了一下类型转换。

为什么转换var x uint = uint(-1) 的时候会报错？

而var y int = -1

var x uint = uint(y)就不会报错呢？

2018-08-30

作者回复

-1是负数，编译器看出来，帮你挑出来。y是int类型的变量，编译器不知道里面存的是不是负数，没法帮你挑出来。转换会成功结果会不正确。

2018-09-01



金阳

0

定义了一个name string类型变量，在init函数写

name = flag.string("name" , " jobs" , " des")

为啥报错了？

2018-08-16

作者回复

你需要给出完整代码别人才能帮你，另外报错信息也很重要。

2018-08-17



金阳

0

想请教一下老师，这种go 语言自带的api哪里可以查看

2018-08-16

作者回复

godoc.org

2018-08-17



AlittleChange

0

pdf由 我爱学it(www.52studyit.com) 收集并免费发布



大家都已经实践了吗？

2018-08-13



Dragoonium

0

我试着把参数增加到两个，然后试试运行结果

```
func init() {
    flag.StringVar(&name, "name1", "ladies", "The greeting object 1")
    flag.StringVar(&name, "name2", "gentlemen", "The greeting object 2")
}
```

```
# go run test.go
```

Hello gentlemen!

和想像的一样，name2的默认值覆盖了name1的默认值

```
# go run test.go -name1=Robert
```

Hello Robert!

和想像的略有不同，只指定了name1，没有指定name2，输出了name1的指定值，name2的默认值没有生效

```
# go run test.go -name2=Jose
```

Hello Jose!

没毛病

```
# go run test.go -name1=Robert -name2=Jose
```

Hello Jose!

没毛病

```
# go run test.go -name2=Jose -name1=Robert
```

Hello Robert!

这有点奇怪了，输出的值是以参数的先后顺序为准的，而不是以flag.StringVar函数的顺序为准的

2018-08-13



@XP

0

回答问题

1、string bool int这三种基本类型

2、使用func Set(myset,value string) error 等定义一个类型，var myflag myset定义变量，flag.Var(&myflag, 'default value' , 'doc')

不知道理解的对不对

2018-08-13



ylck

0

很详细。

2018-08-13

