Go语言学习(八)获取命令行参数

原创 2016年03月27日 12:52:16

标签：

[go语言](http://so.csdn.net/so/search/s.do?q=go%E8%AF%AD%E8%A8%80&t=blog" \t "http://blog.csdn.net/mchenys/article/details/_blank)

6884

部署golang项目时难免要通过命令行来设置一些参数，那么在golang中如何操作命令行参数呢？可以使用flag库和os库。

1.flag库的使用

Go语言标准库提供了用于快迅解析命令行参数的flag包,大致的使用步骤如下:

a.通过flag.String(), flag.Bool(), flag.Int()等方式来定义命令行中需要使用的参数。

b.在定义完flag后，通过调用flag.Parse()来进行对命令行参数的解析。

c.获取flag.String(), flag.Bool(), flag.Int()等方法的返回值,即对应用户输入的参数.

需要注意的是flag.Xxx()返回的值是变量的内存地址,要获取值时要通过在变量前加\*(星号)获取.

说明:

像flag.Int、flag.Bool、flag.String这样的函数格式都是一样的，调用的时候需要传入3个参数

参数的说明如下:

第一个arg表示参数名称，在控制台的时候,提供给用户使用.

第二个arg表示默认值，如果用户在控制台没有给该参数赋值的话,就会使用该默认值.

第三个arg表示使用说明和描述,在控制台中输入-arg的时候会显示该说明,类似-help

1

2

3

4

5

6

7

8

9

10

11

12

13

看个完整的使用例子:

package main

import (

"flag"

"fmt"

)

func main() {

married := flag.Bool("married", false, "Are you married?")

age := flag.Int("age", 22, "How old are you?")

name := flag.String("name", "", "What your name?")

var address string

//flag.StringVar这样的函数第一个参数换成了变量地址，后面的参数和flag.String是一样的。

flag.StringVar(&address, "address", "GuangZhou", "Where is your address?")

flag.Parse() //解析输入的参数

fmt.Println("输出的参数married的值是:", \*married)//不加\*号的话,输出的是内存地址

fmt.Println("输出的参数age的值是:", \*age)

fmt.Println("输出的参数name的值是:", \*name)

fmt.Println("输出的参数address的值是:", address)

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

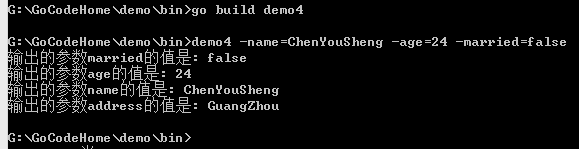
19

20

21

22

23

运行结果:   


从运行结果中可以看出,address参数我们并没有指定值,则输出的就是默认值.

另外,-arg后面的=号也不是必须的,可以用空格代替.

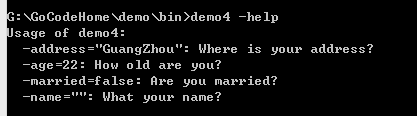
如果要查看该程序的所有参数的使用,可以使用-help来查看,如下:

1

2

3

4



2.os库的使用

直接看例子

package main

import (

"os"

"fmt"

)

func main() {

args := os.Args //获取用户输入的所有参数

if args == nil || len(args) <2{

Usage()//如果用户没有输入,或参数个数不够,则调用该函数提示用户

return

}

name := args[1] //获取输入的第一个参数

age := args[2] //获取输入的第二个参数

fmt.Println("your name is:",name,"\nyour age is:",age)

}var Usage = func() {

fmt.Println("you name?");

fmt.Println("you age?");

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

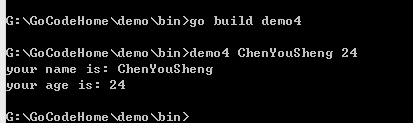
20

21

22

23

24

运行结果:   


个人感觉还是flag获取参数好用一点,因为flag更贴切参数的用法,而且可以查看详情.

1

2

最后补充下如何获取变量的内存地址和接收方式

package main

import (

"fmt"

)

func main() {

var name string = "abc"

var nameAddress \*string = &name; //如果要接收变量的内存地址,变量对应的类型前要加\*号

fmt.Println("name变量的内存地址:",&name)

fmt.Println("nameAddress的值:",nameAddress)

fmt.Println("&name==nameAddress? ",&name == nameAddress)

fmt.Println("nameAddress对应变量的值: ",\*nameAddress)//内存地址前加\*号则可以看到对应的值了.

fmt.Println("name变量的值: ",\*&\*&name)//\*号和&可以多次组合来使用,阅读的时候从右往左

}

1

2

3

4

5

6

7

8

9

10

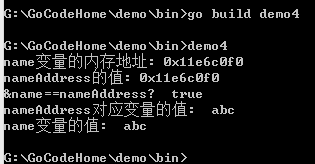
11

12

13

14

15

运行结果:   


[理论]

&符号的意思是对变量取地址，如：变量a的地址是&a

\*符号的意思是对指针取值，如:\*&a，就是a变量所在地址的值，当然也就是a的值了

1

2

[简单的解释]

\*和 & 可以互相抵消,同时注意，\*&可以抵消掉，但&\*是不可以抵消的 a和\*&a是一样的，都是a的值，值为1 (因为\*&互相抵消掉了)

同理，a和\*&\*&\*&\*&a是一样的，都是1 (因为4个\*&互相抵消掉了)

1

2

3

文章来源：http://blog.csdn.net/mchenys/article/details/50991165

版权声明：本文为博主原创文章，未经博主允许不得转载。