Go语言 标准库 Flag包

Go语言内置的 flag 包实现了命令行参数的解析， flag 包使得开发命令行工具更为简单。

os.Args

如果你只是简单的想要获取命令行参数，可以像下面的代码示例一样使用os.Args来获取命令行参数。程序获取运行他时给出的参数，可以通过os包来实现。

```
func argsDemo() {
    //os.Args是一个[]string
    if len(os.Args) > 0 {
        for index, arg := range os.Args {
            fmt.Printf("args[%d]=%v\n", index, arg)
        }
    }
}
```

将上述代码运行

```
PS D:\GoPro> go run "d:\GoPro\flag\flag.go" a b c d
args[0]=C:\Users\包子\AppData\Local\Temp\go-build3629847899\b001\exe\flag.exe
args[1]=a
args[2]=b
args[3]=c
args[4]=d
```

可以看到，命令行参数包括了程序路径本身，以及通常意义上的参数。

程序中os.Args的类型是 **[]string** ，也就是字符串切片。所以可以在for循环的range中遍历，还可以用 **len(os.Args)** 来获取其数量。

os.Args是一个存储命令行参数的字符串切片，它的第一个元素是执行文件的名称。

flag 包

flag包相比单纯的通过os.Args切片分析命令行参数，提供了更强的能力。

flag 参数类型

flag包支持的命令行参数类型有bool、int、int64、uint、uint64、float、float64、string、duration。

flag参数	有效值
字符串flag	合法字符串
整数flag	1234、0664、0x1234等类型，也可以是负数。
浮点数flag	合法浮点数

flag参数	有效值
bool类型flag	1, 0, t, f, T, F, true, false, TRUE, FALSE, True, False。
时间段flag	任何合法的时间段字符串。如”300ms”、”-1.5h”、”2h45m”。 合法的单位有”ns”、”us” /“μs”、”ms”、”s”、”m”、”h”。

定义命令行flag参数

有以下两种常用的定义命令行flag参数的方法。

flag.Type()

格式：

```
flag.Type(flag名, 默认值, 帮助信息)*Type
```

例如我们要定义姓名、年龄、婚否三个命令行参数，我们可以按如下方式定义：

```
name := flag.String("name", "包子", "姓名")
age := flag.Int("age", 18, "年龄")
married := flag.Bool("married", false, "婚否")
delay := flag.Duration("d", 0, "时间间隔")
```

需要注意的是，此时name、age、married、delay均为对应类型的指针。

flag.TypeVar()

基本格式如下：

```
flag.TypeVar(Type指针, flag名, 默认值, 帮助信息)
```

例如我们要定义姓名、年龄、婚否三个命令行参数，我们可以按如下方式定义：

```
var name string
var age int
var married bool
var delay time.Duration
flag.StringVar(&name, "name", "包子", "姓名")
flag.IntVar(&age, "age", 18, "年龄")
flag.BoolVar(&married, "married", false, "婚否")
flag.DurationVar(&delay, "d", 0, "时间间隔")
```

flag.Parse()

通过以上两种方法定义好命令行flag参数后，需要通过调用flag.Parse()来对命令行参数进行解析。

支持的命令行参数格式有以下几种：

- -flag xxx （使用空格，一个-符号）
- --flag xxx （使用空格，两个-符号）
- -flag=xxx （使用等号，一个-符号）
- --flag=xxx （使用等号，两个-符号）

其中，布尔类型的参数必须使用等号的方式指定。

Flag解析在第一个非flag参数（单个”-“不是flag参数）之前停止，或者在终止符”–“之后停止。

flag其他函数

- flag.Args() //返回命令行参数后的其他参数，以[]string类型
- flag.NArg() //返回命令行参数后的其他参数个数
- flag.NFlag() //返回使用的命令行参数个数

完整示例

使用flag包，首先定义待解析命令行参数，也就是以"-"开头的参数，比如这里的 -b -s -help等。-help不需要特别指定，可以自动处理。

参数中没有能够按照预定义的参数解析的部分，通过flag.Args()即可获取，是一个字符串切片。

需要注意的是，从第一个不能解析的参数开始，后面的所有参数都是无法解析的。即使后面的参数中含有预定义的参数

```
package main

import (
    "flag"
    "fmt"
    "os"
    "time"
)

func main() {
    //定义命令行参数方式1
    var name string
    var age int
    var married bool
    var delay time.Duration
    flag.StringVar(&name, "name", "包子", "姓名")
    flag.IntVar(&age, "age", 18, "年龄")
    flag.BoolVar(&married, "married", false, "婚否")
    flag.DurationVar(&delay, "d", 0, "延迟的时间间隔")

    //解析命令行参数
    flag.Parse()
    fmt.Println(name, age, married, delay)
    //返回命令行参数后的其他参数
    fmt.Println(flag.Args())
    //返回命令行参数后的其他参数个数
    fmt.Println(flag.NArg())
    //返回使用的命令行参数个数
    fmt.Println(flag.NFlag())
}
```

运行效果：

```
PS D:\GoPro> go run "d:\GoPro\flag\flag.go" -name baozi -age 28 -married=false -
d=10s
baozi 28 false 10s
[]
0
4
```