1、基本介绍

fmt包实现了类似C语言printf和scanf的格式化I/O。格式化动作（'verb'）源自C语言但更简单。

2、输出

2.1 Print系列

Print系列函数会将内容输出到系统的标准输出，共有三个函数。区别在于`Print函数直接输出内容`，`Printf函数支持格式化输出字符串`，`Println函数会在输出内容的结尾添加一个换行符`。

```
func Print(a ...interface{}) (n int, err error)
func Printf(format string, a ...interface{}) (n int, err error)
func Println(a ...interface{}) (n int, err error)
```

2.2 Fprint系列

Fprint系列函数会`将内容输出到一个io.Writer接口类型的变量w中`，我们通常用这个函数`往文件中写入内容`。只要`满足io.Writer接口的类型都支持写入`。

```
func Fprint(w io.Writer, a ...interface{}) (n int, err error)
func Fprintf(w io.Writer, format string, a ...interface{}) (n int, err error)
func Fprintln(w io.Writer, a ...interface{}) (n int, err error)
```

```
import (
    "fmt"
    "os"
)

func main() {

    // 向标准输出写入内容
    fmt.Fprint(os.Stdout, "Welcome to the OpenStack project") // 直接按照内容输出
    fmt.Fprintln(os.Stdout, "Welcome to the OpenStack")       // 输出后会换行

    // 向文件中写入内容
    fileObj, _ := os.OpenFile("./xxx.txt", os.O_WRONLY|os.O_CREATE, 0644)
    defer fileObj.Close()

    data := "文件输出Fprintf"
    fmt.Fprintf(fileObj, "往文件中写入--> %s", data) // 格式化输出
}
```

2.3 Sprint系列

Sprint系列函数会`把传入的数据生成并返回一个字符串`。

```
func Sprint(a ...interface{}) string    // 直接拼接字符串
func Sprintf(format string, a ...interface{}) string  // 格式化拼接字符串
func Sprintln(a ...interface{}) string  // 在生成的字符串末尾添上换行符
```

2.4 Errorf

Errorf函数根据format参数生成格式化字符串并返回一个包含该字符串的错误。

```
func Errorf(format string, a ...interface{}) error
```

```
    err := fmt.Errorf("这是一个错误")
    fmt.Println(err)

    // Go1.13版本为fmt.Errorf函数新加了一个%w占位符用来生成一个可以包裹Error的Wrapping
Error。
    e := errors.New("原始错误e")
    w := fmt.Errorf("Wrap了一个错误 --> %w", e)
    fmt.Println(w)
```

3、输入

Go语言fmt包下有`fmt.Scan、fmt.Scanf、fmt.Scanln`三个函数，可以在程序运行过程中`从标准输入获取用户的输入`。

3.1 fmt.Scan

- Scan从标准输入扫描文本，读取由空白符分隔的值保存到传递给本函数的参数中，`换行符视为空白符`。
- 本函数返回成功扫描的数据个数和遇到的任何错误。`如果读取的数据个数比提供的参数少，会返回一个错误报告原因`。

```
func Scan(a ...interface{}) (n int, err error)
```

```
func main() {

    var (
        name    string
        age     int
        married bool
    )
    fmt.Scan(&name, &age, &married)
    fmt.Printf("扫描结果 name:%s age:%d married:%t \n", name, age, married)
}
```

3.2 fmt.Scanf

- Scanf从标准输入扫描文本，根据format参数指定的格式去读取由空白符分隔的值保存到传递给本函数的参数中。
- 本函数返回成功扫描的数据个数和遇到的任何错误。

```
func Scanf(format string, a ...interface{}) (n int, err error)
```

```go
func main() {
    var (
        name    string
        age     int
        married bool
    )
    fmt.Scanf("1:%s 2:%d 3:%t", &name, &age, &married)
    fmt.Printf("扫描结果 name:%s age:%d married:%t \n", name, age, married)
}
```

```
$ ./scan_demo
1:小王子 2:28 3:false
扫描结果 name:小王子 age:28 married:false
```


3.3 fmt.Scanln

- Scanln类似Scan，它在`遇到换行时才停止扫描`。最后一个数据后面必须有换行或者到达结束位置。
- 本函数返回成功扫描的数据个数和遇到的任何错误。

```go
func Scanln(a ...interface{}) (n int, err error)
```

```go
func main() {

    var (
        name    string
        age     int
        married bool
    )
    fmt.Scanln(&name, &age, &married)
    fmt.Printf("扫描结果 name:%s age:%d married:%t \n", name, age, married)

}
```

```
PS F:\tools\golang\fmt> go run .\fprint.go
nihao 12 true
扫描结果 name:nihao age:12 married:true
```


3.4 Fscan系列

这几个函数功能分别类似于fmt.Scan、fmt.Scanf、fmt.Scanln三个函数，只不过它们不是从标准输入中读取数据而是`从io.Reader中读取数据`。

func Fscan

```
func Fscan(r io.Reader, a ...interface{}) (n int, err error)
```

Fscan从r扫描文本，将成功读取的空白分隔的值保存进成功传递给本函数的参数。换行视为空白。返回成功扫描的条目个数和遇到的任何错误。如果读取的条目比提供的参数少，会返回一个错误报告原因。

func Fscanln

```
func Fscanln(r io.Reader, a ...interface{}) (n int, err error)
```

Fscanln类似Fscan，但会在换行时才停止扫描。最后一个条目后必须有换行或者到达结束位置。

func Fscanf

```
func Fscanf(r io.Reader, format string, a ...interface{}) (n int, err error)
```

Fscanf从r扫描文本，根据format 参数指定的格式将成功读取的空白分隔的值保存进成功传递给本函数的参数。返回成功扫描的条目个数和遇到的任何错误。

3.5 Sscan系列

这几个函数功能分别类似于fmt.Scan、fmt.Scanf、fmt.Scanln三个函数，只不过它们不是从标准输入中读取数据而是`从指定字符串中读取数据`。

func Sscan

```
func Sscan(str string, a ...interface{}) (n int, err error)
```

Sscan从字符串str扫描文本，将成功读取的空白分隔的值保存进成功传递给本函数的参数。换行视为空白。返回成功扫描的条目个数和遇到的任何错误。如果读取的条目比提供的参数少，会返回一个错误报告原因。

func Sscanf

```
func Sscanf(str string, format string, a ...interface{}) (n int, err error)
```

Sscanf从字符串str扫描文本，根据format 参数指定的格式将成功读取的空白分隔的值保存进成功传递给本函数的参数。返回成功扫描的条目个数和遇到的任何错误。

func Sscanln

```
func Sscanln(str string, a ...interface{}) (n int, err error)
```

Sscanln类似Sscan，但会在换行时才停止扫描。最后一个条目后必须有换行或者到达结束位置。


```go
import (
    "fmt"
    // "os"
    // "errors"
)

func main() {

    var data1, data2, data3, data4 string

    fmt.Sscanln("starting1 end Ruda", &data1, &data2, &data3, &data4)

    fmt.Println(data1)
    fmt.Println(data2)
    fmt.Println(data3)
    fmt.Println(data4)

}
```

```
问题 1  输出  调试控制台  终端  COMMENTS

PS F:\tools\golang\fmt> go run .\fprint.go
starting1
end
Ruda

PS F:\tools\golang\fmt>
```