# **Go 语言字符串**

## **实验简介**

本节介绍 Go 语言中的字符串以及常用的两个字符串处理包。

## **一、实验说明**

本课程所有源代码，可以在XfceTerminal中通过以下方式克隆到实验环境:

$ git clone http://git.shiyanlou.com/shiyanlou/Golang\_Programming

## **二. 字符串**

Go语言中的字符串是 [UTF-8](http://zh.wikipedia.org/wiki/UTF-8" \t "https://www.shiyanlou.com/courses/11/labs/58/_blank) 字符的一个序列（当字符为 ASCII 码时则占用 1 个字节，其它字符根据需要占用 2-4 个字节）。UTF-8 是被广泛使用的编码格式，是文本文件的标准编码，其它包括 XML 和 JSON 在内，也都使用该编码。由于该编码对占用字节长度的不定性，Go 中的字符串也可能根据需要占用 1 至 4 个字节，这与其它语言如 C++、Java 或者 Python 不同。Go 这样做的好处是不仅减少了内存和硬盘空间占用，同时也不用像其它语言那样需要对使用 UTF-8 字符集的文本进行编码和解码。

Go语言中字符串的可以使用双引号( " )或者反引号( ` )来创建。双引号用来创建可解析的字符串字面量，所谓可解析的是指字符串中的一些符号可以被格式化为其他内容，如"\n"在在输出时候会被格式化成换行符， 如果需要按照原始字符输出必须进行转义。而反引号创建的字符串原始是什么样，那输出还是什么，不需要进行任何转义。以下是几个例子:

t1 := "\"hello\"" //内容： "hello"

t2 := `"hello"` //内容：和t1一致

t3 := "\u6B22\u8FCE" //内容：你好

Go语言中的部分转义字符如下表所示：

| **转义字符** | **含义** |
| --- | --- |
| \\ | 表示反斜线 |
| \' | 单引号 |
| \" | 双引号 |
| \n | 换行符 |
| \uhhhh | 4个16进制数字给定的Unicode字符 |

在Go语言中单个字符可以使用单引号( ' )来创建。之前的课程中，我们有学习过rune类型，它等同于unint32，在Go语言中，一个单一的字符可以用一个单一的rune来表示。这也是容易理解的，因为Go语言的字符串是UTF-8编码，其底层使用4个字节表示，也就是32 bit。

在Go语言中，字符串支持切片操作，但是需要注意的是如果字符串都是由ASCII字符组成，那可以随便使用切片进行操作，但是如果字符串中包含其他非ASCII字符，直接使用切片获取想要的单个字符时需要十分小心，因为对字符串直接使用切片时是通过字节进行索引的，但是非ASCII字符在内存中可能不是由一个字节组成。如果想对字符串中字符依次访问，可以使用range操作符。另外获取字符串的长度可能有两种含义，一种是指获取字符串的字节长度，一种是指获取字符串的字符数量。字符串支持以下操作：

| **语法** | **描述** |
| --- | --- |
| s += t | 将字符串t追加到s末尾 |
| s + t | 将字符串s和t级联 |
| s[n] | 从字符串s中索引位置为n处的原始字节 |
| s[n:m] | 从位置n到位置m-1处取得的字符（字节）串 |
| s[n:] | 从位置n到位置len(s)-1处取得的字符（字节）串 |
| s[:m] | 从位置0到位置m-1处取得的字符（字节）串 |
| len(s) | 字符串s中的字节数 |
| len([]rune(s)) | 字符串s中字符的个数，可以使用更快的方法utf8.RuneCountInString() |
| [ ]rune(s) | 将字符串s转换为一个unicode值组成的串 |
| string(chars) | chars类型是[]rune或者[]int32, 将之转换为字符串 |
| [ ]byte(s) | 无副本的将字符串s转换为一个原始的字节的切片数组，不保证转换的字节是合法的UTF-8编码字节 |

让我们尝试一个例子， 使用vim创建源文件string\_t.go，然后输入以下源代码：

package main

import (

"fmt")

func main() {

t0 := "\u6B22\u8FCE\u6765\u5230" // t0内容：欢迎来到

t1 := "\u5B9E\u9A8C\u697C" // t1内容：实验楼

t2 := t0 + t1

for index, char := range t2 {

fmt.Printf("%-2d %U '%c' %X %d\n",

index, char, char, []byte(string(char)), len([]byte(string(char))))

}

fmt.Printf("length of t0: %d, t1: %d, t2: %d\n", len(t0), len(t1), len(t2))

fmt.Printf("content of t2[0:2] is: %X\n", t2[0:2])

}

然后通过以下方式运行，在这里一起显示了程序的输出（由于console可能不支持中文显示，可能显示乱码）:

$ go run string\_t.go0 U+6B22 '欢' E6ACA2 33 U+8FCE '迎' E8BF8E 36 U+6765 '来' E69DA5 39 U+5230 '到' E588B0 312 U+5B9E '实' E5AE9E 315 U+9A8C '验' E9AA8C 318 U+697C '楼' E6A5BC 3

length of t0: 12, t1: 9, t2: 21

content of t2[0:2] is: E6AC

说明：通过前面的课程我们知道通过\uhhhh的方式我们可以通过创建Unicod字符，在以上程序中，首先通过:=符号创建了变量t0，其值为\u6B22\u8FCE\u6765\u5230，是欢迎来到中文字符的unicode编码，然后以同样的方式创建了变量t1，其值为实验楼，然后通过+操作符将t0和t1拼接赋值给t2。然后我们通过range操作符号对unicode字符串t2中的每一个unicode字符依次操作，我们这里只是简单的打印出每个字符在t2中的位置，每个字符的unicode码值，每个字符的字面量，每个字符的十六进制值，以及每个字符的字节长度。这里我们使用fmt包种支持的格式指令，如果读者学习过C语言的话就一目了然。接着，我们通过len操作符计算出了每个字符串的字节长度。最后，我们使用切片访问了字符串t2的第0-1个字节，也就是前两个字节，其内容为E6AC。前面我们说到不能使用切片的方式访问非ASCII字符串中的字符，原因在这里一目了然。字符欢其底层使用了三个字节表示，内容是E6ACA2，如果只是简单的使用切片（只取切片中的一项)访问的是不能访问到整个字符的，因为字符的切片是通过字节数来索引的。

## **三. 格式化字符串**

Go语言标准库中的fmt包提供了打印函数将数据以字符串形式输出到控制台，文件，其他满足io.Writer接口的值以及其他字符串。目前为止我们使用了fmt.Printf和fmt.Prinfln，对于前者的使用，就像C语言中的printf函数一样，我们可以提供一些格式化指令，让Go语言对输出的字符串进行格式化。同样的我们可以使用一些格式化修饰符，改变格式化指令的输出结果， 如左对齐等。常用的格式化指令如下：

| **格式化指令** | **含义** |
| --- | --- |
| %% | %字面量 |
| %b | 一个二进制整数，将一个整数格式化为二进制的表达方式 |
| %c | 一个Unicode的字符 |
| %d | 十进制数值 |
| %o | 八进制数值 |
| %x | 小写的十六进制数值 |
| %X | 大写的十六进制数值 |
| %U | 一个Unicode表示法表示的整形码值，默认是4个数字字符 |
| %s | 输出以原生的UTF-8字节表示的字符，如果console不支持UTF-8编码，则会输出乱码 |
| %t | 以true或者false的方式输出布尔值 |
| %v | 使用默认格式输出值，或者使用类型的String()方法输出的自定义值，如果该方法存在的话 |
| %T | 输出值的类型 |

常用的格式化指令修饰符如下：

* 空白 如果输出的数字为负，则在其前面加上一个减号"-"。如果输出的是整数，则在前面加一个空格。使用%x或者%X格式化指令输出时，会在结果之间添加一个空格。例如fmt.Printf("% X", "实")输出E5 AE 9E
* #
  + %#o 输出以0开始的八进制数据
  + %#x 输出以0x开始的十六进制数据
* + 让格式化指令在数值前面输出+号或者-号，为字符串输出ASCII字符（非ASCII字符会被转义），为结构体输出其字段名
* - 让格式化指令将值向左对齐（默认值为像右对齐）
* 0 让格式指令以数字0而非空白进行填充

让我们练习一下，使用vim创建源文件fmt\_t.go，输入以下源码：

package main

import (

"fmt"

)

func main() {

text := "\u5B9E\u9A8C\u697C"

fmt.Printf("bool output:\n%t\n%t\n\n", true, false)

fmt.Println("number output, origin value: 64")

fmt.Printf("|%b|%8b|%-8b|%08b|% 8b|\n", 64, 64, 64, 64, 64)

fmt.Printf("|%x|%8x|%-8x|%08X|% 8X|\n\n", 64, 64, 64, 64, 64)

fmt.Println(`text output, origin value: \u5B9E\u9A8C\u697C`)

fmt.Printf("content: %s\n", text)

fmt.Printf("hex value: % X\nUnicode value: ", text)

for \_, char := range text {

fmt.Printf("%U ", char)

}

fmt.Println()

bytes := []byte(text)

fmt.Printf("value of bytes: %s\n", bytes)

fmt.Printf("hex value of bytes: % X\n", bytes)

fmt.Printf("origin value of bytes: %v\n", bytes)

}

运行代码，输出如下：

$ go run fmt\_t.gobool output:truefalse

number output, origin value: 64

|1000000| 1000000|1000000 |01000000| 1000000|

|40| 40|40 |00000040| 40|

text output, origin value: \u5B9E\u9A8C\u697C

content: 实验楼

hex value: E5 AE 9E E9 AA 8C E6 A5 BC

Unicode value: U+5B9E U+9A8C U+697C value of bytes: 实验楼

hex value of bytes: E5 AE 9E E9 AA 8C E6 A5 BC

origin value of bytes: [229 174 158 233 170 140 230 165 188]

代码一目了然，就不详细解释了。

## **四. 字符串处理相关的包**

Go语言处理字符串的强大之处不仅限于对索引和切片的支持，很多官方包提供了大量的实用函数，可以对字符串进行很方便的操作。在这里我们简单的介绍几个常用的包。

### **1. strings 包**

[strings](http://golang.org/pkg/strings/" \t "https://www.shiyanlou.com/courses/11/labs/58/_blank)包提供了如查找字符串，分割字符串，判断前后缀，判断字符串包含，字符串替换，统计字符串出现的次数等常用操作，完整的方法列表可以参考官方包说明。下面我们通过一个小练习来感受下。使用vim创建文件strings\_package.go，输入以下源码：

package main

import (

"fmt"

"strings"

)

func main() {

var str string = "go\_lang"

fmt.Printf("T/F? Does the string \"%s\" have prefix \"%s\"? ", str, "go")

fmt.Printf("%t\n", strings.HasPrefix(str, "go"))

fmt.Printf("T/F? Does the string \"%s\" contains \"%s\"? ", str, "-")

fmt.Printf("%t\n", strings.Contains(str, "-"))

str\_new := strings.Replace(str, "go", "python", 1)

fmt.Printf("Origin string: \"%s\", after replace: \"%s\"\n", str, str\_new)

fmt.Printf("Number of 'n' in \"%s\" is: %d\n", str\_new, strings.Count(str\_new, "n"))

}

执行代码，输出如下：

$ go run strings\_package.go

T/F? Does the string "go\_lang" have prefix "go"? true

T/F? Does the string "go\_lang" contains "-"? false

Origin string: "go\_lang", after replace: "python\_lang"

Number of 'n' in "python\_lang" is: 2

代码依然很简单，函数的功能就从函数名就可以看出。值得注意的地方是函数func Replace(s, old, new string, n int) string中的参数n指明了将字符串s中的前n个old字符串替换为new字符串，如果n = -1则提供所有匹配到的字符串。

### **2. strconv 包**

[strconv](http://golang.org/pkg/strconv/" \t "https://www.shiyanlou.com/courses/11/labs/58/_blank)包提供了许多可以在字符串和其他类型的数据之间进行转换的函数。例如可以将数字转换为字符串，将数字样式的字符串转换为数值（将字符串"12345"转换int类型的整数）。我们还是直接通过例子学习，创建源文件strconv\_package.go，输入以下代码：

package main

import (

"fmt"

"strconv")

func main() {

var ori string = "123456"

var i int

var s string

fmt.Printf("The size of ints is: %d\n", strconv.IntSize)

i, \_ = strconv.Atoi(ori)

fmt.Printf("The integer is: %d\n", i)

i = i + 5

s = strconv.Itoa(i)

fmt.Printf("The new string is: %s\n", s)

}

以上代码中，需要注意的地方是strconv.IntSize是一个常量，其值是int类型的所占的bit数，运行代码输出如下：

$ go run strconv\_package.go

The size of ints is: 64

The integer is: 123456

The new string is: 123461

## **作业**

请通过查阅资料掌握字符串的常用处理包。