Go语言入门到精通

江洲老师云课堂

主讲：江洲老师

切片2

目 录

append函数

1.append函数

append函数

append() 函数可以向切片尾部添加数据，可以自动为切片扩容，常常会返回新的切片对象。

```
var s1 []int          //创建nil切片，或者：s1 := make([]int, 0)
s1 = append(s1, 1)    //追加1个元素
s1 = append(s1, 2, 3) //追加2个元素
s1 = append(s1, 4, 5, 6) //追加3个元素
fmt.Println(s1)       //[1 2 3 4 5 6]
s2 := make([]int, 5)
s2 = append(s2, 6)
fmt.Println(s2)       //[0 0 0 0 0 6]
s3 := []int{1, 2, 3}
s3 = append(s3, 4, 5)
fmt.Println(s3)       //[1 2 3 4 5]
```

append函数会智能的将底层数组的容量增长，一旦超过原底层数组容量，通常以2倍（1024以下）容量重新分配底层数组，并复制原来的数据。

因此，使用append 给切片做扩充时，切片的地址可能发生变化，但数据都被重新保存了，不影响使用。

```
func main() {
  s := make([]int, 0, 1)
  c := cap(s)
  for i := 0; i < 100; i++ {
    s = append(s, i)
    if n := cap(s); n > c {
      fmt.Printf("cap: %d -> %d\n", c, n)
      c = n
    }
  }
}
```

输出结果如下：

```
cap: 1 -> 2
cap: 2 -> 4
cap: 4 -> 8
cap: 8 -> 16
cap: 16 -> 32
cap: 32 -> 64
```

copy函数

1.copy函数

copy函数

函数 copy 在两个切片间复制数据，复制长度以 len 小的为准，两个切片指向同一底层数组，直接对应位置覆盖。

```
data := [...]int{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
  s1 := data[8:]       //{8, 9}
  s2 := data[:5]       //{0, 1, 2, 3, 4}
  copy(s2, s1)         // dst:s2, src:s1

  fmt.Println(s2)              //[8 9 2 3 4]
  fmt.Println(data)            //[8 9 2 3 4 5 6 7 8 9]
```

切片做函数参数

1.切片做函数参数

切片做函数参数

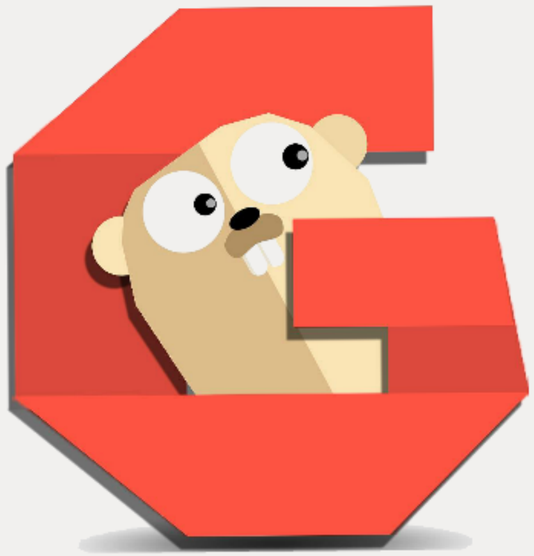
切片作为函数参数时，是怎样传递呢？是传值，还是传引用呢？

```
func testFunc(s []int) {   // 切片做函数参数
  s[0] = -1                // 直接修改 main中的 slice
}

func main() {
  slice := []int{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
  fmt.Println(slice)

  testFunc(slice)       // 传引用
  fmt.Println(slice)
}
```

实例演示



Go语言入门到精通

江洲老师云课堂

主讲：江洲老师

感谢您的聆听和观看