

Array

array数组是值类型。|

* [10]int 和 [20]int是不同类型
* 调用func f(arr [10]int) 会 拷贝 数组
* go语言中一般不直接用数组

```
// 使用指针(引用传递)
func change_arr(arr *[5]int) {

}

arr1 := [5]int{1, 2, 3, 4, 5}
change_arr(&arr1)
```

```
// arr是slice
func change_arr(arr []int) {
}
// arr是数组
func change_arr(arr [5]int) {
}
```

Slice

slice本身没有数据，是对底层array的一个view

slice定义：
arr := [...]int{0, 1, 2, 3, 4, 5, 6, 7} // 初始化数组
s := arr[2:6] // slice

* s的值为 [2, 3, 4, 5]
* 半开半闭原则

slice是引用传递：

参数: slice

```
func update(arr []int) {
    arr[0] = 100
}

func main() {
    arr := [...]int{0, 1, 2, 3, 4, 5, 6, 7} // 初始化数组
    s := arr[2:6] // 初始化slice
    update(s)
    fmt.Println("s =", s)
    fmt.Println("arr =", arr)
}
```

* s = [100 3 4 5]
* arr = [0 1 100 3 4 5 6 7]

如何快速取得slice：

```
func update(arr []int) {
}

func main() {
    arr := [...]int{0, 1, 2, 3, 4, 5, 6, 7} // 初始化数组
    update(arr[:])
}
```

Reslice：

```
func main() {
    arr := [...]int{0, 1, 2, 3, 4, 5, 6, 7} // 初始化数组
    s2 := arr[:]  
    fmt.Println("s2 =", s2)  
    s2 = s2[:5]  
    fmt.Println("s2 =", s2)  
    s2 = s2[2:]  
    fmt.Println("s2 =", s2)
}
```

* s2 = [0 1 2 3 4 5 6 7]
* s2 = [0 1 2 3 4]
* s2 = [2 3 4]

问题一: Slice的扩展?

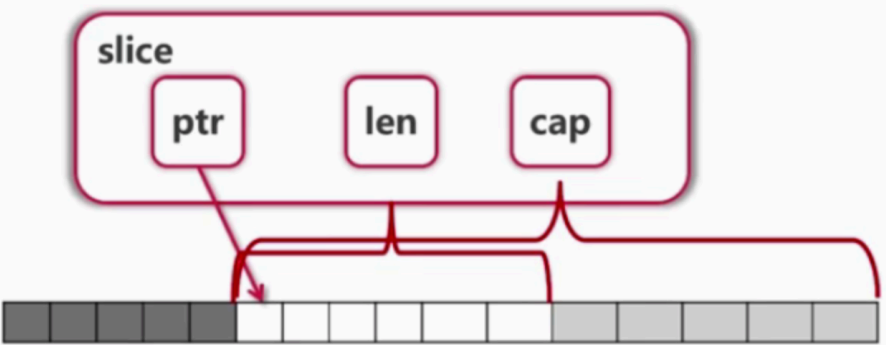
```
arr := [...]int{0, 1, 2, 3, 4, 5, 6, 7}
s1 := arr[2:6]
s2 := s1[3:5]
```

* s1的值为?
* s2的值为?

```
fmt.Printf("s1=%v, len(s1)=%d, cap(s1)=%d\n", s1, len(s1), cap(s1))
fmt.Printf("s2=%v, len(s2)=%d, cap(s2)=%d\n", s2, len(s2), cap(s2))
```

* s1=[2 3 4 5], len(s1)=4, cap(s1)=6
* s2=[5 6], len(s2)=2, cap(s2)=3

Slice的实现



Slice的扩展

```
arr := [...]int{0, 1, 2, 3, 4, 5, 6, 7}
s1 := arr[2:6]
s2 := s1[3:5]
```

- ◆ s1的值为[2 3 4 5]，s2的值为[5 6]
- ◆ slice可以向后扩展，不可以向前扩展
- ◆ s[i] 不可以超越 len(s)，向后扩展不可以超越底层数组cap(s)