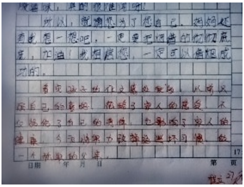
Golang：值类型与引用类型

与蟒唯舞 简书作者

0.163 2017-11-16 18:20 打开App

> Golang is a **pass by value** language, so any time we pass a value to a function either as a receiver or as an argument that data is copied in memory and so the function by default is always going to be working on a copy of our data structure. We can address this problem and modify the actual underlying data structure through the use of pointers

> and memory address.



加盟信100字作文

写作加盟

1899阅读

广告 ×

值类型

先来看一段代码：

```
package main

import (
    "fmt"
)

type person struct {
```

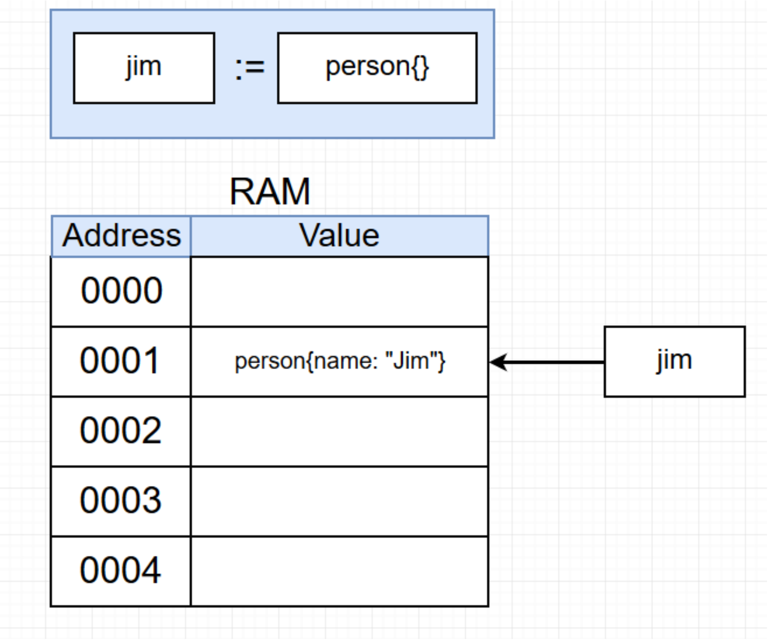
```go
    name string
}

func (p person) print() {
    fmt.Printf("%+v", p)
}

func (p person) updateName(newName string) {
    p.name = newName
}

func main() {
    jim := person{name: "Jim"}
    jim.updateName("Jimmy")
    jim.print()
}
```

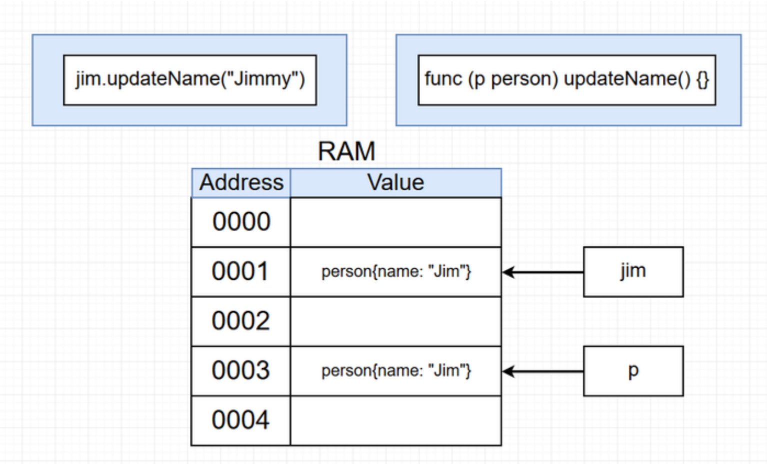
运行结果：

```
{name:Jim}
```

没有达到我们期待的结果：

```
{name:Jimmy}
```

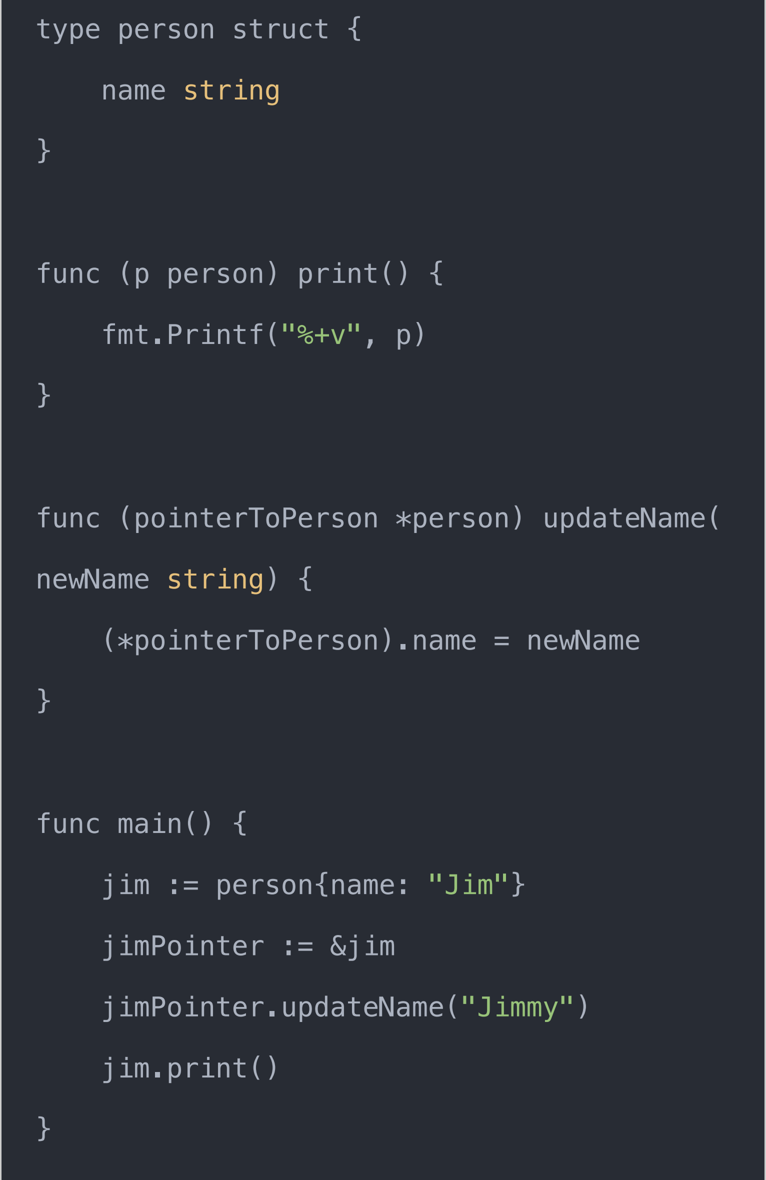




值传递意味着每当我们将某个值传递给函数时，golang 就会取那个值或那个结构。

为了达到我们的预期，修改源代码：

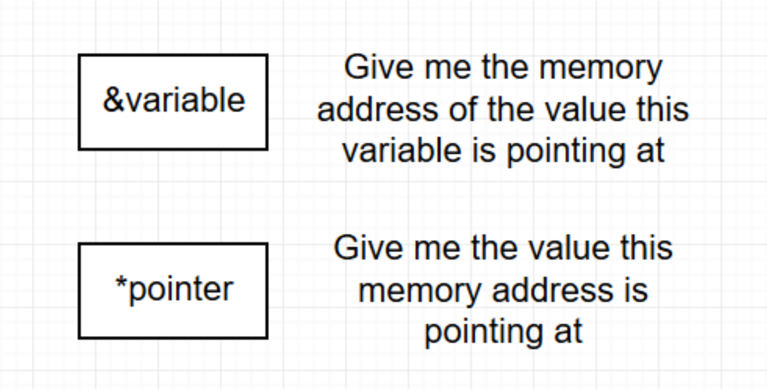
```
package main

import (
    "fmt"
)
```



```go
type person struct {
    name string
}

func (p person) print() {
    fmt.Printf("%+v", p)
}

func (pointerToPerson *person) updateName(
newName string) {
    (*pointerToPerson).name = newName
}

func main() {
    jim := person{name: "Jim"}
    jimPointer := &jim
    jimPointer.updateName("Jimmy")
    jim.print()
}
```

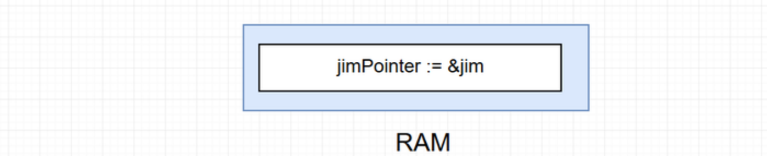
运行结果：

```
{name:Jimmy}
```

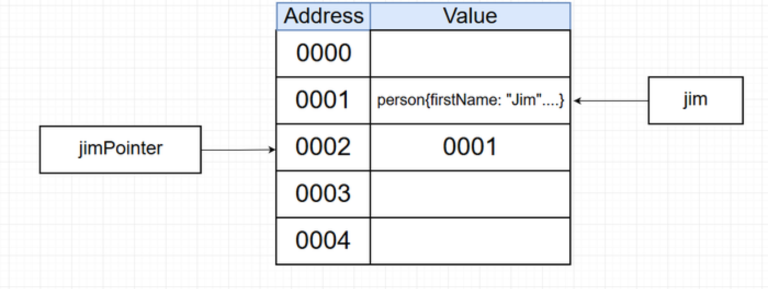


Give me the memory address of the value this variable is pointing at

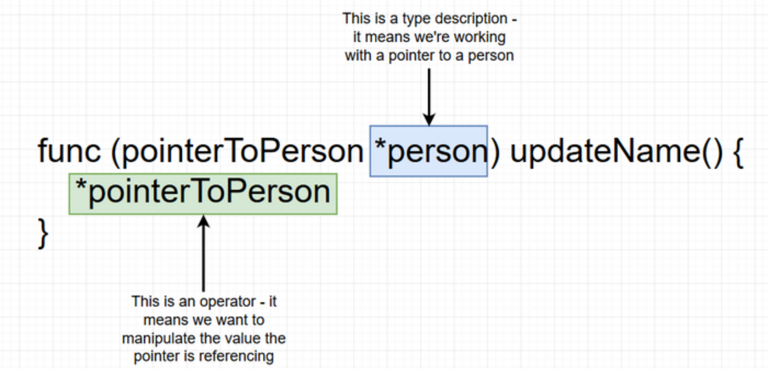
Give me the value this memory address is pointing at



```
jimPointer := &jim
```



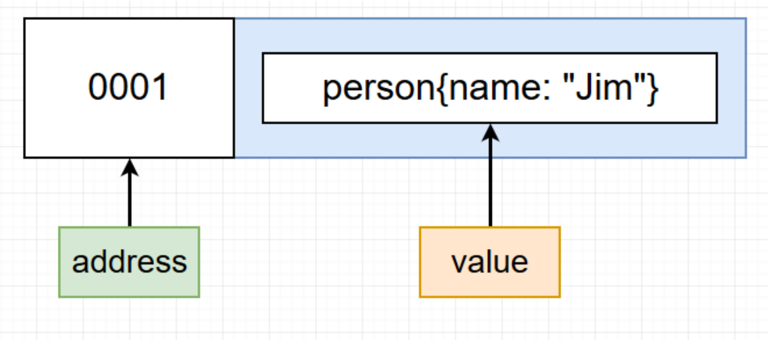
Address	Value
0000	
0001	person{firstName: "Jim"....}
0002	0001
0003	
0004	

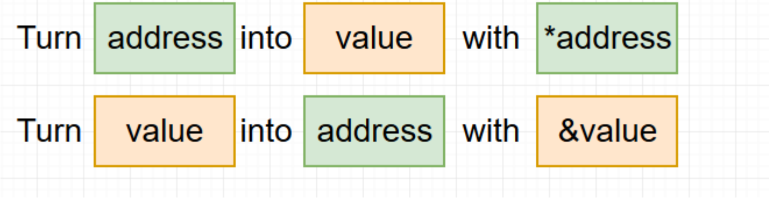


```
func (pointerToPerson *person) updateName() {
    *pointerToPerson
}
```

This is a type description - it means we're working with a pointer to a person

This is an operator - it means we want to manipulate the value the pointer is referencing





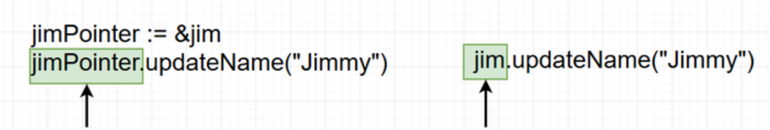
Turn address into value with *address

Turn value into address with &value

最后一点，小的改进：

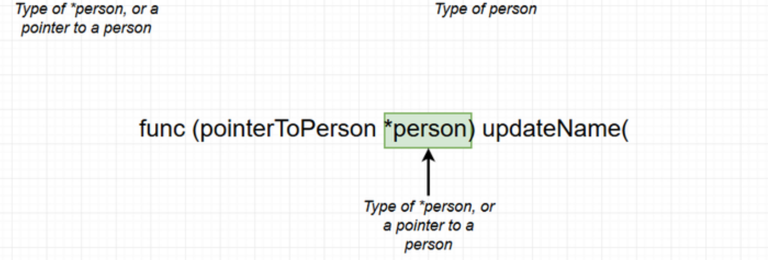
```
func main() {
    jim := person{name: "Jim"}
    jim.updateName("Jimmy")
    jim.print()
}
```

pointer shortcut, go will automatically turn your variable of type person into pointer person for you.



```
jimPointer := &jim
jimPointer.updateName("Jimmy")
```

```
jim.updateName("Jimmy")
```



引用类型

先看一段代码：

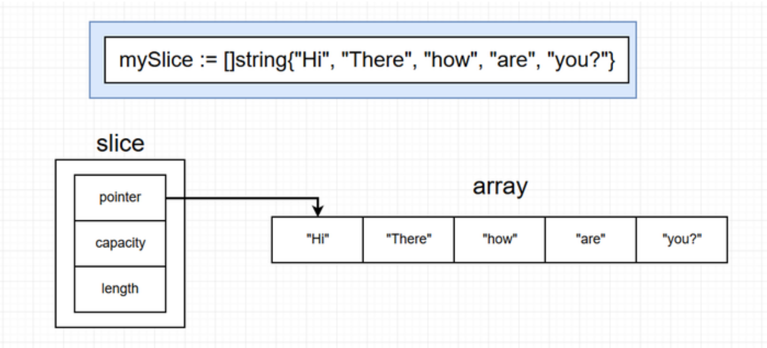
```
package main

import "fmt"

func updateSlice(s []string) {
    s[0] = "Bye"
}

func main() {
    mySlice := []string{"Hi", "There", "ho
```

```
w", "are", "you?"}
    updateSlice(mySlice)
    fmt.Println(mySlice)
}
```

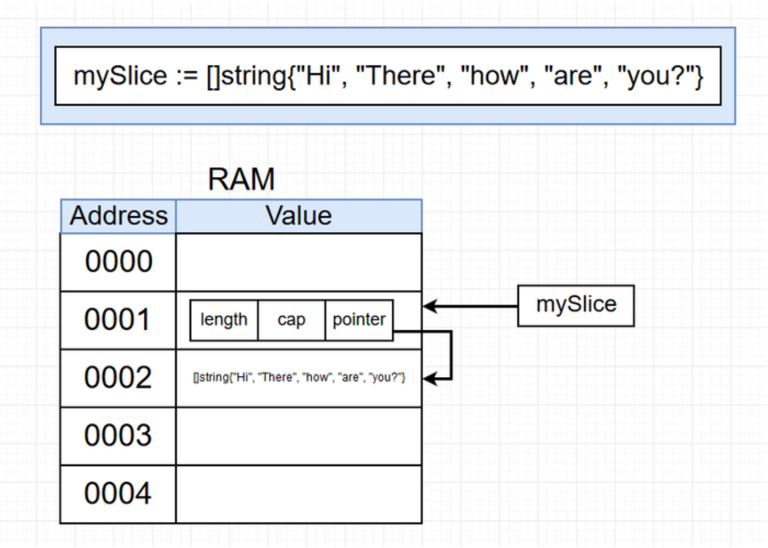
运行结果：

```
[Bye There how are you?]
```

运行结果为什么是这样?



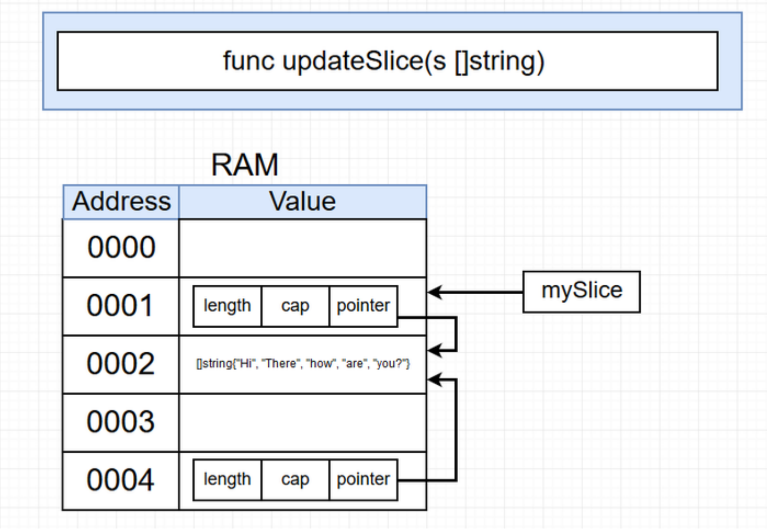
```
mySlice := []string{"Hi", "There", "how", "are", "you?"}
```



```
mySlice := []string{"Hi", "There", "how", "are", "you?"}
```

RAM

Address	Value
0000	
0001	length | cap | pointer
0002	[]string{"Hi", "There", "how", "are", "you?"}
0003	
0004	



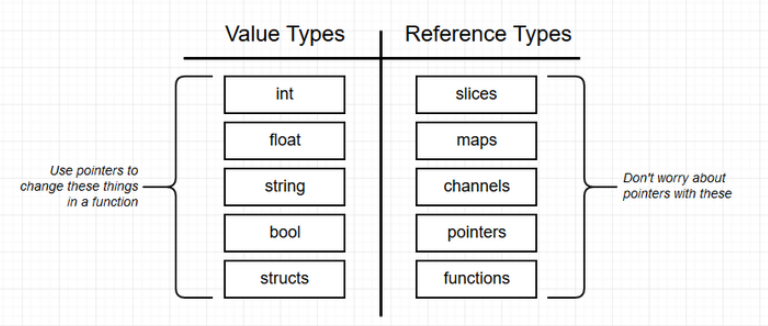
```
func updateSlice(s []string)
```

RAM

Address	Value
0000	
0001	length | cap | pointer
0002	[]string{"Hi", "There", "how", "are", "you?"}
0003	
0004	length | cap | pointer

When we make the slice of string golang internally is creating two separate data structures. What we call a function and pass my slice into it, golang is still behaving as a pass by value language. Even though the slice data structure is copied it is still pointing at the original array in memory, because the slice data structure and the array data structure are two separate elements in memory.

总结



Whenever you pass an integer, float, string, or struct into a function, golang will creates a copy of each argument, and these copies are used inside of the function.

© 著作权归作者所有,转载或内容合作请联系作者

点赞赚钻 最高日赚数百元

赞 (5)

与蟒唯舞

一起摇摆

赞赏

暂无评论

写评论

智慧如你，不想 发表一点想法 咩~

推荐阅读

更多精彩内容 >

又是夜半惊坐起 App中阅读 👁 4254 💬 0 ♥ 5

蒲公英获取UDID

咩咩sheep App中阅读 👁 1303 💬 2 ♥ 1

清欢

宛丘之 App中阅读 👁 321 💬 3 ♥ 7

简安

踏曦行 App中阅读 👁 42 💬 0 ♥ 0

我说所有的酒，都不如你

72滩 App中阅读 67 0 1

如何打造一个顾客、员工、经营者心目中一致的店铺形象

柳二白 App中阅读 571 0 8

简书
Jianshu.com

创作你的创作，
接受世界的赞赏

登录 | 打开App | 热门文章

下载简书，创作你的创作