整明白 Golang struct 实例化和赋值

Mr_houzi
958 • 3 • 28 • 46

发布于
2021-11-19

> Golang 中 struct 各种实例化和赋值方式，一会儿是值传递，一会儿又是指针，让人一头雾水，于是我决定梳理一下，整个明白。

先定义一个结构体，下面结合代码进行讲解。

```
package main

import "fmt"

type Person struct {
    Name string
    Age int
    Descprtion string
}
```

实例一

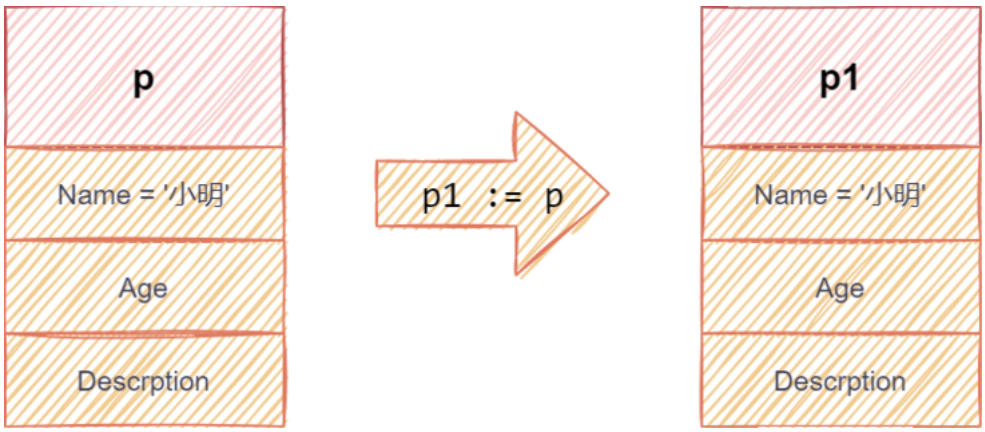
p 以最常规的方式实例化一个 struct，变量 p 得到一个 Person 结构体。

```
p := Person{}
p.Name = "小明"

fmt.Printf("p:%+v 变量地址：%p\n", p, &p)
fmt.Println("============")

// result:
// p:{Name:小明 Age:0 Descprtion:} 变量地址：0xc000078480
// ============
```

实例二

变量 p1 由 p 赋值而来，由于 Golang 语言是值传递，赋值后，对 p1 的修改并不会影响到 p；

从第一个输出也可以看得出，Golang 的赋值并不存在像PHP变量赋值时的写时复制（copy on write）机制。

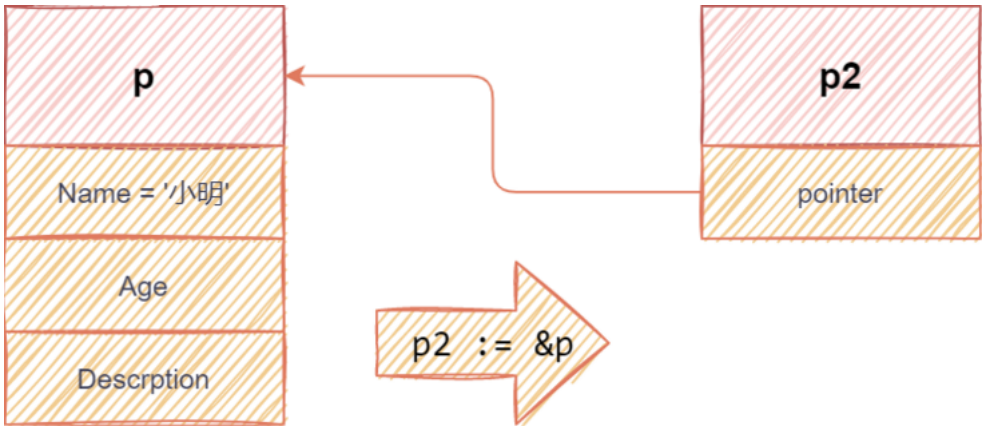



```
p1 := p
fmt.Printf("p1:%+v 变量地址：%p\n", p1, &p1) // 不存在写时复制
p1.Name = "小明p1"
fmt.Printf("p:%+v 变量地址：%p\n", p, &p)
fmt.Printf("p1:%+v 变量地址：%p\n", p1, &p1)
fmt.Println("============")

// result:
// p1:{Name:小明 Age:0 Descprtion:} 变量地址：0xc0000784e0
```

```
// p:{Name:小明 Age:0 Descprtion:} 变量地址：0xc000078480
// p1:{Name:小明p1 Age:0 Descprtion:} 变量地址：0xc0000784e0
// ===========
```

实例三

利用取地址符将 p 的地址赋值给 p2，变量 p2 是一个指针，存放着指向 p 的地址。当 p2 修改了结构体中元素 Name 时，通过 p 访问结构体对应的值



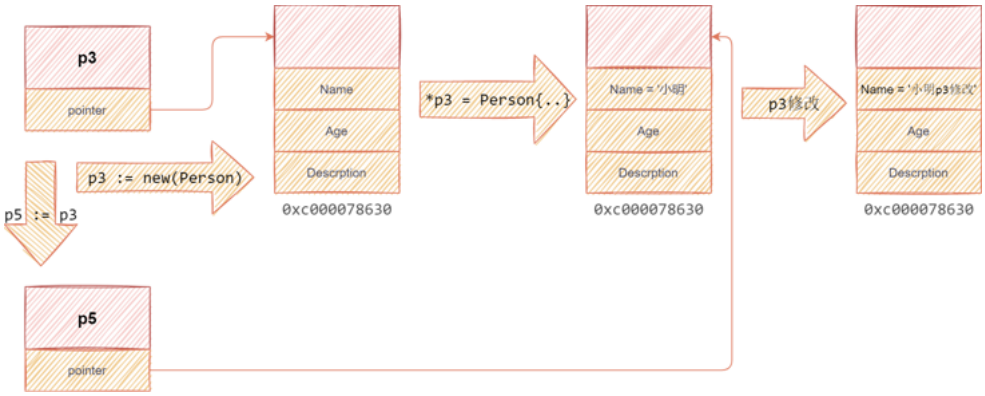

```
p2 := &p // 等同于 var p2 *Person = &p
fmt.Printf("p2:%+v 指针变量指向地址（变量值）：%p 变量地址：%p\n", p2, p2, &p2)
p2.Name = "小明p2"
fmt.Printf("p1:%+v 变量地址：%p\n", p, &p)
fmt.Printf("p2:%+v 指针变量指向地址（变量值）：%p 变量地址：%p\n", p2, p2, &p2)
fmt.Println("===========")

// result:
// p2:&{Name:小明 Age:0 Descprtion:} 指针变量指向地址（变量值）：0xc000078480 变量地址：0xc000006030
// p1:{Name:小明p2 Age:0 Descprtion:} 变量地址：0xc000078480
// p2:&{Name:小明p2 Age:0 Descprtion:} 指针变量指向地址（变量值）：0xc000078480 变量地址：0xc000006030
// ===========
```

实例四

变量 p3 由 new(Person) 得来。new 将开辟一块内存，返回内存地址给 p3，也即 p3 是一个指向这块内存的指针。

p3 是指向结构体的指针，它有两种方式可以操作结构体，`p3.Age = 3` 和 `*p3 = Person{Name: "小明p3"}`, 如果第二种方式后操作，将会覆盖第一种方式对

由于 p3 是指针，当 p3 赋值给 p5 时，p5 也将指向这块内存地址。

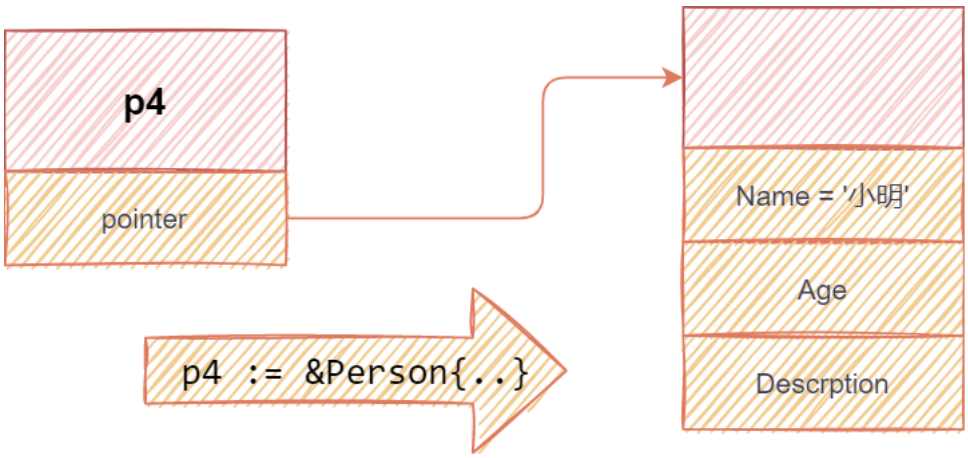



```
p3 := new(Person)
fmt.Printf("p3:%+v 指针变量指向地址（变量值）：%p 变量地址：%p\n", p3, p3, &p3)
p3.Age = 3 // 等同于 (*p3).Age = 3
fmt.Println("================ 操作 Age ================")
fmt.Printf("p3:%+v 指针变量指向地址（变量值）：%p 变量地址：%p\n", p3, p3, &p3)
*p3 = Person{
    Name: "小明p3",
```

```
}
fmt.Println("================ 操作 Name ================")
fmt.Printf("p3:%+v 指针变量指向地址（变量值）: %p 变量地址: %p\n", p3, p3, &p3)
p5 := p3
fmt.Println("================ p5 := p3 ================")
fmt.Printf("p3:%+v 指针变量指向地址（变量值）: %p 变量地址: %p\n", p3, p3, &p3)
fmt.Printf("p5:%+v 指针变量指向地址（变量值）: %p 变量地址: %p\n", p5, p5, &p5)
p3.Name = "小明p3修改"
fmt.Println("================ p3 修改 ================")
fmt.Printf("p3:%+v 指针变量指向地址（变量值）: %p 变量地址: %p\n", p3, p3, &p3)
fmt.Printf("p5:%+v 指针变量指向地址（变量值）: %p 变量地址: %p\n", p5, p5, &p5)
fmt.Println("===========")

// result:
// p3:&{Name: Age:0 Descprtion:} 指针变量指向地址（变量值）: 0xc000078630 变量地址: 0xc000006038
// ================ 操作 Age ================
// p3:&{Name: Age:3 Descprtion:} 指针变量指向地址（变量值）: 0xc000078630 变量地址: 0xc000006038
// ================ 操作 Name ================
// p3:&{Name:小明p3 Age:0 Descprtion:} 指针变量指向地址（变量值）: 0xc000078630 变量地址: 0xc000006038
// ================ p5 := p3 ================
// p5:&{Name:小明p3 Age:0 Descprtion:} 指针变量指向地址（变量值）: 0xc000078630 变量地址: 0xc000006040
// ================ p3 修改 ================
// p3:&{Name:小明p3修改 Age:0 Descprtion:} 指针变量指向地址（变量值）: 0xc000078630 变量地址: 0xc000006038
// p5:&{Name:小明p3修改 Age:0 Descprtion:} 指针变量指向地址（变量值）: 0xc000078630 变量地址: 0xc000006040
// ===========
```

示例五

p4 的实例化方式也将得到一个指针，这种实例化方式与 p3 的实例化是相同的，但 p4 的写法更常使用。




```
p4 := &Person{
    Name: "小明p4",
}
fmt.Printf("%+v %p\n", p4, &p4)

// result:
// &{Name:小明p4 Age:0 Descprtion:} 0xc000006048
```

附完整代码：

```
package main

import "fmt"

type Person struct {
    Name string
    Age int
    Descprtion string
}

func main() {
    p := Person{}
```

```
    p.Name = "小明"

    fmt.Printf("p:%+v 变量地址：%p\n", p, &p)
    fmt.Println("===========")

    p1 := p
    fmt.Printf("p1:%+v 变量地址：%p\n", p1, &p1) // 不存在写时复制
    p1.Name = "小明p1"
    fmt.Printf("p:%+v 变量地址：%p\n", p, &p)
    fmt.Printf("p1:%+v 变量地址：%p\n", p1, &p1)
    fmt.Println("===========")

    p2 := &p
    fmt.Printf("p2:%+v 指针变量指向地址（变量值）：%p 变量地址：%p\n", p2, p2, &p2)
    p2.Name = "小明p2"
    fmt.Printf("p1:%+v 变量地址：%p\n", p, &p)
    fmt.Printf("p2:%+v 指针变量指向地址（变量值）：%p 变量地址：%p\n", p2, p2, &p2)
    fmt.Println("===========")

    p3 := new(Person)
    fmt.Printf("p3:%+v 指针变量指向地址（变量值）：%p 变量地址：%p\n", p3, p3, &p3)
    p3.Age = 3 // 等同于 (*p3).Age = 3
    fmt.Println("================ 操作 Age ================")
    fmt.Printf("p3:%+v 指针变量指向地址（变量值）：%p 变量地址：%p\n", p3, p3, &p3)
    *p3 = Person{
        Name: "小明p3",
    }
    fmt.Println("================ 操作 Name ================")
    fmt.Printf("p3:%+v 指针变量指向地址（变量值）：%p 变量地址：%p\n", p3, p3, &p3)
    p5 := p3
    fmt.Println("================ p5 := p3 ================")
    fmt.Printf("p3:%+v 指针变量指向地址（变量值）：%p 变量地址：%p\n", p3, p3, &p3)
    fmt.Printf("p5:%+v 指针变量指向地址（变量值）：%p 变量地址：%p\n", p5, p5, &p5)
    p3.Name = "小明p3修改"
    fmt.Println("================ p3 修改 ================")
    fmt.Printf("p3:%+v 指针变量指向地址（变量值）：%p 变量地址：%p\n", p3, p3, &p3)
    fmt.Printf("p5:%+v 指针变量指向地址（变量值）：%p 变量地址：%p\n", p5, p5, &p5)
    fmt.Println("===========")

    p4 := &Person{
        Name: "小明p4",
    }
    fmt.Printf("%+v %p\n", p4, &p4)
}
```

End!

后端 go php