**接口初始化和赋值机制**

### 接口内部数据结构：

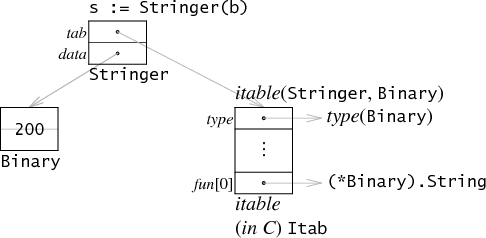
1. **接口内部数据结构：**

|  |
| --- |
| *$GOROOT/src/pkg/runtime/runtime.h*  *178 struct Iface*  *179 {*  *180 Itab\* tab;*  *181 void\* data;*  *182 };*    *449 struct Itab*  *450 {*  *451 InterfaceType\* inter;*  *452 Type\* type;*  *453 Itab\* link;*  *454 int32 bad;*  *455 int32 unused;*  *456 void (\*fun[])(void);*  *457 };* |
|  |

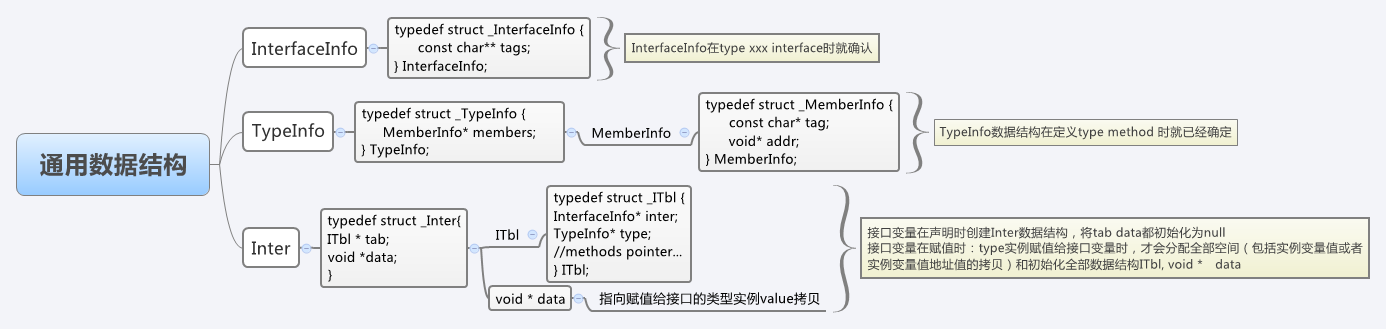
c模拟go接口底层实现原理：



1. **interface 实例典型的内部数据结构图：**



1. **内部数据结构关系图：**





注意本图定义中：

struct Inter 对应 *Iface；InterfaceInfo对应InterfaceType ；TypeInfo 对应Type*

### 接口方法调用原理：

一个type 实例(对象)：实际上就是 TypeInfo + void \* data ，对于同一种类型type 实例，TypeInfo 是共享的的，原因是代码段是不变的可以共享，但是数据段是每个实例开辟一个空间：



type变量赋值给接口时，做了如下处理：

1. 判断type 实例 的MemberInfo 中的method 签名set 是否是接口method 签名set 的超集；如果不是则直接报错；

type 实例变量类型是T，则其方法集是：Reciever为T的method集合

type 实例变量类型是\*T，则其方法集是：Reciever为T和\*T的method集合

（type T 实例变量的method set 是所有receiver type为 T的mothod集合；

type \*T 实例变量的method set 是所有receiver type为 T或者\*T的mothod集合；

）

**接口调用不会做 receiver 自动转换，目标方法必须在接口实现方法集中**

1. 如果是，则初始化接口变量的*数据结构*

|  |
| --- |
| *struct Itab*  *{*  *InterfaceType\* inter; //------可以初始化*  *Type\* type; //------初始化*  *Itab\* link;*  *int32 bad;*  *int32 unused;*  *void (\*fun[])(void); //-----通过TypeInfo里面*MemberInfo中保存的methods地址初始化  *};* |
| *struct Iface*  *{*  *Itab\* tab; // 初始化 上面已经赋值的变量*  *void\* data; // 复制实例变量值拷贝或者实例变量值地址值的拷贝*  *};* |

**接口赋值初始化逻辑图：**



**struct 匿名字段调用规则**

**匿名字段⽅方法集规则:**

• 如果 S 嵌⼊入匿名类型 T，则 S ⽅方法集包含 T ⽅方法集。

• 如果 S 嵌⼊入匿名类型 \*T，则 S ⽅方法集包含 \*T 的⽅方法集 (T + \*T)。

• 如果 S 嵌⼊入匿名类型 T 或 \*T，则 \*S ⽅方法集包含 \*T 的⽅方法集 (T + \*T)。

**通俗描述：**

type struct S{

T

}

==>> s 和 \*s 可以调receiver T.的method

==>> \*s 也可以调receiver T和\*T的method

type struct S{

\*T

}

==>> s 和 \*s 可以调receiver是 T和\*T 的method