# 环境设置

GOPATH环境变量要设置为本地工作目录。在执行go build时，如果Import的包没有明确指明路径，则会在GOPATH环境变量所指定的路径中查找。

# 关于包管理

每个目录可以有多个.go文件，但是均必须属于同一个包。

import导入包的时候，导入的是包所在的目录的路径。比如在目录AAA中有多个.go文件，均属于包bbb。此时在于AAA目录平级的main.go中引用bbb包，需要使用import “./AAA”,而不能使用import “bbb”

虽然上面是import AAA，但是在使用包bbb中的内容是要使用bbb.xxx。

# Go语言基础

## type

type关键字可以实现如下操作：

1. 定义结构体
2. 定义接口
3. 类型定义
4. 类型别名
5. 类型查询

* 类型定义

|  |
| --- |
| type name string  //这里 name 和 string 是不同的数据类型。如果一个函数的入参是name类型，则不可以传入string 类型的变量 |

* 类型别名

|  |
| --- |
| type name = string  // 这里name 和 string 是相同的类型。name 只是 string类型的一个别名。相当于C++中的typedef 。如果一个函数的入参是name类型，那么也可以将一个string类型的参数传入该函数当中。 |

* 类型查询

|  |
| --- |
| var a interface{} = “abc”  switch a.(type)  case string:  // do something  Case int:  // 都something  Default:  // do something |

注意：只能对interface{}类型的变量执行.(type)操作

## interface{}类型变量

空接口类型，既可以是任何类型。

在函数参数中，可以将入参设置位interface{}类型，这样就可以向函数传入任意类型的参数。

但是在函数内部，interface{}类型依然是interface{}类型。使用的时候需要通过类型断言进行转换。

## 类型断言

接口类型向基本类型的转换成为类型断言。形态为x.(T)。其中x是interface{}类型，而T是要转换的目标类型。例如：

|  |
| --- |
| func TypeAssert(a interface{}) {  b,of := a.([]int) // 将空接口类型转换成为整形数组，并初始化b。ok为错误信息  if (ok) {  …  }  } |

## defer

延迟调用。defer后只能跟函数调用语句，否则会报错。

defer声明的函数调用会在其函数执行完成后再被调用。

典型场景：

1. 打开文件描述符后调用defer在函数结束后关闭。
2. Mutex锁定后，调用defer在最后Unlock

如果希望defer声明的函数调用能够在中间执行，可以采用方案：

1. 匿名函数。在函数中创建匿名函数，在匿名函数内部使用defer

如果在函数中声明了多个defer函数调用，其调用顺序是反序，既后声明的先被调用。

defer函数调用的所使用的**参数值**以defer声明时的参数值为准。只有函数参数适用这条规则。

## 切片

切片有点类似通过new创建不固定长度的数组。可能创建在堆上，需要使用make函数。

1. 通过数组创建切片

通过数组创建切片，其内存可能是在栈上。采用的方法：

|  |
| --- |
| var arry [5]int = [5]int{1,2,3,4,5} // 数组  var a []int = arry[:3] //切片 数组arry的第0到3个元素，3是开区间。既1,2,3  var b []int = arry[1:3] // 切片 数组arry的第1到3个元素。3是开区间。既2,3  var c []int = arry[3:] // 切片 数组arry第3到最后一个元素。既4，5 |

1. 通过make创建切片

|  |
| --- |
| Var sl []int = make([]int, 3,5) // len为3,cap为5 |