GoLang学习文档

姚中凯

|  |  |  |
| --- | --- | --- |
| 作者 | 日期 | 内容 |
|  |  |  |
|  |  |  |
|  |  |  |

目录

[基础篇 4](#_Toc513059269)

[1. 安装与配置 4](#_Toc513059270)

[1.1. xxxxx 4](#_Toc513059271)

[2. 程序结构 4](#_Toc513059272)

[2.1. xxxxx 4](#_Toc513059273)

[3. 调试 4](#_Toc513059274)

[3.1. xxxxx 4](#_Toc513059275)

[4. 数据类型 5](#_Toc513059276)

[4.1. 基本数据类型 5](#_Toc513059277)

[4.1.1. 整型 5](#_Toc513059278)

[4.1.2. 浮点型 5](#_Toc513059279)

[4.1.3. 布尔型 6](#_Toc513059280)

[4.2. 复合数据类型 6](#_Toc513059281)

[4.3. 指针 6](#_Toc513059282)

[4.4. unsafe 6](#_Toc513059283)

[5. 过程 8](#_Toc513059284)

[5.1. xxxxx 8](#_Toc513059285)

[6. 函数 8](#_Toc513059286)

[6.1. xxxxx 8](#_Toc513059287)

[7. 接口 9](#_Toc513059288)

[7.1. xxxxx 9](#_Toc513059289)

[8. 并发 9](#_Toc513059290)

[8.1. xxxxx 9](#_Toc513059291)

[9. 性能 9](#_Toc513059292)

[9.1. xxxxx 9](#_Toc513059293)

[10. 包 9](#_Toc513059294)

[10.1. xxxxx 9](#_Toc513059295)

[11. 测试 10](#_Toc513059296)

[11.1. xxxxx 10](#_Toc513059297)

[应用篇 11](#_Toc513059298)

[12. 命令行 11](#_Toc513059299)

[12.1. xxxxx 11](#_Toc513059300)

[13. 缓冲区 11](#_Toc513059301)

[13.1. xxxxx 11](#_Toc513059302)

[14. 编解码 11](#_Toc513059303)

[14.1. xxxxx 11](#_Toc513059304)

[15. 调教 11](#_Toc513059305)

[15.1. xxxxx 11](#_Toc513059306)

[高级篇 11](#_Toc513059307)

[16. 反射及对象 11](#_Toc513059308)

[16.1. xxxxx 12](#_Toc513059309)

[17. 内存及Gc 12](#_Toc513059310)

[17.1. xxxxx 12](#_Toc513059311)

[18. 启动及初始化 12](#_Toc513059312)

[18.1. xxxxx 12](#_Toc513059313)

[19. Cgo/调用/回调 12](#_Toc513059314)

[19.1. xxxxx 12](#_Toc513059315)

[20. 命令&工具 13](#_Toc513059316)

[20.1. xxxxx 13](#_Toc513059317)

# 基础篇

# 安装与配置

## xxxxx

# 程序结构

## xxxxx

# 调试

## xxxxx

# 数据类型

## 基本数据类型

### 整型

|  |  |  |  |
| --- | --- | --- | --- |
| 数据类型 | 符号 | 长度(字节) | 说明 |
| int | Y | 4 or 8，由系统决定 | 有别名rune |
| uint | N | 4 or 8，由系统决定 |  |
| int8 | Y | 1 |  |
| uint8 | N | 1 | 有别名byte |
| int16 | Y | 2 |  |
| uint16 | N | 2 |  |
| int32 | Y | 4 |  |
| uint32 | N | 4 |  |
| int64 | Y | 8 |  |
| uint64 | N | 8 |  |

### 浮点型

|  |  |  |
| --- | --- | --- |
| 数据类型 | 长度 | 说明 |
| float32 |  |  |
| float64 |  |  |
| complex64 |  |  |
| complex128 |  |  |

### 布尔型

|  |  |  |
| --- | --- | --- |
| 数据类型 | 长度 | 说明 |
| bool |  |  |

## 复合数据类型

|  |  |  |
| --- | --- | --- |
| 数据类型 | 定义 | 说明 |
| string |  |  |
| array |  |  |
| slice |  |  |
| map |  |  |
| struct |  |  |
| chan |  |  |
| func |  |  |
| interface |  |  |
| ptr |  |  |

## 指针

## unsafe

|  |  |  |
| --- | --- | --- |
| 数据类型 |  | 说明 |
| uintptr |  |  |
| unsafe.Point |  |  |

在unsafe包中，定义了一个类型类型：

unsafe.Point 是个任意类型数据的指针

和三个函数：

unsafe.Sizeof() 对任意数据类型求尺寸

unsafe.Offsetof() 结构体成员的偏移位置

unsafe.Alignof() 数据类型的对其值

图 1 unsafe.Pointer作用示意图

uintptr

unsafe.Pointer

任意指针类型

* 任何指针数据类型和unsafe.Pointer之间互转
* unsafe.Pointer和uintptr之间互转

unsafe.Pointer 和 unitptr 的主要区别，unsafe.Pointer是golang中的一种通用数据类型，对象取指针到unsafe.Pointer将持有对象，会影响GC的回收。而uintptr底层是int类型，仅仅是一个数值，不持有对象，所指向的对象也可能会被GC回收。

注意：unsafe转换存在风险，请谨慎使用。

|  |
| --- |
| package main  import (  "fmt"  "unsafe"  )  type st struct {  x uint8  y uint32  }  func main() {  data := uint32(0)  fmt.Printf("%p\n", unsafe.Pointer(&data))  fmt.Printf("%#x\n", uintptr(unsafe.Pointer(&data)))  stdata := st{}  fmt.Printf("%d\n", unsafe.Sizeof(stdata))  fmt.Printf("%d\n", unsafe.Alignof(stdata))  fmt.Printf("%d\n", unsafe.Offsetof(stdata.y))  }  0xc420014094  0xc420014094  8  4  4 |

# 过程

## xxxxx

# 函数

## xxxxx

# 接口

## xxxxx

# 并发

## xxxxx

# 性能

## xxxxx

# 包

## xxxxx

# 测试

## xxxxx

# 应用篇

# 命令行

## xxxxx

# 缓冲区

## xxxxx

# 编解码

## xxxxx

# 调教

## xxxxx

# 高级篇

# 反射及对象

## xxxxx

# 内存及Gc

## xxxxx

# 启动及初始化

## xxxxx

# Cgo/调用/回调

## xxxxx

# 命令&工具

## xxxxx