GO 语言面试重点范围：

（保护每个人的个人利益！！绝对禁止外传！！！）

任务一：学习 Golang 合辑视频**（必做）**

线上学习链接如下：https://www.bilibili.com/video/BV1hv411x7we

任务二：阅读 GO 语言相关书籍

必做：《GO 学习笔记第四版》

选读：《深入解析 GO 内核实现》

任务三：checklist 学习**（必做）**

基础 Go-checklist：附件 1：Go-checklist（所有 Go 语言的代码都需遵守此规范）

https://go-zh.org/doc/effective_go.htm

任务四：学习模块设计相关资料（**必做，**因为线下工作中项目经理指导你们工作的时候，API 设计和模块设计部分更加侧重于实践和真实案例的讲解，知识性内容讲解较少）

1、Solid 原则：https://www.cnblogs.com/wuyuegb2312/p/7011708.html

2、设计模式： https://www.runoob.com/design-pattern/design-pattern-intro.html

L

GO 语言设计初衷

设计初衷：为了解决当时 google 开发遇到的问题

1）大量的 c++代码，同时又引入了 python 和 java

2）数以万行的代码

3）分布式的编译系统

4）数百万的服务器

google 开发中的痛点：

1）编译慢

2）失控的依赖

3）每个工程师只是用了一个语言里面的一部分

4）程序难以维护

5）更新的花费越来越长

6）交叉编译困难

如何解决当前的问题和痛点?

设计 go 语言

GO 语言的优点和缺点

优点

并发编程

自动垃圾回收

丰富的标准库

函数多返回值

匿名函数

开发速度相对快

上手容易

自带工具丰富

缺点

部分语法陷阱

缺少成熟框架

内存管理(自动垃圾回收)

GO 的安装配置

下载地址：https://golang.org/dl/

linux：

export GO111MODULE=on

export GOPROXY=https://goproxy.io,direct

export GOROOT=”安装根目录路径”

export GOPATH=”工作区目录路径”

export GOBIN=”放置 go 程序生成的可执行文件路径”

windows：

go env -w GOPROXY=https://goproxy.io,direct

go env -w GO111MODULE=on

set GOOS GOOS=linux

set GOARCH=amd64

常用go工具

go fmt

go vet

go tool pprof

go test

常用库介绍

fmt
time
sync
strconv
encoding/json
strings
context
panic、revonver
runtime

常用库练习一：

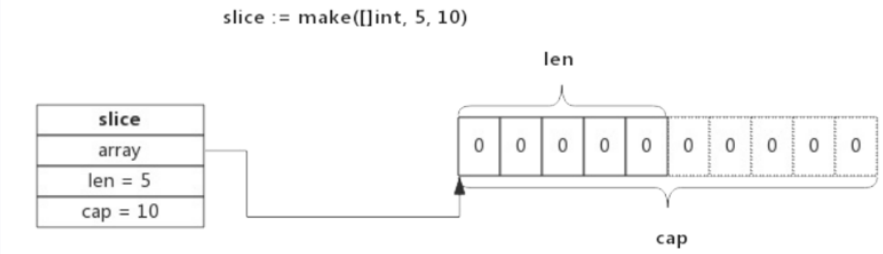
```
fmt.Print("a", "\n")
fmt.Print("a", "b", "\n")
fmt.Print('a', "\n")
fmt.Print('a', 'b', "\n")
fmt.Print(12, "\n")
fmt.Print(12, 13, "\n")
```

golang丰富的内置类型

- Slice：依托数组实现，底层数组对用户屏蔽，在底层数组容量不足时可以实现自动重分配并生成新的Slice。

源码包中 `src/runtime/slice.go:slice` 定义了Slice的数据结构：

```
1. type slice struct {
2.     array unsafe.Pointer
3.     len   int
4.     cap   int
5. }
```

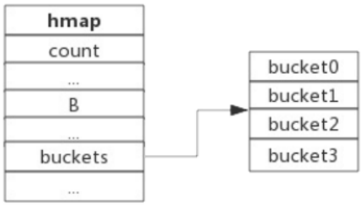


golang丰富的内置类型

- Map：使用哈希表作为底层实现，一个哈希表里可以有多个哈希表节点，也即bucket，而每个bucket就保存了map中的一个或一组键值对。

map数据结构由 `runtime/map.go/hmap` 定义：

```
type hmap struct {
    count     int // 当前保存的元素个数
    ...
    B         uint8  // 指示bucket数组的大小
    ...
    buckets    unsafe.Pointer // bucket数组指针，数组的大小为2^B
    ...
}
```



bucket数据结构由 `runtime/map.go/bmap` 定义：

```
type bmap struct {
    tophash [8]uint8 //存储哈希值的高8位
    data    byte[1]  //key value数据:key/key/key/.../value/value/value...
    overflow *bmap   //溢出bucket的地址
}
```

golang丰富的内置类型

map和sync.map 性能对比

golang丰富的内置类型

sync.map 源码、原理

golang并发控制

goroutine 常用管理和控制

select

```go
func Producer(queue chan<- int) {
    for i := 0; i < 10; i++ {
        queue <- i
        fmt.Println("Producer:", i)
    }
}
func Consumer(queue <-chan int) {
    for i := 0; i < 10; i++ {
        v := <-queue
        fmt.Println("Consumer:", v)
    }
}
func main() {
    queue := make(chan int, 1)

    go Producer(queue)
    go Consumer(queue)

    time.Sleep(10)
}
```

golang并发控制

Channel: 使用channel控制子协程

WaitGroup：使用信号量机制控制子协程

Context: 使用上下文控制子协程

golang常见坑

map[string]map[string]chan bool

切片

range

sync.WaitGroup 拷贝

string 和 slice

go defer func

golang并发案例

案例：使用go的并发查找素数