操作系统：Linux/Mac OSX

- 系统管理命令

```
ps M <pid> ：列出pid进程下的所有线程
```

- 进程/线程/协程

```
线程：同步原语 Lock RLock Semaphore(信号量)
with lock 语句更优雅安全(忘记释放 或在执行异常时都能释放锁)

一个线程在需要获取多个锁时，解决死锁问题的一种方案是为程序中的每一个锁分配一个唯一的id，然后只允许按照升序规则来使用多个锁

保存线程的状态：threading.local() 实例为每个线程维护着一个单独的实例字典。 所有普通实例操作比如获取、修改和删除值仅仅操作这个字典。 每个线程使用一个独立的字典就可以保证数据的隔离了。

难点：线程同步 控制多个线程的执行顺序。

原子操作：硬件CPU原子操作(独条指令、关中断、锁L1/L2缓存、锁内存总线)
锁：利用硬件原子操作，读取、设置内核对象，该对象有加锁、空闲两种状态。
```

- File I/O

```
文件描述符(文件、管道、设备的抽象 0开始的整数表示 指向`文件`对象)：
    进程获取文件描述符方式：打开文件、目录、设备；创建一个管道；复制已存在的描述符。
    系统调用close会释放一个文件描述符，使得它未来可以被open, pipe, dup等调用重用。
    一个新分配的文件描述符永远都是当前进程的最小的未被使用的文件描述符。
    按照惯例，进程从文件描述符0读入（标准输入），从文件描述符1输出（标准输出），从文件描述符2输出错误（标准错误输出）
IO多路复用 select/poll/epoll 模型
```

容器Docker

```
应该保证在一个容器中只运行一个进程。将多个应用解耦到不同容器中，保证了容器的横向扩展和复用。例如 web 应用应该包含三个容器：web应用、数据库、缓存。
```

网络协议:HTTP/Websocket/TCP

- Web 服务器 Nginx
- Http 事务

```
  HTTp:
      http1.0 content-length, http1.1 content-length、Transfer-
encoding: chunked.
      如果客户端知道body长度，则客户端先断开TCP连接，
      否则客户端一直接受数据直到服务端主动断开连接。
  Https:
      服务端：CA + 服务端公钥 + 数字签名
      客户端：CA认证 生成随机对称加密秘钥并使用服务端公钥加密
      服务端：用私钥解密对称秘钥 双方用该对称秘钥通信
```

- connect close/keep-alive
- TLS/SSL 传输层加密
- I/O 多路复用 select/epoll
- TCP 报文

数据库： PgSQL/MySQL/SQLite/Redis

- c/s 数据库与嵌入式数据库

```
  c/s数据库：数据库客户端通常通过数据库驱动程序如JDBC(psql/mysql/sqlserver)、
ODBC(oracle)等访问数据库服务器，数据库服务器再操作数据库文件。
  数据与应用程序分离 通过TCP/IP通讯 便于对数据访问的控制和管理(可以是非开法人员
DBA管理控制)

  嵌入式数据库：不需要数据库驱动程序，通过API访问数据库
  数据库与应用程序部署在一起 访问控制完全交给应用程序
```

- 事务/事务隔离级别
- 索引

```
  索引：类比书本目录，系统先查找索引，找到相应记录所在的磁盘块，然后读取磁盘块，取出
数据。
  索引结构与特定的搜索码关联
  索引项：搜索码值和指向具有该搜索码值的一条或多条记录的指针结构(磁盘块标识和块内记
录偏移量)
  稠密索引/稀疏索引 查询效率/节省空间 折中：每一个块(数据存储单元)建立索引项的稀疏
索引
  顺序索引：按顺序存储索引码值
```

```
散列索引：索引码值平均分布到若干个散列桶中
多级索引：外/内层
```

- 并发控制
- SQL标准支持/系统自定义SQL

Web后端框架：Flask/Django

- WSGI-- Web Server Gateway Interface

```
  Python语言定义的Web服务器和Web应用程序或框架之间的一种简单而通用的接口。这是一个规范，描述了web server如何与web application交互、web application如何处理请求，该规范的具体描述在PEP3333。
  WSGI server: Gunicorn/uwsgi
  WSGI app: flask app/ django app
  Werkzeug: WSGI 工具集，路由处理、request/response 封装处理、自带的开发测试 WSGI server。Python Web后端框架底层库
```

异步消息：Celery

工具：Docker/Jenkins/Git

- 容器/镜像 linux alpine

```
容器：镜像运行的逻辑单元
镜像分成构建
```

- 数据卷volumes

```
容器数据持久存储、主机文件目录与容器文件目录的映射。
```

- networks 容器之间通信
- docker-compose：Docker 官方容器编排工具

前端框架：Bootstrap/JQuery/Vue

前端工具：Bower/Gulp/SaSS

区块链：以太坊/智能合约

Python 基础

文件操作(大文件)

read(all lines) readline(line) readlines(line list)

for line in f其实是将文件对象f视为一个迭代器，自动的采用缓冲IO和内存管理，所以不必担心大文件

递归打印目录中的文件路径(包含子目录里面的文件路径)

数组切片

```
list[<start>:<stop>:<step>]
反转数组 list[::-1]
```

集合相同不同元素

```
l1 = [1,3,5]
l2 = [2,3,6]
set1 = set(l1)
set2 = set(l2)
set1 & set2
set1 ^ set2
```

内置数据类型

```
int float str list tuple set dict
```

元类： type 是 python的元类，元类是用于创建类对象(class)的类

```
    class xxx(type):
        __call_():
            pass

    class Foo(metaclass=xxx):
        pass
```

可变类型、不可变类型

```
    list dict 传递内存地址 指向的内存中的值可以被改变 一般不需要开辟新内存
    number str tuple 值改变会开辟新的内存
```

函数作用域的LEGB顺序

```
    L:  local 函数内部作用域
    E: enclosing 函数内部与内嵌函数之间
    G: global 全局作用域
    B:  build-in 内置作用
```

装饰器 使用 functools wraps

```
    避免被装饰的函数/类 名称、属性被改变。
    本质上是修改函数的属性
    def my_decorator(func):
        @wraps(func)
        def wrapper(*args, **kwargs):
            if xxx:
                return func(*args, **kwargs)
            else:
                do something
        return wrapper
```

闭包

```
    在函数内部再定义一个函数，并且这个函数用到了外边函数的变量，那么将这个函数以及用到的一些变量称之为闭包。
```

迭代器、生成器

只读属性

```
class MyCls(object):
    __weight = 50

    @property
    def weight(self):
        return self.__weight
```

golang

语言特性

```
没有类和继承的概念 通接口实现多态
类型安全?(不允许隐式类型转换)
垃圾回收自动内存管理
```

不支持函数重载和操作符重载
不支持动态加载代码、动态链接库、泛型
不支持断言、静态变量
参数都是值传递(值拷贝传递，即使是指针，也是拷贝的指针的值(指向具体值的地址))

go 需要运行时(runtime)
可以实现go 解释器

包pkg(包含.go文件) main 包 可独立执行的程序 小写字母

反射、类型断言