# 操作系统导论

## 多道批处理系统

作业调度：作业现场的保护和恢复

资源共享：资源的竞争和同步 --互斥机制

内存使用：提高内存使用效率

内存保护：系统存储区和应用存储区不可冲突

文件系统管理：文件随机存放

## 分时操作系统

分时：两个或两个以上的事件按时间划分轮流使用计算机系统中的某一资源

分时系统:多个用户分时的使用同一计算机的系统

特点：

同时性或多路性：

独占性：

及时性：

交互性：

调进和调出：

主存-后援存储方式：

内存只存放一个当前执行作业，其他放在外存上

多流调入调出方式：

主存中同时存放一个当前执行作业和多个后备作业

分时系统的相应时间：

一条指令从开始到响应所需要的时间

与当前系统的用户数和时间片的大小有关

## 实时操作系统：

主要用于过程控制、事务处理

要求实时性和可靠性

## 进程

定义：一个进程就是一个程序针对某个数据集合（处理对象）的一次执行过程

#### 程序和进程的关系：

程序是静态的，进程是程序的一次运行活动，是一条条指令按顺序执行的过程

程序一般是长期存在，进程占用cpu和存储空间，存在时间很短

一个程序可以被多个进程同时执行，一个进程可以包含多个程序

### 中断技术和通道技术：

#### 通道：

是一台结构简单的IO处理机

与cpu单独通信，直接控制外部设备，与内存进行数据传输

#### 中断：

Cpu收到外部中断信号，停止原来工作，转去处理中断事件，完毕后回到原来断点继续 工作

中断请求、中断响应、中断点、中断处理例程、中断返回

操作系统的首要任务：管理计算机中的硬件和软件资源

跟踪资源状态

分配资源

回收资源

保护资源

进程管理是操作系统的核心工作

操作系统功能：

进程管理

作业和用户界面管理

存储管理

设备管理

文件管理

## 用户与操作系统：

操作系统给用户操作计算机提供的两种接口

系统命令接口：

完成用户作业的组织和控制

系统调用接口

系统提供给开发人员的接口

作业及其管理：

作业:用户在一次事务中要求计算机工作的共和

作业和进程的关系

每个作业拥有一个独立的根进程和多个子进程

作业是用户提交任务的实体

进程是任务的执行实体

进程管理：

一个进程从建立到结束的过程的管理

作业管理：

对整个作业的组织、调度和控制

作业的处理流程：

提交 后备作业 作业调度（进程调度） 完成

作业调度：

根据当前系统资源，从后备作业队列中选取一个合适作业投入运行

吞吐量：单位时间里处理的作业个数

作业调度算法评价因素:

吞吐量 资源利用率 对各作业的公平程度

作业调度算法：

先来先服务（FIFO）

优点：简单 公平

缺点：没考虑资源利用率和特殊作业

短作业优先：

易于实现，强调资源利用率，保证吞吐量

不公平。长作业长期等待

高响应比优先： 前两者的结合

公平、吞吐量大

增加计算，增加开销

高优先级优先

资源的均衡调度

系统调用：

操作系统提供给软件开发人员的唯一接口

## 进程的控制

进程的顺序执行和并发执行

顺序执行：

顺序性，封闭性、可再现

并发执行

异步、不封闭、不可再现、相互竞争

进程的概念：

程序的一次执行，是系统进行资源分配和调度的独立单位

线程的特征：

动态性

并发性

独立性和异步性

相互制约

结构性

线程的组成

程序

数据集合

进程控制块：（PCB Process Controller Block）

进程的描述信息和控制信息

进程控制块的内容：

进程描述信息

进程控制信息

资源占用信息

Cpu线程保存结构

## 线程

是一个执行过程、处理器调度的基本单位

使用线程的优点：

一个进程有多个线程 效率高

资源共享，一个进程，各个线程共享资源

减少系统开销

线程和进程的关系：

线程是进程的组成部分

资源分配给进程， 线程共享进程资源

处理器调度的基本单位是线程

进程调度程序：

按照调度算法，处理机处理线程

程序调度算法设计：

非抢占式： 进程执行时，进程结束才开始下一个进程

抢占式： 进程执行时，系统根据算法把cpu给其他进程

进程调度算法的评价标准：

吞吐量： 单位时间内的完成作业数

处理机利用率

设备的均衡利用

面向用户的调度性能准则

响应时间：请求-系统给出响应

周转时间：作业提交到结束的时间

截止时间：开始截止时间，完成截止时间

公平性：

优先级：

进程调度算法：

先来先服务

短作业优先

优先权调度：

高响应比优先

时间片轮转

多级反馈队列

## 进程线程与锁

### 死锁：

定义：若干进程相互等待被对方占有的资源

原因：多个线程，某些去申请资源时，资源被占用，自己的资源无法释放

死锁的条件：

资源互斥（独享）

资源不可抢占

资源部分分配等待（占有并等待）

循环等待

### 进程的同步与互斥

相关进程

功能、逻辑彼此有某种联系

无关进程

功能、逻辑没有联系

间接制约

共享同一资源而产生的制约关系

直接制约

进程之间按照一定的顺序去执行

同步与互斥

广义同步：保证进程之间协调一致

狭义互斥：线程合作完成任务时，有一定的先后顺序

互斥：使用共享资源时，资源一个时刻只能被一个线程使用

临界资源：每次只允许一个进程使用的资源

临界区：程序中涉及临界资源的程序段

临界区使用的几个原则

无空等待

有空让进

多中选一

有限等待

让权等待

进程互斥的一种实现方法：

平等协商方式：

进入临界资源设置一些标志，退出时标志该为原始状态

申请进入临界资源的进程进入之前等待，直至其他资源退出临界区

### 进程通信

低级通信：

进程间传递的信息仅为一个信号，使用低级通信原语对这个信号进行处理

高级通信：

进程间传递的信息量比较大，

信号量：

是一个数据结构，包括一个整型变量（S）和一个与之关联的进程队列(queue)

S>0 表示有资源可以分配

S<0 表示该资源被占用完

低级通信的原语

P： 加锁

S值减一

If(S>=0){

进程继续执行

}else {

进程进入等待状态，并进入等待队列

}

V：释放已持有资源

S值+1

If( S<=0) {

等待队列的第一个进程获取资源，继续执行

}

PV原语的互斥分析：

同一个信号量的P、V操作必须成对出现

同一个信号量的操作在一个线程内，

P操作在进入临界区之前

V操作在退出临界区之后

PV原语的同步分析：

同一个信号量的P、V操作必须成对出现

同一个信号量的操作出现在两至多个线程中

P操作出现在被动的，进程需要等待对方信号通知的那个线程

V操作出现在主动的，让对方等待自己通知的那个进程

### Window 中的同步与互斥

进程间的高级通信：

进程间传递较大量信息的机制

场景的高级通信：

消息： 进程间传递的一组信息

消息缓冲通信

直接的通信方式

信箱通信

间接的通信方式

一个“信箱”可以存放若干信件

管道

供两个进程共享的文件

一端负责写入一端负责读取

# Redis

### SDS（simple dynamic String）：简单动态字符串

Redis 底层所使用的的字符串表示

用途：

实现字符串对象

在redis程序内部用做char\*类型的替代品

小结：

Redis 的字符串表示为 sds ，而不是 C 字符串（以 \0 结尾的 char\*）。

对比 C 字符串，sds 有以下特性：

– 可以高效地执行长度计算（strlen）；

– 可以高效地执行追加操作（append）；

– 二进制安全；

sds 会为追加操作进行优化：加快追加操作的速度，并降低内存分配的次数，代价 是多占用了一些内存，而且这些内存不会被主动释放

### 双端链表：

是 Redis 列表结构的底层实现之一

小结：

Redis 实现了自己的双端链表结构。

双端链表主要有两个作用：

– 作为 Redis 列表类型的底层实现之一；

– 作为通用数据结构，被其他功能模块所使用；

双端链表及其节点的性能特性如下：

节点带有前驱和后继指针，访问前驱节点和后继节点的复杂度为 O(1) ，并且 对链表的迭代可以在从表头到表尾和从表尾到表头两个方向进行；

– 链表带有指向表头和表尾的指针，因此对表头和表尾进行处理的复杂度为 O(1) ；

– 链表带有记录节点数量的属性，所以可以在 O(1) 复杂度内返回链表的节 点数量（长度）；

# 数据结构：

数据的逻辑结构、物理结构

逻辑结构：元素之间的逻辑关系

集合、线性结构，树状结构、网状结构

物理结构：在计算机中的存储方法（表现和实现）

数据类型：

一个值得集合和定义在该集合的操作

抽象数据类型：

数据类型+定义在该结构上的操作

表示(D、S、P)

D:数据

S：D上的关系

P:D上的基本操作

算法：

问题求解步骤的一种描述

时间复杂度：

算法执行时间随规模增长而增长的趋势

空间复杂度：

算法执行过程所需的最大空间

线性表：

N个数据元素的有限序列

除却第一个和最后一个元素，没个元素都有唯一一个前驱元素和后继元素

# 名词解析

token：

登录凭证、令牌

JWT（Json Web Token）： 同类：OAUTH2

是JSON风格轻量级的授权和身份认证规范，可实现无状态、分布式的Web应用授权；

官网：https://jwt.io

非对称加密 RSA

原理：同时生成两把密钥：私钥和公钥，私钥隐秘保存，公钥可以下发给信任客户端

私钥加密，持有私钥或公钥才可以解密

公钥加密，持有私钥才可解密

优点：安全，难以破解

缺点：算法比较耗时

对称加密 AES

基本原理：将明文分成N个组，然后使用密钥对各个组进行加密，形成各自的密文，

最后把所有的分组密文进行合并，形成最终的密文。

优势：算法公开、计算量小、加密速度快、加密效率高

缺陷：双方都使用同样密钥，安全性得不到保证

不可逆加密：eg:MD5、SHA

无状态登录：

服务端不保存任何客户端请求者信息

客户端的每次请求必须具备自描述信息，通过这些信息识别客户端身份

好处？

- 客户端请求不依赖服务端的信息，任何多次请求不需要必须访问到同一台服务

- 服务端的集群和状态对客户端透明

- 服务端可以任意的迁移和伸缩

- 减小服务端存储压力

有状态登录：

服务端需要记录每次会话的客户端信息，从而识别客户端身份

根据用户身份进行请求的处理

缺点？

- 服务端保存大量数据，增加服务端压力

- 服务端保存用户状态，无法进行水平扩展

- 客户端请求依赖服务端，多次请求必须访问同一台服务器

axios 类似ajax的异步查询工具

# 简单的描述

操作系统：

是一个系统软件

作用：

操作计算机硬件

为程序员调用计算机硬件提供系统调用（操作计算机硬件）

虚拟机：

是一个软件

作用：

模拟一台计算机，帮助学习新的系统