操作系统

1 概述

1.1 操作系统的特点和功能 ※※※

特点：**并发**（进程）、**共享**（系统资源供并发进程使用）、**虚拟**（虚拟多个处理器，虚拟存储器）、**异步**（进程速度不可预知）

- **并发和共享是最基本的特征**：资源共享依赖于并发，并发需要资源共享

功能：进程管理，内存管理，文件系统管理，设备驱动和输入输出管理，提供用户界面

1.2 并发和并行 ※※

并发：多个任务在**同一时间间隔**内执行。宏观上同时执行，微观上分时交替执行

并行：多个任务在**同一时刻**执行。需要具备多个处理器，每个任务互不干扰

1.3 中断和系统调用的区别 ※※

（CPU指令外部）**中断**：包含可屏蔽中断（INTR线）和不可屏蔽中断（NMI线，例如电源掉电等硬件紧急故障）

（CPU指令内部）**异常**：包含故障（缺页、除0、溢出等），自陷（系统调用），终止（控制器异常）

系统调用：操作系统提供给应用程序使用的接口，可从用户态向内核态完成转换

2 进程与线程

2.1 进程和线程的概念和区别 ※※※※※※

进程：计算机中正在运行的程序的实例，**每个进程有自己独立的地址空间**，包括**代码、数据、堆栈**等；不会共享内存

线程：是进程中的一个执行单元，进程可包含多个线程，共享相同的地址空间和资源。**线程间通过共享变量来通信和同步**

区别：

- 进程是**资源分配**的基本单位，线程是**调度**的基本单位
- 进程间可并发，同一进程间的线程可并发，不同进程的不同线程亦可并发
- 进程并发比线程并发**系统开销大，切换慢**

2.2 进程的状态，进程状态间的转换 ※※※※

进程的基本状态：就绪、运行、阻塞，此外还有创建态、终止态

- **就绪态**：进程获得了除CPU以外所有需要的资源，一旦获得CPU即可立即执行
- **运行态**：进程正在CPU上执行
- **阻塞态**：进程正在**等待**某一事件（例如等待IO完成）而放弃CPU（即使CPU空闲），**暂停**运行

进程状态间的转换：

- 就绪 → 运行：进程被调度，获得CPU资源
- 运行 → 就绪：进程时间片用完，让出CPU
- 运行 → 阻塞：请求资源使用和分配，或等待事件（如IO）完成，是主动行为
- 阻塞 → 就绪：IO完成或中断结束

2.3 进程调度策略，进程调度性能指标 ※※※※※

进程调度策略：

1. **先来先服务**（FCFS）：每次从后备队列中选择**最先进入**的作业，直至完成或阻塞才释放CPU
    - 实现**简单**，**但不利于短作业。**不可抢占（暂停一个正在执行的进程而交给其他进程）
2. **短作业优先**（SJF）：每次调度前选择当前后备队列中**执行时间最短的作业**
    - **平均等待、周转时间最优**；但长时间会产生**饥饿现象**（长作业长时间得不到调度）。适用于批处理系统。
3. **优先级调度**：设置优先级，优先级可动态变化，可抢占
4. **高响应比优先**：引入**响应比**：$R_p = (\text{等待时间} + \text{要求服务时间}) / \text{要求服务时间}$
    - 综合FCFS和SJF，兼顾长短作业，但计算响应比开销大
5. **时间片轮转**（RR）：FCFS排成队列，**规定一个时间片长度**，执行完则时钟中断，当前程序排至队尾
    - 兼顾长短作业，但**平均等待时间长**，且**上下文切换浪费时间。**适用于分时系统
6. **多级反馈队列**：设置多级队列，优先级递减，每个队列时间片递增。每个队列都用FCFS，按优先级调度队列，一个时间片没执行完的任务放在下一个队列的队尾
    - 无需知道进程执行时间，兼顾长短作业，相当通用，但最复杂

进程调度性能指标：

1. **CPU利用率**（忙碌时间/总时间）
2. **系统吞吐量**（单位时间CPU完成作业的数量）
3. **周转时间**（作业**提交到完成**经历的时间），平均、带权（周转时间 / 实际运行时间）、平均带权周转时间
4. **等待时间**：进程处于等待被服务的时间之和
5. **响应时间**：用户提交请求到首次产生响应所用时间

2.4 进程间通信方式 ※※※※

1. **共享存储**：两进程共享**数据结构**（低级，如队列）或**存储区**（高级，如内存中划分共享区），借助这些空间进行通信
2. **消息传递**：以格式化的消息为单位，封装通信的数据，利用OS提供的一组原语（发送、接收）在进程间进行数据交换
3. **管道通信**：连接读写进程之间通信的共享文件，必须用被互斥地访问（半双工方式），必须进行同步（写满后才读，或读空后才写）

2.5 死锁，产生原因，产生死锁的必要条件 ※※※※※

> **同步**：多个进程运行次序的直接制约；**互斥**：两进程都使用临界资源，需要阻塞其中一个直到释放
>
> **临界区**：访问临界资源（一次仅允许一个进程使用）的代码
>
> **临界区互斥的准则**：空闲让进，忙则等待，有限等待，让权等待（非必须，进程不能进入临界区则释放CPU）
>
> 实现临界区互斥的方法：软件（Peterson），硬件（中断屏蔽，TestAndSet，Swap）
>
> **信号量**：P（wait，相当于进入区），V（signal，相当于退出区）

死锁定义：多个进程因为**竞争资源而造成相互等待**的情况，无外力干涉则无法向前推进

产生原因：系统资源（**不可剥夺资源**）的竞争，进程推进顺序非法

死锁产生的必要条件：缺一不可

1. **互斥条件**：一段时间内某资源仅为一个进程占有（排他性使用）
2. **不可剥夺条件**：进程使用完资源前不可被其他进程夺走
3. **请求并保持条件**：进程已经保持至少一个资源，又提出新的资源请求
4. **循环等待条件**：死锁时必然有一个进程资源的循环等待链

2.6 解决死锁，预防死锁 ※※※※※

1. **预防**：破坏四种必要条件之一
    - 不必剥夺，但效率低
2. **避免**：银行家算法
    - 不必剥夺，但必须知道将来的资源需求
3. **检测与解除**：系统状态的资源分配图可完全简化则无死锁；解除需要撤销进程
    - 允许对死锁现场处理，但需要剥夺接触死锁

2.7 银行家算法 ※※※

用于解决资源分配问题和预防死锁

- **安全序列**：系统能按某种进程推进顺序为每个进程分配所需资源

申请资源时**先看资源是否足够**；然后试探是否会使系统**处于安全状态**，若安全则分配资源，否则不分配资源并等待进程释放资源

2.8 哲学家进餐 ※※※

五个哲学家五根筷子，随机开始拿筷子吃饭，目的是防止死锁。实现方式有：

- 至多n-1名哲学家同时进餐，使至少一人能拿到两根筷子
- 仅当一个人两边筷子都能用时才拿筷子
- 编号，奇数哲学家先拿左边的，偶数哲学家先拿右边的

3 内存管理

3.1 内存管理的功能 ※※※

1. **内存的分配与回收**
2. **地址转换**：逻辑地址转化为物理地址
3. **内存保护**：保证各个进程在各自存储空间运行，互不干扰
4. **内存共享**：允许多个进程访问内存同一部分
5. **内存扩充**：虚拟存储**在逻辑上**扩充内存

3.2 内存空间的分配 ※※※※※

连续分配方式：

- **单一连续分配**：系统区+用户区，仅有一道用户程序。
  - 无外碎片，但仅适用单用户单任务，利用率极低
- **固定分区分配**：把用户内存空间划分为固定大小分区，每个分区给一个进程。
  - 无外碎片，但有内碎片，利用率低
- **动态分区分配**：进程装入内存时，根据进程实际需要动态分配空闲内存块
  - 无内碎片，但有外碎片。分配和回收算法见3.3和3.4

非连续分配方式：

- **分页存储**：将**物理内存**和**进程的逻辑地址空间**分成**大小固定**的页（如4KB），从而将进程的页映射到物理内存的任意位置
  - 可以减少外碎片，但需要两次访问内存：页表一次，取数据一次。可以增加快表TLB
  - 逻辑地址=页号+页内偏移，页表项=块号（页号隐含）
- **分段存储**：将**进程的逻辑空间**划分为多个逻辑段，每个逻辑段映射到物理内存的任意位置
  - **便于实现共享**，只需设置一张共享段表即可（相较而言，分页存储需要每个进程的页表设置N个页表项）
  - 逻辑地址=段号+段内偏移，段表项=段长+基址（段号隐含）
- **段页式存储**：把进程的逻辑地址空间分为多个逻辑段，每个段划分为固定大小的页

- 逻辑地址=段号+页号+页内偏移，段表项=页表长度+页表始址，访问需要**访存三次**

3.3 动态分区分配算法 ※※※※※

基于顺序搜索：

1. **首次适应算法**（First Fit）：空闲内存块列表中**从头**顺序查找**第一个能满足大小**的空闲分区。有利于大作业，性能最好
2. **邻近适应算法**（Next Fit）：**从上次查找结束的地方**开始查找，高低地址空闲分区分配概率同等，性能较差
3. **最佳适应算法**（Best Fit）：找到空闲内存块列表中最小且满足大小的空闲分区。碎片最多，利于大作业
4. **最差适应算法**（Worst Fit）：找到空闲内存块列表中最大的空闲分区。碎片最少

基于索引搜索：大中型系统采用，根据大小对空闲分区分类，每一类单独设置空闲分区链

- **伙伴系统**：所有分区**大小为2的幂。**分配时先找满足大小的最小2^i的分区链，若存在空闲分区则分配，否则查找2^{i+1}大小，若存在，则**等分为两个分区（伙伴），一个分配，一个加入空闲分区链。回收时需要将伙伴合并。**

3.4 动态分区回收算法 ※※※

> 实际上就是链表合并问题，准备插入新链表项看是否有必要新建。有四种情况

1. 前邻空闲区，合并，修改前空闲区长度
2. 后邻空闲区，合并，修改后空闲区长度和首地址
3. 前后均邻空闲区，合并，修改长度和首地址，取消后分区表项
4. 均不相邻，新建链表项并插入

3.5 分页和分段的区别 ※※※※※

分页：

- 页是信息的**物理单位**，**目的是提高内存利用率**，分页对用户**不可见**
- 页的大小**固定**，且由系统决定
- 页的地址空间是一维的，只给页号即可求出页内偏移

分段：

- 段是信息的**逻辑单位**，**目的是更好满足用户需求**，分段对用户**可见**
- 段的大小**不固定**，且取决于用户程序
- 段的地址空间是二维的，需要显式给出段号和段内偏移

3.6 虚拟内存，共享内存 ※※※

虚拟内存：在逻辑上**扩展物理内存容量**，允许把一个作业**分多次调入内存**，通过**请求调页**把页面从外存调入内存，通过**页面置换**把暂时不用的页面换到外存。优点：

- 提供了比物理内存更大的地址空间。空间大小只由虚拟地址的位数决定
- 允许多个进程并发执行
- 简化了内存管理

共享内存：允许多个进程访问同一物理内存区域，实现数据的**共享和交换**

3.7 页面置换算法 ※※※※※

1. **最佳置换算法**（OPT）：每次淘汰**以后最长时间内不再访问**的页面。是最理想的，但无法实现，可作为算法好坏的衡量标准
2. **先进先出置换算法**（FIFO）：每次淘汰**最先进入内存**的页面。保证公平性，但会出现Belady现象，即内存空间增大时，页命中率不降反升
3. **最近最久未用置换算法**（LRU）：每次淘汰**最近最久未被使用**的页面，记录每个页面自上次访问以来经过的时间
4. **时钟置换算法**（Clock）：为每个页面设置访问位，被访问则置为1，每次淘汰访问位为0的页面。每次淘汰时循环检查访问位，为1则置0，为0则淘汰。该算法给了页面留存内存的第二次机会
5. **改进时钟算法**：访问位+修改位，四种组合，00最先淘汰，01次佳。优势在于可减少磁盘IO操作，但增加了实现算法本身的开销

4 文件系统的组织 ※※※

- **逻辑结构**：分为无结构文件和有结构文件。
  - 无结构文件是有序字节流，没有结构，用读写指针指出下一个要访问的字节
  - 有结构文件在逻辑上被视为一组连续记录的集合。分为定长记录文件和不定长记录文件
- **有结构文件**：
  - **顺序文件**：一系列记录**按照某种顺序排列**所形成的文件
  - **索引文件**：为变长记录建立一张**索引表**，每个记录设置一个表项，从而加速对记录的检索
  - **索引顺序文件**：只为**一组记录的第一个记录**建立一个**索引项**
- **物理结构**：与逻辑结构不同。磁盘中的存储单元是一个个磁盘块，常与页面大小相同
  - 连续分配：每个文件占有磁盘上一组连续块，只需访存一次但有外碎片，且无法动态增长
  - 链接分配：链接各物理块，无外碎片，但只能顺序访问
  - 索引分配：为每个文件分配一个索引块，所有盘块号记录在索引中。可随机访问，但增大开销

5 IO管理

5.1 磁盘调度算法 ※※※※

1. **先来先服务**（FCFS）：公平，根据访问磁盘的先后顺序调度
2. **最短寻道时间优先**（SSTF，Shortest Seek Time First）：每次调度离当前磁头最近的磁道，**使每次寻道时间最短。**可能产生饥饿现象
3. **扫描算法**（SCAN）：先满足一个方向所有请求，再调头满足另一个方向的请求。可以提前调头（LOOK）
4. **循环扫描算法**（C-SCAN）：**磁头只能单向移动**，到头快速回到起始端，可以提前到起始端（C-LOOK）