OS 期末复习

Karry

题型：

- 填空题（小标题一级的知识点） < 20 分 = 0.5 * 40 >
  > 重视起来，要认识基础知识点
- 单选题（） < 30 分 = 2 * 15 >
- 解答题 < 50 分 = 10 * 5 ~ 6 道 >

比较常规：

- 访问存储器 在什么样的情况下需要多少次访存 必考点
  > 可能产生的访存的次数

1 第一章

1. 操作系统的作用
   - OS 作为用户与计算机硬件系统之间的接口
   - OS 作为计算机系统资源的管理者
   - OS 实现了对计算机资源的抽象
2. 推动操作系统发展的动力
   - 不断提高计算机资源的利用率
   - 方便用户
   - 硬件不断更新
   - 计算机体系结构在不断发展
   - 不断提出新的应用需求
3. 各种操作系统的特征
   - 基本的一些特征
     - 方便性 —— 易学易用
     - 有效性 —— 利用率高 + 吞吐率高
     - 可扩充性 —— 能不断地加新的模块（模块化）
     - 开放性 —— 有和国际统一的标准
   - 单道批处理的优点缺点
     - 一定程度上解决人机矛盾、CPU与I/O设备速度不匹配的矛盾
     - 资源得不到充分的利用（CPU、内存）（一个程序在等待IO的时候 CPU 空闲不能得到任何利用）

- 多道批处理的优缺点
    - 资源利用率高、系统吞吐量大
    - 平均周转时间长、无交互能力
- 分时系统的特征是什么
    - 多路性：多个用户共享主机
    - 独立性：独立的环境和其他使用者互不干扰
    - 及时性：用户请求能够很快得到反馈
    - 交互性：人机可以广泛对话
- 实时系统|基本定义、类型、特征|
    - 所谓“实时”是表示“及时”，系统的正确性，不仅由计算的逻辑结果来确定，而且还取决于产生结果的时间。将时间作为关键参数
    - 类型
        - 工业（武器）控制系统
        - 信息查询系统
        - 多媒体系统
        - 嵌入式系统
    - 特征
        - 多路性
        - 独立性
        - 及时性
        - 交互性
        - **可靠性**
- 操作系统的基本特征
    - 并发
    - 共享
    - 虚拟
    - 异步

4. 操作系统的主要功能
    - 处理器的管理功能
    - 存储器的管理功能
    - 设备管理功能
    - 文件管理功能
    - 操作系统与用户之间的接口
5. 操作系统几种常见的结构
    - 无结构

- 模块化
- 分层
- 微内核 是现代广泛应用的
  - 微内核结构的优点和缺点

2 第二章

1. 程序的执行顺序

   在执行时，都需要按照某种先后次序顺序执行，仅当前一程序段执行完后，才运行后一程序段。

2. 程序顺序执行的特征
   - 顺序性
   - 封闭性
   - 可再线性
3. 程序并发执行的特征（给代码分析出来组合结果）<汤老师上课有过例子>（可能有几种结果？）
   - 间断性
   - 失去封闭性
   - 不可再现性
4. 进程的基本概念

   为了能使程序并发执行，并且可以对并发执行的程序加以描述和控制，引入“进程”，为了使参与并发执行的每个程序都能独立地运行，在操作系统中必须为之配置一个专门的数据结构，称为PCB，程序段 + 相关数据段 + PCB = 进程

5. 进程的基本特征
   - 动态性
   - 并发性
   - 独立性
   - 异步性
6. 线程的基本特征
7. 进程的三状态以及三状态之间的转化
   - 就绪：没有 CPU其他所有的都有了
   - 阻塞：从CPU上滑下来了

- 执行：在CPU上运行

8. 进程的五状态以及五状态之间的转化
    - 创建状态：PCB 初始化
    - 终止状态：等待操作系统进行善后处理，最后将其PCB清零，并将PCB空间返还系统
9. 进程控制块中含有的信息
    - 为了便于系统描述和管理进程的运行，在OS的核心未每个进程专门定义了一个结构 —— PCB 用于描述进程的当前情况以及管理进程的全部信息
        - 作为独立运行基本单位的标志
        - 能实现间断性运行方式
        - 提供进程管理所需要的信息
        - 提供进程调度所需要的信息
        - 实现与其他进程的同步与通信
    - 含有的信息;
        - 进程标识符：外部标识符 + 内部标识符
        - 处理机状态：处理机种各种寄存器中内容组成的（现在硬件到底是个上面状态）
        - 进程调度信息：状态 + 优先级 + 进程调度所需的其他信息 + 事件 + 其他信息
        - 进程控制信息（怎么样控制一个进程）：程序和数据的地址 + 进程同步和通信机制 + 资料清单 + 链接指针
10. 什么叫做进程的挂起和激活
    - 挂起操作引入的原因：为了系统和用户观察和分析进程的需要。
        - 终端用户的需要：发现有问题，希望使程序暂停使其停止以便用户观察 debug
        - 父进程的需求：父进程有时希望挂起自己的某个子进程，以便考查和修改或者协调各个子进程之间的活动
        - 负荷调节的需要：实时操作系统的工作负荷很严重了，可能影响到系统的运行了，可以先把不重要的进程挂起，保证系统能正常运行
        - 操作系统的需要：操作系统有时希望挂起某些进程，以便检查运行资源使用情况和几张
11. 进程的同步
    - 基本概念

        - 是对多个相关进程在执行次序尚进行协调，使并发执行的诸进程之间能按照一定的规则共享系统资源，并能很好地相互合作，从而使程序的执行具有可再现性。
    - 两种制约关系
        - 同步（直接相互制约）有相互合作关系<生产者消费者>
        - 互斥（间接相互制约）由于共享资源所导致的相互制约关系<打印问题>
    - 什么叫临界资源
        - 就是诸如数据、内存、设备硬件等资源
    - 什么叫临界区
        - 每个进程中访问临界资源的那段代码称为临界区
    - 进程同步应当遵循的规则
        - 空闲让进<临界资源空闲 允许别人使用>、忙则等待<临界资源繁忙 等待着不要进去>、优先等待<总要有个头>、让全等待<等的时候让别人干事情 不要赚着资源不放手>
12. 信号量（重点！！——要重视 可能是一道写程序的大题）

    关中断、Test-and-Set指令实现互斥、使用Swap 指令实现进程互斥——让全等待不满足

    - 整形信号量、记录形信号量最基本的原理
        - 找一个整型的值来表示有多少信号量
        - 有一个记录 + 一个列表（可以 block 列表中的内容）
        - AND 型信号量 => 一次有申请多种信号
        - 信号量集 => 一次申请多个多种信号
    - 如何用信号量解决一些具体的问题（如何解决前驱关系 有向图 => 信号量的语言<书上有的 学会表达 汤老师那次课也上去尝试过 >）
        - 学会进程互斥同步问题（仓库 + 切水果 + 做面条）
        - 前趋图（学会设置信号量）
13. 管程的基本概念和操作（工作原理）
    - 为什么要有管程：每一个进程要想同步就必须都有 PV 操作太杂乱无章了，所以我们要批量管理他们 —— 管程
    - 管程的定义：代表共享资源的数据结构以及由对该共享数据结构实施操作的一组过程所组成的资源管理程序共同构成了一个操作系统的资源管理模块。

- 包含的内容：管程的名称 + 共享数据结构说明 + 对该数据结构操作的过程 + 对共享数据的初始化语句

14. 经典的同步问题（写程序）
    - 生产者消费者问题及其变种（吃面条等）

      > 信号量初始值的设置

    - 读者写者问题（一定要回）
15. 进程的通信方式有哪些？
    - 以书上的为准（不要以实验课的内容来答题）
      - 进程同步和互斥本来就已经是一种进程通信了（低级） - 效率低 + 通信对用户不透明
      - 高级通信工具：使用方便 + 高效透明
    - 传统的操作系统最经典的部分的通信方式（重点掌握）
      - 共享存储器的通信方式
        - 基于共享数据结构的通信方式 buffer
        - 基于共享存储区的通信方式 划出来一块共享存储区域
      - 管道通信方式：做一个公用文件
      - 消息传递系统
        - 进程不需要借助任何共享存储区或数据结构，而是以格式化的消息为单位，将通信封装在消息中进行发送和接收。
      - 客户机-服务器系统
        - 以上的所有操作虽然可以实现不同计算机间进程的双向通信，但是客户机-服务器系统的通信机制仍然难以实现
        - 套接字、远程过程调用和远程方法调用
16. 线程的基本概念

    线程是比进程更小一级的单位，如果说进程的引入是为了使程序能够并发执行，那么现成的引入就是为了减少程序在并发执行时所付出的时空开销，并发性更好。

    - 线程与进程的比较（差异）
      - 调度的基本单位：线程——这一比进程更小的调度和分派单位，在切换时仅需保存和设置少量寄存器内容，切换代价远低于进程

        - 并发性：进程中的线程还可以并发执行，使得OS具有更好的并发性
        - 拥有资源：线程本身并不拥有系统资源，而是仅有一点必不可少的、能保证独立运行的资源。
        - 独立性：在同一进程中的不同线程之间独立性要比不同进程之间的独立性低得多
        - 系统开销：创建和撤销进程时，OS为此所付出的开销明显大于线程
        - 支持多处理机系统：在多处理机系统中，单线程进程只能在一个处理器尚运行，但队友多线程进程可以在多个处理器上并行执行
    - 线程的三状态（和进程完全相同）
        - 执行状态
        - 就绪状态
        - 阻塞状态
    - 线程的实现方式
        - 内核支持线程（KST）：为了对内核线程进行控制和管理，在内核空间也为每一个内核线程设置了一个线程控制块，内核根据该控制块而感知某线程的存在并加以控制
        - 用户级线程：用户级线程是在用户空间中实现的，对线程的创建、撤销、同步与通信等功能，都无需内核的支持，即用户级线程是与内核无关的，内核根本不知道用户级线程的存在。

3 第三章

> 进程数目多余处理器数目，那该怎么去分配处理器呢，就需要处理器调度了
>
> 调度的实质是一种资源分配，处理机调度是对处理机资源进行的分配

1. 处理器的调度几种层次（三级调度）
    - 高级调度：对象是作业，决定将外存上处于后备队列中的那几个作业调入内存，用于多道批处理中，实时和分时中是没有的
    - 低级调度：对象是进程，决定就绪队列中哪个进程该获得处理机 —— 最基本的调度，所有 OS 中都有

- 中级调度：提高内存的利用率和系统吞吐量，将暂时不能运行的程序调至外存等待，挂起

2. 处理器调度的基本指标

- 资源利用率
- 公平性：诸进程都获得合理的 CPU 时间
- 平衡性：所有类型的作业都能够处于忙碌
- 策略强制执行：只要有需要就可以打破规则

3. 几种调度的算法

- 先来先服务
- 短作业优先
  - 作业越短优先级越高
  - 优点：较比先到先服务有了良好的提升
  - 缺点：1. 必须直到作业的运行时间（很难预估）2.对长作业很不利 3. 人机无法交互 4. 不能保证紧迫性作业能得到及时处理
- 优先级调度算法
  - 外部赋予作业相应的优先级
- 高相应比优先算法
  - 优先级可以改变，随等待时间延长而增加，使长作业等待期间足够后，必然有机会获得处理机。
  - 可以看出：1.既有利于短作业又有利于长作业 2.做相应比的计算增加开销

4. 非抢占式调度

- 概念：一旦把处理机分配给某进程后，就一直让他运行下去，绝不会因为时钟中断或任何其他原因去抢占当前正在运行进程的处理机，直至该进程完成，或者发生某事件而阻塞时，才把处理机分给其他进程
- 引发的原因：1. 正在执行的进程运行完毕，或因发生某事件而使其无法再继续运行 2. 正在执行中的进程因提出 I/O 请求而暂停执行; 3. 在进程通信或进程同步中执行了某原语操作
- 优缺点：实现简单，系统开销小，适用于大多数批处理系统，不能用于分时系统和大多数实时系统

5. 抢占方式

- 原则：优先权原则；短进程优先原则；时间片原则
- 优点：更加公平，可以防止一个长进程长时间占用处理器，只有采用抢占方式才有可能实现人机交互。缺点：抢占方式比较

复杂，所需付出的时间开销比较大
    - 进程饥饿就是指部分进程长时间得不到响应。
6. 什么叫死锁？
    - 定义：如果一组进程中的每一个进程都在等待仅由该组进程中的其他进程才能引发的事件，那么该组进程是死锁的。
    - 产生死锁的原因：多个进程被阻塞，互相都想让对方释放出自己所需要的资源，但他们谁都因不能获得自己所需要的资源去继续运行，从而无法释放出自己占有的资源。
    - 进程死锁 和 进程饥饿有什么区别？
7. 基本的资源
    - 可抢占：指某进程在获得这类资源后，该资源可以再备其他进程或系统抢占。（CPU、内存）
    - 不可抢占：一旦系统把某资源分配给该进程后，就不能将它强行收回，只能再进程用完后自行释放。（刻录机<刻到一半去干别的了，那不乱套了嘛>、打印机、磁带机）
8. 产生死锁的原因
    - 起因就是多个进程对资源的争夺
        - 竞争不可抢占性资源
        - 竞争可消耗资源
        - 进程推进顺序不当
    - 必要条件
        - 互斥条件：对所分配的资源进行排他性使用，即在一段时间内某资源只能被一个进程占用。
        - 请求和保持条件：进程已经保持了至少一个资源，但又踢出了新的资源请求，而该资源已被其他进程占有。
        - 不可被抢占条件：进程获得的资源在未使用完之前不能被抢占
        - 循环等待条件：存在一个进程资源的循环链。
9. 处理死锁的方法有哪些？
    - 预防死锁
    - 避免死锁
    - 检测死锁
    - 解除死锁
10. 预防死锁的手段

> 核心理念就是打破必要条件中的一条，但是互斥条件又是必须要有的，所以只能破除其他三个

- 破坏“请求和保持条件”
  - 第一种协议（十分硬气）所有进程在开始运行之前，必须一次性地申请其在整个运行过程中所需的全部资源。（要么没有 - “非保持” 要么全部 - “非请求”） —— 资源被严重浪费 + 进程经常发生饥饿现象
  - 第二种协议（很灵活）允许一个进程只获得运行初期所需的资源后，便开始运行。进程运行过程中再逐步释放已分配给自己的、且已用毕的全部资源（“非保持”），然后再请求新的所需资源
- 破坏不可抢占条件
  - 当一个已经保持了某些不可被抢占资源的进程，提出新的资源请求而不能得到满时，它必须释放已经保持的所有资源，等待以后需要时再重新申请
- 破坏循环等待条件 —— 破坏掉环
  - 对系统所有资源类型进行线性排序并赋予不同的序号。必须先申请低的再申请高的，如果申请到高的了还要申请低的，那就只能先放弃高的

11. 什么叫避免死锁（安全状态的概念，如何分析一个状态是否是安全的）

> 在资源动态分配的过程中，防止系统进入不安全状态

- 安全状态和不安全状态
  - 安全状态是指系统能按某种进程推进顺序（P1，P2，…… Pn）为每个进程 Pi 分配其所需资源，直至满足每个进程对资源的最大需求，使每个进程都可以顺利地完成。此时的（P1，P2，…… Pn）为安全序列，如果系统找不到这样的一个安全序列，则称系统处于不安全状态，反之如果能找到就是安全状态
  - 进入不安全状态不一定会进入死锁，但在安全状态下一定不会进入死锁。
- 银行家算法 （一定要按照之前作业一般熟练掌握）

12. 如何实现死锁的检测（检测的方法有哪些？）

> 为了能够监测必须要有以下两个条件：1. 保存有关资源的请求和分配信息；2. 提供一种算法，能够根据这些信息来监测系统是否进入死锁状态

- 资源分配图法：一定要会资源分配图简化算法

13. 如何实现死锁的解除

    - 抢占资源：来个更狠的，去找到别人没有给到我的
    - 终止（或撤销）进程，终止系统中的一个或多个死锁进程，直至打破循环环路（终止所有死锁进程、逐个终止进程、付出代价最小的死锁解锁算法）

4 第四章 —— 内存的管理

> 存储器虽然很大，但是仍然满足不了需要，对其有效的管理直接影响到存储器的利用率，对系统性能也有重大影响。（本章我们重点关注内存）

1. 基础概念还是要了解一下：

    - 对存储器的三个要求：存储器速度必须快；容量必须大；价格必须便宜
    - 存储器的多层结构：至少三级 CPU 寄存器，主存，辅存
    - 寄存器和主存（内存）是OS管理的范围，但是辅存属于设备管理的范围
    - 寄存器和主存储器又被成为可执行存储器，CPU 只需要对其进行 load 和 store 两条命令就可以访问和运行，和外存访问数量级差别很大
    - 高速缓存：Cache 在寄存器和主存之间（的确是一个独立的存储空间）
    - 磁盘缓存：主存和磁盘自建（主存的一部分，并无独立的存储空间）

2. 程序的装入方式有哪些？

    > 程序执行的三大步骤：编译（形成若干个目标模块）链接（由连接程序江边以后形成的一组目标模块以及他们所需要的库函数连接在一起，形成装入模块）装入（由装入程序将装入模块装入内存）

    - 绝对装入方式：当内存很小并且仅能运行单道程序时，我们完全可能直到程序将驻留在内存的什么位置，按照模块的地址直接将程序和数据装入已知的指定内存部分就可以了。
    - 可重定位装入方式：在装入时对目标程序中指令和数据地址的修改过程称为重定位。又因为地址变换通常是在进程装入时一次完成的，以后不再概念，故称为静态重定位。可以将装入模块装入到内存中任何允许的位置，不允许程序运行时在内存中移动。

- 动态运行时的装入方式：运行过程中程序在内存中的位置可能经常要改变，程序装入后并不立即把装入模块的逻辑地址转换为物理地址，而是把这种地址转换推迟到成勋真正执行时才进行。

3. 程序的链接方式有哪些？

- 静态链接方式：在程序运行之前先将各目标模块及它们所需的库函数链接成一个完整的装配模块以后不在拆开。 A B C是编译后产生的3各块在装入前直接链接起来（解决两个基本问题1.每一个块当中的【操作地址】都要做出改变 2.每一个块的【起始地址】都要做出改变）
- 装入时动态链接：用户源程序编译后所得到的一组目标模块，在装入内存时，采用边装入边链接的链接方式。也就是说不像静态链接那样直接把各个块链接在一起（做一个不可拆分的大家伙）而是说发现外部模块调用事件，将引起装入程序去找出相应的外部目标模块，并将他装入内存（**便于修改和更新 + 便于实现对目标模块的共享**）
- 运行时动态链接：前两种方式的共同缺点：链接起来的大块中有的的是根本不会运行的。运行时动态链接要求 将对某些模块的链接推迟到程序执行时才进行，在执行程序过程中，当发现一个被调用模块尚未装入内存时，立即由OS去找到该模块，并将之装入内存，将其链接道调用者模块上。凡是在执行过程中给未被用到的目标模块，都不会被调入内存和被链接到装入模块上，加快程序的装入过程 + 节省大量时间。

4. 常规存储器管理方式（最基本的一些概念要掌握）

> 既然要将程序装入内存，怎么装？装入哪？

- 连续分配存储管理方式（背对背 肩对肩）
  - 单一连续分配：在单道程序环境下，存储器管理方式是把内存分为系统区和用户区两部分，系统区给OS 使用，用户区中有且仅有一道用户程序，整个内存的用户空间由该程序独占
  - 固定分区分配:将整个用户空间划分为若干个固定大小的区域,在每个分区中只装入一道作业,这样就形成了最早的也是最简单的一种可运行多道程序的分区式存储管理方式
    - 划分分区的方法:等分 or 不等分

        - 内存分配:将分区按到小进行排队并为之建立一张分区使用表,各种信息全部涵盖在其中,检索该表找到能满足要求的尚未分配的分区
        - 缺点:内存浪费现象严重
    - 动态分区分配：又称为可变分区分配,根据进程的实际需要,动态地为之分配内存空间。涉及到数据结构（拿什么分配）分区分配算法（怎么分配）分区的分配与回收操作（具体分配方式）
        - 数据结构：空闲分区表（顺序），空闲分区链（链接）
        - 动态分区分配算法（下一个问题中做具体阐述）
        - 分区分配操作：
            - 分配内存： P138
            - 回收内存： 4 种情况：

                1） 回收区前有一个空闲区

                2） 回收区后有一个空闲区

                3） 回收区前后都有空闲区

                4） 回收区前后都没有空闲区
    - 动态可重定位分区分配：紧凑 + 动态重定位

5. 动态分区分配的四种算法
    - 基于顺序搜索的动态分区分配算法
        - 首次适应算法
        - 循环首次适应算法
        - 最佳适应算法
        - 最坏适应算法
    - 基于索引搜索的动态分区分配算法
        - 快速适应算法
        - 伙伴系统
        - 哈希算法
6. 什么叫对换？
    - 概念：所谓对换是指把内存种暂时不能运行的进程或者暂时不用的程序和数据换出到外存上，以便腾出足够的空间，再把已具备运行条件的进程或进程所需要的程序和数据换入内存。
7. 离散分配方式 —— 分页（重中之重）

> 和连续分配做呼应，既然连续分配这么容易产生“碎片”，并且尽管可以通过“紧凑”等方式来拼接，但是仍然存在巨大的开销 + 浪费
>
> 如果允许将一个进程直接“离散”地装入到许多不相邻的分区中，便可充分地利用内存空间，而无须再进行“紧凑”等复杂分配
>
> 分页 —— 将内存块分成一块又一块和页面一一对应（页号和块号通过中介表一一对应） 所以需要物理地址的部分只需要指定页号，通过页号就能访问到块号。

- 页面和物理块
- 地址结构（会从逻辑地址转化为分页地址）
- 页表（索引表 或者说是字典）实现从页号到物理块号的地址映射（页表中还可以涵盖很多其他的信息）
- 地址变换机构的使用方法（P150）
- 具有快表的地址变换机构
    - Motivation : 现在要想访问物理地址必须经过两次访存，得不偿失
    - Method ： 把经常使用到的放在快表（缓冲寄存器）中，这样就不需要两次访问内存了
- 访问内存的有效时间
    - 加上页表和不加页表
- 两级和多级页表 P 152（离散分配页表）
- 反置页表

8. 对分段存储管理方式的理解

> 以上的各种内存分配方式对程序员太不友好了，所以我们要做分段
>
> - 方便编程
> - 信息共享
> - 信息保护
> - 动态增长
> - 动态链接

- 如何从逻辑地址到物理地址：其实和页本质上没什么区别，只不过是外加了一个索引字典罢了
- 分页和分段的区别

        - 页是【信息】的物理单位，分页仅仅是系统管理上的需要，完全是系统的行为，对用户不可见；分段的目的主要是方便用户
        - 页的大小固定且由系统固定，由硬件实现的。而段的长度不固定，决定于用户所编写的程序，通常由编译程序在对源程序进行编译时按照信息性质来划分
        - 分页的用户程序地址空间是一维的，分页系统中程序员只需利用一个记忆符即可表示一个地址；分段式用户行为，分段系统中用户程序的地址空间是二维的。
    9. 段页式

5 第五章 虚拟存储器

> motivation: 第四章所讲到的各种存储器管理方式有一个共同特点，一次性把作业装入内存后方能运行，但是很明显会出现以下两种情况
>
> - 有的作业很大，要求的内存空间超过了总的内存空间，作业不能全部被装入内如此呢，致使该作业无法运行
> - 有大量作业需求运行，但由于内存容量不足以容纳所有这些作业，只能将少数作业装入内存让它们先运行，而将其他大量的作业停留在外存上等待。
>
> 一个显而易见的解决方式：加内存嘛 但是要是这么简单，别人早做了。
>
> 另一种方法：并非从物理上扩充内存，而是在逻辑上扩充内存（虚拟存储器）

1. 常规存储器管理方式的特征
    - 一次性：作业必须一次性地全部装入内存后才能开始运行
    - 驻留性：作业被装入内存后，整个作业都一直驻留在内存中，其中任何部分都不会被患处，知道作业运行结束。
2. 什么是局部性原理
    - 概念：在一较短时间内，程序的执行仅局限于某个部分，相应地，他所访问的存储空间也局限于某个区域。
    - 表现：
        - 时间局部性：如果程序中的某条指令被执行，则不久以后该指令可能再次执行，如果某数据被访问过，则不久

以后该数据可能再次被访问。（程序中存在着大量的循环操作）

- 空间局部性：一旦程序访问了某个存储单元，在不久之后，其附近的存储单元也将被访问，即程序在一段时间内所访问的地址可能集中在一定的范围之内。（程序顺序执行）

3. 虚拟存储器的概念和虚拟存储器的特征

- 整体思路：应用程序运行之前没有必要将之全部装入内存 只需要执行以下步骤：1. 将那些当前要运行的少数页面或段装入内存便可运行，其余部分暂留在攀上，程序在运行时，如果它索要访问的页已经调入内存，便可以运行下去。2.如果程序访问的页尚未调入内存，便发出缺页（段）请求。3. 此时OS 将利用请求调页功能将他们调入内存 依次类推，如果此时内存已满，无法再装入新的页，OS 还须再利用页的置换功能，将暂时不用的页调到外存上
- 定义：是指具有请求调入功能和置换功能，能从逻辑上对内存容量加以扩充的一种存储器系统。其逻辑容量是由内存容量和外存容量之和所决定，其运行速度接近于内存速度，而每位的成本却又接近外存。
- 特征：
  - 多次性：相对于“一次性” —— 程序被允许分成多次调入内存运行
  - 对换性：相对于“驻留性” —— 那些暂时不使用的代码和数据从内存调至外存
  - 虚拟性：能够从逻辑上扩充内存容量，使用户所看到的内存容量远大于实际内存容量（以多次性和对换性为基础）

4. 请求分页基本的过程理解

> 虚拟存储器的实现全部都是建立再离散分配存储管理方式的基础上。

请求分页系统是建立在基本分页基础上的，为了能支持虚拟存储器功能，而增加了请求调页功能和页面置换功能。（+ 请求分页的页表机制 + 缺页中断机构 + 地址变换机构）

- 请求页表机制（主要数据结构是：请求页表）
  - 状态位（存在位）指示该页是否已调入内存？
  - 访问字段：用于记录本页在一段时间内被访问的次数。

        - 修改位：该页在调入内存后是否被修改，若没有被修改就不用再被重新写回到外存。
        - 外存地址，通常 = 物理块号
    - 缺页中断系统

        > 缺页中断是一种特殊的中断，即会像普通中断一样做好保护

        - 在指令执行期间产生和处理中断信号，而非一条指令执行完后
        - 一条指令在执行期间可能产生多次缺页中断
    - 地址变换结构：P170 讲请求分页讲的很清楚
    - 怎么为 请求分页 分配内存呢？
        - 最小块数的确定：保证程序正常运行所需的最小物理块数，当系统为进程分配的物理块数少于此时，进程将无法运行。
        - 内存分配策略：1. 固定分配局部置换 2. 可变分配全局置换 3.可变分配局部置换
        - 物理块分配算法：采用固定分配的话 => 平均分配算法 + 按比例分配算法 + 考虑优先权的分配算法
    - 怎么进行页面调入策略呢？（事先将要执行的那部分程序和数据所在的页面调入内存）
        - 啥时候调入？
            - 预调页策略——预测哪些页可能会被使用，然后将这一部分一次性调入
            - 请求调页策略——进程在运行中需要访问某部分程序和数据时，若发现其所在的页面不在内存，便立即提出请求，由OS将其所需页面调入内存。
        - 从哪调入？

            > 将请求分页系统中的外存分为两部分：
            >
            > - 存放文件的文件区（离散分配）
            > - 存放用于对换页面的对换区（连续分配 很香）

            - 如果有足够的对换区：直接全部从对换区调入所需页面
            - 如果缺少足够的对换区：不被修改的从文件区调入
            - UNIX 方式：未被运行过的页面从文件区调入；被运行过的从对换区调入
        - 调入的过程？P173 学会描述！！

5. 缺页率的计算

   - $$f = \frac{F}{A}$$

     > 其中 f 为缺页率 F 为从内存访问失败的次数 A 为总访问次数

   - 影响缺页率的几个原因：1. 页面大小 2. 进程所分配物理块的数目 3. 页面置换算法 4. 程序固有属性

6. 页面的置换算法 ***

   - 最佳置换算法
   - 先进先出置换算法
   - 最近最久未使用
   - 最少使用
   - 访问内存的有效时间
     - 被访问的页在内存中，且其对应的页表项在快表中
     - 被访问的页在内存中，且其对应的页表不再快表中
     - 被访问的页不在内存中

7. 什么叫做抖动

   - 随着进程数目的增加，处理机的利用率急剧增加，但后续会明显利用率下降
   - 产生抖动的原因：同时在系统中运行的进程太对，由此分配给每一个进程的物理块太少，不能满足进程正常运行的基本要求，至使每个进程在运行时，频繁地出现缺页，必须请求系统将所缺的页调入内存。

8. 工作集的概念

   - 所谓工作集，是指在某段时间间隔 Δ 里，进程实际所要访问页面的集合

6 第六章

> I/O 系统管理的主要对象是 I/O 设备和相应的设备控制器。
>
> 其主要的任务是：
>
> - 完成用户提出的 I/O 请求

> - 提高 I/O 速率
> - 提高设备的利用率

1. 最基本的功能
    - 隐藏物理设备的细节
    - 与设备无关性
    - 提高处理器和 I/O 设备的利用率
    - 对 I/O 设备进行控制
    - 确保对设备的正确共享
    - 错误处理
2. IO系统的层次结构（四个层次）
    - 用户层 I/O 软件
    - 设备独立性软件
    - 设备驱动程序
    - 中断处理程序
3. 输入输出系统的控制方式

> 设备控制器：控制一个或多个 I/O 设备，以实现 I/O 设备和计算机之间的数据交换，是 CPU 和 I/O 设备之间的接口。

    - 设备控制器的基本功能：
    - 设备控制器的组成：
4. 什么叫做与设备无关（基本概念）
    - 一方面，用户不仅可以使用抽象的 I/O 命令，还可使用抽象的逻辑设备名来使用设备
    - 另一方面，也可以有效地提高 OS 的可移植性和易适应性，对于OS本身而言，应允许在不需要将整个操作系统进行重新编译的情况下，增添新的设备驱动程序，以方便新的 I/O 设备的安装。
    - 如 Windows 中，系统可以未新的 I/O 设备自动安装和寻找驱动程序，从而做到即插即用。
    - 与设备无关的I/O软件：
        - 基本概念：
            - 早期：以物理设备名使用设备（应用程序与系统中的物理设备直接相关）
            - 引入了逻辑设备名（抽象的设备名）没有指定那一个 I/O

        - 所含内容：
            - 设备驱动程序的统一接口
            - 缓冲管理
            - 差错控制
            - 对独占设备的分配与回收
            - 独立于设备的逻辑数据块
        - 设备分配：为了实现对独占设备的分配，必须在系统中配置相应的数据结构
            - 设备控制表 + 控制器控制表 + 通道控制表 + 系统设备表

5. Spooling（假脱机系统） 技术本质上是什么
    - 引入了多道程序技术后，可以利用其中的一道程序，来模拟脱机输入时外围控制机功能，把低速 I/O 设备上的额数据传送到高速磁盘上。
    - 当系统中引入了多道程序设计后，完全可以利用其中的一道程序，来模拟脱机输入时外围控制机的的功能，把低速 I/O 设备上的数据传送到高速磁盘上。再用另一道程序模拟脱机输入时的外围控制机功能，把数据从磁盘传送低速输出设备上，这样便可在主机的直接控制下，实现以前的脱机输入输出功能。
    - 此时的外围操作与CPU对数据的处理同时进行，我们把着这种在联机情况下实现的同时外围操作的技术称为 Spooling
    - 四部分构成：
        - 输入井和输出井：这是在磁盘上开辟出来的两个存储区域，输入井模拟脱机输入时的磁盘，输出井模拟脱机输出时的磁盘
        - 输入缓冲区和输出缓冲区
        - 输入进程和输出进程
        - 井管理程序
    - SPOOLING 系统的特点：
        - 提高了 I/O 速度
        - 将独占设备改造为共享设备
        - 实现了虚拟设备功能

6. 缓冲区的基本概念和原理
    - 引入的原因：
        - 缓和 CPU 与 I/O 设备间速度不匹配的矛盾
        - 减少对 CPU 的中断频率

        - 解决数据粒度不匹配的问题
        - 提高 CPU 和 I/O 设备之间的并行性
    - 缓冲区类型
        - 单缓冲区
        - 双缓冲区
        - 环形缓冲区
        - 缓冲池

7. 磁盘的访问时间（由哪些部分组成）

    - 每条磁道上可存储相同数目的二进制位，这样磁盘密度即每英寸中所存储的位数，显然内层密度大外层小。
    - 磁盘格式化完成 => 磁盘分区 => 磁盘高级格式化
    - 访问时间由三部分构成：三步走战略——磁头必须能移动到指定的磁道上； 2.等待指定的扇区的开始位置旋转到磁头下；3.开始读写。
        - 寻道时间： $T_s = m \times n + s$ n 是移动磁道数目，m 是磁盘的属性 s 为启动磁道的时间
        - 旋转延迟时间：等待删去的开始位置旋转到磁头下 T_σ 跟磁盘转动速度相关
        - 传输时间： $T_t = \frac{b}{rN}$ 因此可将访问时间:

$$T_a = T_s + T_\sigma + T_t$$

8. 磁盘调度算法

    - 先来先服务
    - 最短寻道时间优先
    - 扫描算法：磁头来回地双向移动
    - 循环扫描算法：磁头只往一个方向移动

7 第七章

> 文件系统的管理功能是：奖其管理的程序和数据组织为一系列的文件的方式实现的。数据项 => 记录 => 文件 三级

1. .文件的逻辑结构（几种都要掌握）

> - 文件的逻辑结构：是从用户观点出发所观察到的文件组织形式，即文件是由一系列的逻辑记录组成的，是用户可以直接

> 处理的数据及其结构，它独立于文件的物理特性，又称为文件组织
> - 文件的物理结构，是系统将文件存储在外村上所形成的一种存储组织形式，使用户不能看见的。

- 按照是否有结构进行分类
  - 有结构文件（每个记录都用于描述实体集中的一个实体，各记录有着相同或不同数目的数据项）
    - 定长记录（所有记录长度相同）
    - 变长记录（所有记录长度不同）
  - 无结构文件（流式文件，文件长度以字节为单位，利用读写指针来指出下一个要访问的字符）
- 组织方式来看
  - 顺序文件（记录按顺序排列形成的文件）
  - 索引文件（一张索引表，查询索引表来锁定文件）
  - 索引顺序文件

2. 文件的物理结构
3. 文件的检索方法
   - 单级 ：首先将变长记录顺序文件中的所有记录分为若干个组，如 50 个记录一组。然后为顺序文件建立一张索引表，并为每组中的第一个记录在索引表中建立一个索引项，其中含有该记录的关键字和指向该记录的指针。
   - 多级 ：为低级索引表再建立一个索引表

8 第八章 磁盘存储器的管理（文件的物理存储方式）

> 磁盘存储器不仅容量大，存取速度快而且可以实现随机存取，故是实现虚拟存储器和存放文件的最理想外存，三大要求：
>
> - 有效地利用存储空间；
> - 提高磁盘的 I/O 速度
> - 提高磁盘系统的可靠性

8.1 外存的组织方式

8.1.1 连续组织方式

> 把逻辑文件中的记录顺序地存储到【邻接】的各物理盘块中

优点：

- 顺序访问容易
- 顺序访问速度快

缺点：

- 要求为一个文件分配连续的存储空间
- 必须实现知道文件的长度
- 不能灵活地删除和插入记录
- 很难为哪些动态增长的文件分配空间

8.1.2 链接组织方式

> 十分类似于链表
>
> - 消除了磁盘的外部碎片，提高了外存的利用率
> - 对插入、删除和修改记录都非常容易
> - 能适应文件的动态增长，无需实现知道文件的大小

1. 隐式链接
2. 显示链接

8.1.3 索引组织方式

1. 单机索引组织方式
2. 多级索引组织方式
3. 增量式索引组织方式

8.2 文件存储空间的管理