*注：标黄的为往年考察过的知识点，标蓝的为背诵自用不用管，蓝字为题目中出现过但没找到具体出处的东西，紫色为ppt里翻到的东西*

**Unit 1 导论**

**计算机系统分类**

单处理器系统、多处理器系统、集群系统

**多处理器系统：**

由多个紧密通信的CPU组成，这些CPU共享总线，有时还有时钟、内存和外设。

优点：吞吐量大，规模经济，可靠性高

①非对称多处理：每个处理器执行不同的任务，一个主处理器控制系统，其余处理器要么执行分配到的任务要么向主处理器要任务

②对称多处理：处理器间没有主从之分，所有处理器都要执行系统的所有任务

**集群系统：**

集群系统和多处理器系统类似，都由多个CPU集中起来完成计算任务。但与多处理器系统不同的是，集群系统由两个或多个独立的系统耦合而成。集群计算机共享存储，通过局域网连接或更快的内部连接

**中断机制作用**

程序管理：

①实现进程/线程切换：通过时钟中断定期打断进程调度其他任务，确保多道程序并发执行（U1 分时系统）

②响应外部事件：用户输入、I/O完成，用中断通知CPU及时处理（U13 I/O控制方式-中断方式）

资源管理：

①管理I/O设备：避免CPU忙等，利用中断通知完成数据传输，提高资源利用率

②协调硬件资源：通过中断优先级处理多个设备请求，实现资源分配的公平高效

③处理异常：触发异常中断，保护系统安全（U1 硬中断软中断）

**操作系统结构**

**多道程序multiprogramming：**

操作系统同时将多个任务放入内存，如果一个执行中的任务需要等待某事件的完成，则CPU切换到另一个任务并执行而不是空等待原任务完成

意义/优点：提高CPU利用率，减少空闲等待时间，提高系统吞吐量，优化资源使用

缺点：系统复杂性增加，内存管理压力大，程序之间需隔离

**批处理系统、分时系统、实时系统**

**批处理系统：**

用户将作业交给系统操作员，系统操作员将许多用户的作业组成一批作业后输入到计算机，在系统中形成一个自动转接的连续作业流，系统自动、依次执行每一个作业。最后由操作员将作业结果交给用户

优点：作业流程自动化，效率高，吞吐量高

缺点：无交互手段，调试程序困难

**分时系统：**

操作系统将CPU时间分成多个小段，称为时间片。操作系统以时间片为单位，在用户间快速切换，轮流为终端用户服务，每次服务一个时间片。系统的快速切换让用户感到整个系统只为自己所用。是多道程序设计的延申。由于切换频率很高，用户可以在程序运行期间与之进行交互。

意义/优点：提高响应速度，提高交互性，支持多用户同时使用系统

缺点：频繁上下文切换可能引入额外开销

**实时系统：**

当外界事件或数据产生时，能够快速接收并作出处理，其处理结果又能在规定时间内控制生产过程或对处理系统做出快速响应，并控制所有实时任务协调一致的操作系统

①硬实时系统：在规定时间内必须完成操作，否则意味着失败

②软实时系统：只要按照任务的优先级，尽可能快地完成操作即可

**什么是用户态和内核态：**

用户态：用户程序运行时的状态，较低的特权级别。当CPU处于用户态时程序只能执行非特权指令

内核态：操作系统管理程序运行时的状态，较高的特权级别。处于内核态时程序可执行特权指令，访问所有资源，并可改变处理器的状态

**通过什么途径可以实现用户态到内核态的转变：**

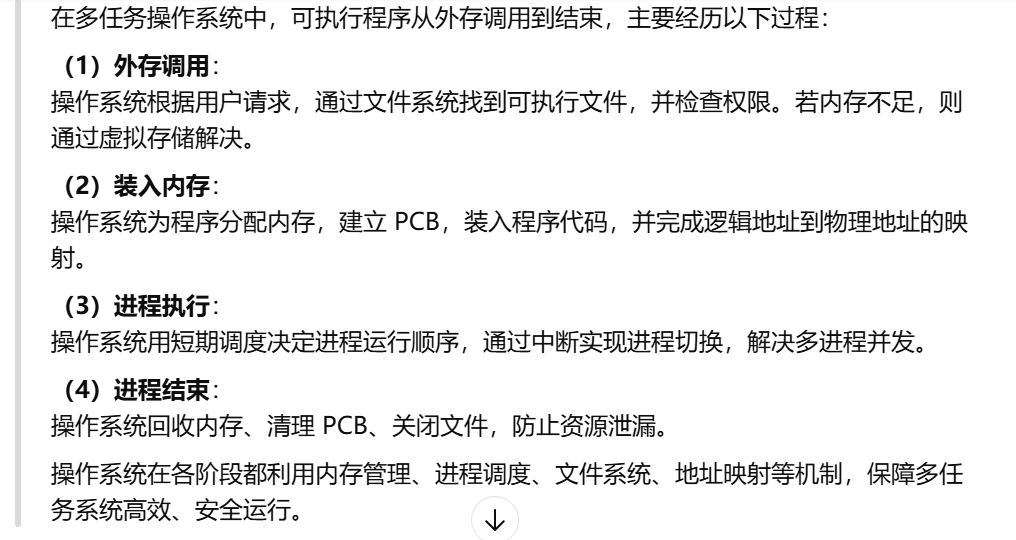
用户态->内核态：通过中断（系统调用、中断事件）

内核态->用户态：设置PSW（修改程序状态字）

**特权指令：**

将能引起损坏的硬件指令称为特权指令，用户模式想执行特权指令，硬件不会执行，会认为是非法指令，并以trap的形式通知操作系统

**可执行程序从外存调用，执行，结束各环节发生了什么，并如何解决的**



**Unit 2 操作系统结构**

**系统调用**

操作系统内核提供一系列预定功能，通过一组称为系统调用的接口呈现给编程人员。系统调用把应用程序的请求传给内核，系统调用内核函数完成所需处理，并将处理结果返回给应用程序

优点：简单，可移植性

**向操作系统传递参数的三种方法：**

①通过寄存器传递参数

②若参数数量比寄存器多，参数通常存在内存的块和表中，并将块的地址通过寄存器传递

③参数也可以通过程序放在或压入栈中，并通过操作系统弹出

**操作系统结构**

简单结构、分层方法、微内核、模块、虚拟机

**简单结构**：较小，简单，功能有限。MS-DOS，原始的UNIX操作系统

**分层方法：**

操作系统分成若干层，最底层（层0）为硬件，最高层（层N）为用户接口

优点：构造和调试的简单化

缺点：难以对层进行详细定义

**微内核：**

将所有非基本部分从内核中移走，并将它们实现为系统程序或用户程序，从而得到更小的内核。微内核通常包括最小的进程和内存管理以及通信功能。

用户程序和系统服务通过微内核以消息传递形式进行通信，并不会直接交互

优点：便于扩充操作系统；操作系统容易移植；更好的安全性和可靠性

缺点：由于系统功能总开销的增加而导致系统性能下降

**模块：**

采用面向对象的特点，将各个功能模块化，每一个模块之间采用接口进行通讯，必要的时候可以将一部分内容加载到内核中进行操作。比层次结构更灵活，比微内核更有效率

**虚拟机：**

通过软件模拟的具有完整硬件系统功能的、运行在一个完全隔离环境中的“完整”计算机系统

优点：不同的系统资源具有完全的保护；每个虚拟机独立于其他虚拟机，没有安全问题；兼容性好

缺点：没有直接资源共享

解决方法：①通过共享小型磁盘来共享文件 ②定义虚拟机网络来传递消息

**Unit 3 进程**

**进程的五个特征**

动态性：是进程最基本的特征，表现在进程由创建而产生，由调度而执行，因得不到资源而暂停执行，由撤销而消亡

并发性：多个进程实体同时存在于内存中，能在一段时间内同时执行

独立性：进程实体是能独立运行的基本单位，同时也是系统中能独立获得资源和独立调度的基本单位

异步性：进程按各自的、不可预知的速度向前推进

结构特征：进程由程序段、数据段及PCB组成，三部分也称为进程影像

**进程状态**

新建：进程正在被创建

运行：指令正在被执行

等待：进程等待某个事件的发生

就绪：进程等待分配处理器

终止：进程完成执行

**（进程状态图）**

**进程控制块PCB中的项目有哪些？**

进程标识符：唯一标识进程的编号

进程状态：新建、就绪、运行、等待、停止

程序计数器：计数器表示进程要执行的下个指令的地址

CPU寄存器：包括累加器、索引寄存器、堆栈指针、通用寄存器和其它条件码信息寄存器

CPU调度信息：包括进程优先等级、调度队列的指针和其他调度参数

内存管理信息：基址或界限寄存器的值、页表或段表

记账信息：包括CPU时间、实际使用时间、时间界限、记账数据、作业或进程数量等

I/O状态信息：分配给进程的I/O设备列表、打开的文件列表等

**PCB：**

系统为了管理进程状态设立的一个专门的数据结构，用它来记录进程的外部特征，描述进程的运动变化过程

**进程调度**

解决多道程序环境下选择哪个进程在CPU上执行的问题，能够使多个进程合理执行，提高CPU利用率

**调度程序**

长期调度：从缓冲池中选择进程，装入内存中准备执行。不频繁。控制多道程序设计的程度，平衡IO和CPU的利用率

短期调度：从准备执行的进程中选择进程，并为其分配CPU。频繁，速度快。影响CPU利用率和系统吞吐量

中期调度：将进程从内存中移出，从而降低多道程序设计的程度

**进程分类**

IO密集型：在IO上花费的时间比计算多

CPU密集型：花费更多时间进行计算

**独立进程和合作进程**

独立进程不能影响或受其他进程影响

合作进程会影响或受其他进程影响

合作进程的优势：信息共享、计算速度加快、模块化、方便

**上下文切换**

将CPU切换到另一个进程需要保存当前进程的状态并恢复另一个进程的状态，这一任务称为上下文切换

CPU在进程间的切换由中断或系统调用引起，需要保存状态至原进程的PCB，并从新进程的PCB恢复状态

**Fork()的功能**

①为子进程做父进程的上下文拷贝 ②子进程与父进程共享子进程创建之前父进程的所有资源 ③父进程和子进程在不同的地址空间运行

要点：子进程继承父进程的所有资源，包括父进程的私有变量，但继承之后互相不能访问

**进程间通信的两种方式**

共享内存：进程通过读写公共区域来交换信息。允许以最快的速度进行方便的通信，在计算机中可达到内存的速度

消息传递：通过彼此之间交换消息来交换信息。对于交换较少的数据很有用，因为不需要避免冲突。比共享内存更易于实现

**Unit 4 线程**

**线程**

线程是CPU使用的基本单元，，由线程ID、程序计数器、寄存器集合和栈组成。它与属于同一进程的其他线程共享代码段、数据段和其它操作系统资源，如打开文件和信号

**多线程优点：**

①响应度高：对一个交互程序采用多线程，即使其部分阻塞或执行冗长操作，该程序仍能继续执行，从而增加了对用户的响应程度

②资源共享：线程默认共享它们所属进程的内存和资源

③经济：创建和切换线程比较经济

④充分利用多处理器体系结构

**进程和线程的区别：**

①一个进程可以有多个线程，但至少有一个线程；一个线程只能在一个进程的地址空间内活动。每当创建一个进程时，至少要同时为该进程创建一个线程，否则该进程无法被调度执行

②地址空间和其他资源：进程间互相独立，同一进程的各线程共享，某进程内的线程在其他进程不可见

③通信：进程间通信采用IPC，线程间可以直接读写进程数据段来进行通信；所有线程可共享进程的主存，不需要特殊的通信机制

④调度：线程上下文切换比进程上下文切换快得多

⑤引入进程的目的：使多个程序并发执行，以改善资源使用率和提高系统效率；引入线程的目的：减少程序并发执行时所付出的时空开销，并发性更好

**多线程模型：**

**多对一模型：**将多个用户级线程映射到一个内核线程。线程管理由线程库在用户空间进行，效率高。但如果一个线程执行阻塞了系统调用，那么整个进程会阻塞，因为任意时刻只有一个线程能访问内核，多个线程不能并行运行在多处理器上**（可创建任意多用户线程，不并发，不并行）**

**一对一模型：**将每个用户线程映射到一个内核线程。更好的并发性。允许多个线程并行的运行在多处理器系统上**（限制线程数量，并发，并行）**

**多对多模型：**多路复用了许多用户线程到同样数量或更小的内核线程上，可以创建任意多用户线程，并且相应内核线程能在多处理器系统上并发执行。当一个线程执行阻塞系统调用时，内核能调度另一个线程来执行**（可创建任意多用户线程，并发，并行）**

**二级模型：**在多对多模型的基础上也允许将一个用户线程绑定到某个内核线程上**（可创建任意多用户线程，并发，并行）**

**Unit 5 CPU调度**

**调度准则**

CPU使用率（CPU使用时间/运行总时间）

吞吐量（一个时间单元内完成进程的数量）

周转时间（从进程提交到彻底完成）

等待时间（进程等待的总时间）

响应时间（从进程提交到系统第一次响应进程的时间）

**非抢占和抢占：**

**非抢占调度：**就绪进程不可以从运行进程手中抢占CPU。一旦进程处于运行状态，它就不断执行直到终止或者为等待IO或请求某些操作系统服务而阻塞自己，才把CPU让给别人

**抢占调度：**就绪进程可以从运行进程手中抢占CPU。允许调度程序根据某种策略终止当前运行进程的执行，将其转移到就绪状态，并选择另一进程投入运行

**CPU调度决策的发生情况：**

①运行->等待

②等待->就绪

③运行->就绪

④终止

其中①④没有选择只有调度，为**非抢占**的；②③有选择，为**抢占**的

**多级队列和多级反馈队列**

**多级队列**：将就绪进程根据**类别或优先级**分成多个队列，每个队列有自己的调度算法，**进程进入某个队列后，永远待在该队列中**

**优点：结构清晰，易于管理；调度效率高；高优先级任务响应快**

**缺点：缺乏灵活性；可能导致饥饿；不适应动态任务负载**

**多级反馈队列：**将进程划分为多个队列，但**允许进程在不同队列之间移动**

**优点：灵活性高；兼具响应性和公平性；适应性强；防止饥饿**

**缺点：实现复杂，可能调度不稳定，调优困难**

**Unit 6 同步**

**进程的同步和互斥：**

进程的同步（直接制约）：指系统中一些进程需要相互合作，共同完成一项任务。一个进程运行到某一点时要求另一伙伴进程为它提供消息，在未获得消息之前，该进程处于等待状态，获得消息后被唤醒进入就绪态

进程的互斥（间接制约）：由于各进程要求共享资源，而有些资源需要互斥使用，因此各进程之间竞争使用这些资源

**临界区设计原则**

①互斥：任意时刻最多只能有**一个进程/线程**进入临界区

②进程有空让进：不应无故阻止进程进入临界区

③有限等待：若一个进程想进入临界区，总有机会在有限时间内进入

④让权等待：不忙等占用CPU，阻塞时自动让出CPU

**信号量如何实现这些原则：**



**Unit 7 死锁**

**死锁定义：**

一组进程中，每个进程都无限等待被该组进程中另一进程所占有的资源，因而永远无法得到资源，这种现象称为进程死锁，这一组进程称为死锁进程

**死锁的必要条件：**

①互斥：至少一个资源要求互斥地共享

②占有并等待：一个进程至少占有一个资源并等待另一资源，该资源为其它进程所占有

③非抢占：资源不能被抢占，只有进程完成任务后自动释放

④循环等待：互相等待形成一个环

**死锁处理方法：**

①死锁预防：限制申请资源方法，使至少一个条件不成立

②死锁避免：动态监测资源分配状态以确保循环等待条件不可能成立

③死锁检测和恢复：通过算法来检测并恢复

④忽略死锁：认为死锁不可能在系统内发生

**死锁预防对各条件的打破方式：**

①互斥：对于非共享资源，必须要有互斥条件，但共享资源不要求互斥访问

②占有并等待：确保当一个进程申请一个资源时，它不能占有其他资源

③非抢占：如果一个进程占有资源并申请另一个不能立即分配的资源，那么其现已分配的资源都可被抢占

④循环等待：对所有资源类型进行完全排序，且要求每个进程按递增顺序来申请资源

**安全状态：**

系统能按某种顺序，如<P1, P2, ... , Pn>，为每个进程分配所需资源，直到最大需求，使每个进程都可顺序完成，称系统处于安全状态

不安全状态可能导致死锁，死锁一定是不安全状态

**死锁恢复方法：**

进程终止：终止所有的死锁进程/一次只终止一个进程直到取消死锁循环为止

资源抢占：逐渐从进程中抢占资源给其他进程使用，直到死锁循环被打破为止

**Unit 8 内存**

**分配算法**

首次适应：分配找到的第一个足够大的可用分区，寻找的起始位置可以是表头，也可以是上次首次适应结束时的位置

最佳适应：分配足够大的里面最小的孔，可以产生最小的剩余孔。必须要遍历整个表，除非表已按顺序排列

最差适应：分配最大的表，可以产生最大的剩余孔。也必须遍历整个表

邻近适应：从上次找到合适孔的位置继续往后找此次合适的孔

**TLB作用：**

类似于高速缓存，是一种快速内存。TLB中只包含页表的一部分条目，CPU产生逻辑地址后，先在TLB中查找是否有相应页号，若命中则得到相应帧号，再根据帧号访问内存即可；若未命中，则访问两次内存，并把对应的页号和帧号放入TLB。若TLB已满，则操作系统选择一个进程进行替换

**哈希分页**

虚拟地址的虚拟页号根据哈希函数找到哈希表的某一条目，将虚拟页号与链表中的每个元素的虚拟页号比较，找到匹配的元素即获得对应帧号，形成物理地址

优点：①内存利用率高 ②快速定位，减少页表搜索时间 ③满足大规模内存管理需求

缺点：①哈希冲突 ②增加时间复杂度、空间利用率 ③cache未命中率增加

**什么是碎片？碎片可以分为哪几类？**

碎片是内存或存储系统中由于频繁的分配释放操作，导致可用空间被分割成许多不连续的小块，使得系统无法有效利用这些零散空间的现象

①内部碎片：分配给进程的内存块中未被实际使用的部分，由于进程需求小于内存分配的固定大小而产生

②外部碎片：内存中分散的、未被分配的小块空闲内存，无法合并以满足大内存请求。由于频繁动态分配/释放导致空闲内存分散。可通过压缩解决

**分页的基本工作原理；硬件支持及作用；地址映射过程；该存储管理方案的优缺点：**

**基本工作原理：**

①逻辑地址分页：进程的逻辑地址空间被划分为固定大小的页

②物理内存分块：物理内存划分为相同大小的帧

③页表映射：通过页表记录逻辑页号到物理帧号的映射关系

④按需加载：仅当进程访问某页时，才将其从磁盘调入内存

**硬件支持：**

①MMU：逻辑地址->物理地址

②TLB：缓存常用页表项，加速转换

③缺页中断：要访问的页不在内存时，触发操作系统加载

**地址映射过程：**

①CPU生成逻辑地址（页号+页内偏移） ②查询页表 ③生成物理地址

**优缺点：**

1. 优点：无外部碎片、内存利用率高、支持虚拟内存
2. 缺点：页表占用空间大、地址转换可能慢（需TLB加速）、内部碎片

**分页和分段：**

1. 页是信息的物理单位，分页实现离散分配方式，以消减内存的外碎片，提高内存的利用率，满足系统管理需要；段是信息的逻辑单位，包含一组意义完整的信息，满足用户需求
2. 页的大小固定，由系统决定，把逻辑地址划分为页号和页内地址两部分，由硬件实现；段的大小不固定，取决于用户编写的程序，通常由编译器根据信息性质来划分
3. 分页的作业地址空间是一维的，只需一个地址记忆符即可表示一个地址；分段是二维的，标识地址时既需要给出段名又需要给出段内地址

**页表项包含内容**

页表项主要包括物理块号、存在位、访问权限、脏位、访问位等，用于记录虚拟页与物理页的映射关系以及内存管理的状态信息

**二级页表项里面有什么**

一级页表项保存二级页表的物理地址以及存在位、访问权限等信息

二级页表项保存物理块号以及存在位、访问权限、脏位、访问位等信息，用于完成虚拟地址到物理地址的映射

**Unit 9 虚拟内存**

**虚拟内存**

虚拟存储器指仅把作业的一部分装入内存便可运行作业，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系统

优点：使一个大程序可以在小内存中运行；用户在一个虚拟内存中编程，使编程不再受内存容量限制；内存可装入更多程序并发执行；提高系统的并发度和吞吐量，减少IO时间

**按需调页 demand paging**

定义：在需要的时候调入相应页，懒惰交换

优点：I/O时间低，所需内存减少，更快响应，并发度更高

改进：可预调页（避免进程初始时大量页错误）

新增有效-无效位，因为调页程序只把必需的页调入内存，所以用这个位区分哪些页在内存、哪些页在磁盘

修改位（脏位）：页是否被写过，如果换出要不要写回磁盘

访问位：最近有没有访问过

磁盘位置：记录该页在磁盘哪里

**处理页错误陷阱：**

①检查进程的内部页表，以确定该引用是合法的还是非法的地址访问

②引用非法，终止进程；引用有效但未调入页面，则现在调入

③找到一个空闲帧

④调度一个磁盘操作，以便将所需要的页调入刚分配的帧

⑤当磁盘读操作完成后，修改进程内部表和页表，表示该页已在内存中

⑥重新开始因陷阱而中断的指令

**什么是系统颠簸？阐述原因和解决方法**

定义：频繁的页调度行为称为颠簸。如果一个进程在换页上用的时间多于执行时间，那么这个进程就在颠簸

原因：马上要访问的页因为资源紧张被换出去，导致接下来需要频繁换入换出

解决方法：

①采用局部置换或优先级置换算法：局部置换算法可保证进程颠簸时不会从其他进程所占用的帧中选择置换帧，不会干扰其他进程；优先级置换算法保证进程只能从优先级低的其他进程中选择置换帧，减少从其他进程中调度帧

②工作集合模型：定义了进程执行的局部模型。局部是一个经常使用页的集合，进程由多个局部构成，执行时从一个局部移到另一个局部。为了防止抖动，需要提供给进程所需的足够多的帧。应根据进程的局部所占内存大小分配帧

**系统如何检测颠簸：**

系统通过 CPU低利用率 + 高缺页率 + 磁盘I/O过载 综合判断颠簸。

①CPU利用率骤降：CPU大部分时间用于处理页面调度，导致实际计算利用率极低

②磁盘I/O激增：大量进程因缺页而频繁读写交换区

③高缺页率：若远高于正常水平（如每秒数千次），可能发生颠簸。

**全局替换和局部替换**

1. 全局替换

特点：替换页面时，从所有进程的内存页中选择牺牲页（不考虑页面属于哪个进程）。

优点：

①系统整体效率高，优先淘汰最不活跃的页面，有更好的系统吞吐量。

②避免某个进程占用过多内存，公平性较好。

缺点：

①可能导致某些进程的页面被过度替换，影响其性能（“抖动”风险）。

②不能控制页错误率

1. 局部替换

特点：替换页面时，仅从当前进程的内存页中选择牺牲页（每个进程独立管理）。

优点：

①进程间隔离性好，避免相互干扰。

②适合多任务系统，保障关键进程的稳定性。

缺点：

①不能使用其他进程的不常用内存，所以会阻止一个进程。

②全局内存利用率较低（无法动态平衡负载）。

**页大小**

页表：对于给定虚拟内存，页越大，相应的页表和页的数量越小，所以较大页比较理想

内存利用率：较小的页可以更好的利用内存

页读写所需时间：为最小化I/O时间，需要较大页

页错误：为了降低页错误数量，需要较大页

**近似LRU页置换**

**附加引用位算法**

为位于内存内的每个表中的页保留一个8位的字节，包含该页在最近8个时间周期内的使用情况

**二次机会算法**

当要选择一个页时，检查其引用位。若为0，则直接置换；若为1，则给该页第二次机会并将引用位清零（可采用循环队列实现）

**增强二次机会算法**

将引用位和修改位作为一有序对来考虑。

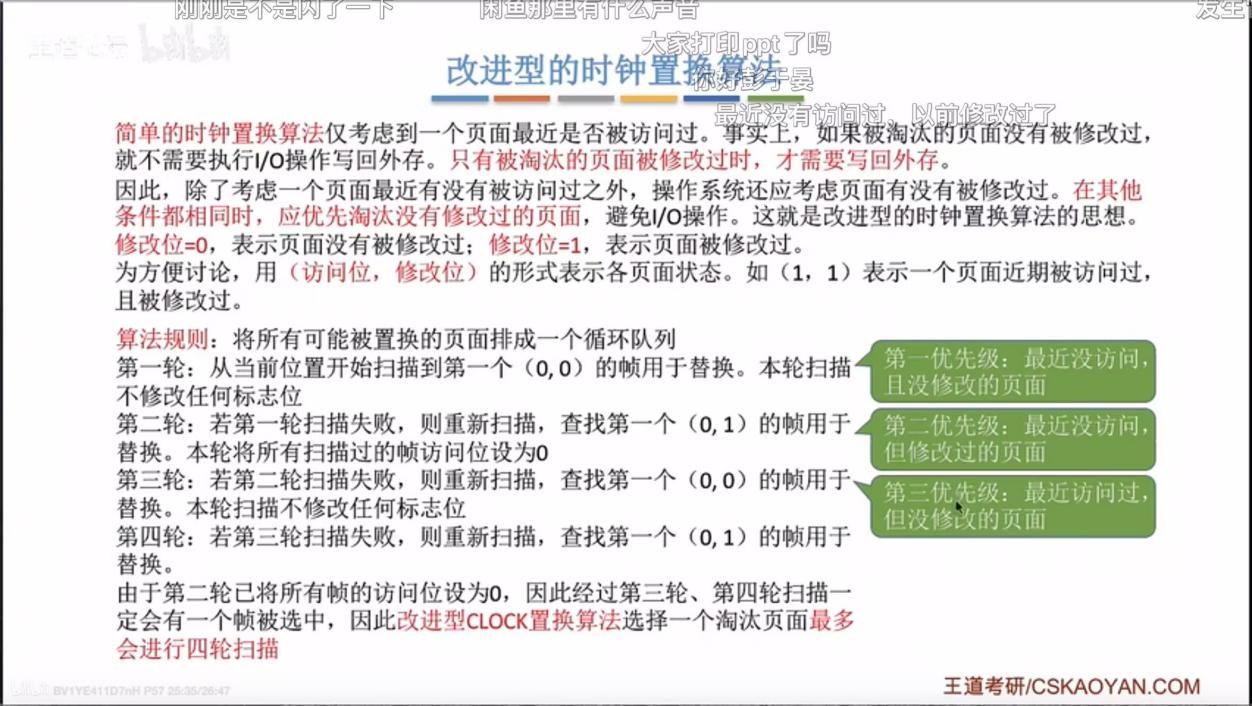
①（0，0）最近没有使用也没有修改——用于置换的最佳页

②（0，1）最近没有使用但修改过——不是很好，因为置换前需要将页写出到磁盘

③（1，0）最近使用过但没有修改——它有可能很快又要被使用

④（1，1）最近使用过且修改过——它有可能很快又要被使用，且置换前需将页写出到磁盘

（置换优先度：（0，0）>（0，1）>（1，0）>（1，1））



**Unit 10 文件系统**

**什么是当前目录？设置当前目录的好处是什么？**

当前目录是用户或进程当前所在的目录路径，系统默认在此目录下查找或操作文件（如不指定完整路径）。

优点：简化路径输入，提高操作效率，避免路径错误

**目录结构**

**单层结构目录：**所有文件都包含在同一层目录中

优点：便于理解，便于支持，搜索高效

缺点：命名不能重复，用户分组问题，不同用户间文件共享问题

**双层结构目录：**为每个用户建立独立目录，避免不同用户间命名重复问题。第一层目录叫主文件目录，第二层用户的目录叫用户文件目录

优点：命名问题解决，搜索高效

缺点：不同用户间文件共享问题；系统文件多次备份的问题

解决方案：定义一个特殊用户包含所有系统文件，本地UFD搜索不到文件时默认去特殊用户下搜索

**树状结构目录**：将目录结构扩展为任意高度的树，不再按照用户区分不同子目录，所以可以通过路径访问其他用户的文件

优点：易于管理，搜索高效

缺点：多用户对同一文件的共享问题

**无环图目录：**允许目录含有共享子目录和文件，共享文件不同于文件的复制，对于一个共享文件，只存在一个真正的文件。当多用户在一个组工作时，只需要把共享文件目录设为每个用户目录下的子目录即可完成分组共享

优点：实现了文件共享、悬空指针问题

**实现共享文件和目录**

1. 创建链接。链接是另一文件或目录的指针，当需要访问一个文件时，搜索目录，如果目录条目标记为链接，就可获得真正文件的名称
2. 在共享目录中重复所有共享文件的信息

**Unit 11 文件系统实现**

**FCB**

FCB是一种数据结构，包含了一个文件的所有结构信息，可以根据文件名获取该文件的FCB，后期对所有单一文件的操作所需的信息本质上都来源于FCB。FCB包括：

文件权限

文件日期

文件所有者、组、访问控制列表

文件大小

文件数据块

**文件系统包含结构：**

1. 卷的引导控制块：该卷如果有操作系统，则存放引导信息，没有则为空
2. 卷控制块：包括卷的详细信息：卷分区块数、块的大小、空闲块的数量和指针、空闲FCB的数量和指针
3. 每个文件系统的目录结构：存放目录。打开文件时这一块需要先加载入内存
4. 每个文件的FCB
5. 具体文件的数据块

**open()**

用户给open()传入文件的逻辑路径名，这时先将文件系统的目录结构加载进内存，根据文件名，操作系统会首先对系统范围内的打开文件表进行搜索

如果该文件已被其他进程打开，则直接将该进程的打开文件表中的指针指向系统范围打开文件表的这一项，同时系统打开文件表该文件引用计数加一

如果该文件不在系统范围文件表，说明该文件第一次打开，则对文件系统的目录表进行搜索，依次查找到叶节点，叶节点包含了一个该文件控制节点（inode）号，即控制节点的物理位置指针，将这个指针返回给用户，同时在系统范围打开文件表中新注册一行这个文件的信息，将该进程的打开文件表中的指针指向这条新信息

**read()**

通过open()操作返回的该文件索引节点号从进程的打开文件表中的指针找到系统的打开文件表中该文件的inode物理位置指针，将该FCB读入内存，通过FCB中文件存储类型和存储地址的信息算出数据块的存储地址，将数据块读入

**open()和close()的作用：**

open：在进程打开表中增加一个条目，并指向整个系统表的相应条目，文件打开计数器+1。初始化文件操作环境，获取文件描述符，管理文件状态

close：递减打开计数器。释放资源，确保数据写入磁盘，解除文件锁定

**文件分配方法**

**连续分配：**

每个文件占有一组连续的快，可用第一块磁盘地址和连续块的数量来定义

优点：实现简单，随机存取速度快，效率高，适合文件内容不进行变动的情况

缺点：难以扩展文件，需提前声明文件大小，产生外碎片，不能随机访问

**链接分配：**

每个文件是磁盘块的链表，磁盘块分布在磁盘的任何地方，目录块包含文件的第一块和最后一块指针

优点：实现简单，无外部碎片问题，文件可以扩展，不需要提前声明大小

缺点：每个文件快都有指针，占用空间，无法随机读取，可靠性差，一个中间数据块中指针的丢失会导致链断裂

**FAT（文件分配表）：**

每个卷的开始部分用来存储该卷的FAT，该表可以通过块号码进行索引。FAT使用和链表相似，目录条目含有文件首块的块号，根据块号索引的FAT条目包含文件下一块的块号，这条链一直继续直到最后一块，这块对应的FAT条目的值为文件的结束值，存在特殊结尾符

优点：实现对文件任意位置的随机访问

缺点：如果不用cache将FAT读入内存，每次访问FAT表会造成时间浪费

**索引分配：**

把所有指针放到统一的索引块上，上面存放着磁盘块地址的数组，索引块第i个条目指向文件的第i个块

优点：随机访问，可扩展，不需提前声明大小，无外碎片

缺点：浪费空间

**空闲空间管理**

1. 位图：将空闲空间表现为位图，每块空闲空间用一位表示，空闲为1，已分配为0

优点：实现简单，易产生连续空间分配的文件

缺点：如果磁盘块数多，位图也会很大

1. 链表：将所有空闲磁盘块用链表连接起来，并将指向第一个空闲块的头指针保存在磁盘的超级块上，同时也缓存在内存中

优点：查询时占用空间小

缺点：可靠性低，查找费时

1. 组：对空闲链表而言，可将n个空闲块的地址存在第一个空闲块中，之后的n-1块为空，最后一块包含另外n个空闲块的地址

优点：减少访问磁盘次数，适合大批量分配

缺点：实现稍复杂

1. 计数：建立一个类似分区内存管理中的分区表，每个表记录第一块的地址与第一块连续的空闲块的数量

优点：高效管理连续空闲空间，节省存储空间

缺点：不适合高度碎片化的磁盘

**操作系统是如何实现不同进程对同一个文件的访问和操作的？**

1. **文件描述符与文件表**：进程有自己的文件描述符，记录进程和文件的关联；操作系统有全局文件表，用来和描述符建立联系
2. **文件共享和锁机制**：多个进程打开同一个文件时可共享文件描述符和数据，但每个进程都有独立的文件指针；使用文件锁防止进程对同一文件的冲突操作
3. **虚拟内存与内存映射文件**：内存映射文件机制允许多个进程直接将文件映射到它们的虚拟内存空间

**操作系统是如何实现多个用户对同一个文件的访问和操作的？**

1. 文件权限控制机制：每个文件都带有访问权限位，用来表示文件所有者、所属组、其他用户对该文件的r（读）w（写）x（执行）权限
2. 访问控制列表：一些系统支持 ACL（Access Control List）机制，允许为每个用户或用户组单独设置权限，实现更精细的权限控制操作
3. 多目录指向同一文件：多个用户可通过不同路径访问同一个物理文件
4. 用户身份和文件系统安全模型：根据用户UID判断是否有权限访问文件
5. 并发访问控制机制

**Unit 12 大容量存储器**

**磁盘存取操作的内容和时间：**

1. 请求：读/写，磁盘地址（设备号，柱面号，磁头号，扇区号），内存地址（源地址、目标地址）
2. 时间：

①存取时间（磁盘定位时间，Positioning Time）=寻道时间+旋转延迟时间

②访盘时间=寻道时间+旋转延迟时间+传送时间

传送时间：从指定扇区读写数据的时间

排队延迟时间：发出I/O请求的进程须首先在队列中等待该设备可用

**SSTF不一定比LOOK好的原因：**

1. 饥饿问题：SSTF 优先服务最近的请求，可能导致边缘磁道的请求长期等待；LOOK 按方向扫描所有请求，公平性更高。
2. 高负载性能下降：SSTF 在请求密集时频繁改变磁头方向，增加寻道时间；LOOK 单向移动（类似电梯），减少抖动，适合高负载。
3. 旋转延迟影响：SSTF 仅优化寻道时间，但目标扇区可能位于磁盘远端，旋转延迟更高；LOOK 顺序扫描可能更接近扇区物理位置，减少总定位时间。
4. **SCAN比SCAN好的原因：**
5. 更公平的响应时间

SCAN（LOOK）：磁头双向移动，可能导致中间磁道的请求被优先处理，边缘磁道的请求等待时间较长。

C-SCAN：磁头单向移动，所有磁道的请求等待时间更均衡，避免饥饿问题。

2. 更低的平均寻道时间（高负载场景）

SCAN（LOOK）：在磁头折返时，可能导致频繁改变方向，增加寻道开销。

C-SCAN：单向扫描 + 快速回程，减少磁头抖动，适合高负载磁盘。

3. C-SCAN 的顺序访问模式更接近物理磁盘布局，减少旋转延迟（相比 SCAN 的来回扫描）。

**RAID：**

RAID技术将数据按位级或块级分散写入多个磁盘，多个磁盘可同时读写存取操作，实现数据的并行存取，提高系统性能；再结合数据冗余技术及校验技术，又提高系统的可靠性

**Unit 13 I/O**

**I/O子系统**

**I/O 子系统** 是操作系统中的一部分，负责处理与 I/O 设备相关的所有操作，包括数据传输、设备管理、错误处理、缓冲区控制、驱动程序接口等

**ppt原文：**

I/O的内存管理，包括缓冲（buffering）、缓存（caching），假脱机（spooling）

通用设备驱动程序接口

特定硬件设备的驱动程序

**引入原因：**

①设备差异性大，需要统一接口来屏蔽差异 ②实现多任务环境下的并发I/O管理 ③支持异步I/O，提高CPU利用率 ④优化性能，提高安全性可靠性

**缓冲buffer**

缓冲区是用来保存两个设备之间或设备和应用程序之间所传输数据的内存区域。使用缓冲的优点有：

1. 处理数据流中生产者和消费者之间的速度差异。使用双缓冲，一个缓冲接收生产者数据，另一个缓冲写给消费者，第一个缓冲写满后两个缓冲调换，完成生产者与消费者的分离解耦
2. 协调传输数据大小不一致的设备
3. 支持应用程序i/o的复制语义。当用作buffer的页面被修改后，复制一个新的页面，原有的页面用于write()，新的页面用于修改

**SPOOLing技术**

1. 作业执行前用慢速设备将作业预先输入到输入井中，称为预输入
2. 作业运行后，使用数据时，从输入井中取出
3. 作业执行不必直接启动外设输出数据，只需将这些数据写入输出井中
4. 作业全部运行完毕，再由外设输出全部数据和信息，称为缓输出

目的：解决独占设备的并发访问问题以提高设备利用率

**Buffering和Spooling的区别**

Buffering：用内存临时存储，数据量小，等待较短

Spooling：用磁盘临时存储，适合大批量数据，用户不用等待外设

**I/O控制方式**

**轮询方式**

CPU不断主动查询设备状态寄存器是否准备好读/写，一旦设备准备好，CPU就执行数据传输指令

优点：实现简单，不需要额外硬件支持

缺点：占用CPU时间，效率低

**中断方式**

CPU下达I/O指令后不再等待，去做别的事，当设备就绪产生中断信号，CPU会保存当前工作现场转去处理I/O，数据传输完成后返回原程序继续执行

优点：减少CPU忙等，响应速度快

缺点：中断频繁时可能产生中断开销，实现较为复杂

**DMA方式**

CPU向DMA控制器发出I/O指令（数据的源地址、目的地址、传输长度），DMA控制器独立完成数据搬运，搬运结束后再向CPU通知传输完成

优点：CPU减轻负担，速度快，提高系统并行度

缺点：硬件实现复杂，成本高；对内存总线占用较多