**第一章：概述**

**什么是操作系统？**

是一段一直运行在计算机上的程序

是资源的分配者

向上管理软件向下管理硬件

为用户提供良好接口

**中断的概念？**

中断指当出现需要时，CPU暂时停止当前程序的执行转而执行处理新情况的程序和执行过程。

**中断向量的概念？**

各种设备的中断处理子程序的地址数组

**什么是系统调用？**

由操作系统实现提供的所有系统调用所构成的集合即(Application Programming Interface，API)。是应用程序同系统之间的接口。

**内存是什么？**

内存是处理器可以直接访问的唯一的大容量存储区域，他通常是用被称为动态随机访问内存的半导体技术来实现的，是一组内存字的数组，每个字都有其地址。

**存储系统的速度**

寄存器>高速缓存>主存>电子磁盘 >磁盘> 光盘 >磁带

**什么是DMA及其工作原理？（课本第十页有解释）**

DMA即直接内存访问模式，简单来说，总线控制权在CPU“手上”，外连设备无权直接访问内存，需要CPU参与，但[DMA控制器](https://www.baidu.com/s?wd=DMA%E6%8E%A7%E5%88%B6%E5%99%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvuhPWmhc1mHT4PHnsmWIB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHmsP1bsPHnLn1nkP1mYPjcvPs" \t "http://zhidao.baidu.com/_blank)从CPU那“偷出”几个时钟来[控制总线](https://www.baidu.com/s?wd=%E6%8E%A7%E5%88%B6%E6%80%BB%E7%BA%BF&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvuhPWmhc1mHT4PHnsmWIB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHmsP1bsPHnLn1nkP1mYPjcvPs" \t "http://zhidao.baidu.com/_blank)，让外连设备可以直接访问内存，这样外连设备的读写就不需要CPU参与，降低了CPU的占用率。**（通俗解释版本）**

是一种不经过[CPU](http://baike.baidu.com/view/2089.htm)而直接从[内存](http://baike.baidu.com/view/1082.htm" \t "http://baike.baidu.com/_blank)存取数据的数据交换模式，在DMA模式下，CPU只须向DMA控制器下达指令，让DMA控制器来处理数据的传送，数据传送完毕再把信息反馈给CPU，这样就很大程度上减轻了CPU资源占有率，可以大大节省系统资源。**（答题版本）**

**什么是****多道程序系统？（课本11页）**

多处理器系统也称并行系统或者是紧耦合系统，这类系统有多个紧密通信的CPU，他们共享计算机总线，有时还有时钟、内存和外设等。

**多道程序系统的优点？**

1. 增加吞吐量
2. 规模经济
3. 增加可靠性

**非****对称多处理系统**

每个处理器都有各自特定的任务，一个主处理器控制系统，其他处理器或者向主处理器要任务或者完成预定任务

**对称多处理系统**

每个处理器都要完成操作系统的任务，所有处理器对等，没有主从关系

**什么是多道程序设计？**

是在计算机内存中同时存放几道相互独立的程序，使它们在管理程序控制之下，相互穿插的运行。 两个或两个以上程序在计算机系统中同处于开始到结束之间的状态。

**目的**

是为了提高CPU的利用率，充分发挥计算机系统部件的[并行性](http://baike.baidu.com/view/149970.htm" \t "http://baike.baidu.com/_blank)

**什么是分时系统？**

分时系统是多道程序设计逻辑上的一个延伸。把处理机时间划分成很短的时间片轮流地分配给各个联机作业使用。如果某个作业在分配给他的时间片用完之前计算还未完成，该作业就暂时中断，等待下一轮继续计算。此时处理机让给另一个作业使用。此时，多个用户分享使用同一台计算机。多个程序分时共享硬件和软件资源，分时系统具有多用户性和交互性。**（宝宝结合课本15页和百度百科加百度知道暖心归纳的）**

**作业池**

在分时和多道程序设计中需要在存储器中同时保存多个作业，但主存较小不能容纳太多作业，所以这些作业开始储存在磁盘上，这个储存地址叫作业池

**作业调度**

在作业池中选择作业进入内存，这样的决策叫做作业调度

**CPU调度**

如果有多个任务要执行，系统必须做出选择让其中一个执行，这个决策叫做CPU调度

**双重模式操作（重点中的重点)**

指用户模式和（内核模式或者系统模式或者特权模式）

模式位的设立是用来表示当前模式（1代表用户模式，0代表内核模式）

特权指令

特权指令指具有特殊权限的指令。这类指令只用于操作系统或其他系统软件，一般不直接提供给用户使用

它主要用于系统资源的分配和管理，包括改变系统工作方式，检测用户的访问权限，修改[虚拟存储器](http://baike.baidu.com/view/51844.htm)管理的段表、[页表](http://baike.baidu.com/view/2143270.htm)，完成任务的创建和切换等。

**常见的特权指令**

有关对I/O设备使用的指令 如启动I/O设备指令、测试I/O设备工作状态和控制I/O设备动作的指令等。

有关访问程序状态的指令 如对[程序状态字](http://baike.baidu.com/view/5499368.htm" \t "http://baike.baidu.com/_blank)（PSW）的指令等。

存取特殊[寄存器](http://baike.baidu.com/view/6159.htm)指令 如存取中断寄存器、时钟寄存器等指令。

转换到用户模式就是一个特权指令（**课本17页**）

**第二章：操作系统结构**

**系统调用（重点）**

课本41页图

系统调用类型分为五大类：

进程控制、文件管理、设备管理、信息维护、通信

**操作系统的结构**

1. **简单结构**

可以访问硬件，不稳定

1. **分层方法**

系统模块化，分层法的优点在于构造和调试的简单化

1. **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者是用户程序，

微内核通常包括最小的进程和内存管理以及通信功能

优点：便于扩充操作系统，所有的新服务可以在用户空间增加，因而不需要修改内核。

**第三章：进程**

**什么是进程（也叫作业）？**

进程是执行中的程序，是具有某一功能的程序，是在某一数据集上的一次执行过程，是资源分配和调度的独立单元。还包括有程序计数器、处理器寄存器、进程堆栈段等。

**进程的特性：**并发性和动态性

进程的状态：

**重点：（73页进程状态图）**

新的、运行、等待、就绪、终止

**进程控制块（PCB）**

**重点图（74页+82页代码）(产生中断时PCB怎么活动，也就是上下文切换）**

包括：**进程状态**、进程编号、**程序计数器**、寄存器...

**3.4（85页——90页）全是重点**

**进程间通信**

进程间关系分为独立进程和协作进程

协作进程分为共享内存和消息传递

While(true)

While(((in+1)%size)==out)

;

Buffer[in]=nextproducer;

In=(in+1)%size;

}

生产者进程

While（true)

While(in==out)

;

Nextconsumer=buffer[out];

Out=(out+1)%size;

}

消费者进程

**第四章：线程**

**什么是线程？**

线程是CPU使用的基本单元，它由线程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、数据段和其他资源。

**多线程的优点？**

1. 响应度高
2. 资源共享
3. 经济
4. 多处理器体系结构的利用（增加了并发功能）

**什么是线程？**

线程是CPU使用的基本单元，它由线程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、数据段和其他资源。

**如果直接使用进程并发，会产生什么问题？**

进程创建很耗时间与资源，使系统性能下降

**进程与线程的对比**

**从调度方面：**

线程作为调度和分派的基本单位，而进程作为资源拥有的基本单位

**从资源方面：**

进程间相互独立，同一进程的各线程间共享资源。线程自己不拥有系统资源，某进程内的线程在其它进程不可见。

**从并发方面：**

在引入线程的操作系统中，不仅进程之间可以并发执行，而且在一个进程中的多个线程之间亦可以并发执行，使得操作系统具有更好的并发性，从而能更加有效地提高系统资源的利用率和系统的吞吐量。但进程并发代价大，线程代价小

**从执行方面：**

忘了....(哈哈..如果知道补充好告诉偶哦）

**用户级线程**

指不需要内核支持而在用户程序中实现的线程，其不依赖于操作系统核心，应用进程利用线程库提供创建、同步、调度和管理线程的函数来控制用户线程。

**内核级线程**

由操作系统内核创建和撤销。内核维护进程及线程的上下文信息以及线程切换。

**以下是用户级线程和内核级线程的区别**

用户级线程的程序实体是运行在用户态下的程序，而内核支持线程的程序实体则是可以运行在任何状态下的程序

**内核线程的优点:**

当有多个处理机时，一个进程的多个线程可以同时执行

**缺点：**

由内核进行调度

**用户进程优点：**

线程的调度不需要内核直接参与，控制简单

可以在不支持线程的操作系统中实现

代价比内核线程小

**缺点：**

多个处理机下，同一个进程中的线程只能在同一个处理机下时分复用

**线程池**

为了限制线程的数量，在进程开始时，先创建好一定数量的线程，放到池中，等待服务的使用

**优点：**

先创建好了线程，处理速度快；

线程池限制了线程的数量，对那些不能支持数量大线程并发的系统非常重要；

**第五章：CPU调度**

**CPU调度的背景和概念（重点）**

为了解决单处理器CPU利用率低的问题，采用多道程序设计，此时会有多个进程在内存中，当其中一个进程执行一段时间后必须等待时，此时操作系统会夺走CPU的使用权交给另一个进程，这就是CPU调度

**分派程序（重点）**

其实就是将CPU使用权交给短期调度选择的进程的过程

**功能**

切换上下文

切换到用户模式

跳转到用户程序的合适 位置，以重启程序

**周转时间：**进程完全结束的时间减去进程到达的时间

**调度算法（重点）**

**FCFS**

**Easy......**

**了解护航效应（convoy effect）的概念**

**SJF（最小等待时间）**

1. **非抢占**

**若进程到达时间都是同一时间：**

则操作系统会直接根据作业时间的大小进行选择（必须完整的执行完一个进程再换到另一个进程）

**若进程到达时间都不是同一时间：**

则操作系统在最短时间作业选择的时候，开始只考虑已经到了的进程，当执行完一个进程后（必须完整的执行完一个进程再换到另一个进程），又有进程到达，则把该进程纳入考虑范围内，继续进行最短时间作业选择...

1. **抢占**

**若进程到达时间都是同一时间：**

则与非抢占是一样的效果.....

**若进程到达时间都不是同一时间：**

则执行抢占的方法（参考141页的例子，有抢占和非抢占）

**优先级调度算法**

**若进程到达时间都是同一时间：**

直接按照优先级进行选择

**若进程到达时间不是同一时间：**

开始只考虑已经到了的进程，当执行完一个进程后（必须完整的执行完一个进程再换到另一个进程），又有进程到达，则把该进程纳入考虑范围内，继续按照优先级进行选择...

**轮转法调度（最快响应）**

**Easy...不做详解**

**多级队列调度（了解概念即可）**

进程分配到独立的队列中，每个队列有自己独立的调度算法，只能在这个队列中

**多级反馈队列调度（了解概念即可）**

进程分配到独立的队列中，每个队列有自己独立的调度算法，但进程可以根据执行效果在不同队列中流动

**了解概念即可**

**亲和性**

课本148

**负载平衡**

将工作平均分配到多个等待的处理器中，防止其中一个处理器压力山大...

**确定模型**

**1、分析评估法**

用一套评估方法去给模型打分

**2、确定模型法**

直接用数据去测试模型，看看结果

1. **：进程同步**

**临界区问题（重点 critical section，理解概念）**

互斥、前进、有限等待（对应洋文一定要知道呀）

（mutual exclusion、progress、bounded waiting）

**硬件同步（了解即可）**

**信号量（重点中的重点，算法大题应该就是它）**

Wait（s) signal (s){

While(s<=0) s++;

; }

S--;

}

**该方法常用，但出现了**

**实现（重点）**

**忙等待：**

就是当有一个进程在临界区的时候，任何试图进入其临界区的进程都必须在进入代码连续循环

**三个重点内容 经典同步问题**

1. **有限缓冲问题**

该算法适用于解决生产者消费者问题，一般需要定义一个缓冲区大小的变量empty 初始化为n ,一个full初始化为0，表示正在缓冲区的个数，mutex初始化为1，用于实现互斥，用这三个变量就能解决这类问题 177页

1. **读者写者问题**

课本中程序同时达到目的为

1. 没有写的时候，后续读的直接进入
2. 有一个在写，后续的读和写都得等待
3. 有一个在读，后续的读继续读，写进入等待

**具体怎么实现的看178页代码啦~**

1. **哲学家进餐问题**

**简单来说，就是一个****wait（a[i]）;一个wait（a[i+1]%5）**

这两个wait后如果哲学家进入了临界区，就表示这个哲学家此时占用了他相邻的两根筷子，别人就不能用了，知道signal(a[i]) 和signal(a[i+1]%5) ，说明吃完了放下筷子进入思考...

**管程（了解）**

1. **：死锁**

**死锁和死锁状态的概念**

在多道程序环境下，多个进程可能竞争一定数量的资源。某个进程申请资源，如果这时资源不可用，那么该进程进入等待状态。如果所申请的资源被其他等待进程占有，那么该等待进程可能再也无法改变其状态。这种情况称为**死锁 deadlock**。

当一组进程的每个进程都在等待一个事件，而这一事件只能有这一组进程的另一个进程所引起，那么这组进程就处于**死锁状态**。

**必要条件**

互斥、占有并等待、非抢占、循环等待

**资源分配图**

没有环，一定没有死锁；有环，只是有可能有死锁（若每个资源只有一个实例，则有环就必然死锁，这也是 为什么只有一个实例的时候，可以采用资源分配图算法，而有多个的时候一般使用银行家算法）

**死锁处理方法**

1. **死锁预防**

让四个必要条件其中一个不满足就行，在必要条件之前加上否定就行

1. **死锁避免**

**安全序列**

系统能按某个顺序给每个进程分配资源而能避免死锁，这个顺序就是安全序列

安全状态无死锁，不安全状态只是可能导致死锁

安全序列不是唯一的，满足条件即可，但考试基本老师会给只有唯一的安全序列的套路，哈哈，便于批卷

**资源分配图算法：**

用于每个资源有单个实例的情况

**银行家算法：**

安全性算法

就是寻找能给所有进程分配资源的一个安全序列

**课本222页的举例是必考题**

资源请求算法

当一个进程请求资源的时候，先判断有没有那么多给它，如果有，再判断如果给它，新状态下有没有安全序列

**死锁恢复办法**

进程终止

资源抢占

1. **：内存管理**

**内存概念：**

内存是处理器可以直接访问的唯一的大容量存储区域，他通常是用被称为动态随机访问内存的半导体技术来实现的，是一组内存字的数组，每个字都有其地址。

**输入队列：**也叫作业池，在磁盘上等待调入进内存的进程

CPU产生的地址叫逻辑地址，也叫虚地址、可重定位地址

**MM****U：**内存管理单元，完成虚地址到物理地址的映射

逻辑地址+基地址（存在于重定位寄存器也叫基地址寄存器中)=物理地址

**动态加载：课本240**

**滚入、滚出了解概念**

**连续内存分配**

**外部碎片：**

进程块之间的空闲内存

**内部碎片：**

分配给进程的内存大于它所需要的，多出来的那部分

**重点（分页）**

**分页使得内存非连续**

**物理内存中的块叫帧，逻辑内存上的块叫页**

**将逻辑地址通过页表映射到物理内存地址的计算：**

1. 首先找到逻辑地址的页号p(也就是在逻辑地址上是第几块）
2. 用找到的p通过页表直接找到物理地址上帧的块数m
3. 页偏移是指这个逻辑地址在其所在的那个页块偏移的数，不是从最开始数，只是从其所属的块开始数，在第一就偏移0....
4. m乘以帧的大小加上页偏移就是物理内存地址

**分页技术不会产生外部碎片**

**内部碎片的计算：**

比如 100大小的进程

页大小是30（帧和页一样大也是30），需要3个帧，但还有10的内存，也需要分配一个帧，所以产生了30-10=20的内部碎片

**TLB：**

**转换表缓冲区**

**有效内存访问时间：和求期望类似**

**效内存访问时间=概率一\*时间一+概率二\*时间二...自己体会**

**页表结构（了解）**

**分段（重点内容）**

**也是一种非连续分配**

**其逻辑地址由<segment-number,offset>组成**

**段表的概念 还有图（课本261页)不做解释了...**

**段表的目的：**

将二维的用户定义地址映射为一维地址

.页表的作用是实现从页号到物理块号的地址映射。

在页式管理系统中，用户程序中使用的地址称为 逻辑地址 ，实际访问主存时由系统将它转化为 物理地址 。

分页管理是把内存分为大小相等的区，每个区称为页帧(或页框)，而把程序的逻辑空间分为若干页，页的大小与页帧的大小 相等 。

在分页存储管理中，为了加快地址变换速度，页面大小的值常取2的整数次幂。

在请求式分页系统中，被调出的页面又立刻被调入，这种频繁的调页现象称为颠簸。

分段管理中，若逻辑地址中的段内地址大于段表中该段的段长，则发生 地址越界中断。

段页式存储管理中，每道程序都有一个 段 表和若干个 页 表。

页式管理系统的逻辑地址结构由 页号 和 页内位移 组成。

分段管理中的地址映射过程是：首先找到该作业段表的 起始地址 ，然后根据逻辑地址中的 段号 去查找段表得到该段的内存起始地址，再与逻辑地址中的 段内位移 相加得到物理地址。

请求分页存储管理也称为动态页面管理，不是把一个进程映象的所有页面一次性全部装入内存，而只装入一部分，其余部分在执行中动态调入。

在段页式管理中，逻辑地址分解为段号、页号、页内位移 三部分。

**第九章：虚拟内存**

**背景（重点）**

**将用户看到的逻辑内存和物理内存分开**

只将部分程序放入内存就能执行

逻辑地址空间可以比物理内存空间大

许多情况下整个程序不是必须的

**优点：**比实际空间大

不必担心内存空间的限制

允许地址空间被多个进程共享

**虚拟内存的实现：**

按需调页（demand paging）

顾名思义，在需要某段程序的时候将其调人内存的技术

**275页重点内容，老师说应该背下来**

**纯粹按需调页**

**支持按需调页的硬件**

页表

次级存储器

**产生页错误有两种情况**

1. 不允许访问
2. 未调入内存，需要在后备存储中将其调入

**写时复制（了解原理）**

**页面置换（重点）**

**基本页置换**

如果没有空闲帧就查找当前没有使用的帧，并将其释放（牺牲帧），然后将其内容写到交换空间，并改变页表和帧表

**引用串的计算（重点）：**

会给定一组地址

如

0100,0432,0103,0104，0890....

如果页大小是100B

则将所有地址除以页大小100，向下取整后

得1、4、1、1、8

第三步，将相邻的相同的数合为一个就行（也就是将两个相邻相同的1，保留一个就行）

得1、4、1、8

**FIFO页置换：**

替换已经在内存的最久没有被使用的，也就是最早进入的

这个...不好说明，很简单，课本284页

**最优置换（OPT)：**

替换最久将会被使用的

虽然名字为最优，但后期预测其实是很难的，所以2很多时候

课本285页

**LRU页置换：（重点）**

反正我感觉和FIFO一样....这个再说

**帧分配（了解即可）**

**系统颠簸（thrashing 重点中的重点）**

**什么是颠簸？产生颠簸的原因是什么？**

（1）颠簸是由于内存空间竞争引起的。当需要将一个新页面调入内存时，因内存空间紧张，不得不将一个旧页面置换出去，而刚刚置换出去的旧页面可能又要被使用，因此需要重新将它调入。若一个进程频繁地进行页面调入调出，势必加大系统的开销，使系统运行效率降低。通常称这种现象为该进程发生了颠簸。（2）产生颠簸的原因主要有：系统内的进程数量太多，致使一个进程分得的存储块过少；系统采取的置换算法不够合理。

**工作集合模型：**

没听，那节课后边睡着了

1. **：文件（这章以后基本只考概念）**

**文件属性：**

名称、标识符、类型、位置、大小、保护、时间、日期和用户标识

**文件操作：**

创建文件、写文件、读文件（维护一个读位置的指针）、在文件中重定位（也叫文件寻址）、删除文件（释放空间，也就是全部删除）、截短文件（删除内容保留属性）

首次使用文件时，调用open（），操作系统维护一个包含打开文件的信息表（打开文件表）

系统调用open（）通常返回一个指向打开文件表中的一个条目的指针。

**解决多个进程打开同一文件的问题：**

操作系统采用两级内部表，分别是进程的表和整个系统的表

**四个概念324页**

文件指针

文件打开计数器

文件磁盘位置

**访问权限**

**文件类型（了解）**

**访问方法（重点）**

**顺序访问**

磁带类型，文件信息按顺序一个一个处理，并且自动前移文件指针，以跟踪IO位置

适用于顺序访问设备，也适用于随机访问设备

**直接访问（相对访问）**

基于文件的磁盘模型

文件由固定长度的逻辑记录组成，以允许程序按任意顺序进行快速读写

对访问大量信息极为有用（数据库经常使用）

**其他访问方式**

在直接访问的基础上，建立一个文件索引（索引包里包括文件块的指针），查找文件时，首先搜索索引，再根据指针直接访问文件

**目录结构**

**存储结构**

**磁盘分区：**

一个磁盘上装有多个文件系统，或一部分用于文件系统而另一部分用于其他地方，如交换空间或非格式化的磁盘空间

**卷：**

带有文件系统的磁盘分区叫卷

**目录概述**

记住相关操作，除了基本的创建、删除...还有跟踪文件系统，也就是定期备份整个文件系统到磁盘

**单层目录结构**

所有文件位于同一目录

特点：

便于了解和支持

缺点：

随着文件数目的增加，单层目录不能重名，会使用户难以记住所有文件名称

**双层目录结构**

第一层是**主文件目录（MFD)**，也就是用户目录，每个用户目录都有自己飞**用户目录文件（UFD)**，也就是第二层;

当一个用户引用特定文件时，只需要搜索他自己特定的UFD，不同用户可具有相同文件名

双层目录结构其实就是高度为2的树

**树状结构目录**

将双层目录结构继续扩展

目录包括一组文件和子目录，每个子目录有相同结构（树），创建和删除目录条目都需要调用特定的系统调用

通常情况下每个进程都有一个当前目录，进程需要文件时首先查看当前目录（也就是先看相对地址），如果没找到，再根据指定路径或者改变当前目录

**绝对路径名：**

从树根开始给出目录名知道文件

**相对路径名：**

从当前目录开始定义路径

**无环图目录**

树状结构目录的一个扩展，树状结构目录不允许共享文件和目录，无环图目录可以

实现共享的方法

如Unix采用创建一个链接，实际上是另一文件或目录的指针

**通用图目录**

339页了解一下

**文件系统安装（了解）**

1. **：文件系统实现**

**分层设计的文件系统：**

**容易考选择题**

应用程序——逻辑文件系统——文件组织系统——基本文件系统——I/O控制

——设备

**概述**

**引导控制块**，包括系统从该引导操作系统所需要的信息，通常为均卷的第一块，

UFS称之为引导快，NTFS为分区引导扇区

**每个卷的控制块**，包含卷或者分区的详细信息，如分区块数，块大小，空闲块的数量和指针，UFS称之为超级块，在NTFS中它存在于主控文件表中

**系统范围内的打开文件表**

包含每个发开文件的FCB副本和其他信息

**单个进程的打开文件表**

包括一个指向系统范围内已经打开卷文件表中合适**条目的指针**和**其他信息**

**考点**

**创建文件的主要过程**

应用程序调用逻辑文件系统，逻辑文件系统知道目录结构形式，它将分配一个新的FCB，然后系统将相应的目录信息读入内存，用新的文件名更新该目录和FCB

**考点**

**打开和关闭文件的过程**

356页

分区安装（不考）

虚拟文件系统（不考）

**目录实现**

**目录的实现方法**

最为简单的目录实现方法是使用存储文件名和数据块指针的线性列表（数组、链表等）

容易实现

但运行费时

采用线性搜索来查找特定条目（缺点）

许多操作系统采用软件缓存来存储最近访问过的目录信息

Hash表：采用Hash数据结构的线性表

减少了目录搜索时间

碰撞：两个文件名哈希到相同的位置

哈希表的最大困难是其通常**固定的大小**和**哈希函数对大小的依赖性**

**分配方法**

**考选择题**

分配方法指的是如何为文件分配磁盘块，常用的分配方法有以下三类

**连续分配**：每个文件占据磁盘上的一组连续的块

特点：1简单 － 只需要记录文件的起始位置（块号）及长度。2访问文件很容易，所需的寻道时间也最少

存在的问题：1为新文件找空间比较困难（类似于内存分配中的连续内存分配方式）文件很难增长

**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。

优点：1简单 － 只需起始位置2.文件创建与增长容易。

缺点：1.不能随机访问2.块与块之间的链接指针需要占用空间3. 存在可靠性问题

簇：将多个连续块组成簇，磁盘以簇为单位进行分配

**索引分配**：将所有的数据块指针集中到索引块中。

1.索引块中的第i个条目指向文件的第i块。2目录条目包括索引块的地址

索引分配支持直接访问，且没有外部碎片问题

索引块本身可能会浪费空间

链接方案：一个索引块通常为一个磁盘块。对于大文件，可以将多个索引块链接起来。

多层索引：类似于内存的间接寻址方式（一级、二级间接…）

组合方案：如Unix的inode

**空闲空间管理（了解）**

**效率与性能（不考）**

后面都不考

**十二章：大容量存储器的结构**

简介和磁盘结构

了解

磁盘调度

了解

**1.磁盘调度算法有哪些？每种方法的优缺点。**

答:FCFS、SSTF、扫描(SCAN)算法 、循环扫描(CSCAN)算法、look调度

FCFS：先来先服务，它根据进程请求访问磁盘的先后次序进行调度。

SCAN:扫描算法，磁头不停的往复运动，由边缘至中心然后返回，沿途执行已经到来的访问。

CSCAN：循环扫描算法，在SCAN算法的基础上规定磁头单向移动。

在朝一个方向移动时会看是否有请求

磁盘管理

不考

交换空间管理

了解

**考点**

**RAID结构（考点）**

**磁盘冗余阵列**

一个磁盘损坏并不会导致数据的丢失，这里的多种磁盘组织技术，统称为磁盘冗余阵列,用于提高性能和可靠性

**镜像**

引入冗余复制整个磁盘，最为昂贵

通过并行处理改善性能、

**位级分散**

通过在磁盘上分散数据，可以改善传输率，数据分散是在多个磁盘上分散每个字节的各个位，这种分散就是位级分散。

**RAID级别**

镜像法和分散法的结合使用

其他知识点不考

**十三章：I/O**

**考点**

**Io硬件**

**426页图**

轮询和中断

**直接内存访问（年年考）**

**DMA**

在前面介绍过DMA工作过程

书432页也有

主机向内存中写入DMA命令块，包含源地址指针、目的指针、字节数等等信息，CPU将其写入DMA控制器后，DMA继续下去直接操作内存总线，无需CPU帮助

**IO应用接口**

**考点，选择题**

属于操作系统的是设备控制器以上，不包括设备控制器这一层

435页图

缓冲（考点）

**什么是缓冲什么是缓存？**

buffer与cache操作的对象就不一样。

buffer缓冲是为了提高内存和硬盘或其他I/0设备之间的数据交换的速度而设计的。

cache缓存是为了提高cpu和内存之间的数据交换速度而设计。

**为什么引入缓冲（目的是什么？）**

答：（1） 缓和CPU与I/O设备间速度不匹配的矛盾（2） 减少对cpu的中断频率，放宽对cpu中断响应时间的限制（3）提高cpu和I/O设备之间的并行性

**试从调度性，并发性，拥有资源和系统开销几个方面对线程与进程进行比较**

**调度**  
● 在传统的操作系统中，作为拥有资源的基本单位和独立调度、分派的基本单位都是进程。  
● 在引入线程的操作系统中，把线程作为调度和分派的基本单位，而进程作为资源拥有的基本单位，把传统进程的两个属性分开，使线程基本上不拥有资源，这样线程便能轻装前进，从而可显著地提高系统的并发程度。  
● 在同一进程中，线程的切换不会引起进程的切换，但从一个进程中的线程切换到另一个进程中的线程时，将会引起进程的切换。

并发性  
在引入线程的操作系统中，不仅进程之间可以并发执行，而且在一个进程中的多个线程之间亦可并发执行，使得操作系统具有更好的并发性，从而能更加有效地提高系统资源的利用率和系统的吞吐量。

3) 拥有资源  
 ● 不论是传统的操作系统，还是引入了线程的操作系统，进程都可以拥有资源，是系统中拥有资源的一个基本单位。  
 ● 一般而言，线程自己不拥有系统资源(也有一点必不可少的资源)，但它可以访问其隶属进程的资源，即一个进程的代码段、数据段及所拥有的系统资源，如已打开的文件、I/O设备等，可以供该进程中的所有线程所共享。

4) 系统开销  
● 在创建或撤消进程时，系统都要为之创建和回收进程控制块，分配或回收资源，如内存空间和I/O设备等，操作系统所付出的开销明显大于线程创建或撤消时的开销。  
● 就切换代价而言，进程也是远高于线程的。此外，由于一个进程中的多个线程具有相同的地址空间，在同步和通信的实现方面线程也比进程容易。在一些操作系统中，线程的切换、同步和通信都无须操作系统内核的干预。