操作系统期末

第一章

第一部分 基本概念

操作系统的定义（也即 OS 提供的服务）（从两方面：用户的观点，OS 给用户提供什么样的服务；计算机系统方面：为计算机系统提供什么样的服务）：

计算机系统中的一个系统软件，一些程序模块的集合；管理和控制计算机系统中的软件和硬件资源；合理地组织计算机工作流程；在计算机与其用户之间起到接口作用。（PPT）（没有公认的标准定义）

四大核心功能：进程管理、内存管理、文件管理和设备管理。

第二部分 操作系统提供的接口

三类接口：命令行程序、图形用户界面、系统调用。其实就是操作系统提供的服务，给用户提供界面：命令行界面和图形用户界面，给程序提供系统调用。

这三个接口里面最重要的就是系统调用，系统调用是提供一个什么样的接口，在哪种情况下可以使用？

系统调用提供操作系统服务接口

系统调用的实现：函数库提供了系统调用接口

操作系统内核函数

- 用户程序通过访管指令，请求操作系统提供某种服务。
- 当 CPU 执行访管指令时，引起访管中断；
- 处理器保存中断点的程序执行上下文环境（PSW、PC 和其它寄存器），CPU 切换到管态(硬件自动完成)。
- 中断处理程序开始工作，调用相应的系统服务；
- 结束后，恢复被中断程序的上下文环境，CPU 恢复为目态，回到中断点继续执行。

那么涉及这个系统调用的时候又涉及到**操作系统模式的切换**，就是**从用户态转移到这个内核态**，然后去进行实施具体的这个操作的，那么这个整个的一个过程哈，就是它是怎么样从一个模式切换到另外一个模式的。这个时候我们在系统调用这一块儿也是需要了解掌握的哈。

第三部分 操作系统的双模式

OS 和多个进程共享计算机系统的软硬件资源，为确保 OS 的正确运行，区分 OS 代码和用户代码的运行，多数系统将处理器划分为管态和目态。

根据运行程序对资源和机器指令的使用权限划分

当用户应用通过系统调用，请求操作系统服务时，系统必须从用户模式切换到内核模式。

计算机硬件可以通过一个模式位来表示当前状态：内核模式（0）和用户模式（1）

内核模式（监视模式、系统模式、特权模式、管态）

是操作系统运行时的状态，有较高的特权级别。

可以执行所有的指令（包括**特权指令**）、使用所有的资源，并具有**改变处理器状态**的能力。

用户模式（目态、普态、普通态、用户态）

是用户程序运行时的状态，有较低的特权级别。

禁止使用特权指令，不能直接使用系统资源与改变 CPU 状态，只能访问用户程序所在的存储空间。

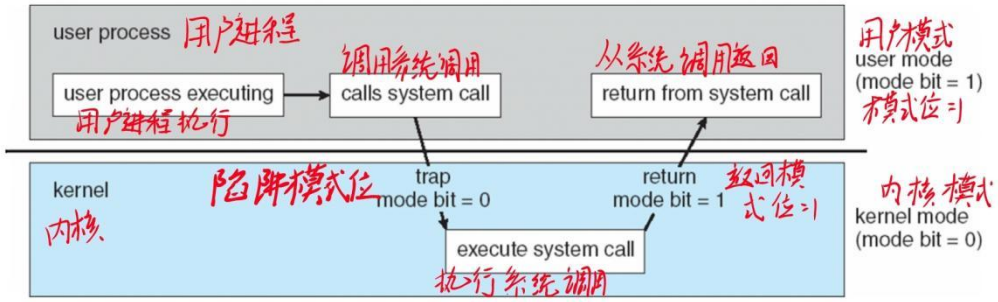


图 1-10 用户模式到内核模式的转换

特权指令和非特权指令

概念：将可能引起损害的机器指令作为特权指令，并且硬件只有在内核模式下才允许执行特权之指令。

受保护的指令（特权指令）只有操作系统才有权使用。比如：访问某些硬件资源的指令（如对 I/O 设备直接访问的指令，如磁盘、打印机等）；对内存管理状态进行操作的指令；某些特殊的状态为的设置指令、停机指令、切换到用户模式的指令

CPU 通过程序状态字 PSW（Program Status Word）来判断当前运行的程序是系统程序还是用户程序。PSW 包括：**状态码：管态还是目态，决定是否可以使用特权指令或拥有其他的特殊权力**、条件码：指令执行后的结果特征、中断屏蔽码：是否允许中断

CPU 工作状态之间如何转换：

管态→目态 ：设置 PSW

目态→管态 ：用户程序无法直接修改程序状态字，需通过**系统调用**

最后部分 了解操作系统的类型

那么现在我们接触到的操作系统呢可能有这种**虚拟机器**哈，虚拟的概念呢是**今年在考研大纲里面新加入的概念**，所以说我们也有可能去以这个考研去进行接轨哈。就可以有什么是虚拟这个技术？什么是虚拟机器？那虚拟这个操作系统又是大概是怎么样的？大家**有所了解**就可以。

Page28 1.11.6

另外部分

分时操作系统、实时操作系统、单用户操作系统、多用户操作系统这样的种种的不同的分

类，那就是我们现在目前目前常用的比如说 **windows、linux** 它们分别是对应的哪一种操作系统类型，这个也是大家去做一个大概的**了解**。

那么另外还有一个部分

就是由于我们这个工程认证的要求呢,它也是会考大家**一部分扩展**的就是关于**实事的这个进展**，上节课也都给大家提到了，就是比如说我们目前国产操作系统**发展到什么样的地步了**。然后其他的操作系统现在发展到什么地步了？可能会问大家一些**科普性的内容常识**性的问题哈，这个不会说的很深，但是就是让大家**对目前最先进的这个技术有有大概的一个了解**。

那这个就是第一章的内容，这部分呢应该都是以**小题**的这种概念的形式出现的。

第三章 进程管理

第一部分 进程的定义、基本特征

进程和程序之间的关系或者区别：

程序本身不是进程。程序只是**被动**实体，如存储在磁盘上包含一系列指令的文件（经常成为可执行文件）。相反，进程是**活动**实体，具有一个程序计数器用于表示下个执行命令和一组相关资源。当一个可执行文件被加载到内存是，这个程序就成为进程。

进程是一个动态的概念,程序是一个静态的概念。

程序是指令的有序集合，进程是程序在处理机上的一次执行过程；

程序是永久的，而进程是有生命期的、暂时的。

进程更能真实地描述并发，而程序不能。

同一个程序在执行过程中也可以产生多个进程

这个是大家首先要了解的哈，也是我们为什么要引入这个进程？

为解决如何称呼所有 CPU 活动问题。

进程的定义（教材）：进程是执行的程序，包括程序计数器、寄存器、堆、栈、数据段。

进程的定义（描述性）：并发执行的程序在执行过程中分配和管理资源的基本单位。

进程的特征

结构特征

进程：由程序段、数据段及进程控制块构成。

动态性

由“创建”而产生，由“调度”而执行；由得不到资源而阻塞；由“撤消”而消亡。

并发性

多个进程可同存于内存中，在一段时间内同时运行。

独立性

独立获得资源的基本单位。

异步性

进程的执行顺序和执行时间的不确定性。

这个进程定义这部分，大家**重点要掌握的就是 PCB。**

PCB：进程控制块

进程控制块 PCB 和进程是一一对应的，每一个进程只有一个进程控制块。它包含许多与某个特定进程相关的信息。PCB 简单地作为这些信息的仓库，这些信息随着进程的不同而不同。

作用：进程控制块是进程存在的唯一标志

第 2 点

进程的基本状态：

就绪状态：已得到除 CPU 外的其他资源

运行状态：占用 CPU

等待（阻塞）状态：正在等待某一事件发生。

根据原因可设置多个阻塞队列

就绪→执行；运行→等待；

等待→就绪；运行→就绪

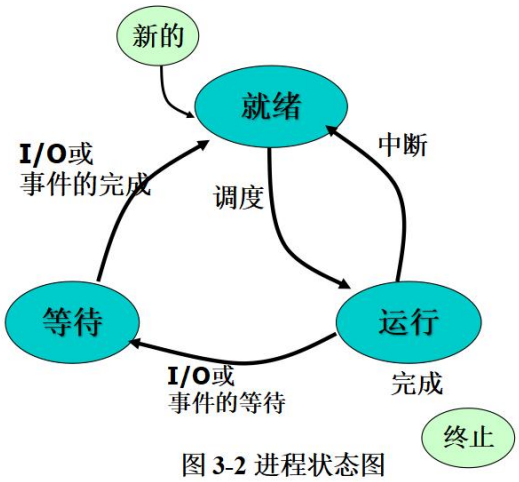


图 3-2 进程状态图

进程状态转换的时机，背过进程状态转换图。

有就绪队列和等待队列，没有执行队列。就绪状态和运行状态可以双向转化，其他只能单向转化。

第 3 部分 进程控制

那么首先我们这个进程控制都是通过**原语操作**来实现的。

另外这里面**最重要的**呢就是**进程创建**的这一部分，因为我们讲到了一个系统调用函数，就是这个 **fork 函数**。那这个当时在课堂上我们也是给大家作为重点讲述的一个例题哈，就是用系统调用之后呢我们产生了一个子进程，那么子进程和父进程呢又是怎么样？独立的向前推进的，会导致他们的结果发生什么样的变化？那这个部分呢是大家需要去了解的。

fork()执行两次返回。功能：建立一个子进程，**与自己独立地并发地运行**。这导致父子进程**哪一个先执行是不确定的**，这取决于内核调度。

父子进程拥有各自独立的地址空间。可以拥有同名变量

fork()的返回值

-1：进程创建不成功

大于 0：表明现在在父进程的上下文中，返回值即为子进程的标识号

0：表明在子进程的上下文中

进程的封锁等待

这个就涉及到那个 **wait** 系统调用。使父进程就陷入封锁，直到子进程运行结束之后，父进程才会被唤醒。就是子进程结束了之后这个信号会传递给这个 wait 函数，那么 wait 函数一接触一接收到这个信号之后，它就知道子进程运行结束，父进程就被唤醒了。所以说这个进程封锁等待这儿涉及的主要就是那个 wait 函数，大家知道它是什么样的含义。

wait()由父进程调用，以保证子进程运行结束之后才运行父进程。

例题：

Page104--page105 3.14：A:0　B:2603　C:2603　D:2600、3.15: X:0 -1 -4 -9 -16 Y:0 1 2 3 4；

PPT p35--p42

第 4 点 进程通信

进程通信的几种方式这个大家都**简单的记一下**，这个我们也都讲过这个软中断呀，还有消息传递这个消息，就是那个 send receive message 这个消息通信还有共享内存通过共享内存的这种方式去进行通信。另外呢还有管道通过管道，就是那个 pipe 这个大家在实验课上可能也做过哈，就是那个 pipe 它也是一种这个进程通信的这个方式，那么在我们**理论课程上呢可能就会考察基本通讯有哪几种方式，然后可能会对它进行一个简单的描述**哈，这个大家注意看一下。

两种基本模型：消息传递和共享内存

消息传递：交换较少量的数据；在分布式系统上常用；内核开销大（因为经常调用系统调用）；比共享内存量小、速度慢；（共享内存只在开辟共享内存时调用系统调用）

共享内存解决生产者——消费者问题

生产者进程生成消息以供消费者进程消费；缓冲区是生产者和消费者进程空间共享内存区；缓冲区分为无界缓冲区和有界缓冲区：无界缓冲区不限制缓冲区大小，生产者进程无需等待；有界缓冲区若满则生产者进程需等待。常用有界缓冲区。生产者进程和消费者进程中的 buffer 在各自的进程空间里 OS 必须提供内存共享机制，才能使两个进程访问同一段数据。Page86

消息传递：page87

直接通信需要用原语：send()、receive()

send(P,message):向进程 P 发送 message

receive(Q,message)：从进程 Q 接收 message

普通管道（pipe）：单向的；有亲缘关系的进程之间才能建立；只能一个进程写，另一个进程读；读数据是一次性操作；page98

命名管道（FIFO）：任何两个进程之间可用；提供了一个路径名与之关联，以 FIFO 文件的形式存储于文件。Page100

第 5 部分 进程的同步和互斥

那这一部分**考察的内容会比较多**，等会儿我们在后面给大家列出来，这部分会考察哪个哈，另外就是这个**死锁，这个死锁也是一个重点。**

重点的部分

调度算法：先来先服务（FCFS）（非抢占）、（最）短作业优先（SJF）（非抢占式）（优先调度下次 CPU 时间最短的）、最短剩余时间优先调度（SRTF）（抢占式 SJF）（保证最小的平均等待时间）、优先级：时间片轮转（RR）（用于分时系统），还有**高响应比优先**那个**多级反馈队列调度**，就会涉及到一些他们融合在一起的这个让大家去求，比如说**周转时间或者是等待时间或者是平均周转时间，平均等待时间**就是有这样的一些**相关的计算**。

周转时间=运行时间+等待时间　　等待时间=在就绪队列中的时间，这个是典型的这个调度算法，出小题，因为它的计算比较简单。**PPT 第六章 8--26　page212 6.3、6.16、6.17**

并发进程的制约关系：**进程同步（直接制约）和进程互斥（简介制约）**

进程同步：两个进程存在直接制约关系

进程共享对方的私有资源；进程间有必然的联系、合作关系，如输入进程、计算进程和打印进程，比如 a 进程的输出作为 b 进程的输入，然后 b 进程的输出又作为 a 进程的这个输入是它们之间相互去传递信号的这样的一个进程同步机制。

进程互斥：间接制约关系

进程竞争**公有资源**的使用权，在临界区内不允许交叉执行；竞争没有固定的必然联系，如：A、B 两进程都要向屏幕输出。

解决临界区问题的 Peterson 方案（了解思想和流程，不用背代码）PPT 第五章 7

实现临界区的互斥，三种策略：开关中断、锁变量、信号量。最重要的是信号量。

信号量

涉及两方面的内容。第一方面是信号量它本身包含哪些属性？信号量既包含这个信号量本身的值，这个值是可以初始化的。初始化是根据我们这个具体应用来决定它初始的值是一呢还是说这个资源的总量。如果要对一个资源去进行互斥访问的话，那么这个互斥信号量的初始值通常我们都给他设为一。如果这个资源有很多有很多的话，那么我们这个信号量的初始值就设为这个资源的总数。在生产者消费者问题中，信号量的初始值还可能设为零。就是那个表示的是空闲的这个缓冲区的时候，或者是满的缓冲区的时候，他们一个初始值为缓冲区的总数一个初始值为零，这个是我们在给大家讲到这个生产者消费者问题的时候，涉及到这个信号量的操作，这个是第一部分，就是信号量本身的这个值。

第二个呢就是信号量还包含一个由于当前等待当前信号量而陷入阻塞的队列。那所以说**信号量它本身的这个值的绝对值呢也能表示当前有多少进程由于等待这个信号量而陷入阻塞**。所以这个信号量的值它是有三个值，一个是大于零，一个是等于零，还有一个是小于零。那么当它大于零的时候就表明系统中还有多少个资源可以被利用，可以被利用。那如果信号量等于零的时候，表明当前系统中没有该资源可利用，并且呢这个等待队列里面是没有进程的。那如果信号量的值是小于零的话，那么信号量的绝对值就表明当前有多少个进程由于等待该资源而陷入阻塞这个信号量的绝对值。那这个是信号量的一些基本概念。

信号量

信号量是一个整数值。

信号量的物理含义：

S>0：有 S 个资源可用；S=0：无资源可用；S<0：|S|表示等待队列中的进程个数。

除初始化外，只能通过标准原子操作访问（wait()即 P 操作、signal()即 V 操作）

初始化置 1，相当于锁变量；P 操作：申请资源；V 操作：分配资源；

原语：系统态下执行的具有特定功能的程序段。目的是保证进程控制和通信的可靠性。机器指令级的（原子指令）：不允许中断；功能级：不允许并发执行。

常用的原语有：进程的创建、撤销、阻塞、唤醒、挂起、激活、P 操作、V 操作

公有信号量和私有信号量

公有信号量

互斥信号量（互斥锁、互斥量）

联系着一组并发进程，初值为 1，每个进程均可施加 P、V 操作

私用信号量

用于进程同步，在制约和被制约的进程之间。初值：0 或某个正整数 n，表示资源的数目。只允许拥有它的进程施加 P 操作。

互斥锁（自旋锁）：优点：等待锁时没有上下文切换；缺点：“忙等待”，占用 CPU 时间；

适用于：临界区持锁时间短、多处理器系统。详见 page144 5.5

优先级调度+忙等待=优先级反转 PPT 第五章 13

另外呢

经典同步问题：出大题

PPT 第五章 21--37 **会默写 page149 的生产者和消费者进程结构伪代码**

生产者进程结构：

```
do{
    …
    /* produce an item in next_produced*/
    …
    wait(empty);
    wait(mutex);
    …
    /* add next_produced to the buffer */
    …
    signal(mutex);
    signal(full);
}while(true);
```

消费者进程结构：

```
do{
    wait(full);
    wait(mutex);
    …
/*remove an item from buffer to next_consumed*/
    …
    signal(mutex);
    signal(empty);
    …
    /* consume the item in next_consumed */
    …
    }while(true);
```

信号量的优缺点：优点：简单；表达能力强：用 P、V 操作可以解决任何同步互斥问题。缺点：不够安全。P、V 操作使用不当会出现思索；遇到复杂同步问题时实现复杂

P、V 操作讨论：

P、V 操作必须成对出现。互斥操作：处于同一进程，同步操作：不在同一操作中

两个 P 操作在一起时，顺序至关重要，一个同步 P 操作与一个互斥 P 操作在一起时，同步 P 操作在前。两个 V 操作顺序无关紧要（一般把临界区限制在最小范围）

生产者消费者问题 PPT 第五章 24--26

读者写者问题 PPT 第五章 28--32 例 5.5

哲学家就餐问题 PPT 第五章 33--37 例 5.6

死锁

死锁的概念：（简答）

一组并发进程彼此互相等待对方所拥有的资源，在得到对方的资源之前不释放自己所拥有的资源，若无外力驱动，各进程不向前推进的僵死状态。

死锁产生的四个必要条件，就这四个必要条件一定要记清楚，一定要记清楚。

互斥

涉及的资源是临界性

不可抢占

进程占用的资源，不会被强制性拿走，必须由该进程主动释放。

部分分配（占有并等待）（请求和保持）

进程在等待新资源时继续占有已分配的资源。

环路等待（循环等待）

存在一种进程的循环链，链中的每一个进程已获得的资源同时被下一个进程请求。

死锁的处理策略有三种：

第一种 死锁的预防（静态的策略）：破坏死锁的必要条件（之一即可）

限制并发进程对资源的请求；可能导致系统资源利用率和系统吞吐量的降低。

第二种 死锁的避免（动态检测动态分配）PPT 第五章 67--75

死锁有发生的可能，要发生的一瞬间，不让它发生。

代表算法：**银行家算法**

在进程提出资源请求时判断：如果满足请求，系统会不会死锁。

安全状态：自身不存在死锁问题；存在着某种调度顺序：即使在最坏的情况下（所有的进程请求它们最大数目的资源），每一个进程都还能够顺利地运行结束。

首先尝试去给这个进程按照他所要求的资源数量去进行分配。那如果分配完了之后呢系统处于不安全的状态，那么我们就不给这个进程分配资源。试着分配之后呢系统还是处于安全状态，那么我们就真正的去给这个进程分配资源，这个是银行家算法

不安全状态不一定导致死锁

第三种 死锁的检测和解除（PPT 第五章 77--80）

当死锁发生时采取应对措施

死锁的检测：资源分配图 RAG 和死锁定理

死锁定理：死锁的充分条件是资源分配图不可完全化简（化简后依旧有环）

化简之前有环，有可能没死锁，但化简之后还有环的一定死锁。

存在死锁时有三种解除策略

剥夺资源、进程回滚、撤销进程

第四种死锁处理方法：鸵鸟政策：装作没看见。假装在系统中不会出现死锁现象。原因：进一步则代价高昂，退一步则影响不大。交给程序设计人员处理。绝大多数操作系统采用鸵鸟政策，如 Unix、Linux、Windows

第四章 线程

第四章呢，这个是我们原来的这个大纲里面不做要求，但是在新的大纲里面我们对这个线程也做出了一些要求。那么这个**线程和进程通常是混在一起去考察的**。

首先

引入线程的原因，就是我们如果有进程的为什么还要引入线程？引入线程的原因以及线程的概念，还有线程与进程的关系。

线程的概念：进程当中的一条执行流程，是 CPU 调度的基本单位。

每个线程**独享**线程 ID、程序计数器、寄存器组和堆栈。与同一进程的其他线程**共享**代码段、数据段和其他操作系统资源。

线程与进程的关系与区别：进程是资源分配的基本单位，线程是调度的基本单位。

进程=线程+资源平台

一个进程之内的多线程之间的调度所花费的时间消耗要远远小于跨越进程之间的线程调度消耗，因为跨越进程的线程之间的调度就相当于是进程调度，涉及到上下文环境的切换，在上下文环境切换中，要把 PCB 里面的一系列的信息全都进行保存。

并发与并行：

并行性：两个或多个事件在同一时刻发生；

并发性：两个或多个事件在同一时间间隔内发生。

并行类型：数据并行与任务并行

数据并行注重将数据分布于多个计算核上，并在每个核上执行相同操作。

任务并行涉及将任务（线程）而不是数据分配到多个计算核。每个线程都执行一个独特的操作。不同线程可以操作相同的数据，也可以操作不同的数据。

大多数情况下，应用程序混合使用这两个策略。

那么还有一点比较重要的

线程的两种分类：内核级线程、用户级线程以及它们两者之间的关系，我们前面讲到了这个一对一模型，一对多模型，还有多对多模型，它们就是这两种不同的线程之间会有不同的这个对应关系。内核级线程和用户级线程他们的分别都是在哪里运行的，通过什么来进行线程支持的。

用户线程：解决上下文切换的开销；TCB 在进程的用户空间；管理无需内核支持，由用户级的线程库函数支持。

优点：无需用户态/核心态切换、速度快

缺点：内核以进程为单位调度，无法发挥多核优势：一个线程阻塞，整个进程等待；一个线程在运行，进程中其他线程无法运行；时间片分配给进程，多线程则每个线程就慢。

内核线程：发挥多核处理器的并发优势；TCB 在内核空间；管理由 OS 内核支持；几乎所有现代 OS 都支持，包括 Windows Linux MacOS X 和 Solaris

优点：内核以线程为单位调度，实现同一进程内的多个线程并行执行。

缺点：线程的调度和同步通过系统调用实现，开销大。

多线程模型：用户级线程和内核级线程的连接方式

多对一 Many-to-One;一对一 One-to-One;多对多 Many-to-Many

Page116 4.3

第五章 内存管理

也是一个**非常重要**的部分，**内存管理的概念**，内存管理要涉及到内存的分配，还有空闲空间的管理，另外呢就是这个地址重定位。

7.1.2 存储器管理方式

连续分配：每个作业分配一个连续的空间

单一连续区

分区（用于多道程序的一种较简单方式）

固定分区

可变分区

离散分配：将一个用户进程离散地分配到内存中的多个不相连接的区域中。

分页存储、分段存储、段页式存储

虚拟存储管理方式

请求分页、请求分段、请求段页式

7.1.3 地址绑定

内存地址（物理地址、绝对地址），逻辑地址（程序地址、虚地址）

将指令和数据绑定到物理地址有三个时机：1.编译时绑定：编译时内存位置已知（不常见、生成绝对代码（地址确定））2.加载时绑定（静态重定位）：编译时生成可重定位代码 3.执行时绑定（动态重定位）：需要硬件支持，大多数现代操作系统支持。

地址重定位（地址映射、地址转换）：把虚拟空间中已链接好的程序装入内存，并将逻辑地址转换成物理地址的工作。编译器将符号地址绑定到可重定位的地址（逻辑地址），链

接或加载程序将逻辑地址加载到绝对地址（物理地址）。

静态地址重定位：程序执行前由重定位装入程序一次完成，按起始地址调整程序指令中的地址。

优点：由软件实现，不需要硬件支持

缺点：程序一旦装入内存，就不能再移动

动态地址重定位：执行指令时转换地址，对指令本身不做任何修改。

优点：可以进行非连续分配。

缺点：需要硬件地址变换机构

一个(或多个)基地址寄存器 BR; 一个(或多个)程序虚拟地址寄存器 VR。

指令或数据的内存地址 MA =(BR)+ (VR)

7.1.4 内存保护（不是重点 PPT 第七章 16--20）：限制各用户进 w 程只能在自己的存储区活动，不能破坏别的进程的程序数据和系统。

7.2 内外存数据传输的控制（不是重点 PPT 第七章 21）覆盖

7.3 **连续内存分配**（不是重点 PPT 第七章 29）现代 OS，物理空间已经没有连续分配

固定分区 优点：易于实现开销小。缺点：分区总数固定，限制了并发程序数目；有内碎片，影响主存的利用率。

内碎片：作业占用的分区内的未被利用的空间。

可变分区

可变分区的建立时机：在作业装入内存时动态建立；

可变分区的大小：与作业大小相等；

可变分区的个数变化不定。

可变分区太大可以分割为小的，空闲可变分区若相邻可合并

可变分区的数据结构：请求表、已分配分区表、空闲区表。

可变分区的分配（PPT 第七章 38--46）

最先适应法、最佳适应法、最坏适应法

可变分区的回收 PPT 第七章 47

可变分区的碎片问题

外碎片：经过一段时间的分配与回收后，内存中存在着不连续的很小的空闲分区。

内存紧缩：把所有的作业尽可能往低地址端移动，空闲的小分区往高地址端移动，形成一个大的空闲区。

进程的移动需要大量 CPU 时间；使用动态重定位

7.3.4 分区存储管理的分析（不是重点）：优点：实现了多道程序设计，硬件支持少、管理算法简单，实现简单。缺点：产生碎片问题；属于连续分配，作业大小受分区大小控制，除非配合采用覆盖和交换技术。

离散分配

7.4 分段 纯粹的分段系统越来越少，分段与分页结合起来，大多数存储管理问题都是关于分页方面的。

段式管理：程序员将**程序**按含义**分成若干段，**

用户程序的逻辑地址是二维的（<段号，段内偏移>）

以段为单位分配内存

内存用**可变分区**管理；每段分配一个连续的内存区；各段长度不等、不要求连续（可变分区咯）

创建进程时建立进程段表

段基址：段的物理地址的起始地址；段界限：段的长度

分段的硬件支持：

段表基址寄存器：存放段表在内存中的始址（用来找段表）

段表限长寄存器：存放段表的长度。

描述段地址映射（简答）：PPT 第七章 56

首先根据段表基址寄存器中的值找到段表，再由段号找到当前逻辑地址对应的段表项，找到其在内存的起始地址，判断给出的段偏移与该段的段界限。若段偏移<段界限，则起始地址+段偏移→找到该指令；若段偏移>段界限，则陷入操作系统，报出越界错误。

PPT 第七章 58 例 2

7.5 分页（PPT 第七章 60--95）目前我们能接触到的计算机、手机，都采用页式

页式管理，我们说这个**最可能考到的就是这个页式管理的这种**大题，通常都是跟这个**虚拟页式管理去融合在一起**进行考察的。

那这部分呢

那么根据这个一维的地址呢，计算这个页号以及页内偏移，也需要知道个每一页的大小是多少。总共有多少页。那么通过这个一维的地址都可以得到它，就是通过它的位数就能够得出每一个页的大小是多少？然后一共包含多少页。

页式管理的基本原理：

把逻辑地址空间和物理地址空间划分为若干大小相等的页（page）

硬件自动完成，**页的大小：2 的整数次幂 2^10=1K、2^11=2K、2^12=4K……2^20=1M……**

逻辑地址：页号 p、页内地址 d

例：页长为 1K，拥有 64 页的虚拟空间地址（逻辑地址空间）结构：

高 6 位表示页号（2^6=64y 页），低 10 位表示页内地址（2^10=1K 字节）

内存地址空间（物理地址空间）的分页**称为块**（内存块、物理页面、**页框**、帧），与虚拟地址空间（逻辑地址空间）的页大小相等。

以页为单位、按进程的页数分配内存。地址变换机构：逻辑上连续的页映射到物理上不连

续的主存块中去。PPT 第七章 63

PPT 第七章 64 例 3、**例 4**

静态页式管理：进程执行之前，按页将整个进程装入内存。与分区的相同点是将进程整个装入内存；不同点是不需要将进程连续存放。

页式管理的数据结构：页表、请求表、存储页面表

页表：

页表项：页号、页面号（页框号）、其他相关信息。（每页一个页表项）

每个进程一个页表，页表的长度和首地址放在进程控制块 PCB 中。

对于占用 CPU 的当前进程的页表来说，它的首地址放在页表始址寄存器中，长度放在页表长度寄存器中。

页表的大小由进程的长度决定。如：一个每页长 1K，大小为 20K 的进程，如果一个内存单元放一个页表项，需要 20 个存储单元。（该进程占 20 页，一页一个页表项）

请求表：整个系统一张，存储进程的页表起始地址、长度和页面数等。

存储页面表（存储块表）： 整个系统一张

7.5.3 页式管理的地址映射 由 MMU 自动完成

页表基地址寄存器：存放页表的始址；

页表长度寄存器：存放页表的大小；

PPT 第七章 75--79 页式地址映射示例（1）、（2）、例 5

页式地址映射示例（1）：由虚地址 100 计算：100/1024=0……100，则该指令位于该进程的第 0 页，页内地址为 100。根据页表基地址寄存器，访问内存，查找到页表，由页表可知，页号 0 对应页框号 2，则该指令的物理地址为 1024*2+100=2148。

当按照给定的逻辑地址进行读写时，必须访问 2 次内存，运行速度要下降一半。

第 1 次：访问内存中的页表，按页号读出块号（页框号）；

第 2 次：按计算出的物理地址进行读写。

由此引出**快表 TLB**（看一下基本概念）

快表（TLB）存储在寄存器中，所以价格昂贵

快表存放当前进程的部分页表项。快表的依据是局部性原理。

局部性原理：时间局部性（指令）：一条指令被执行，有可能在较短时间间隔内被重复执行。空间局部性（数据）：某存储单元被访问，其相邻单元在一定时间内也会被访问。

命中率：在 TLB 中查找到感兴趣的页号的次数的百分比

若命中，则一次 TLB 访问加一次内存访问；未命中，则一次 TLB 访问加两次内存访问

有效访问时间=

命中率*（TLB 访问时间+内存访问时间）+（1-命中率）*（TLB 访问时间+内存访问时间）

可以看 PPT 第七章 85--86 例 6 例 7

页表的结构（不是重点 PPT 第七章 87）

多级页表、哈希页表（大于 32 位地址空间的常用方法）、倒置页表

7.5.5 静态页式管理小结

静态页式管理优缺点

优点：打破了连续存储空间的限制；解决了分区管理时的碎片问题。（现在只有最后一页可能会有碎片）

缺点：进程的大小仍受内存可用页面数的限制（因为要求进程全部装入内存）

内碎片：平均一个进程要浪费半页的存储块

页式与段式的比较

页式

用户程序是**一维**逻辑地址空间；

页是信息的物理单位；

分页是系统管理的需要；

页的大小由系统确定，与内存块的大小相同；

页面大到能装下整个程序，退化为**固定分区**。

段式

用户程序是**二维**逻辑地址空间；

段是信息的逻辑单位；

分段是基于用户程序结构，每段在逻辑上是相对完整的一组信息；

段长度取决于用户编写的程序，段长不等；

如果段的个数为 1，退化为**可变分区**。

7.6 段页式管理

分页优点：克服碎片，提高内存利用率；

分段优点：方便用户，易于共享、保护、动态链接。

段页式系统开销大，一般只用在大型机、工作站

段页式地址变换为了获取一条指令或数据，需要访问 3 次内存。

详见 PPT 第七章 96--99

那么

8.1 虚拟存储管理技术：

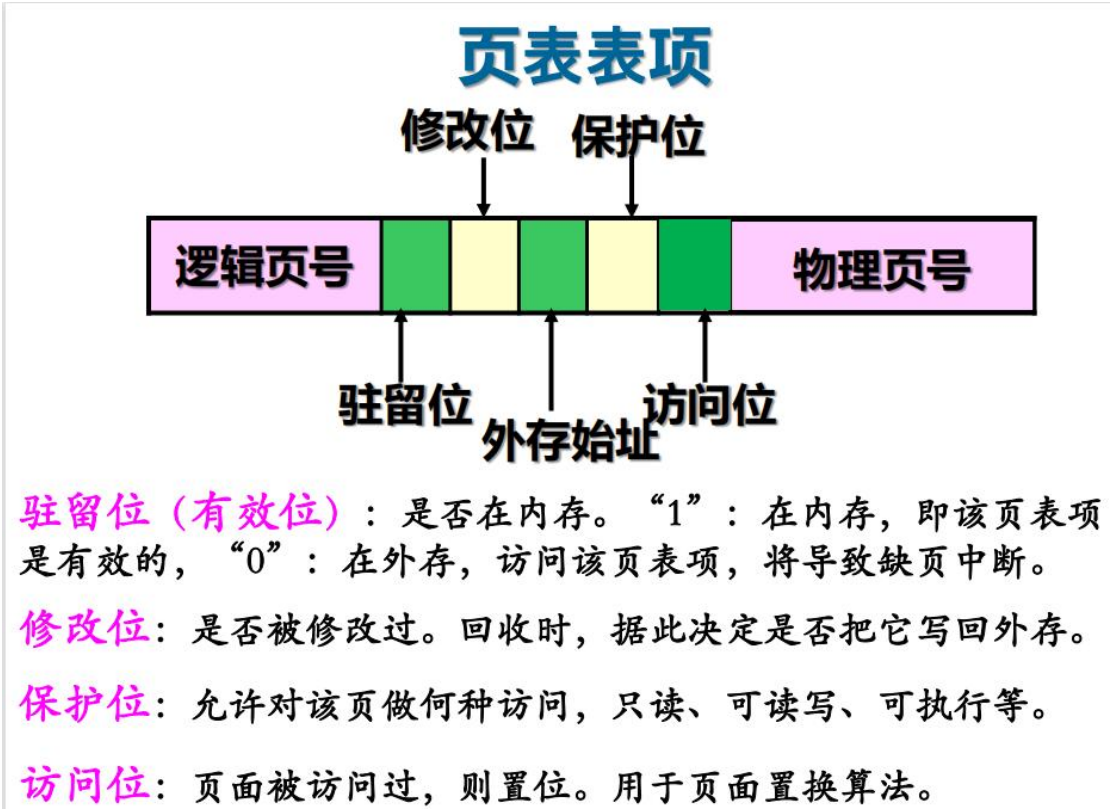
静态的内存管理技术要求进程的所有页全部装入内存才能运行。而虚拟存储技术只装入在一定时间内经常反复执行和调用的代码和数据，其他部分则在执行过程中动态装入。

请求调页、预调页和请求段式

请求调页：访问的逻辑页不在内存，发生缺页异常，硬件发出缺页中断。

优点：容易实现。缺点：每次缺页中断都要 I/O，I/O 次数多，开销大

8.2 请求调页



缺页中断处理过程描述：（要求掌握）

要访问逻辑地址 M，需要知道其物理地址，硬件查询快表，如果查不到就查页表。访问页表始址寄存器→找到页表，再访问页表的限长寄存器，确定页表长度，以此确定 M 的页号是否在页表范围内，若在，检查有效位，若无效，可能有两种情况：违反权限、不在内存，一般认为是不在内存，则发生缺页异常→陷入操作系统，控制权转让给操作系统，操作系统启动缺页异常处理程序，处理程序在磁盘（外存）中查找到逻辑页号对应的内容，启动磁盘把该页写到内存上，若内存中有空闲页框→修改页表（包括把有效位改为 v），结束。若内存中没有空闲页框→发生页面置换→修改页表。若被置换出去的页面被修改过，还需对磁盘内容重写，否则直接从内存中剔除即可。

那另外一个呢

页面置换算法（调用时机：缺页中断发生且内存中没有空闲页面）

随机淘汰算法（少用）

理想淘汰算法（OPT）（最优页面置换算法）

淘汰在访问序列中将来再也不出现的或在离当前最远的位置上出现的页或当前间隔时间最久去访问的，OPT 无法实现，因为其要求预先知道每一个进程的序列。但它可用作其他算法性能评价的依据。

先进先出算法（FIFO）缺点：内存利用率不高（效果不好，缺页次数多）

陷阱（Belady）现象：分配的物理页面多，缺页中断的次数反而增加

只有 FIFO 算法有陷阱现象

最近最久未使用算法（LRU）：换出最近一段时间内最久没有被使用过的页

利用已知的页访问情况，预测将来。

基于局部性原理（时间局部性）（指令）

LRU 是公认很好的页面置换算法

LRU 例题：PPT 第八章 17--18

LRU 的准确实现：（不是重点）

时间戳法（淘汰时需要找到最小值，详见 page264“计数器”）

页码栈

这两种准确实现代价较大

LRU 的近似实现：

最不经常使用 LFU（非重点）：页表设“访问计数器”；页被访问则加 1，淘汰技术支持最小的页，并将所有计数清零。

最近没有使用 NUR（更加非重点）：页表设“访问位”；页被访问则置 1，否则置 0；从最近一个时期内为被访问的页中任选一页淘汰。系统周期性清零。

时钟页面置换算法（Clock）：（FIFO+跳过访问过的页）（第二次机会算法）

页表设“访问位”：页被访问，访问位置 1；

页组织成环形链表（类似时钟），指针指向最老（最先进来）的页

选择淘汰页

访问位为 0：立即淘汰；访问位为 1：置 0，指针往下移动一格。日从辖区直到找到被淘汰的页，把指针移动到它的下一格。

是公认很好的 LRU 近似算法。

抖动：高度的页面调度活动使得进程的调页时间大于它的执行时间。防止抖动的根本手段是给进程分配足够多的帧（也即页框、页面、块）PPT 第八章 24 工作集驻留集应该不是重点，想看可以看一下 PPT 第八章 26 例 3

存储管理非常重要，这一部分在就是基本上这么多年的考研题目里面这个大题也就是考内存管理这个各种计算。

第六章 文件管理

主要考察的是文件的基本概念。那么这个文件它的组成就是一个文件名，然后一个分隔符，还有扩展名，拓展名也表明了这个文件的分类。

文件：一组带标识（即文件名）的、在逻辑上有完整意义的信息项的序列。

（记录在外存上的相关信息的命名组合）

另外呢

文件系统对于用户来说，它所提供的服务就是能够实现按名存取，那么它对于这个文件文

件底层的来说呢就是主要控制这个文件，它的这个存储在物理设备上的一个存储，还有空间的分配以及空间空间的管理。

那么在这个文件区域，在磁盘上进行放置的时候呢肯定要涉及到这个物理结构，那么它对于用户这个角度来说提供的又是这个逻辑结构。所以呢我们要把逻辑结构和物理结构去进行一个对应。

那么

文件的逻辑结构呢主要就是分为两种哈，**一个是字符流式的，没有结构的文件字符流式的无结构式的文件。还有一个呢就是记录式的有结构的文件哈，这个是文件的逻辑结构。**

那么

文件的物理结构：文件在存储设备上的存放方法

（以“块”作为分配何川送信息的基本单位。物理块：文件的存储设备划分为若干个大小相等的块；逻辑块：把文件信息划分为与物理块大小相等的块。）

顺序结构、链接结构。（索引顺序不做要求了？是什么不做要求啦）这个中间有一个链接，有个链接文件，就是顺序和链接，还有索引这三种形式的物理结构，这三种物理结构是怎么样去对文件进行存储的，也各有什么优缺点

连续结构（顺序结构）：逻辑上连续的文件信息依次存放到连续的物理块。目录项：始址、总块数、最后一块字节数。

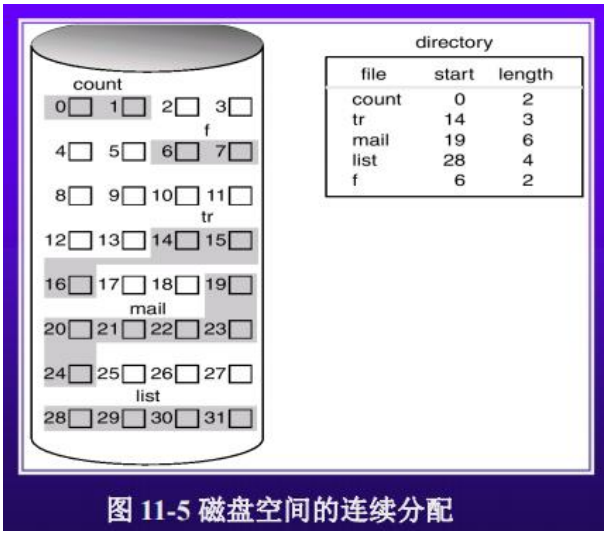


图 11-5 磁盘空间的连续分配

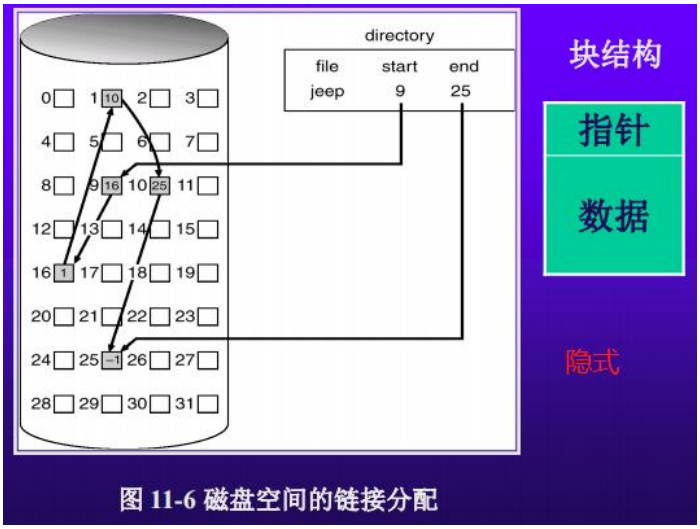
优点：简单；支持顺序存取及随机存取；所需磁盘寻道次数和寻道时间最少；可以同时读取多个块。

缺点：建立文件时确定文件长度，且以后不能动态增长。再删除文件的某些部分后，会留下无法使用的零头空间；不宜存放用户文件、数据库文件等经常被修改的文件。

链接（串联）结构：文件信息存放在不连续的物理块中，通过指针连接。目录项：链表的首指针。

优点：提高了磁盘空间利用率；有利于文件的动态扩充：插入和删除。

缺点：搜索效率较低；适合顺序存取方法、不适于随机存取；为读取某个信息块需要更多的寻道次数和寻道时间。



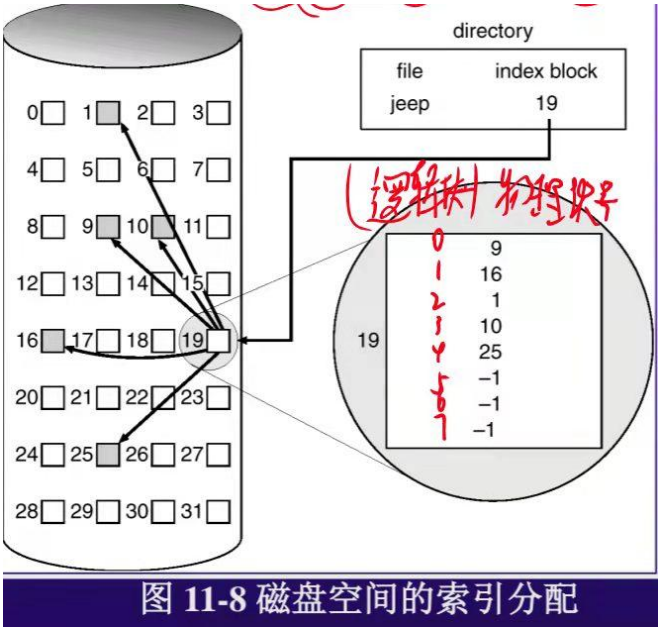
（DOS 系统采用文件分配表（FAT）的形式，将所有指针集中到 FAT，缓存在内存中。（显式））

索引结构：文件的信息存放在若干不连续物理块中（集合了以上两种的优点）

每个文件建立一张索引表

表项：逻辑块号和对应的物理块号

目录项：索引表的物理地址

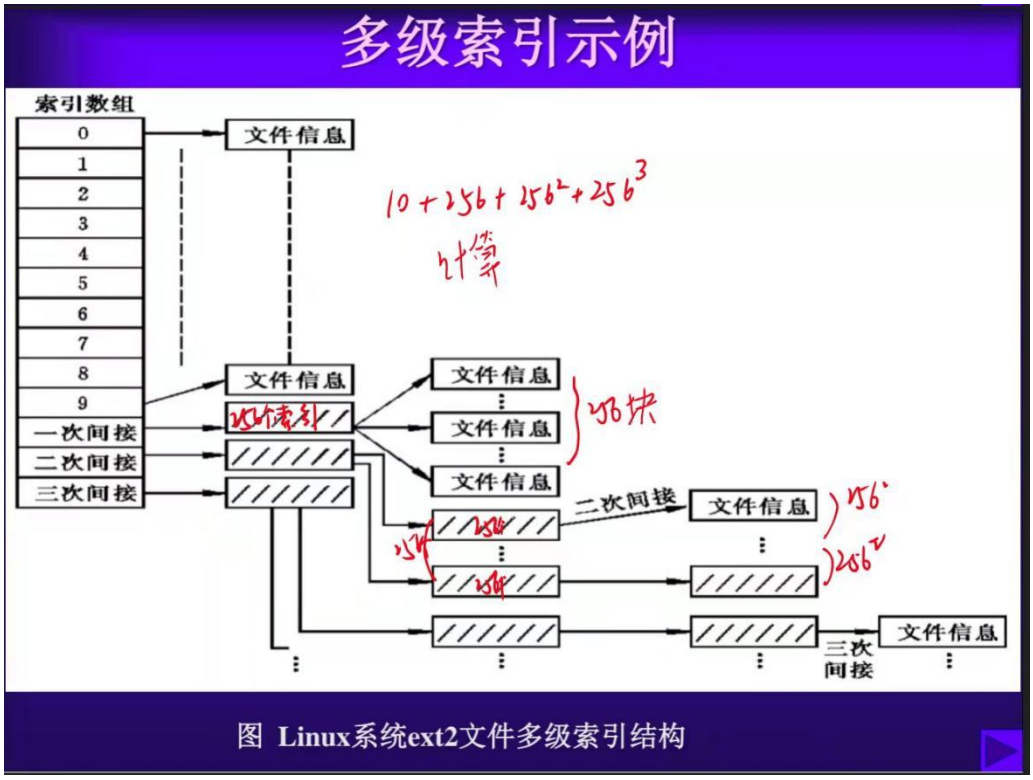


如果索引表的大小超过了一个物理块？

要决定索引表的物理存放方式，不利于索引表的动态增加；

使用二重索引：索引表所指的物理块中存放的是装有文件信息的物理块地址

在实际系统中，索引表的头几项设计成直接寻址方式：所指的物理块中存放的是文件信息；后几项设计成多重索引。在文件较小时，可利用直接寻址方式找到物理块号而节省存取时间。例如 Linux ：多级索引结构



优点：既能顺序存取，又能随机存取；满足了文件动态增长、插入删除的要求；充分利用外存空间。

缺点：索引表增加了存储空间的开销；在存取文件时，至少访问磁盘二次以上。

改进的方法：对文件进行操作之前，预先把索引表放入内存:直接在内存通过索引表确定物理地址块号，访问磁盘的动作只需要一次。

索引结构是最普遍的一种物理结构

那么这一部分还额外的

涉及到**文件的访问方式**，或者叫做文件的存取方式。以及存储介质。

顺序存取设备：前面的物理块被存取访问后，才能存取后续物理块。**磁带**最典型。

特点（不是重点）：容量大；顺序存取方式时存取速度高，按随机方式或按键存取方式，效率低；适合存放连续文件结构且文件大小确定，修改、增删比较少的文件

直接存取设备：磁盘最典型；允许直接存取磁盘上的任意物理块。信息记录在磁道上，多个盘片，正反两面都用来记录信息，每面一个磁头；柱面：所有盘面中处于同一磁道号上的所有磁道组成；物理地址形式：磁头号（盘面号）、磁道号（柱面号）、扇区号

存储设备、存储结构、存取方式间的关系

存储设备	磁　　盘			磁 带
存储结构	连续文件	串联文件	索引文件	连续文件
存取方式	顺序、随机	顺序	顺序、随机	顺序

9.4.1 文件的组成（非重点）：文件说明（文件控制块 FCB）FCB 管理文件所需的所有信息；FCB 是文件存在的标志；FCB 组成目录文件。

9.4.2 目录项的内容（非重点）：直接法：目录项=文件名+FCB；（缺点：占空间大）（MS-DOS、Windows）；间接法：目录项=文件名+FCB 的地址（索引号）（节省空间）（Unix（inode）、Linux）。目录的主要功能：文件各类属性的保存；从路径名到物理位置的转换。

那第三个呢

目录结构：一级目录、二级目录、树形的目录，那树形的目录里面又包含了这个绝对地址、绝对路径和相对路径，那个各种目录结构它们又有什么样的优缺点？

单级目录：所有文件建立一张目录表，每个文件占有一项。在系统初启时或需要时调入内存。

不足：搜索效率低（查找一个文件的范围是整个目录文件的所有目录项。文件越多，速度越慢）；不允许文件重名。只适用于单用户环境。

两级目录：

主目录表（MFD）整个系统一个；给出用户名、用户子目录 UFD 的物理位置。

用户目录表（UFD）每个用户一个，包含该用户所有文件的 FCB。

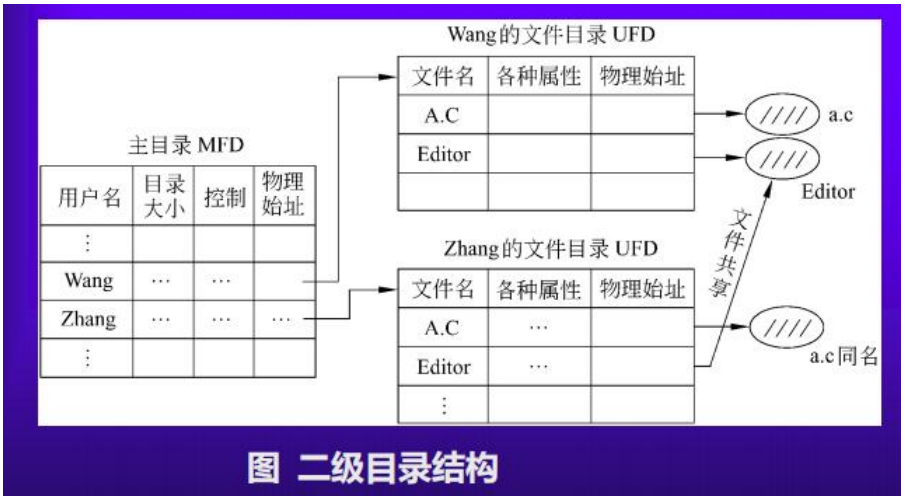
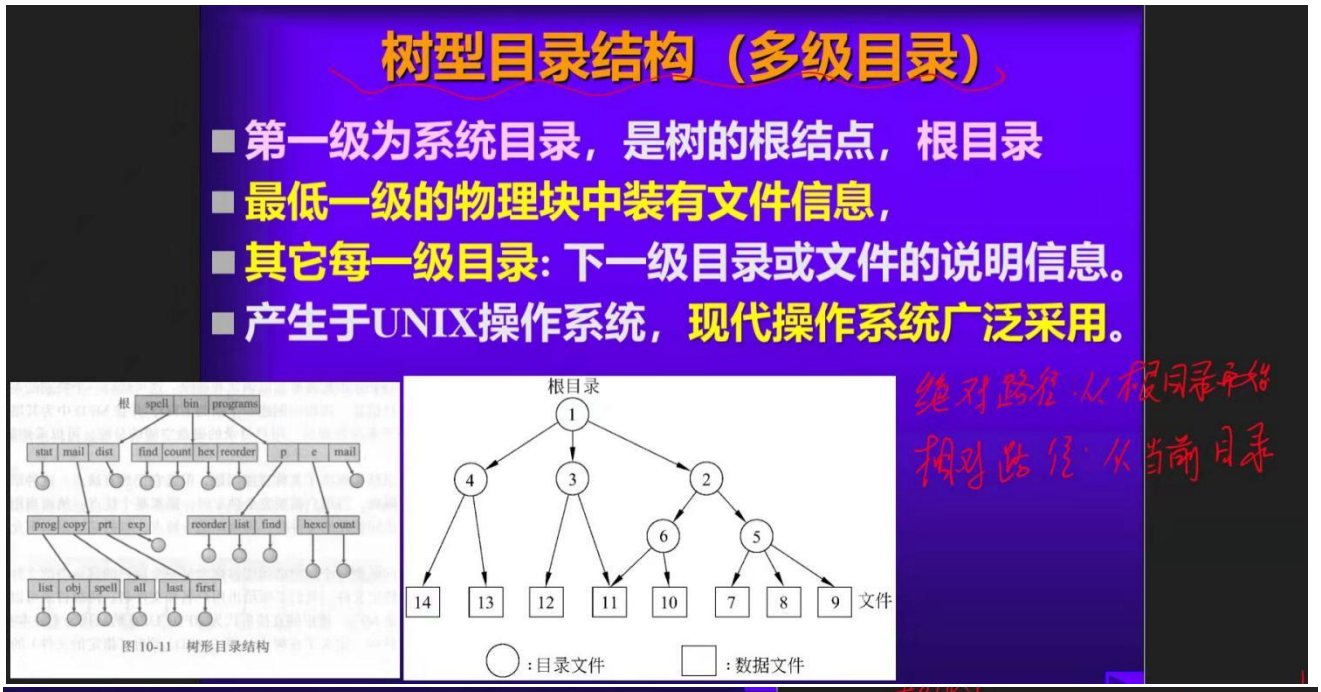


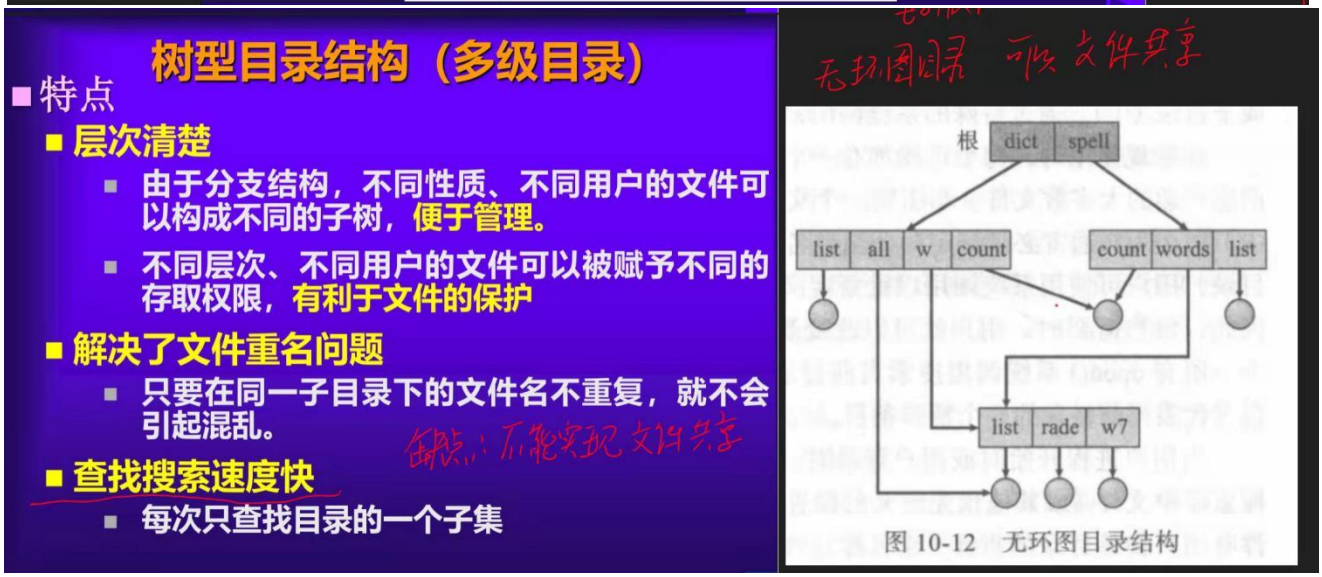
图 二级目录结构

优点：解决文件重名问题；较高的搜索速度（单级目录搜索时间与单级目录表长度成正比、二级目录搜索时间与用户个数+每个用户拥有的文件数成正比）；支持文件共享

缺点：不太适合大量用户和大量文件的大系统。

（树形目录）多级目录：





那第四个呢

空闲块的管理方法，这主要呢是有一个是空闲目录，另外呢还有一个是位图。接着呢我们之前还见过一个题，就是用 fat 这种方式也可以去进行空闲块的管理，他是把指向空闲块的这些列表呢全都集中到一起，集中到一起去进行管理，那么这个就是一种特殊的这个链接式的这个物理结构，物理结构。那么通过这个 fat 呢，我们同时也可以对空闲块还有这个已经存

有文件的这个物理块去进行管理哈。

空闲文件目录：放在一个物理块中；表项：一个由多个空闲块构成的空闲区；**适用于连续文件结构**

空闲块链：**较常用**；把文件存储设备上的所有空闲块链接在一起（用指针）；常用空闲块链的连接方法有：按空闲区的大小顺序链接；按释放先后顺序链接（后释放的放末尾）；按成组链接法（把连续的空间块组成组，如：a 块→b 组→c 组→d 块……）

位示图法：在内存中为每个文件存取设备建一张位示图；用一串二进制位反应磁盘空间中分配情况，每个物理块对应一位（1 已分配，0 未分配）

设备管理

那

最后设备管理这一章呢会考察到**设备的基本分类。**

（5）按资源分配角度：

独占设备：在一段时间内只能有一个进程使用的设备，一般为低速 I/O 设备。且这类设备若分配不当，可能会造成死锁。如打印机、磁带等。

共享设备：在一段时间内可有多个进程共同使用的设备，多个进程以交叉的方式来使用设备，其资源利用率高。如硬盘。

虚拟设备：在一类设备上模拟另一类设备，常用共享设备模拟独占设备，用高速设备模拟低速设备（先把数据放在缓冲区中，打印机从缓冲区的队列中取走数据，提高 I/O 并行效率）。如 SPOOLing 技术。

那第二个

数据传输控制方式：程序直接控制方式、中断控制方式、DMA 方式、通道方式（并行度依次提高）（前三种适用于微机，第四种适用于专用计算机、中、大型机）一个是这个轮询也叫可编程 io，也叫做轮询 io 或者是查询 io，中断的这个方式还有 DNA 的控制方式。还有一种是这个通道的控制方式，那么各种方式之间他们的这个优缺点。

程序直接控制方式（忙——等待方式）：一旦 CPU 启动 I/O 设备，便不断查询 I/O 设备的准备情况，终止原程序的执行，浪费 CPU 时间。 I/O 准备就绪后，CPU 参与数据传送工作， 而不能执行原程序。

优点：控制简单；不需要硬件支持。缺点：CPU 和 I/O 设备串行工作；适用于 CPU 执行速度较慢、且外围设备较少的系统

中断方式：向 I/O 发命令—执行其它任务；I/O 中断产生—CPU 转相应中断处理程序；如：读数据，读完后以中断方式通知 CPU，CPU 完成数据从 I/O—内存。要求：CPU 与设备之间有相应的中断线。

中断方式存在的问题：在一次数据传送过程中，发生中断次数较多，耗去大量的 CPU 处理时间：数据缓冲寄存通常较小，装满数据发生中断。 由于中断次数的急剧增加而造成 CPU 无法响应中断和出现数据丢失现象。

DMA 方式—用于块设备中。基本思想：在外围设备和内存之间开辟直接的数据交换通路。增加硬件：传送字节计数器、内存地址寄存器等。

DMA 流程：（在数据传送过程中，无需 CPU 参与，而是 DMA 控制器控制完成。）

1.预处理：CPU 收到 DMA 请求，作为司令向 DMA 发布命令，启动 DMA，测试 I/O 设备，初始化寄存器等。2.数据传输完全由 DMA 硬件完成。3.后处理：完成数据传送后，DMA 向 CPU 发送中断请求。

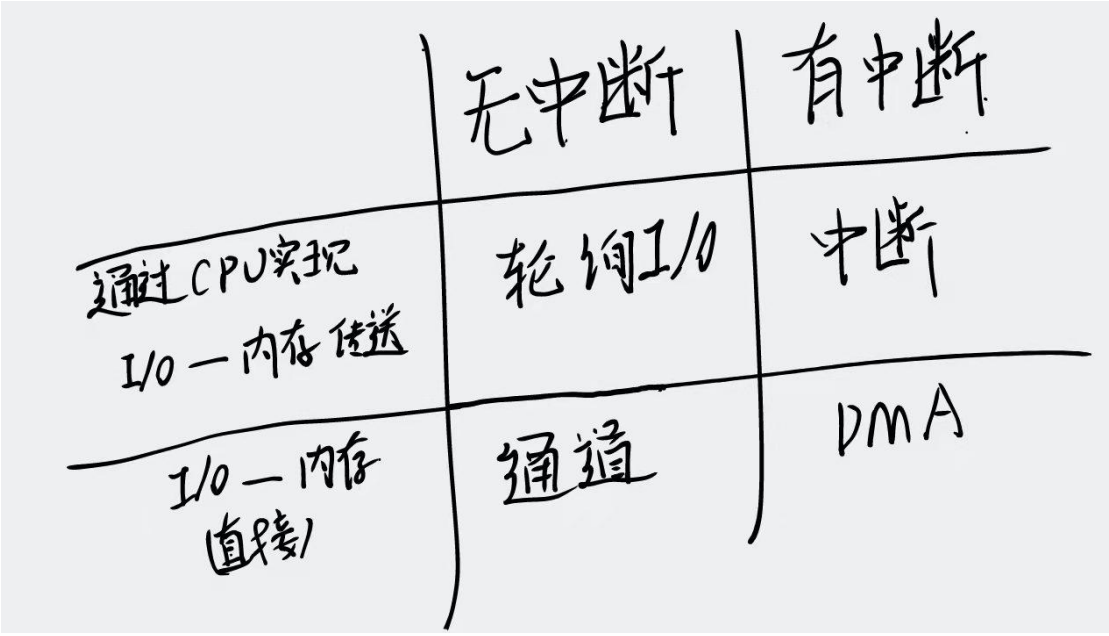
比较

- CPU 进行中断处理的次数
  - 中断：在数据缓冲寄存器满之后
  - DMA：在所要传送的数据块全部传送结束时
- 数据传送的控制
  - 中断：CPU
  - DMA：DMA 控制器

  排除了因并行操作设备过多时 CPU 来不及处理或因速度不匹配而造成数据丢失等现象。
- 局限性
  - 当设备很多，DMA 控制器很多引起内存地址冲突、控制过程复杂；
  - 多个 DMA 控制器的使用不经济。

通道控制方式：通道是一种特殊的执行 I/O 指令的处理机，控制设备与内存直接进行数据交换。CPU 只需发送 I/O 命令给通道，通道调用内存的通道程序完成任务，CPU 只需给出通道程序首地址、要访问的 I/O 设备，就可以完成一组块操作

DMA 与通道的区别：1.一些控制信息，如数据块的大小，内存位置，DMA 方式下由 CPU 控制，但通道方式下由通道控制。2.每个 DMA 控制 1 台设备，与内存交换数据，单通道可以控制多台设备与内存交换数据。



第三个

缓冲技术，就是我们为什么要提出这个缓冲技术？缓冲技术它是为了解决什么问题哈？缓冲技术引入的原因？一定要记住！

1.缓和 I/O 设备与 CPU 间速度不匹配的矛盾。如：计算—打印 buffer—打印

2.减少对 CPU 的中断频率，放宽 CPU 对中断的响应时间。如：buffer 越大，“buffer 满”信号发生频率越低。

3.提高 CPU 和 I/O 并行性，即解决 DMA 或通道方式的瓶颈问题

缓冲有单缓冲（入→出 向外）、双缓冲：有两个缓冲区，输入量大时，可以都用作输入缓冲区，反之亦然。、缓冲池：1.多个缓冲。很多缓冲区都用，如有界缓冲区。2.循环缓冲

设备管理的第二部分呢

磁盘的组织与管理，一是会涉及到一些**磁盘的基本概念**哈，比如说它的最小分配单位呀，最小寻址单位是什么，这样的一些一些那个内容哈，大纲要求**磁盘调度算法**，那么对于磁盘一些比较复杂的知识呢，咱们在这个大纲里面没有给大家做要求哈，这个以后如果大家有需要的话，再把磁盘的那个详细的那些内容都去补一下。

那么

磁盘调度算法四种调度算法：电梯调度算法、循环调度算法、最短距离调度算法、先来先调度。这个磁盘调度算法，那大家就是这四种调度算法就是怎么样去移动这个磁头，然后又去计算一下这个平均的这个寻道长度，或者是平均的这个寻道时间。

PPT 第十章 26--38

隆