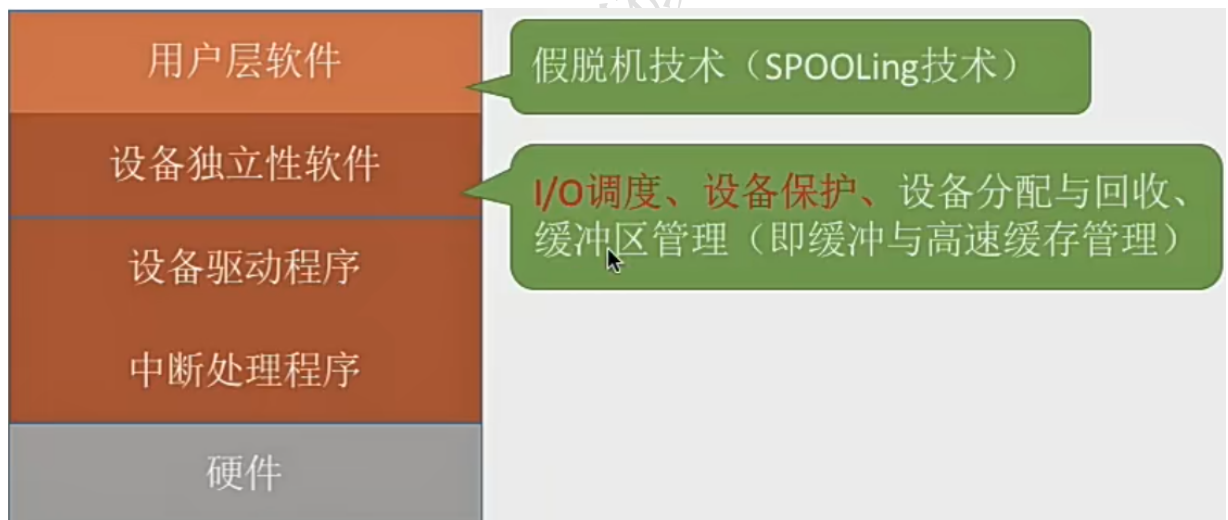


Lecture 12: 知识 - 设备独立性软件

IO 核心子系统

概念: IO 核心子系统

- 概念:
 - 需要重点理解和掌握的功能是 -> I/O 调度、设备保护、假脱机技术（SPOOLing 技术）、设备分配与回收、缓冲区管理（即缓冲与高速缓存）;
- 实现:
 - I/O 调度 -> 用某种算法确定一个好的顺序来处理各个 IO 请求;
 - 设备保护 -> 操作系统需要实现文件保护功能，不同的用户对各个文件有不同的访问权限
 - 如：只读、读和写等;
 - 举例：在 UNIX 系统中，设备被看做是一种特殊的文件，每个设备也会有对应的 FCB;
 - 当用户请求访问某个设备时，系统根据 FCB 中记录的信息来判断该用户是否有相应的访问权限，以此实现“设备保护”的功能
 - 假脱机实现 -> 在用户层实现;
- 图示:



高速缓存与缓冲区

缓冲区基础概念

概念: 缓冲区基础概念

- 概念:
 - 缓冲区是一个存储区域，可以由专门的硬件寄存器组成，也可利用内存作为缓冲区;
 - 使用硬件作为缓冲区的**成本较高，容量也较小**，一般仅用在对速度要求非常高的场合
 - 如存储器管理中所用的联想寄存器，由于对页表的访问频率极高，因此使用速度很快的联想寄存器来存放页表项的副本

- 一般情况下，更多的是利用内存作为缓冲区，“设备独立性软件”的缓冲区管理就是要组织管理好这些缓冲区；
- 缓冲区在逻辑上属于磁盘，但物理上则是驻留于内存中的盘块；
- 目的：
 - 1. 速度一致：
 - 缓和 CPU 与 IO 设备之间速度不匹配的矛盾；
 - 2. 减少中断：
 - 减少对 CPU 的中断频率，放宽对 CPU 中断响应时间的限制；
 - 3. 数据粒度匹配：
 - 解决数据粒度不匹配的问题；
 - 举例：输出进程每次可以生成一块数据但 I/O 设备每次只能输出一个字符；
 - 4. 提供并行性：
 - 缓冲区提高 CPU 与 IO 设备之间的并行性；
 - 5. 减少 IO 次数：
 - 真题：2015年
 - 根据局部性原理，可以在内存中设置 磁盘缓冲区，用于存取磁盘中频繁使用的块来减少磁盘 IO 次数，进而提高磁盘性能；
- 种类：
 - 根据系统设置缓冲区的个数，可以分成单缓冲、双缓冲、循环缓冲区以及缓冲池；
- 补充：
 - (1) 若 IO 所花费的时间远小于 CPU 的处理时间，则缓冲区技术几乎无效；
 - (2) 以下不是缓冲区的目的：
 - 真题：2015年
 - 1. 减少平均寻道时间；
 - 缓冲区在逻辑上属于磁盘，但物理上是驻留于内存中的盘块，因此缓冲区不会减少平均寻道时间；
 - 2. 提高磁盘数据可靠性；
 - 3. 实现设备无关性；
 - (3) 通过延迟写操作，可以优化文件访问速度
 - 真题：2018年
 - 延迟写是指将数据写入缓冲区，并在缓冲区的首部设置延迟写标志，让后将此缓冲区放入空闲缓冲区链表的尾部；
 - 当有进程申请空闲缓冲区而得到该缓冲区时，才将此缓冲区内容写入磁盘；
 - 通过延迟写、可以使得未来在对相同数据进行操作时可以直接访问缓冲区，从而减少了不必要的磁盘访问操作；

单缓冲

概念：单缓冲

- 情况：

- 假设某用户进程请求某种块设备读入若干块的数据，若采用单缓冲的策略，操作系统会在主存中为其分配一个缓冲区；

- 注意：

- (1) 若题目中没有特别说明，一个缓冲区的大小就是一个块；
- (2) 当缓冲区数据非空时，不能往缓冲区传入数据，只能从缓冲区把数据传出；
- (3) 当缓冲区为空时，可以往缓冲区冲入数据，但必须把缓冲区充满以后，才能从缓冲区把数据传出；
- (4) 在“单缓冲”题型中，可以假设**初始状态为工作区满，缓冲区空**；

- 分析：

- 概念：

- T 是数据从 **IO** 设备到缓冲区所需要的实际；
- M 是缓冲区传送到 **CPU** 的工作区所花的时间；
- C 是 **CPU** 处理工作区所花的时间；

- 初始状态：

- 工作区满，缓冲区空

- $T > C$

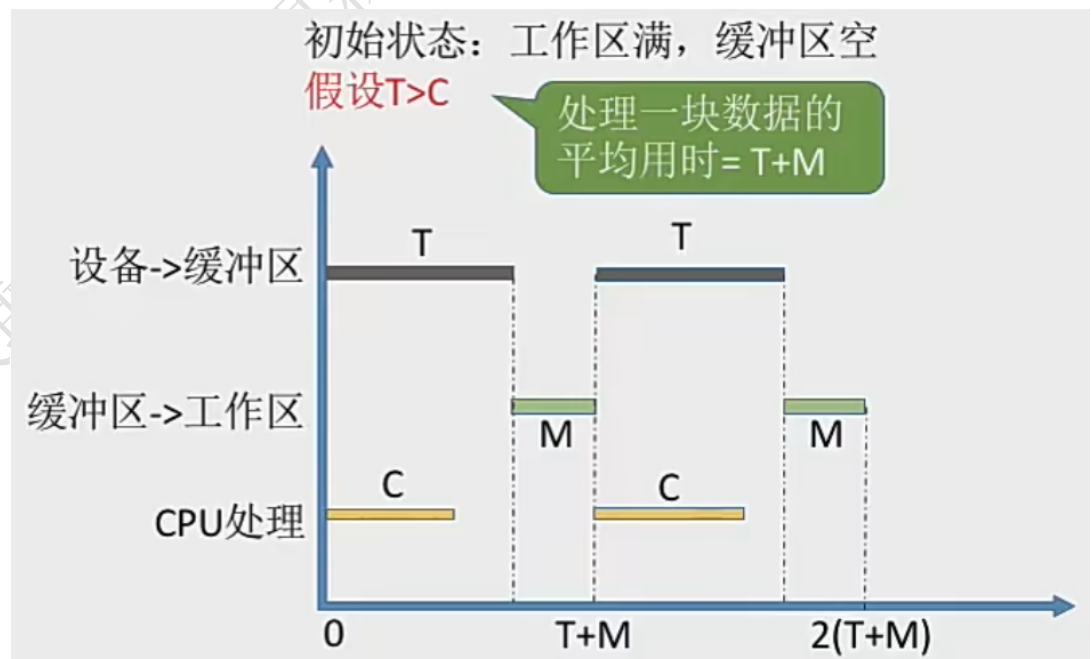
- 概念：

- 数据从 IO 设备到缓冲区所需时间更大；
- CPU 处理完数据后暂时不能将下一块数据传送到工作区，必须等待缓冲区中冲满数据；

- 耗时：

- 处理一块数据的平均用时 = $T + M$

- 图示：



- $T < C$

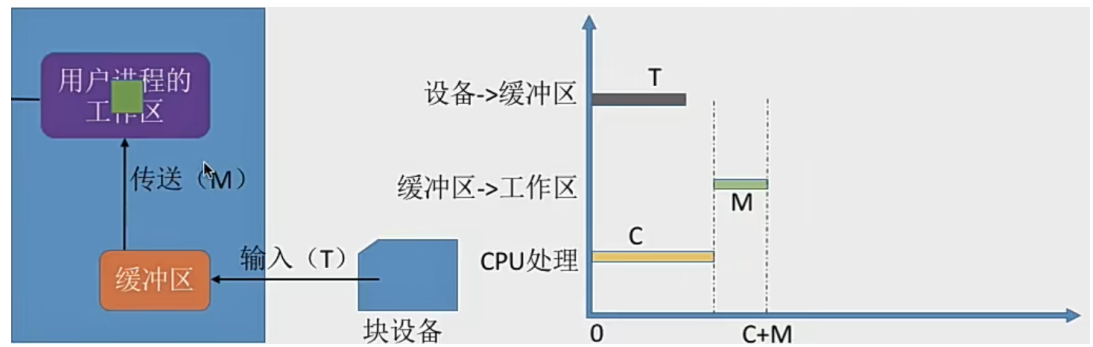
- 概念：

- 缓冲区中冲满数据后暂时不能继续冲入下一块，数据必须等待 CPU 处理结束后将数据从缓冲区传送到工作区；

- 耗时：

- 处理一块数据的平均用时 = $C + M$

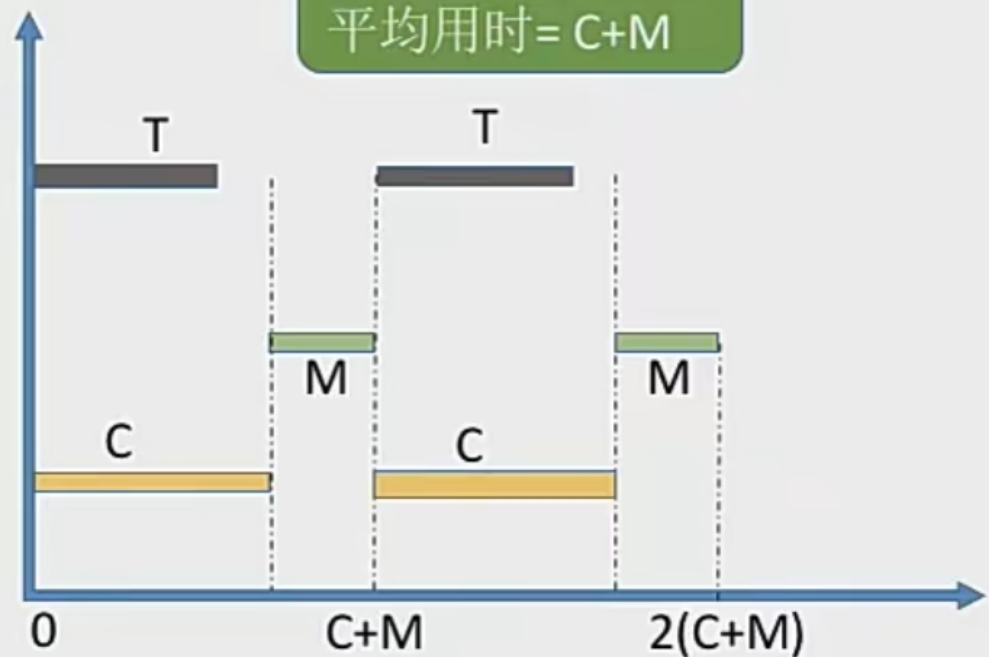
- 图示：



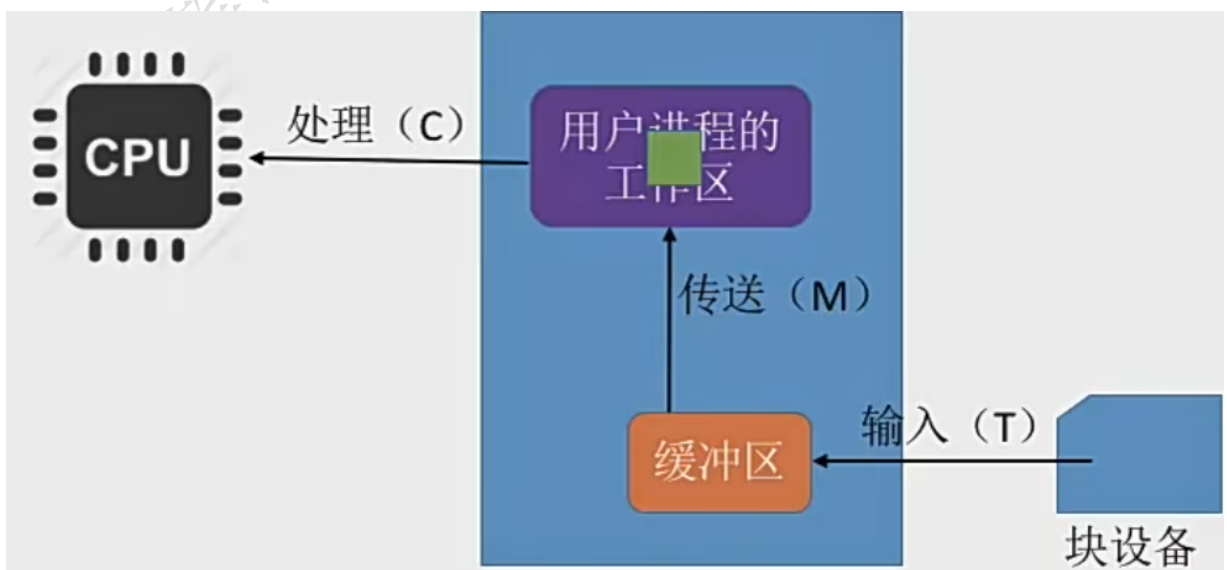
初始状态：工作区满，缓冲区空

假设 $T < C$

处理一块数据的平均用时 = $C + M$



- 图示：



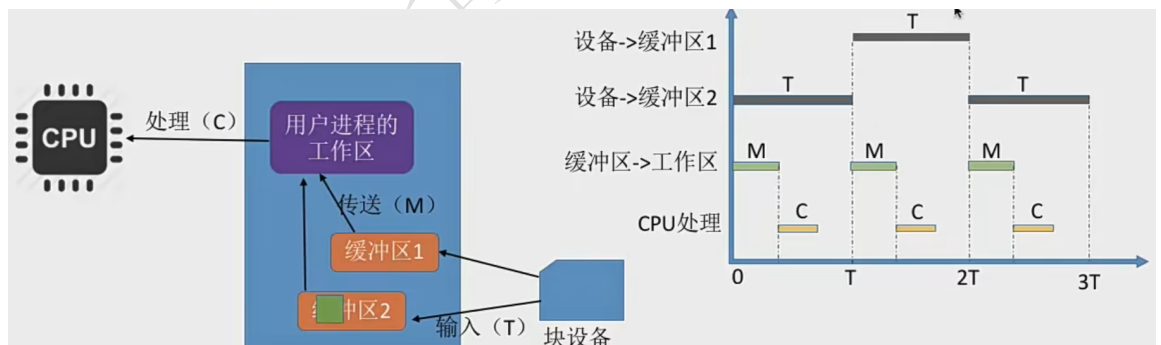
双缓冲

概念：双缓冲

- 概念：
 - 假设某用户进程请求某种块设备读入若干块的数据；
 - 若采用双缓冲的策略，操作系统会在主存中为其分配**两个缓冲区**；
 - 若题目中没有特别说明，一个缓冲区的大小就是一个块；
 - 双缓冲题目中，假设初始状态为 $\rightarrow$ 工作区空，其中一个缓冲区满，另一个缓冲区空；
- 实现：
 - 当设备输入数据时，先将数据送入缓冲区 1，装满后便转向缓冲区 2；
 - 此时，操作系统可以从缓冲区 1 中取出数据，送入用户进程，并由 CPU 对数据进行处理；
 - 当缓冲区 1 中取出的数据处理完后，若缓冲区 2 已冲满，则操作系统又从缓冲区 2 中取出数据送入用户进程处理；
- 结论：
 - 设备的双缓冲机制提高了设备和 CPU 的并行程度；
 - 设备的 C 和 M 是可以和 T 并行的；
- 特性：
 - 1. 双缓冲区支持 **同一时刻** 数据的双向传输；
 - 2. 单缓冲区支持 **一段时间内** 数据的双向传输；
- 分析：

- $T > C+M$

- 处理时间：T



- $T < C+M$

- 处理时间：C + M

- 假设 $2T < 2M+C$ ，则 IO 设备将缓冲区 1 冲满时，缓冲区 2 的数据尚未取空，因此 IO 设备暂时不能冲入数据；

- 平均时间：

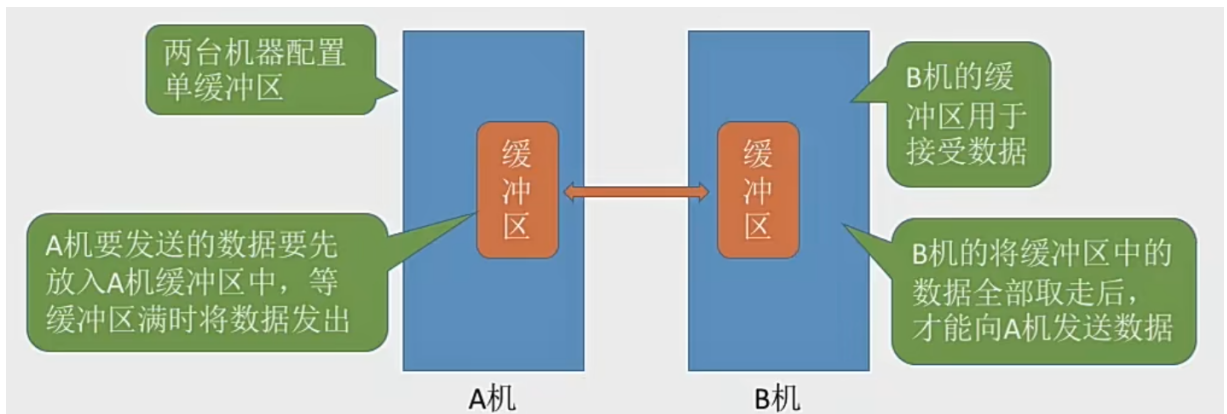
- $\text{Max}(C + M, T)$

分析：使用单/双缓冲在通信时的区别

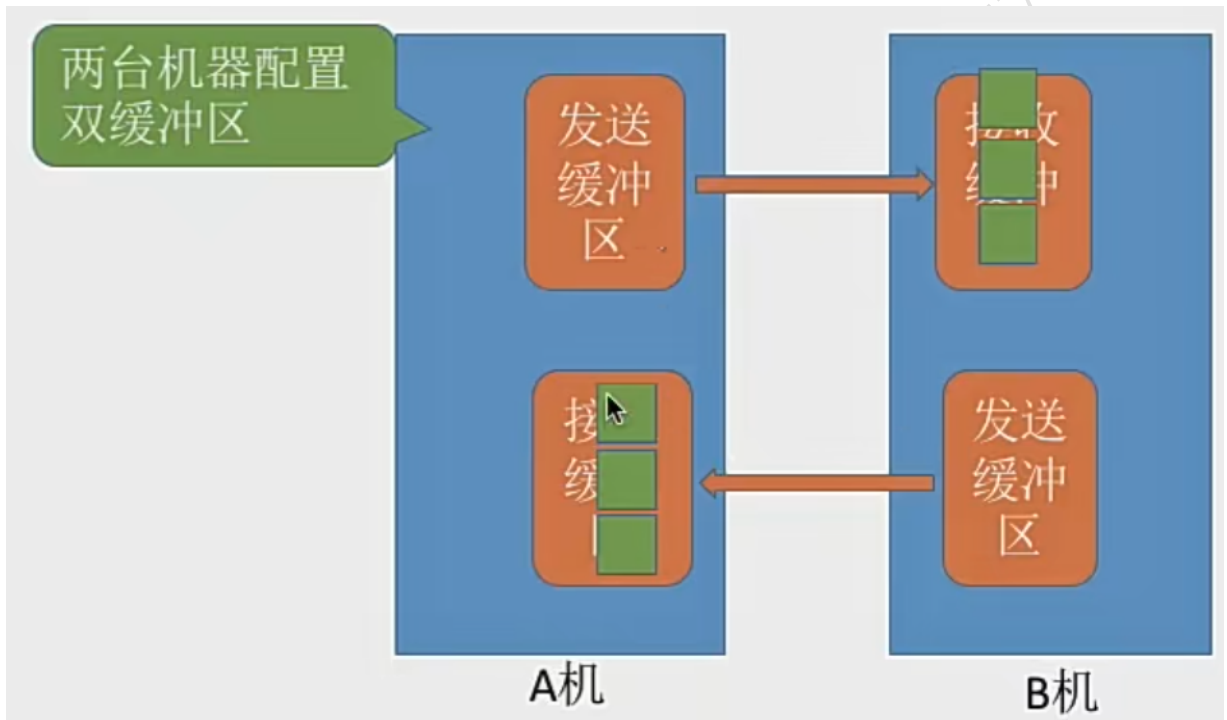
- 概念：
 - 两台机器之间通信时，可以配置缓冲区用于数据的发送和接受；
- 单缓冲区：
 - 若两个相互通信的机器只设置单缓冲区，在任一时刻只能实现数据的单向传输；
- 双缓冲区：

- 若两个相互通信的机器设置双缓冲区，则同一时刻可以实现双向的数据传输；

• 图示：单缓冲区



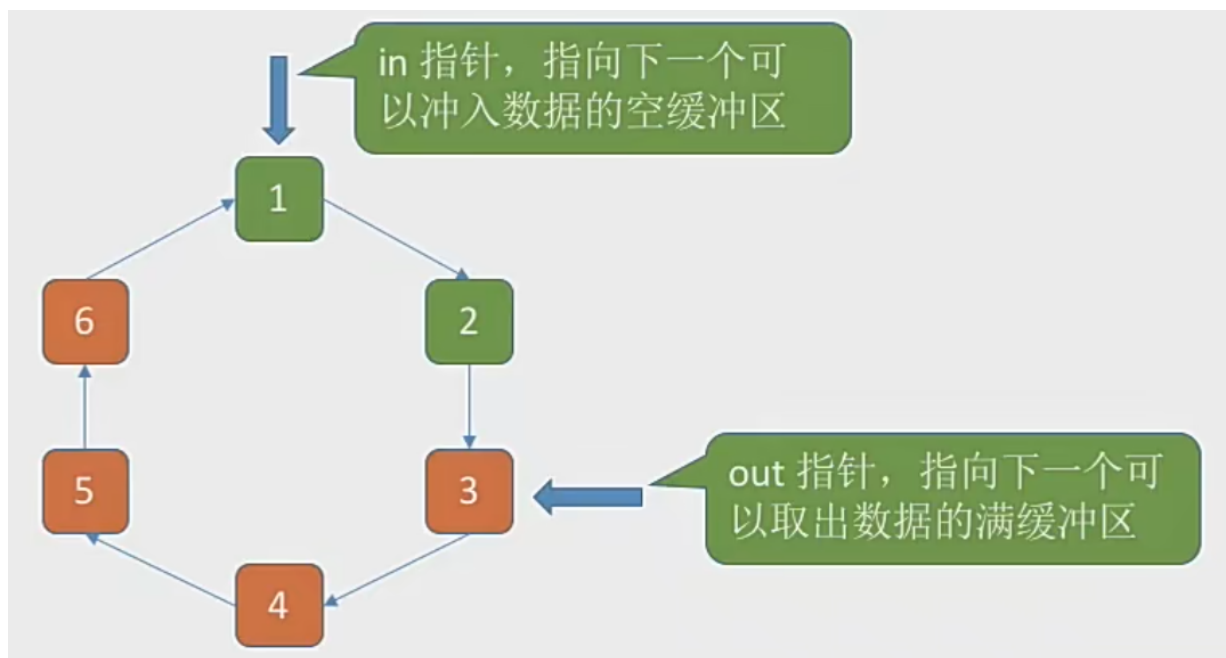
• 图示：双缓冲区



循环缓冲区

概念：循环缓冲区

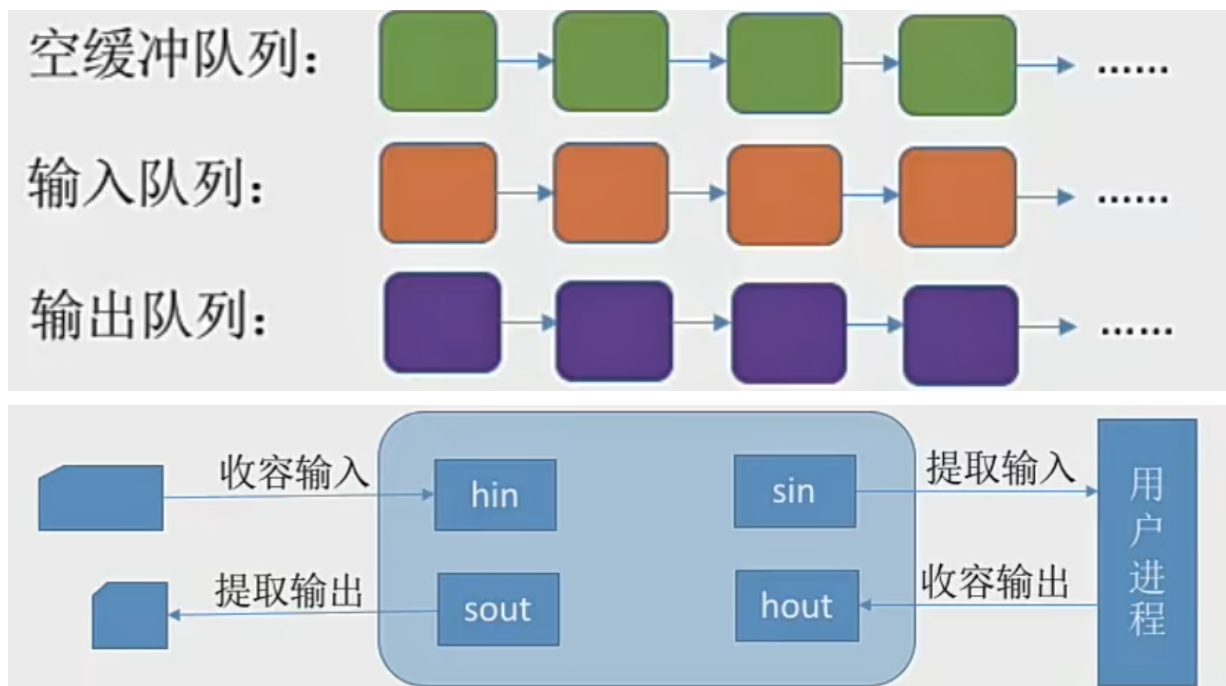
- 概念：
 - 将多个大小相等的缓冲区链接成一个循环队列；
 - 注：以下图示中，橙色表示已充满数据的缓冲区，绿色表示空缓冲区；
- 实现：
 - 循环缓冲包含多个大小相等的缓冲区，每个缓冲区中有一个链接指针指向下一个缓冲区，最后一个缓冲区指针指向第一个缓冲区，多个缓冲区链接成一个循环队列；
 - 循环缓冲中还需设置 `in` 和 `out` 两个指针：
 - `in` 指向第一个可以输入数据的空缓冲区；
 - `out` 指向第一个可以提取数据的满缓冲区；
 - 输入/输出时，`in` 和 `out` 指针沿链接方向循环移动；
- 图示：



缓冲池

概念：缓冲池

- 概念：
 - 由系统中共用的缓冲区组成；
 - 这些缓冲区按使用状况可以分为：
 - (1) 空缓冲队列；
 - (2) 装满输入数据的缓冲队列 (输入队列)；
 - (3) 装满输出数据的缓冲队列 (输出队列)；
 - 另外，根据一个缓冲区在实际运算中扮演的功能不同，又设置了四种工作缓冲：
 - 用于收容输入数据的工作缓冲区 (`hin`)；
 - 用于提取输入数据的工作缓冲区 (`sin`)；
 - 用于收容输出数据的工作缓冲区 (`hout`)；
 - 用于提取输出数据的工作缓冲区 (`sout`)；
- 注意：
 - 不要把循环缓冲和缓冲池的组成搞混了。循环缓冲不属于缓冲池；
- 实现：
 - 从输入队列中取得一块冲满输入数据的缓冲区作为提取输入数据的工作缓冲区 (`sin`) 缓冲区读。空后挂到空缓冲区队列；
- 图示：



设备分配与回收

基础概念

概念：设备分配时的考虑因素

- 设备的固有属性：
 - 独占设备 -> 一个时段只能分配给一个进程 (如打印机);
 - 共享设备 -> 可同时分配给多个进程使用 (如磁盘), 各进程往往是宏观上同时共享使用设备, 而微观上交替使用;
 - 虚拟设备 -> 采用 `SPooling` 技术将独占设备改造成虚拟的共享设备, 可同时分配给多个进程使用 (如采用 `SPooling` 技术实现的共享打印机);
- 设备分配算法：
 - 先来先服务;
 - 优先级高者优先;
 - 短任务优先;
- 设备分配中的安全性：
 - (1) 安全分配方式：
 - 为进程分配一个设备后就将进程阻塞, 本次 `IO` 完成后才将进程唤醒;
 - 举例: 考虑进程请求打印机打印输出的例子;
 - 一个时段内每个进程只能使用一个设备
 - 优点 -> 破坏了“请求和保持”条件, 不会死锁;
 - 缺点 -> 对于一个进程来说, `CPU` 和 `IO` 设备只能串行工作;
 - (2) 不安全分配方式：
 - 进程发出 `IO` 请求后, 系统为其分配 `IO` 设备, 进程可继续执行, 之后还可以发出新的 `I/O` 请求;
 - 只有某个 `IO` 请求得不到满足时才将进程阻塞;
 - 一个进程可以同时使用多个设备;

- 优点 -> 进程的计算任务和 IO 任务可以并行处理，使进程迅速推进
- 缺点 -> 有可能发生死锁（死锁避免、死锁的检测和解除）

• 举例：

• 题目：

【2023 统考真题】下列因素中，设备分配需要考虑的是（ ）。

I. 设备的类型

II. 设备的访问权限

III. 设备的占用状态

IV. 逻辑设备与物理设备的映射关系

- A. 仅 I、II
- B. 仅 II、III
- C. 仅 III、IV
- D. I、II、III、IV

• 解答：

- D

• 分析：

• 设备的类型：

- 设备的类型决定了设备的固有属性，如独占性、共享性、可虚拟性等。不同类型的设备需要采用不同的分配方式，如独占分配、共享分配、虚拟分配等；

• 设备的访问权限：

- 设备的访问权限决定了哪些进程可以使用哪些设备，以保证系统的安全性和保密性，通常系统设备只能由系统进程或特权进程访问，用户设备只能由用户进程或授权进程访问；

• 设备的占用状态：

- 设备的占用状态决定了设备是否可以被分配给请求进程，以及如何处理等待进程，若设备空闲，则通常可以直接分配给请求进程；
- 若设备忙，则需要将请求进程排入设备队列，并按照一定的算法进行调度；

• 逻辑设备与物理设备的映射关系：

- 逻辑设备与物理设备的映射关系决定了如何通过逻辑地址访问物理地址，以提高系统的灵活性和可扩展性，通常系统会为每个物理设备分配一个逻辑名，并建立一个系统设备表来记录逻辑名与物理名之间的对应关系；

概念：静态分配与动态分配

• 静态分配：

• 概念：

- 进程运行前为其分配全部所需资源，运行结束后归还资源；

• 特点：

- 破坏了“请求和保持”条件，不会发生死锁；

• 动态分配：

- 进程运行过程中动态申请设备资源；

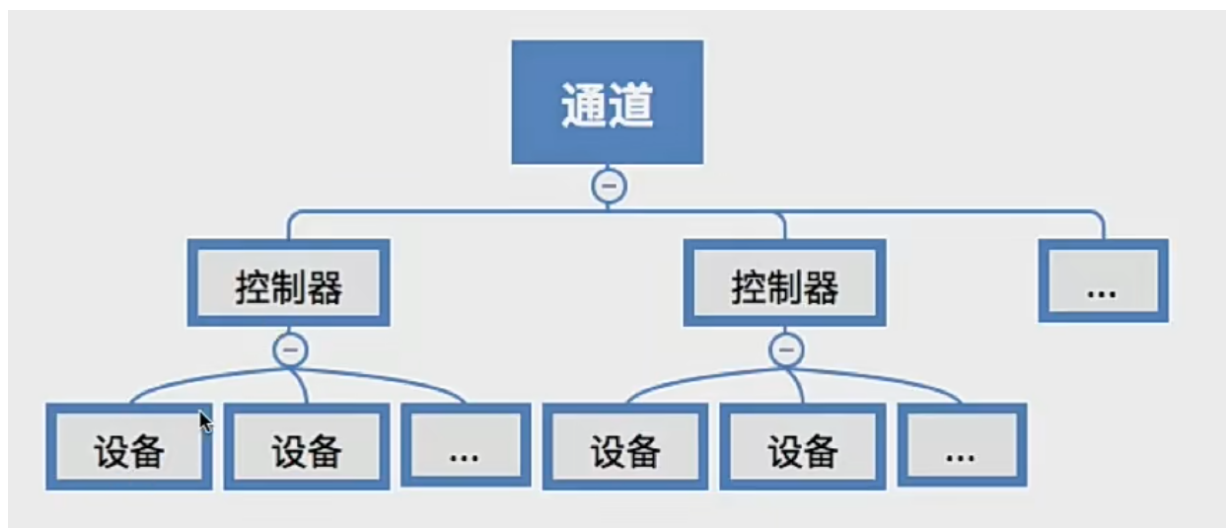
设备分配中的数据结构

关系：“设备、控制器、通道”之间的关系

• 概念：

- 一个通道可控制多个设备控制器，每个设备控制器可控制多个设备；

• 图示：



概念：各个具体数据结构

- 设备控制表：

- 概念：

- 设备控制表 (DCT) -> 系统为每个设备配置一张 DCT，用于记录**设备情况**；

- 图示：

设备控制表 (DCT)	
设备类型	如：打印机/扫描仪/键盘
设备标识符	即物理设备名，系统中的每个设备的物理设备名唯一
设备状态	忙碌/空闲/故障...
指向控制器表的指针	每个设备由一个控制器控制，该指针可找到相应控制器的信息
重复执行次数或时间	当重复执行多次I/O操作后仍不成功，才认为此次I/O失败
设备队列的队首指针	指向正在等待该设备的进程队列（由进程PCB组成队列）

- 控制器控制表：

- 概念：

- 控制器控制表 (COCT) -> 每个设备控制器都会对应一张 COCT，操作系统根据 COCT 的信息对控制器进行操作和管理；

- 图示：

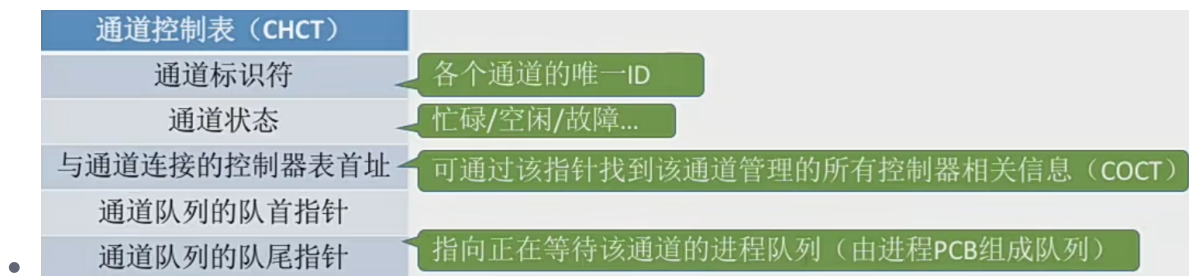
控制器控制表 (COCT)	
控制器标识符	各个控制器的唯一ID
控制器状态	忙碌/空闲/故障...
指向通道表的指针	每个控制器由一个通道控制，该指针可找到相应通道的信息
控制器队列的队首指针	
控制器队列的队尾指针	指向正在等待该控制器的进程队列（由进程PCB组成队列）

- 通道控制表：

- 概念：

- 通道控制表 (CHCT) -> 每个通道都会对应一张 CHCT；
 - 操作系统根据 CHCT 的信息对通道进行操作和管理；

- 图示：



- 系统控制表:

- 概念:

- 系统设备表 (SDT) -> 记录了系统中全部设备的情况, 每个设备对应一个表目;

- 图示:



设备分配过程

概念: 设备分配

- 步骤:

- 1. 根据进程请求的物理设备名查找 SDT (注: 物理设备名是进程请求分配设备时提供的参数);
 - 2. 根据 SDT 找到 DCT, 若设备忙碌则将进程 PCB 挂到设备等待队列中, 不忙碌则将设备分配给进程;
 - 3. 根据 DCT 找到 COCT, 若控制器忙碌则将进程 PCB 挂到控制器等待队列中, 不忙碌则将控制器分配给进程;
 - 4. 根据 COCT 找到 CHCT, 若通道忙碌则将进程 PCB 挂到通道等待队列中, 不忙碌则将通道分配给进程;

- 注意:

- 只有设备控制器、通道三者都分配成功时, 这次设备分配才算成功, 之后便可启动 I/O 设备进行数据传送;

- 顺序: SDT: 系统分配表 -> DCT: 设备分配表 -> COCT: 控制器分配表 -> CHCT: 通道分配表;

概念: 分配方法的改进 - 逻辑设备表

- 引入:

- 原方法的缺点:

- 1. 用户编程时必须使用“物理设备名”, 底层细节对用户不透明, 不方便编程;
 - 2. 若换了一个物理设备, 则程序无法运行;

- 3. 若进程请求的物理设备正在忙碌，则即使系统中还有同类型的设备，进程也必须阻塞等待；

- 改进：

- 建立逻辑设备名与物理设备名的映射机制，用户编程时只需提供**逻辑设备名**；

- 概念：逻辑设备表

- 逻辑设备表 (LUT) -> 建立了逻辑设备名与物理设备名之间的映射关系；
- 某用户进程第一次使用设备时使用**逻辑设备名**向操作系统发出请求，操作系统根据用户进程指定的设备类型 (逻辑设备名) 查找 **系统设备表 SDT**，找到一个空闲设备分配给进程，并在 **LUT** 中增加相应表项；
- 如果之后用户进程再次通过相同的逻辑设备名请求使用设备则操作系统通过 **LUT** 表即可知道用户进程实际要使用的是哪个物理设备了，并且也能知道该设备的驱动程序入口地址；

- 注意：

- 真题：2020年
- (1) 用户可以**使用文件名访问物理设备**；
 - 比如 **Linux** 系统将所有的设备都看作文件来处理，用户可以像使用文件一样使用这些设备，而无需了解它们的具体存在与实现形式；
 - 注意这里说的是文件名，不是物理设备名；
- (2) 更改物理设备后不必修改访问该设备的应用程序；
 - 由于设备独立性的实现，用户进程使用**逻辑设备名**来访问设备，更改物理设备不会对设备访问产生影响，无需修改应用程序；

- 图示：

逻辑设备名	物理设备名	驱动程序入口地址
/dev/printer	3	1024
/dev/tty	5	2046
...	...	...

- 设置分析：

- 整个系统只有一张 **LUT** -> 各用户所用的逻辑设备名不允许重复，适用于单用户操作；
- 系统每个用户一张 **LUT** -> 不同用户的逻辑设备名可重复，适用于多用户操作系统；

- 步骤：

- 1. 根据进程请求的**逻辑设备名**，查找 **SDT** (注：用户编程时提供的逻辑设备名其实就是“设备类型”)
- 2. 查找 **SDT**，找到用户进程指定类型的、并且空闲的设备，将其分配给该进程。操作系统在逻辑设备表 (**LUT**) 中新增一个表项；

- 3. 根据 DCT 找到 COCT，若控制器忙碌则将进程 PCB 挂到控制器等待队列中，不忙碌则将控制器分配给进程；
- 4. 根据 COCT 找到 CHCT，通道忙碌则将进程 PCB 挂到通道等待队列中，不忙碌则将通道分配给进程；
- 顺序：
 - 进程 -> LUT 逻辑设备表 -> SDT：系统分配表 -> DCT：设备分配表 -> COCT：控制器分配表 -> CHCT：通道分配表；

假脱机技术

概念：假脱机技术的起源 - 脱机技术

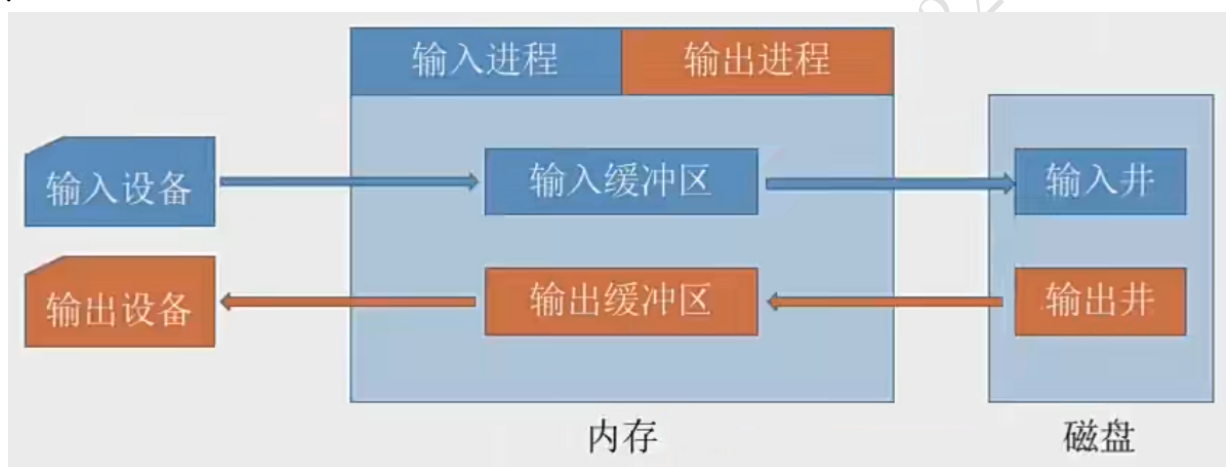
- 起源：
 - 最开始的手工操作阶段 -> 主机直接从 IO 设备获得数据，由于设备速度慢，主机速度很快。人机速度矛盾明显，主机要浪费很多时间来等待设备；
 - 批处理阶段 -> 引入了脱机输入/输出技术 (用磁带完成) -> 在**外围控制机**的控制下，慢速输入设备的数据先被输入到更快速的磁带上之后，主机可以从快速的磁带上读入数据，从而缓解了速度矛盾；
 - 脱机 -> 脱离主机的控制，进行 IO 操作；
- 核心：
 - 用一块访问速度更快的区域（一般是磁盘）作为缓存，将 CPU 和 IO 之间交换的数据缓存到专门的区域；
- 优点：
 - 引入脱机技术后，缓解了 CPU 与慢速 I/O 设备的速度矛盾；
 - 另一方面，即使 CPU 在忙碌，也可以提前将数据输入到磁带；
 - 即使慢速的输出设备正在忙碌，也可以提前将数据输出到磁带；

概念：假脱机技术

- 概念：
 - 又称 SPooling 技术，是用**软件的方式**模拟脱机技术；
 - 由 1. 预输入程序；2. 井管理程序；3. 缓输出程序；三者构成
 - 假脱机技术允许**多个作业**同时共享一台**独占**设备；
- 注意：
 - **真题：**2016年
 - 假脱机技术必须需要**多进程支持**，因为外围控制器需要支持同时有输入进程和输出进程；
- 构成：
 - 输入井：
 - 模拟脱机输入时的磁带，用于收容 IO 设备输入的数据；
 - 输出井：
 - 模拟脱机输出时的磁带，用于收容用户进程输出的数据；
 - 举例：在用户使用打印机时，打印结果首先被送到输出井，然后再进入输出缓冲区；
 - 磁盘：
 - 磁盘上开辟两个存储区域 -> 输入井和输出井；

- 外围控制机：
 - 概念：
 - 由 输入进程 和 输出进程 构成；
 - 又称**并管理程序**，用于控制设备与输入井、输出井之间信息的交换
 - 输出进程：
 - 模拟脱机输入时的外围控制机；
 - 在输出进程的控制下，“输出缓冲区”用于暂存从输出井送来的数据，之后再传送到输出设备上；
 - 输入进程：
 - 模拟脱机输出时的外围控制机；
 - 在输入进程的控制下，“输入缓冲区”用于暂存从输入设备输入的数据，之后再转存到输入井中；

• 图示：



分析：假脱机技术的应用分析 - 共享打印机

- 设备类型：
 - 独占式设备：只允许各个进程串行使用的设备，一段时间内只能满足一个进程的请求；
 - 共享设备：允许多个进程“同时”使用的设备 (宏观上同时使用，微观上可能是交替使用)，可以同时满足多个进程的使用请求；
- 改造：
 - 打印机是种 独占式设备 -> 但是可以用假脱机技术改造成 共享设备；
- 举例：
 - 前提：
 - 若进程 1 正在使用打印机，则进程 2 请求使用打印机时必然阻塞等待；
 - 分析：
 - 当多个用户进程提出输出打印的请求时，系统会答应它们的请求，但是并不是真正把打印机分配给他们，而是由假脱机管理进程为每个进程做两件事：
 - (1) 在 磁盘输出井 中为进程申请一个 空闲缓冲区 (这个缓冲区是在磁盘上的)，并将要打印的数据送入其中；
 - (2) 为 用户进程 申请一张空白的 打印请求表，并将用户的打印请求填入表中 (其实就是用来说明用户的打印数据存放位置等信息的)，再将该表挂到 假脱机文件队列 上；

- (3) 当打印机空闲时，输出进程会从文件队列的**队头**取出一张**打印请求表**，并根据表中的要求将要打印的数据从 **输出井** 传送到 **输出缓冲区**，再输出到打印机进行打印。用这种方式可依次处理全部的打印任务
- 结论：
 - 虽然系统中只有一个台打印机，但每个进程提出打印请求时，系统都会为在输出井中为其分配一个存储区 (相当于分配了一个逻辑设备)，使每个用户进程都觉得自己独占一台打印机，从而实现对打印机的共享；

设备驱动程序接口

概念：设备驱动程序接口

- 概念：
 - 若各公司开发的设备驱动程序接口不统一，则操作系统很难调用设备驱动程序；
 - 操作系统规定好设备驱动程序的接口标准，各厂商必须按要求开发设备驱动程序；
 - 设备驱动程序需要处理与设备有关的中断处理过程；
- 图示：

