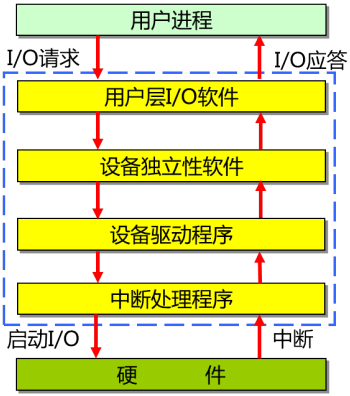

河南大学
操作系统
计算机学院

第 6 章

输入输出系统

大纲

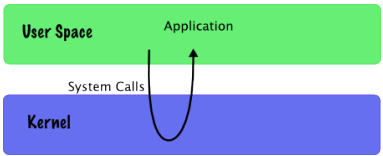
用户层 I/O 软件



- 多数I/O软件都在操作系统中，用户空间中也有一小部分。通常它们以库函数形式出现。
- 用户层软件必须通过一组系统调用来取得操作系统服务。通过调用对应的库函数使用系统调用。
- I/O软件还包括运行在内核之外的守护进程（后台）等。
- 用户空间中另一个重要的I/O软件是SPOOLING系统，可在多道程序系统中实现虚拟设备技术。

系统调用和库函数

- 系统调用（System Call）是操作系统内核提供的一组功能强大的函数，在核心态下运行。
- 系统调用是用户程序取得 OS 服务的唯一途径。
- 库函数是系统调用的一种封装和扩展，工作在用户态。
- 用户层 I/O 软件提供了一些读/写设备和控制/检查设备的库函数，以方便用户取得 OS 的服务。



假脱机技术

- 脱机输入输出技术

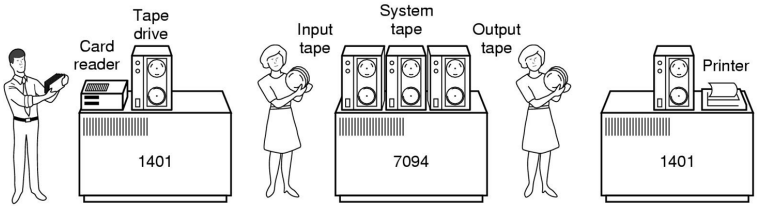
  为了缓和 CPU 的高速性与 I/O 设备的低速性间矛盾而引入，该技术在外围控制机的控制下实现低速的 I/O 设备与高速的磁盘之间进行数据传送。

- SPOOLing（Simultaneous Peripheral Operations On-Line，外部设备联机并行操作），也称假脱机技术。

- SPOOLing 是指在多道程序的环境下，利用多道程序中的一道或两道程序来模拟外围控制机，从而在联机的条件下实现脱机 I/O 的功能。

假脱机技术

- 设计思想：用常驻内存的进程去模拟外围机，从而用一台主机完成脱机技术中需用三台计算机完成的工作。
- 功能:
  1. 把独占设备改造为逻辑共享设备。
  2. 把一台物理 I/O 设备虚拟为多台逻辑 I/O 设备。



SPOOLing 系统的组成

SPOOLing 系统的组成

1. 输入井和输出井
2. 输入缓冲区和输出缓冲区
3. 输入进程和输出进程
4. 井管理程序

SPOOLing 系统的组成

1. 输入井和输出井（外存中）
   - 输入井和输出井是在磁盘上开辟的两个大的存储空间，分别模拟输入设备和输出设备。
   - 输入井用于暂存 I/O 设备输入的数据。
   - 输出井用于暂存用户程序的输出数据。
2. 输入缓冲区和输出缓冲区（内存中）
   - 输入缓冲区和输出缓冲区是为了缓和 CPU、磁盘、I/O 设备之间速度不匹配的矛盾而设置的。
   - I/O 设备 ⇒ 输入 buf ⇒ 输入井
   - 输出井 ⇒ 输出 buf ⇒ I/O 设备

SPOOLing 系统的组成

3 输入进程和输出进程

- 输入进程和输出进程用于控制 I/O 设备与磁盘井之间的信息交换。
- 输入进程 SPi 相当于脱机输入控制器。

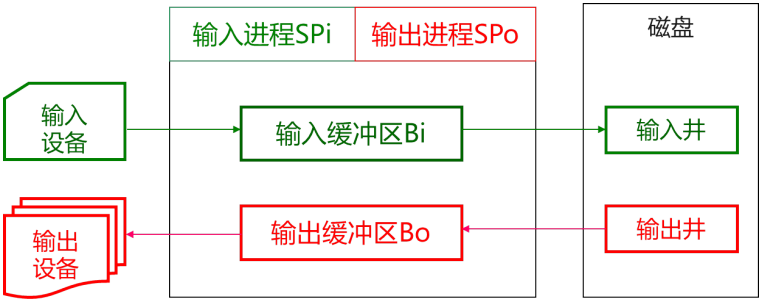
  控制：I/O 设备 ⇒ 输入 buf⇒ 输入井
- 输出进程 SPo 相当于脱机输出控制器。

  控制：输出井 ⇒ 输出 buf⇒I/O 设备

4 井管理程序

- 井管理程序用于控制作业与磁盘井之间的信息交换。
- 磁盘井 ⇔ 进程 buf ⇔ 用户进程

SPOOLing 系统的组成



- CPU 需要输入数据时，由输入进程 SPi控制把数据从输入设备送到输入井，cpu 再从输入井读入内存。
- CPU 需要输出数据时，把数据从内存送到输出井，由输出进程 SPo控制，把数据从输出井送到输出设备。

SPOOLing 举例

输入

- （1）进程 n 请求输入 ⇒ SPi 为进程 n 在输入井中分配空间 ⇒ 设备数据由输入 buf送入输入井 ⇒ 生成输入请求表挂在输入请求队列（允许多个进程同时请求输入）。
- （2）CPU 空闲 ⇒ 取请求表中的任务 ⇒ 进程 buf 。

输出（假脱机打印机系统）

- （1）进程 n 请求输出 ⇒ SPo 为进程 n 在输出井中分配空间 ⇒ 将数据由进程 buf转到输出井 ⇒ 生成打印请求表挂在打印请求队列（允许多个进程同时请求输出）。
- （2）打印机空闲 ⇒ 查打印请求表中的任务 ⇒ 取输出井中的数据 ⇒ 输出 buf ⇒ 打印

SPOOLing 系统的特点

1. 提高了 I/O 速度。
   把对低速设备操作变为对输入/输出井的操作。
2. 将独占设备改造为共享设备。
   分配设备的实质是分配输入/输出井，并为进程分配一个存储区和建立一张 I/O 请求表。
3. 实现了虚拟设备功能。

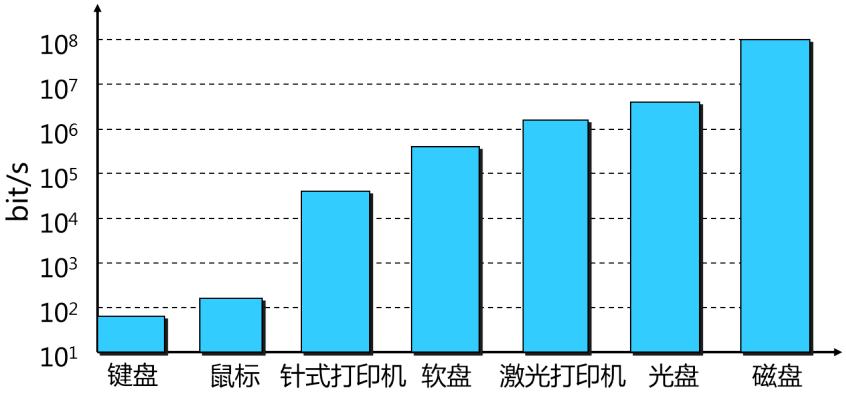
Spooling 系统

- 虚拟设备实现了对独占设备的改造，虚拟分配使进程对独占设备的使用与物理设备分离，使进程与 I/O 设备之间的同步 I/O 方式转变为异步 I/O 方式（在联机的条件下实现脱机 I/O），提高了进程的并发度和执行效率。
- 虚拟分配方式在逻辑上改造了设备特性，提高了设备的利用率，同时也提高了进程的执行效率。
- SPOOLING 付出的代价是外存空间的开销，利用空间换取时间。

守护进程（daemon）

- 守护进程是工作在后台的一种服务进程。可以把守护进程应用于 SPOOLing 技术中。
- 以打印机为例，系统创建 SPOOLing 打印守护进程，它是唯一获准使用打印机的进程，打印的文件首先被放入 SPOOLing 目录，此后当打印机空闲时，守护进程可以定时地或者在 SPOOLing 目录下的内容达到一定容量时，把 SPOOLing 目录下的文件打印输出。
- SPOOLing 技术还有很多应用。例如电子邮件系统。发送 E-mail 时，Send 程序接收要发送的信件并将其送入一个固定的 SPOOLing 目录下，待以后由守护进程将其取出发送。整个 E-mail 系统在操作系统之外运行。

缓冲区管理



- 几乎所有的 I/O 设备都使用缓冲区来缓解 CPU 与 I/O 传输速度不匹配的矛盾。

缓冲的实现方法

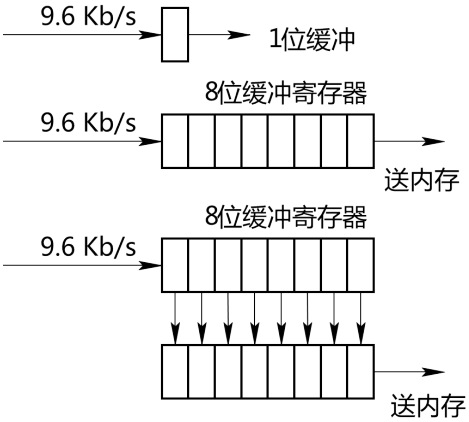
1. 采用硬件缓冲器实现（如联想存储器）
2. 采用软缓冲区来实现
   - 软缓冲区是指在 I/O 操作期间，专门用来临时存放输入/输出数据的一块存储区域，又称为内存缓冲区。
   - 除在关键地方可采用硬件缓冲器外，大都采用软缓冲区来实现。

缓冲技术好比是一个水库，如果上游来的水太多，下游来不及排走，水库就起到“缓冲”作用，先让水在水库中停一段时间，等下游能继续排水时再把水送往下游。

缓冲的引入

1. 缓解 CPU 与 I/O 设备间速度不匹配的矛盾。
2. 减少中断 CPU 的次数。
   buffer 越大，“buffer 满”发生频率越低。
3. 解决数据粒度（即基本数据单元大小）不匹配的问题。
   如：生产者生产的产品数据粒度与消费者消费的数据粒度不一定一致。
4. 提高 CPU 与 I/O 设备的并行性。

缓冲的引入



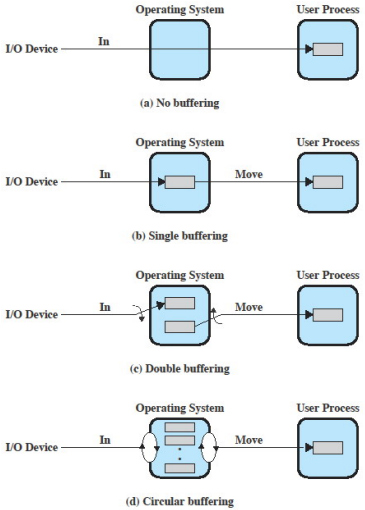
- 速率为9.6 Kb/s的数据通信意味着其中断CPU的频率也为9.6 Kb/s，即每100 μs就要中断CPU一次。
- 而且CPU必须在100 μs内予以响应，否则缓冲区内的数据将被冲掉。
- 若设置一个具有8位的缓冲(移位)寄存器，则可使CPU被中断的频率降低为原来的1/8 。
- 若再设置一个8位寄存器，则又可把CPU对中断的响应时间从100 μs放宽到800 μs。

1μs (微秒) = 10^{-6}秒

缓冲的引入

缓冲技术的分类

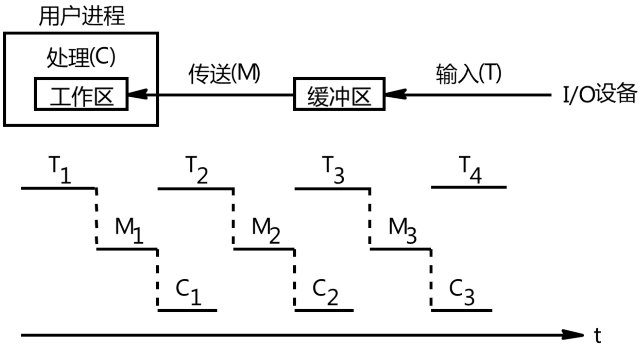
- 单缓冲
- 双缓冲
- 循环缓冲
- 缓冲池



单缓冲区

- 单缓冲区方式是在设备和处理机之间设置一个缓冲区。
- 设备与处理机交换数据时，先把交换的数据写入缓冲区，然后需要数据的设备/处理机再从缓冲区中取走数据。
- 单缓冲区方式可以缓解 CPU 与 I/O 设备间速度不匹配的矛盾。
- 由于缓冲区属于临界资源，不允许多个进程同时对一个缓冲区操作，因此在某一段时间内，缓冲区只能存放输入数据或输出数据（单向传输）。

单缓冲区

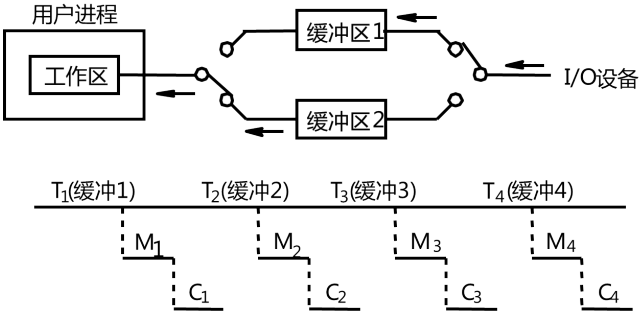


- 无缓冲区时，处理一块数据的时间为 T+C。
- 有缓冲区时，C 和 T 可并行，M 和 T 不能并行，因此处理一块数据的时间：Max(C,T)+M。

双缓冲区

- 双缓冲区方式是在设备和处理机之间设置两个缓冲区。
- 双缓冲提高了设备与处理机并行操作的程度，只有当两个缓冲区均为空时，需要数据的进程才等待。
- 收发可双向同时传送。例如，在双缓冲区方式中，CPU 可把输出数据放入其中一个缓冲区由打印机输出，然后它又可以从另一个缓冲器区中读取所需要的输入数据。

双缓冲区



- M 与 T 可并行，C 与 T 也可并行，因此处理一块数据的时间约为 Max(C+M,T)。通常 M≪T，因此处理一块数据的时间可以粗略地估计为 Max(C,T)。
- 如果 C<T，可使块设备连续输入。

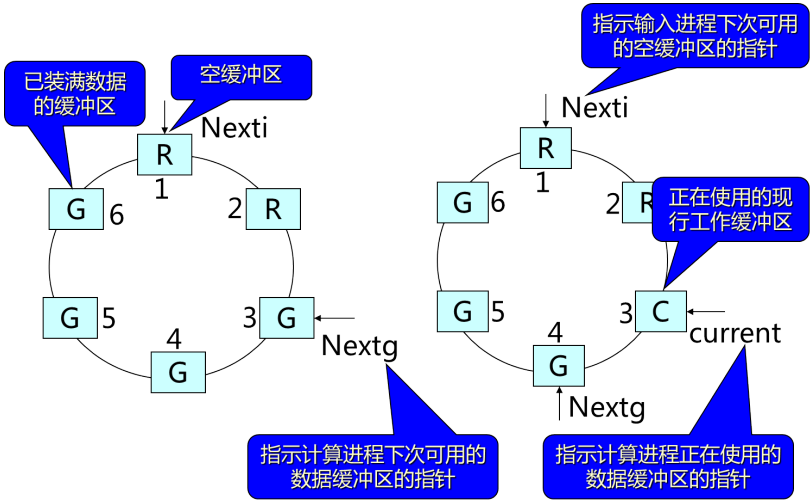
环形缓冲区

- 当输入输出速度基本相等时可以采用双缓冲区方式，否则效果不理想。
- 当输入输出速度相差较多，或者数据量比较大时，必须考虑增加缓冲区的数量以改善系统性能，这就是多缓冲区方式。
- 多个缓冲区通常被组织成循环队列，称为循环缓冲（环形缓冲区）。

环形缓冲区的组成

- 多个缓冲区
  - 在设备和处理机之间设置多个大小相等的缓冲区，这些缓冲区构成环形。
  - 进一步提高了设备与处理机并行操作的程度。
- 多个指针
  - Nexti：指示下次可用的空缓冲区的指针
  - Nextg：指示下次可用的数据缓冲区的指针
  - Current：指示正在使用的缓冲区的指针

环形缓冲区的组成


已装满数据的缓冲区
空缓冲区
Nexti
R
1
2
R
3
G
Nextg
4
G
5
G
6
G
指示计算进程下次可用的数据缓冲区的指针
指示输入进程下次可用的空缓冲区的指针
Nexti
R
1
2
R
正在使用的现行工作缓冲区
3
C
current
4
G
Nextg
5
G
6
G
指示计算进程正在使用的数据缓冲区的指针

环形缓冲区的使用

- Getbuf 过程
  - 计算进程：调用 Getbuf 过程，取 Nextg 对应缓冲区使用，将 Nextg 对应的缓冲区置为空，Nextg=（Nextg+1）Mod N
  - 输入进程：调用 Getbuf 过程，取 Nexti 对应缓冲区使用，将 Nexti 对应的缓冲区置为满，Nexti=（Nexti+1）Mod N
- Releasebuf 过程
  - 输入进程：把 C 装满后，调用 Releasebuf 过程释放缓冲区，改为 G
  - 计算进程：把 C 的数据提取完毕时，调用 Releasebuf 过程释放缓冲区，改为 R

进程之间的同步问题

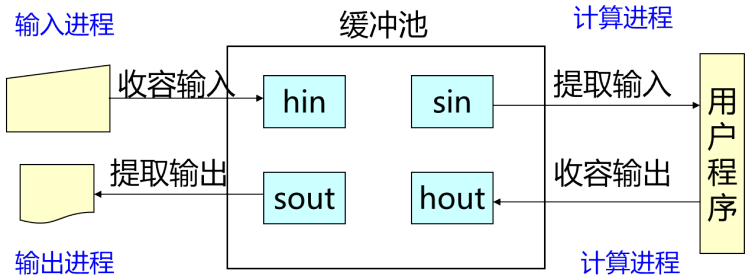
- 指针 Nexti 和指针 Nextg 将不断地沿着顺时针方向移动。
- Nexti 追上 Nextg：表示输入速度 > 输出速度，全部 buf 满，再无空 buf 可用，这时输入进程阻塞。这种情况被称为系统受计算限制。
- Nextg 追上 Nexti：输入速度 < 输出速度，全部 buf 空，这时输出进程阻塞。这种情况被称为系统受 I/O 限制。

缓冲池

- 上述三种缓冲区的组织形式仅适用于某种特定的 I/O 进程和计算进程，属于专用缓冲。
- 当系统中的设备很多时，将会有许多这样的循环缓冲区，消耗大量的内存空间，而且其利用率也不高。
- 为了提高缓冲区的利用率，可以采用公共缓冲池技术，其中的缓冲区可为多个设备和进程服务。
- 缓冲池：将系统内所有的缓冲区统一管理起来，就形成了能用于输入/输出的缓冲池。缓冲池通常由若干大小相同的缓冲区组成，是系统的公用资源，任何进程（输入、输出）都可以申请使用缓冲池中的各个缓冲区。

缓冲池的组成

- 三个队列（链表）：空缓冲队列 emq、装满输入数据队列 inq、装满输出数据队列 outq
- 四种工作缓冲区：收容输入缓冲区 hin、提取输入缓冲区 sin、收容输出缓冲区 hout、提取输出缓冲区 sout



getbuf 过程和 putbuf 过程

- Addbuf(type,number) 过程：用于将参数 number 所指示缓冲区挂在 type 队列上。
- Takebuf(type) 过程：用于从 type 所指示的队列的队首摘下一个缓冲区。
- MS(type)：互斥信号量
- RS(type)：资源信号量（如缓冲队列中缓冲区的数量）

getbuf 过程和 putbuf 过程

```
Procedure Getbuf(type)
  begin
     wait(RS(type));
     wait(MS(type));
     B(number):=Takebuf(type);
     signal(MS(type));
   end
```

```
Procedure Putbuf(type)
   begin
      wait(MS(type));
      Addbuf(type,number);
      signal(MS(type));
      signal(RS(type));
 end
```

MS(type)：互斥信号量　RS(type)：资源信号量（缓冲区的数量）

缓冲池的工作方式

输入进程：

1. 从空缓冲队列的队首取一空缓冲区用作收容输入缓冲区：hin=getbuf(emq)
2. 输入数据：输入设备将数据输入收容输入缓冲区并装满。
3. 将此缓冲区挂到装满输入数据队列队尾：putbuf(inq, hin)

计算进程：

1. 从装满输入数据队列队首取一满缓冲区用作提取输入缓冲区：sin=getbuf(inq)
2. 计算：CPU 从提取输入缓冲区中取出数据至用完。
3. 将空缓冲区挂到空缓冲队列队尾：putbuf(emq, sin)

缓冲池的工作方式

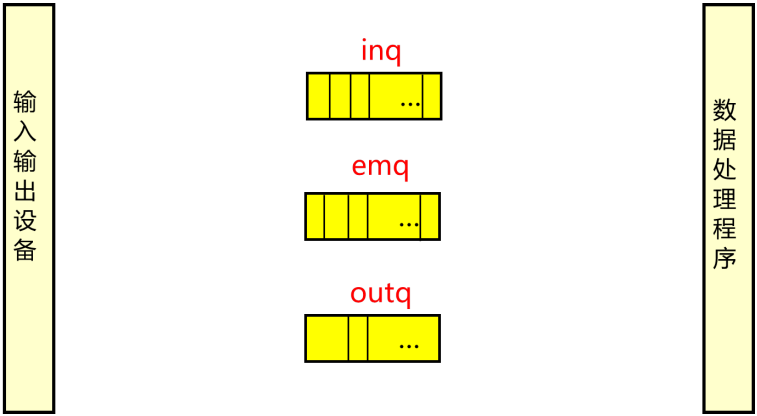
计算进程：

1. 从空缓冲队列的队首取一空缓冲区用作收容输出缓冲区：hout=getbuf(emq)
2. 计算：CPU 将数据输入其中并装满。
3. 挂到装满输出数据队列队尾：putbuf(outq, hout)

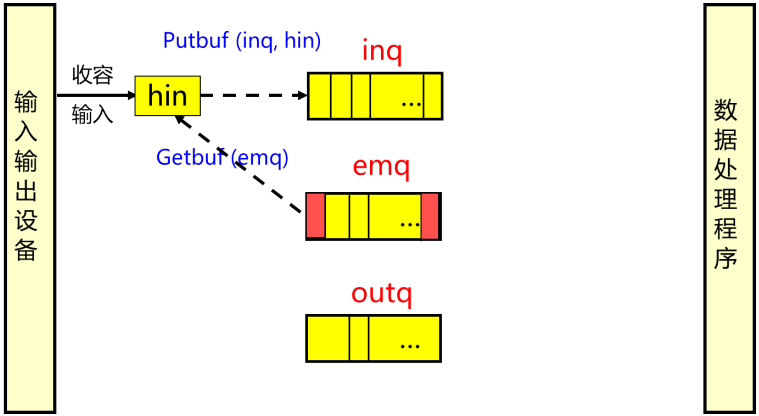
输出进程：

1. 从装满输出数据队列队首取一满缓冲区用作提取输出缓冲区：sout=getbuf(outq)
2. 输出数据：输出设备从中取出数据至用完。
3. 将空缓冲区挂到空缓冲队列队尾：putbuf(emq, sout)

缓冲池的工作方式

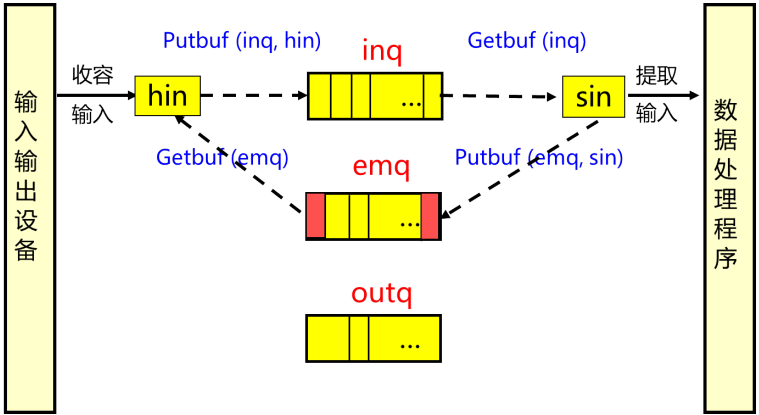


缓冲池的工作方式



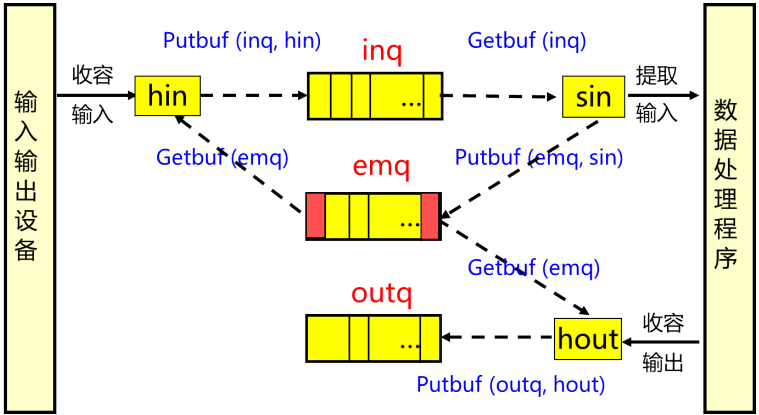
hin 收容输入缓冲区；sin 提取输入缓冲区；hout 收容输出缓冲区；sout 提取输出缓冲区

缓冲池的工作方式



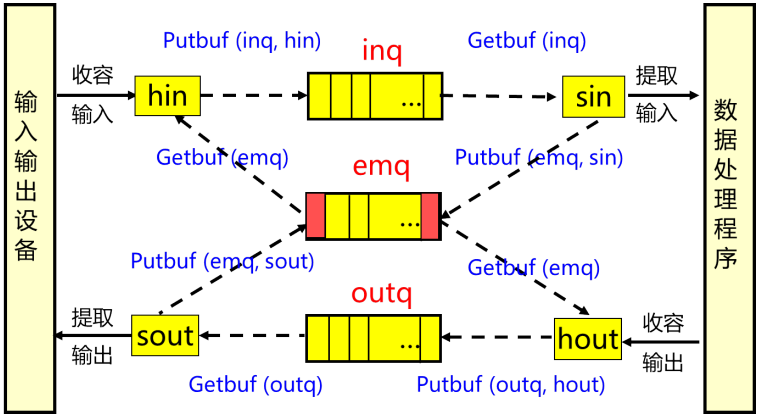
hin 收容输入缓冲区；sin 提取输入缓冲区；hout 收容输出缓冲区；sout 提取输出缓冲区

缓冲池的工作方式



hin 收容输入缓冲区；sin 提取输入缓冲区；hout 收容输出缓冲区；sout 提取输出缓冲区

缓冲池的工作方式



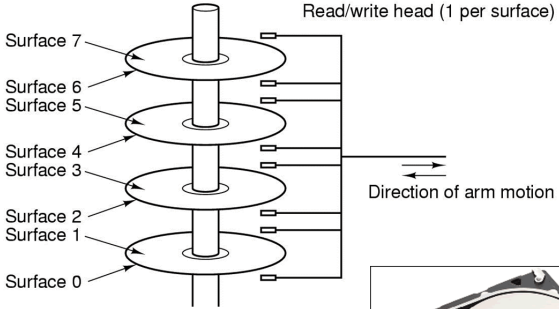
hin 收容输入缓冲区；sin 提取输入缓冲区；hout 收容输出缓冲区；sout 提取输出缓冲区

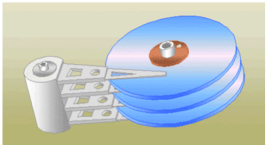
磁盘存储器的性能和调度

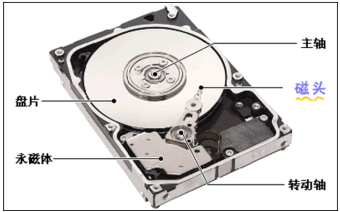
提高磁盘 I/O 速度的主要途径

1. 选择性能好的磁盘
2. 采用好的磁盘调度算法
3. 设置磁盘高速缓存（Disk Cache）
4. 其它方法（提前读、延迟写、虚拟盘）
5. 采用高度可靠、快速的磁盘系统：独立磁盘冗余阵列（RAID）

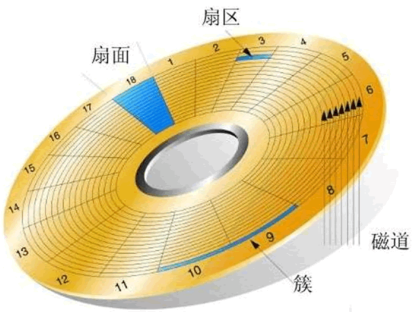
磁盘结构


Read/write head (1 per surface)
Surface 7
Surface 6
Surface 5
Surface 4
Surface 3
Surface 2
Surface 1
Surface 0
Direction of arm motion




主轴
磁头
盘片
永磁体
转动轴

磁盘结构



- 磁道（Track）：磁盘在格式化时被划分成许多同心圆，这些同心圆轨迹叫做磁道。
- 柱面（Cylinder）：不同盘片相同半径的磁道所组成的圆柱称为柱面。
- 扇区（Sector）：磁道被划分成一段段的圆弧，每段圆弧叫做一个扇区。一个扇区称为一个盘块(或数据块)。扇区是磁盘最小的物理存储单元。
- 簇（Cluster）：簇是操作系统使用的逻辑概念（分配的最小单位），相邻的扇区组合在一起，形成一个簇。

磁盘结构

- 磁盘的 0 磁道在最外圈，记录了分区表，主引导记录等一些重要信息。
- 早期的磁盘技术中，内外圈的扇区数是相同的。由于硬盘的主轴的工作方式是恒定角速度，而盘片最外圈的周长比最内圈的周长要长很多，磁头在外圈时，旋转的角度与内圈时一样，但走过的距离就长多了。如果内圈与外圈磁道的扇区数相同，必将造成存储空间的浪费。
- 现代磁盘普遍采用了 ZDR（Zone Data Recording）技术，即区域数据记录技术。可以根据不同的磁道长度来合理设定扇区数量，以达到充分利用磁盘的存储空间的目的。

磁盘性能简述

- 数据的组织
  - 磁盘结构：盘面、磁道、扇区
  - 磁盘物理块的地址：磁头号、柱面号、扇区号
  - 存储容量 = 磁头（盘面）数 × 磁道（柱面）数 × 每道扇区数 × 每扇区字节数

磁盘性能简述

- 数据的组织
  - 磁盘结构：盘面、磁道、扇区
  - 磁盘物理块的地址：磁头号、柱面号、扇区号
  - 存储容量 = 磁头（盘面）数 × 磁道（柱面）数 × 每道扇区数 × 每扇区字节数
- 假定一块硬盘磁头数为 4，柱面数为 2048，每个磁道有扇区 2048，每个扇区记录 1KB（字节），该硬盘储存容量为：

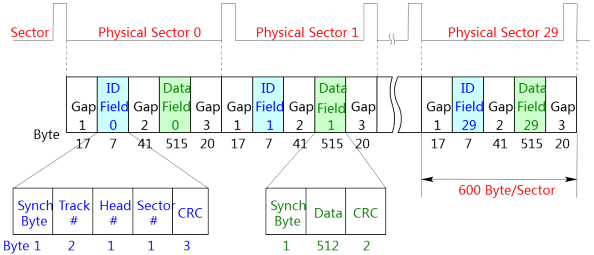
  $4\times2048\times2048\times1024/1024/1024/1024 = 16\ G$

  $4\times2048\times2048\times1024/1000/1000/1000 = 17.2\ G$（厂家）

磁盘性能简述

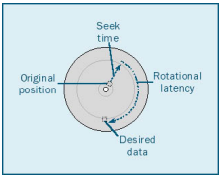
- 磁盘类型
  - 固定头磁盘：每条磁道上都有一个磁头（大容量硬盘）。
  - 移动头磁盘：每个盘面上配有一个磁头。
- 磁盘格式化
  - 低级格式化：物理级的格式化，主要是用于划分硬盘的磁柱面、建立扇区数和选择扇区间隔比。
  - 高级格式化：清除硬盘上的数据、生成引导区信息、初始化 FAT 表、标注逻辑坏道等。

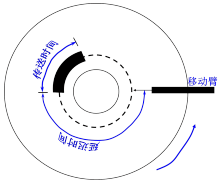
温彻斯特盘低级格式化（一条磁道）



- 每个扇区容量为 600 个字节，其中 512 个字节存放数据，其余用于存放控制信息。每扇区包括两个字段：
  - 标识符字段：其中一个 SYNCH 字节（同步字节）用于标识一个扇区，CRC 字段用于段校验（循环冗余码校验）。
  - 数据字段：其中可存放 512 个字节的数据。

磁盘访问时间





- 磁盘以恒定的速度旋转，只有当磁头位于指定的磁道和该磁道中指定的扇区开始处时才能够进行读或写。
- 把磁头定位到磁道所需时间，称为寻道时间。
- 磁头到达扇区开始位置的时间，称为旋转延迟。
- 一旦磁头被定位，磁头就对旋转通过它下面的扇区执行读操作或写操作，完成数据信息的传输，传输所花费的时间称为传送时间。

磁盘访问时间

- 寻道时间 Ts：将磁头从当前位置移到指定磁道所经历的时间$Ts=m\times n+s$。s 为启动磁臂的时间，n 为移动的磁道数，m 为常数。
- 旋转延迟时间 Tr ：指定扇区移动到磁头下面所经历的时间。设每秒 r 转，则$Tr = 1/2r$（均值）
- 传输时间 Tt：将扇区上的数据从磁盘读出或向磁盘写入数据所经历的时间$Tt = b/rN$。其中 b：读写字节数，N：每道上的字节数。
- 控制器时间 Tc

访问时间 Ta：$Ta=m\times n+s+1/2r+b/rN+Tc$

磁盘访问时间

例：假设磁盘空闲（没有排队延迟）。平均寻道时间是 9ms，传输速度是每秒 4MB，转速是 7200rpm，控制器的开销是 1ms。问读或写一个 512 字节的扇区的平均时间是多少？

磁盘访问时间

例：假设磁盘空闲（没有排队延迟）。平均寻道时间是 9ms，传输速度是每秒 4MB，转速是 7200rpm，控制器的开销是 1ms。问读或写一个 512 字节的扇区的平均时间是多少？

访问时间 = 寻道时间 + 旋转延迟 + 传输时间 + 控制器时间

1 秒 = 1000 毫秒

磁盘访问时间

例：假设磁盘空闲（没有排队延迟）。平均寻道时间是 9ms，传输速度是每秒 4MB，转速是 7200rpm，控制器的开销是 1ms。问读或写一个 512 字节的扇区的平均时间是多少？

访问时间 = 寻道时间 + 旋转延迟 + 传输时间 + 控制器时间

1 秒 = 1000 毫秒

解：访问时间 = 9ms+0.5/7200(rpm)+0.5(k)/4(MB/s)+1ms

= 9ms+0.5/(7200/60/1000)+0.5/ (4×1024/1000)+1ms

≈9+4.2+0.122+1=14.322ms

磁盘访问时间

- 在访问时间中，寻道时间 Ts 和旋转延迟时间 Tr 基本上与所读/写数据的多少无关，且通常占据了访问时间中的大头。
- 例如，假定寻道时间和旋转延迟时间平均为 20 ms，而磁盘的传输速率为 10 MB/s，如果要传输 10 KB 的数据，此时总的访问时间为 21 ms，可见传输时间 Tt 所占比例是非常小的。
- 只计算单个扇区的平均访问时间是不够的。
- 适当地集中数据传输，将有利于提高传输效率。

磁盘访问时间

例：考虑一个磁盘系统。平均寻道时间是 4ms，转速是 7500rpm，忽略控制器的开销。每个磁道有 500 个扇区，每个扇区 512 字节。假设读取一个包括 2500 个扇区的文件（文件大小为 1.28MB），试估计传送需要的总时间。

磁盘访问时间

例：考虑一个磁盘系统。平均寻道时间是 4ms，转速是 7500rpm，忽略控制器的开销。每个磁道有 500 个扇区，每个扇区 512 字节。假设读取一个包括 2500 个扇区的文件（文件大小为 1.28MB），试估计传送需要的总时间。

1. 假设文件尽可能紧凑地保存在磁盘上，占据 5 个相邻磁道的所有扇区（5×500）。读取第一条磁道的时间：

平均寻道时间	4ms	
旋转延迟时间	4ms	0.5/(7500/60/1000)
读 500 个扇区	8ms	1/(7500/60/1000)
合计	16ms	

磁盘访问时间

例：考虑一个磁盘系统。平均寻道时间是 4ms，转速是 7500rpm，忽略控制器的开销。每个磁道有 500 个扇区，每个扇区 512 字节。假设读取一个包括 2500 个扇区的文件（文件大小为 1.28MB），试估计传送需要的总时间。

读取第一条磁道的时间为 16ms。

读取整个文件的时间：

磁盘访问时间

例：考虑一个磁盘系统。平均寻道时间是 4ms，转速是 7500rpm，忽略控制器的开销。每个磁道有 500 个扇区，每个扇区 512 字节。假设读取一个包括 2500 个扇区的文件（文件大小为 1.28MB），试估计传送需要的总时间。

读取第一条磁道的时间为 16ms。

读取整个文件的时间：

后面的 4 个磁道不再需要寻道时间，每个磁道可以在 4+8=12ms 内读入。

$$\text{总时间} = 16 + (4 \times 12) = 64\text{ms} = 0.064\text{s}$$

磁盘访问时间

例：考虑一个磁盘系统。平均寻道时间是 4ms，转速是 7500rpm，忽略控制器的开销。每个磁道有 500 个扇区，每个扇区 512 字节。假设读取一个包括 2500 个扇区的文件（文件大小为 1.28MB），试估计传送需要的总时间。

2 假设文件随机地分布在磁盘上。读取整个文件的时间：

磁盘访问时间

例：考虑一个磁盘系统。平均寻道时间是 4ms，转速是 7500rpm，忽略控制器的开销。每个磁道有 500 个扇区，每个扇区 512 字节。假设读取一个包括 2500 个扇区的文件（文件大小为 1.28MB），试估计传送需要的总时间。

2 假设文件随机地分布在磁盘上。读取整个文件的时间：

平均寻道时间	4ms	
旋转延迟时间	4ms	0.5/(7500/60/1000)
读 1 个扇区	0.016ms	(1/(7500/60/1000))/500
合计	8.016ms	
总时间	2500 × 8.016m = 20040ms = 20.04s	

磁盘访问时间

- 信息在存储空间的排列方式会影响存取等待时间。

例：考虑 10 个记录 A，B……，J 被存于一个磁道上（每道 10 条记录），安排如右图示。假定旋转速度为 20 毫秒/转，处理程序读出每个记录后花 4 毫秒进行处理。请计算依次读出这 10 个记录总共需花多长时间？如何优化最合适？

物理块	逻辑记录
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J

磁盘访问时间

- 由于旋转速度为 20 毫秒/转，且每道 10 条记录，故读取每条记录的时间为 2 毫秒。
- 因为读出并处理记录 A 之后 (4ms) 将转到记录 D 的开始，所以为了读出 B，必须再转一周。
- 于是，处理 10 个记录的总时间为：10 毫秒 (移动到记录 A 的平均时间) + 2 毫秒 (读记录 A) +4 毫秒 (处理记录 A)+9×[16 毫秒 (访问下一记录) +2 毫秒 (读记录)+4 毫秒 (处理记录)] = 214 毫秒

磁盘访问时间

- 按右图方案优化各记录的位置。当读出记录 A 并处理结束后，恰巧转至记录 B 的位置，立即就可读出并处理。按照这一方案，处理 10 个记录的总时间为：
- 10 毫秒 (移动到记录 A 的平均时间) +10×[2 毫秒（读记录）+ 4 毫秒（处理记录）] = 70 毫秒
- 比原方案速度几乎快 3 倍，如果有很多记录需要处理，节省时间更多。

物理块	逻辑记录
1	A
2	H
3	E
4	B
5	I
6	F
7	C
8	J
9	G
10	D

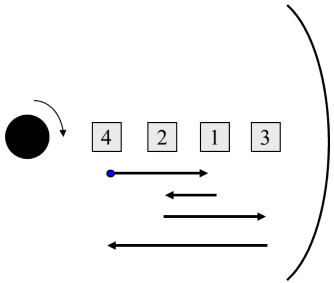
磁盘调度算法

磁盘调度算法

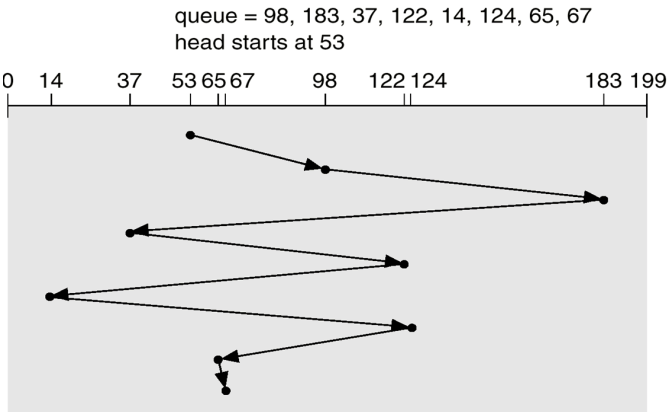
- 早期的磁盘调度算法
  - 先来先服务 FCFS
  - 最短寻道时间优先 SSTF
- 扫描算法
  - 扫描 (SCAN)/电梯 (LOOK) 算法
  - 循环扫描 (CSCAN) 算法
  - N-STEP-SCAN 调度算法和 FSCAN 调度算法

先来先服务调度算法

- 先来先服务算法（FCFS）
- 只考虑申请者申请的先后次序完成磁盘访问操作。
- 可能造成磁头臂来回反复移动，增加了等待时间，而且对机械结构不利。



先来先服务调度算法



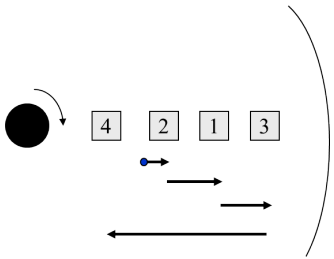
下磁道	移道数
98	45
183	85
37	146
122	85
14	108
124	110
65	59
67	2
总道数	640
平均	80

先来先服务调度算法

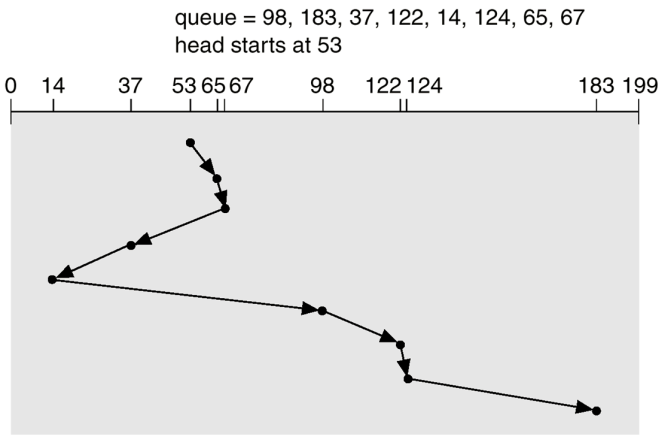
- 磁头臂来回反复移动，增长了等待时间，对机械结构不利。
- 磁头引臂横向移动的速度很慢，若按照请求发出的次序依次读/写各个磁盘块，则磁头引臂在内磁道和外磁道之间频繁地移动，造成较大的时间开销，影响效率。
- 这种算法通常可用于输入/输出负载较轻的系统。

最短寻道时间优先算法

- 最短寻道时间优先算法 (SSTF)
- 以申请者要求磁头移动距离的大小作为优先的因素。
- 靠近当前磁头位置的申请者迅速得到满足，防止了磁头的大幅度来回摆动。可能使一些申请者在较长时间内得不到服务的机会。



最短寻道时间优先算法



下磁道	移道数
65	12
67	2
37	30
14	23
98	84
122	24
124	2
183	59
总道数	236
平均	29.5

最短寻道时间优先算法

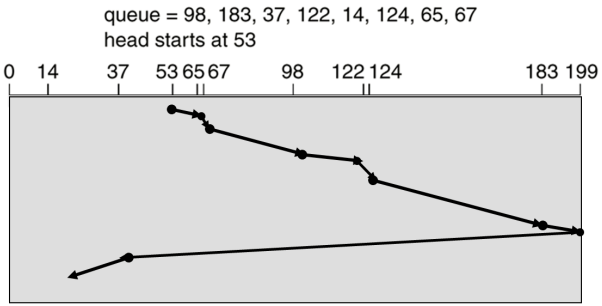
- 与先来先服务调度算法相比，磁头引臂的机械运动明显减少，所需时间大幅度降低。
- 存在的缺点：
  - 磁臂粘着：在最短寻道时间优先调度算法中，可能出现磁臂停留在某处的情况，即反复请求某一磁道，从而垄断了整个磁盘设备，这种现象称为磁臂粘着。
  - 磁道歧视：假设某一时刻外磁道请求不断，则内磁道请求可能长时间得不到满足，这种现象称为“磁道歧视”。因此 SSTF 算法缺乏公平性，存在饥饿和饿死的问题。

硬盘的 0 磁道在最外圈，C 盘在外圈

扫描算法（Scan）

- Scan 算法往复扫描各个柱面（磁道）并为途经柱面（磁道）的请求服务。
- 起始时磁头处于最外柱面，并向内柱面移动。在移动的过程中，如果途经的柱面有访问请求，则为其服务，如此一直移动到最内柱面，然后改变方向由内柱面向外柱面移动，并以相同的方式为途经的请求服务。
- Scan 算法每次都扫描到柱面的尽头，无论最内（最外）柱面处是否有访问请求。

扫描算法（Scan）

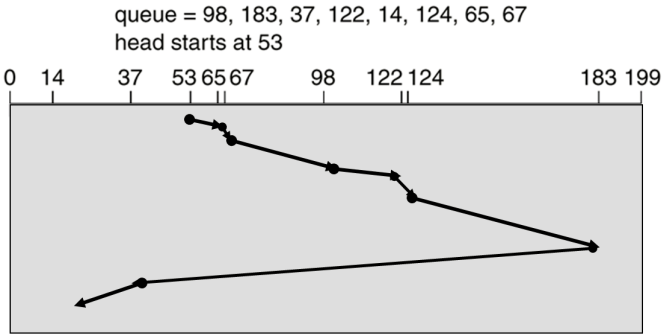


下磁道	移道数
65	12
67	2
98	31
122	24
124	2
183	59
199	16
37	162
14	23
总道数	331
平均	41.4

电梯算法（Look 算法）

- Look 算法也称电梯算法，因其基本思想与电梯的工作原理相似而得名。
- 无访问请求时，磁头引臂停止不动；当有访问请求时，起始时磁头由最外柱面向内柱面移动，并为途经的请求服务。一旦内柱面没有访问请求，则改变移动方向（如外柱面有请求）或停止移动（外柱面也无请求）。

电梯算法（Look 算法）



下磁道	移道数
65	12
67	2
98	31
122	24
124	2
183	59
37	146
14	23
总道数	299
平均	37.4

电梯算法（Look 算法）

- 本教材不区分 SCAN 算法和 LOOK 算法，请大家都按 LOOK 算法解决。

不扫描到头！

电梯算法（Look 算法）

- 本教材不区分 SCAN 算法和 LOOK 算法，请大家都按 LOOK 算法解决。

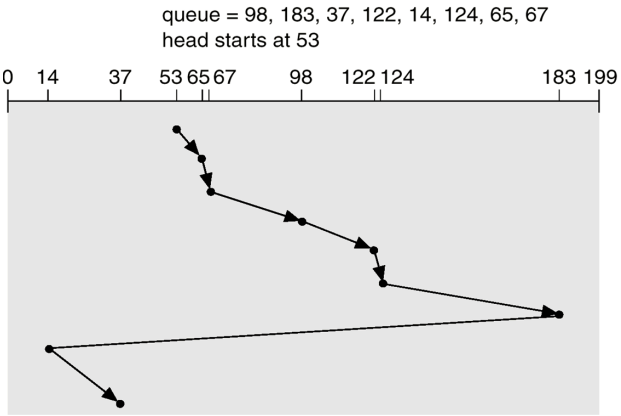
不扫描到头！

- 对于 SCAN(LOOK) 算法来说，位于不同磁道（柱面）的 I/O 请求与获取服务所需的等待时间是不同的。
- 对于靠近边缘的柱面，最坏情况的移动量为2N-1（N 为柱面数）；对于靠近中部的柱面，最坏情况为N-1。平均情况分别约为 N 和 N/2。

循环扫描算法（Circular SCAN, CSCAN）

- 循环扫描算法是为了消除边缘柱面与中部柱面等待时间差异而进行的改进。
- 磁头只在单方向移动过程中才为途经的请求服务，一旦达到边缘，则立即快速移动至另一边缘，在此移动过程中并不处理访问请求，然后重新开始新一轮扫描。
- 特点：消除了对两端磁道请求的不公平。

循环扫描算法（Circular SCAN, CSCAN）



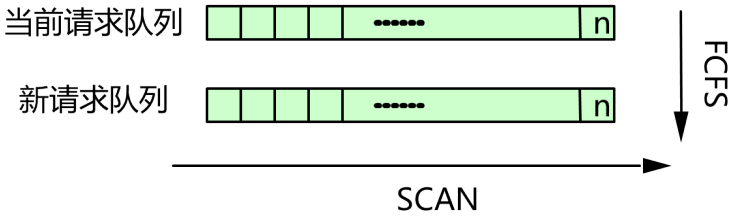
下磁道	移道数
65	12
67	2
98	31
122	24
124	2
183	59
14	169
37	23
总道数	322
平均	40.3

N-STEP-SCAN 调度算法

- N-STEP-SCAN 调度算法将磁盘请求队列分成若干个长度为 N 的子队列，磁盘调度将按 FCFS 算法依次处理这些子队列，而每一子队列按 SCAN 算法处理。
- N=1　　⇒ FCFS 算法
- N 很大　⇒ SCAN 算法

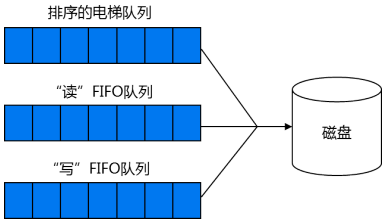
FSCAN 调度算法

- FSCAN 算法实质上是 N 步 SCAN 算法的简化。FSCAN 只将磁盘请求队列分成两个子队列。一个是由当前所有请求磁盘 I/O 的进程形成的队列，由磁盘调度按 SCAN 算法进行处理。
- 在扫描期间，将新出现的所有请求磁盘 I/O 的进程放入另一个等待处理的请求队列。这样所有的新请求都将被推迟到下一次扫描时处理。



Linux 时限调度算法

- 时限（deadline）调度算法：在传统的电梯算法中加入了请求超时的机制。
- 时限调度算法需要维护三个队列。一个是按照扇区或磁道排序的读写请求队列（电梯队列）；另外两个是按照过期时限（deadline）排序的读写请求队列。



时限调度算法

- 系统在处理每个 I/O 请求时，都被附加一个最后执行期限（deadline），也就是为其规定在相应 FIFO 队列里等待的时限。
- 这个时限是可调整的，比如：对于读请求默认时限值是 0.5s，写请求默认时限值是 5s。
- 在一个新 I/O 请求到达时，按所请求块号的大小顺序，在“排序的电梯队列”里排队。
- 此外，如果是读请求，就按到达的时间顺序，排在“读”FIFO 队列末尾；如果是写请求，就按到达的时间顺序，排在“写”FIFO 队列的末尾。

时限调度算法

- 调度程序对排序的电梯队列里的请求进行调度服务。在一个 I/O 请求处理完成时，就将其从排序电梯队列和相应的 FIFO 队列里移走。
- 如果扫描到了电梯的末尾，按传统的电梯算法需要返回到电梯首部。但时限调度算法是返回到等待时间最久的那个 I/O 请求（读写请求队列队首的请求），从那个 I/O 请求开始，沿磁道递增方向继续扫描。
- 上面的过程中，如果有 I/O 请求超时，调度程序就立即对它进行调度服务，处理完成后将其从排序电梯队列和相应的 FIFO 队列里移走。然后从这个 I/O 请求开始，沿磁道递增方向继续扫描。

UNIX 系统中的设备管理

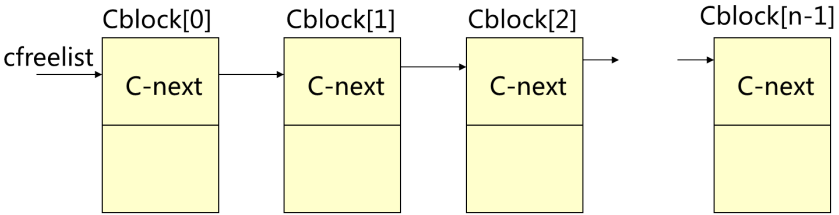
- Unix 系统中，设备被作为一种特殊的文件，由文件系统统一进行管理。
- Unix 系统把设备分为两类
  - 块设备：用于存储信息，如磁盘。
  - 字符设备：用于输入输出程序和数据，如打印机。
- 设备缓冲管理
  - 字符设备缓冲管理
  - 块设备缓冲管理
- 设备处理程序
  - 核心与驱动程序的接口（设备开关表）
  - 磁盘驱动程序、磁盘读写程序（提前读、延迟写）

字符设备缓冲管理

- 字符缓冲区的分配与回收
  - Getcf 过程：从空闲缓冲区链首取一缓冲区，收容设备的 I/O 数据，把指向该缓冲区的指针 bp 返回给调用者。
  - Putcf 过程：将空闲缓冲区归还到空闲缓冲区链中。若此时已有因申请空缓冲区而阻塞的进程，则将它唤醒。
  - 对空闲缓冲区队列的访问应该互斥地进行。

- 字符设备的 Putcf 过程和块设备的 brelse 过程在归还缓冲区时，可以链入空闲链表的队尾，也可以链入队首。
- 如果所释放的缓冲区中的数据被修改过，为使以后某进程需要它时能直接从缓冲区中读出而不必启动磁盘的 I/O 操作，可将该缓冲区链入空闲链表的队尾。
- 如果缓冲区中数据没被修改过，将它链入空闲队列的队首。

字符设备空闲缓冲区队列



Cblock[i]

第一个字符的位置
最后一个字符的位置
指向下个缓冲区的指针
余下64个字符

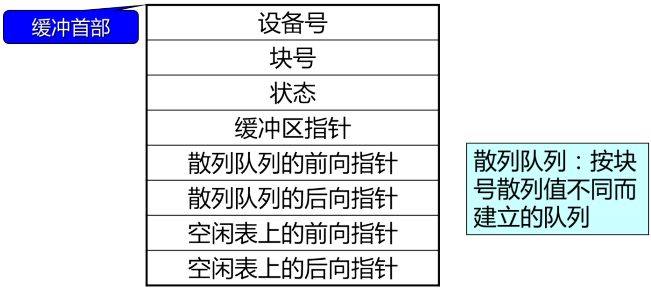
每个缓冲区的大小为70个字节

块设备缓冲区的分配与回收

- 块缓冲区的分配
  - Getblk() 过程：从空闲缓冲区队列中获得空闲缓冲区。
  - Getblk(dev,blkno) 过程：为指定设备 dev 和盘块号为 blkno 的盘块申请一个缓冲区。
- 块缓冲区的回收：brelse 过程
  - 唤醒等待空闲缓冲区的所有进程。
  - 将空闲块缓冲区放入空闲队列。
  - 在缓冲区首置空闲标志，空闲块缓冲区数加 1。
- 块设备缓冲队列的结构
  - 盘块缓冲区及其首部
  - 盘块缓冲池的构成（散列队列、空闲队列）

盘块缓冲区及其首部

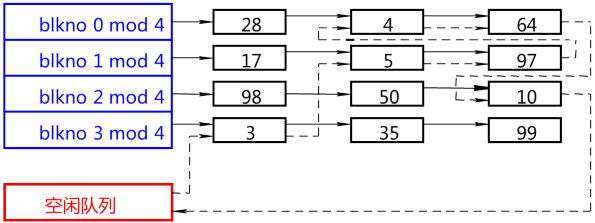
- 缓冲区：存放数据本身。
- 缓冲控制块（缓冲首部）：存放缓冲区的管理信息。
- 缓冲首部与缓冲区一一对应，但两者在物理上并不相连。



盘块缓冲池的构成

- 空闲队列（链表）
  - 空闲缓冲区组成一个双向链接的空闲链表。
  - getblk 过程：从空闲链的链首摘下一个缓冲区。
  - brelse 过程：释放缓冲区，将缓冲区挂在空闲链的末尾。
- 散列队列
  - 缓冲区按块号计算散列值，组织成多个散列队列。
  - 任一缓冲区在某个散列队列中，而空闲缓冲区在空闲队列链中。
  - 因此一个空闲缓冲区可同时链入两个队列。

盘块缓冲池的构成



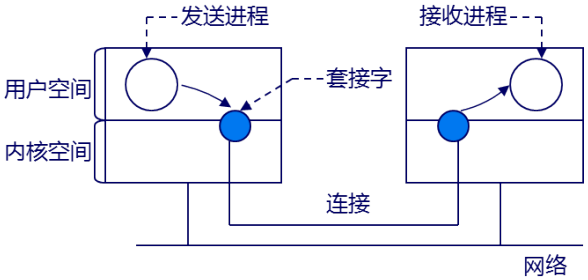
- 散列队列与空闲队列
  - 任一缓冲区一定在某个散列队列中。
  - 一个空闲缓冲区可同时链入两个队列。
- 对空缓冲区的查找方法
  - 要求获得任一空闲缓冲区 -----空闲队列（链首摘取）
  - 要求获得一个特定缓冲区 -----散列队列（搜索）

网络设备

- Linux/Unix 系统中，除了块设备、字符设备，还有网络设备。
- 网络设备是一种经网络接口与主机交换数据的设备。在内核网络子系统的驱动下，网络接口完成对数据包的发送和接收。
- 由于数据传输的特殊性，无法把网络设备纳入到文件系统进行统一管理。
- 用户不能直接把数据交换到网络设备上，而需通过内核网络子系统建立起的连接实现间接通信。

网络设备

- 通过内核网络子系统创建“套接字”，建立起发送进程和接收进程两者间的连接，实现它们之间的通信。
- 创建套接字，返回文件描述符。
- 建立连接、读取数据、写入数据和释放连接。

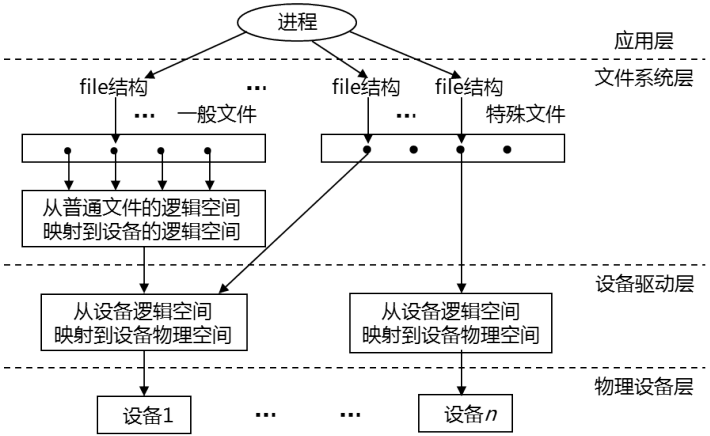


设备驱动的分层结构

- 位于应用层的用户进程，通过打开文件的文件描述符 fd，与其 file 结构相联系。这些 file 结构有的与一般文件对应，有的与特殊文件（设备）对应。
- 在文件系统层，按照文件系统的规则对它们进行分别处理。
  1. 一般文件先从普通文件的逻辑空间映射到相应设备逻辑空间，然后进入设备驱动层，完成从设备逻辑空间到设备物理空间的映射，驱动物理设备层设备执行 I/O 工作。
  2. 特殊文件直接进入设备驱动层，完成从设备逻辑空间到设备物理空间的映射，驱动物理设备层设备执行 I/O 工作。

设备驱动的分层结构

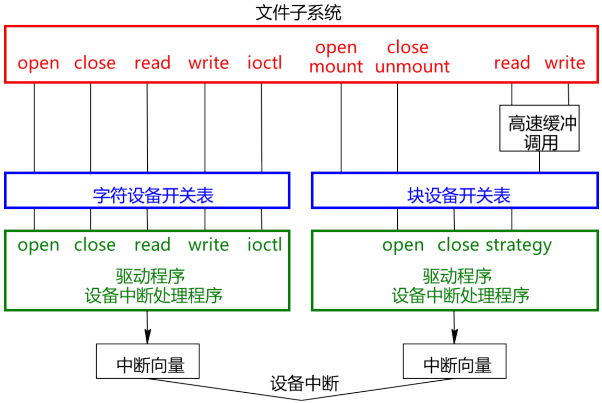
- I/O 是通过设备驱动程序实现的，设备驱动程序是系统内核的一部分。



内核与驱动程序的接口

- 设备驱动程序是内核的一部分，系统为内核与驱动程序提供了一个标准的接口：设备开关表
- 设备开关表
  - 一个驱动程序可能包括多个函数，用于执行不同的操作，如打开、关闭、读或写等。
  - 系统为每类设备提供了一个设备开关，其中含有驱动程序各函数的入口地址。
  - 多种类型的设备开关构成一张设备开关表。表中的每一行是一类设备驱动程序的各函数的入口地址；表中的每一列是执行相同操作的不同设备的函数。

内核与驱动程序的接口



设备开关表是内核与驱动程序间的接口

系统调用通过设备开关表转向相应驱动程序的函数

块设备开关表

表中的每一行是一类设备驱动程序的各函数的入口地址

表中的每一列是执行相同操作的不同设备的函数

字符设备开关表

字符设备开关表					
表项	open	close	read	write	ioctl
0	conopen	conclose	conread	conwrite	conioctl
1	dzbopen	dzbclose	dzbread	dzbwrite	dzbioctl
…	…	…	…		

预置、读取设备参数的函数的入口地址。

磁盘读写程序

- 读方式
  - 一般读方式 bread。
  - 提前读方式 breada：在读当前盘块的同时，提前将下一个盘块中的信息读入缓冲区。
- 写方式
  - 一般写方式 bwrite。
  - 异步写方式 bawrite：进程无需等待写操作完成便返回。
  - 延迟写方式 bdwrite：并不真正启动磁盘，而只是在缓冲首部设置延迟写标志便释放该缓冲区，并将之链入空闲链表的队尾。当有进程申请到该缓冲区时才将其内容写入磁盘。若再有进程需要访问时，可直接从空闲链表中访问，而不必从磁盘读入。减少了不必要的磁盘 I/O。

课堂练习（判断正误）

1 操作系统缓冲技术中的缓冲池属于软件机制。

课堂练习（判断正误）

1. 操作系统缓冲技术中的缓冲池属于软件机制。 ✓
2. 由于独占设备在一段时间内只允许一个进程使用，因此，多个并发进程无法访问这类设备。

课堂练习（判断正误）

1. 操作系统缓冲技术中的缓冲池属于软件机制。 ✓
2. 由于独占设备在一段时间内只允许一个进程使用，因此，多个并发进程无法访问这类设备。 ×
3. 低速设备一般被设置为独占设备。

课堂练习（判断正误）

1. 操作系统缓冲技术中的缓冲池属于软件机制。 ✓
2. 由于独占设备在一段时间内只允许一个进程使用，因此，多个并发进程无法访问这类设备。 ×
3. 低速设备一般被设置为独占设备。 ✓
4. I/O 通道控制方式中不需要任何 CPU 干预。

课堂练习（判断正误）

1. 操作系统缓冲技术中的缓冲池属于软件机制。 ✓
2. 由于独占设备在一段时间内只允许一个进程使用，因此，多个并发进程无法访问这类设备。 ×
3. 低速设备一般被设置为独占设备。 ✓
4. I/O 通道控制方式中不需要任何 CPU 干预。 ×
5. 设备分配算法中一般不采用时间片轮转算法。

课堂练习（判断正误）

1. 操作系统缓冲技术中的缓冲池属于软件机制。 ✓
2. 由于独占设备在一段时间内只允许一个进程使用，因此，多个并发进程无法访问这类设备。 ×
3. 低速设备一般被设置为独占设备。 ✓
4. I/O 通道控制方式中不需要任何 CPU 干预。 ×
5. 设备分配算法中一般不采用时间片轮转算法。 ✓
6. 操作系统中应用的缓冲技术，多数通过使用外存来实现。

课堂练习（判断正误）

1. 操作系统缓冲技术中的缓冲池属于软件机制。 ✓
2. 由于独占设备在一段时间内只允许一个进程使用，因此，多个并发进程无法访问这类设备。 ×
3. 低速设备一般被设置为独占设备。 ✓
4. I/O 通道控制方式中不需要任何 CPU 干预。 ×
5. 设备分配算法中一般不采用时间片轮转算法。 ✓
6. 操作系统中应用的缓冲技术，多数通过使用外存来实现。 ×
7. 数组选择通道和数组多路通道可以支持多个通道程序并发执行，而字节多路通道不支持多个通道程序并发执行。

课堂练习（判断正误）

1. 操作系统缓冲技术中的缓冲池属于软件机制。 ✓
2. 由于独占设备在一段时间内只允许一个进程使用，因此，多个并发进程无法访问这类设备。 ×
3. 低速设备一般被设置为独占设备。 ✓
4. I/O 通道控制方式中不需要任何 CPU 干预。 ×
5. 设备分配算法中一般不采用时间片轮转算法。 ✓
6. 操作系统中应用的缓冲技术，多数通过使用外存来实现。 ×
7. 数组选择通道和数组多路通道可以支持多个通道程序并发执行，而字节多路通道不支持多个通道程序并发执行。 ×

课堂练习

1. 需要 CPU 干预最少的 I/O 控制方式是（　　）。
   - A、程序 I/O 方式
   - B、中断驱动 I/O 控制方式
   - C、直接存储器访问 DMA 控制方式
   - D、I/O 通道控制方式
2. 下面的选项中不是设备驱动程序功能的是（　　）。
   - A、检查用户 I/O 请求的合法性
   - B、及时响应由控制器或通道发来的中断请求
   - C、控制 I/O 设备的 I/O 操作
   - D、了解 I/O 设备的状态，传送有关参数，设置设备的工作方式

课堂练习

1. 需要 CPU 干预最少的 I/O 控制方式是（　　）。
   - A、程序 I/O 方式
   - B、中断驱动 I/O 控制方式
   - C、直接存储器访问 DMA 控制方式
   - D、I/O 通道控制方式
2. 下面的选项中不是设备驱动程序功能的是（　　）。
   - A、检查用户 I/O 请求的合法性
   - B、及时响应由控制器或通道发来的中断请求
   - C、控制 I/O 设备的 I/O 操作
   - D、了解 I/O 设备的状态，传送有关参数，设置设备的工作方式

1D　2C

课堂练习

3 在调试程序时，可以先把所有输出送屏幕显示而不必正式输出到打印设备，其运用了（　　）。

- A、I/O 重定向技术　　B、共享技术
- C、SPOOLing 技术　　D、缓冲技术

4 下列关于通道、设备、设备控制器三者之间的关系叙述中正确的是（　　）。

- A、设备控制器和通道可以分别控制设备
- B、设备控制器控制通道和设备一起工作
- C、通道控制设备控制器，设备控制器控制设备
- D、设备控制器控制通道，通道控制设备

课堂练习

3 在调试程序时，可以先把所有输出送屏幕显示而不必正式输出到打印设备，其运用了（　　）。

- A、I/O 重定向技术　　B、共享技术
- C、SPOOLing 技术　　D、缓冲技术

4 下列关于通道、设备、设备控制器三者之间的关系叙述中正确的是（　　）。

- A、设备控制器和通道可以分别控制设备
- B、设备控制器控制通道和设备一起工作
- C、通道控制设备控制器，设备控制器控制设备
- D、设备控制器控制通道，通道控制设备

3A　4C

课堂练习

5 下列哪一条不是磁盘设备的特点（　　）。

- A、传输速率较高，以数据块为传输单位
- B、一段时间内只允许一个用户（进程）访问
- C、I/O 控制方式常采用 DMA 方式或通道方式
- D、可以寻址，随机地读/写任意数据块

6 在假脱机 I/O 技术中，对打印机的操作实际上是用对磁盘存储器的访问。用以替代打印机的部分通常被称作（　　）。

- A、共享设备　　B、独占设备
- C、虚拟设备　　D、物理设备

课堂练习

5 下列哪一条不是磁盘设备的特点（　　）。

- A、传输速率较高，以数据块为传输单位
- B、一段时间内只允许一个用户（进程）访问
- C、I/O 控制方式常采用 DMA 方式或通道方式
- D、可以寻址，随机地读/写任意数据块

6 在假脱机 I/O 技术中，对打印机的操作实际上是用对磁盘存储器的访问。用以替代打印机的部分通常被称作（　　）。

- A、共享设备　　B、独占设备
- C、虚拟设备　　D、物理设备

5B　6C

课堂练习

7 对于速率为 9.6Kb/s 的数据通信来说，如果说设置一个具有 8 位的缓冲寄存器，则 CPU 中断时间和响应时间大约分别为（　　）。

- A、0.8ms，0.8ms　　B、8ms，1ms
- C、0.8ms，0.1ms　　D、0.1ms，0.1ms

课堂练习

7 对于速率为 9.6Kb/s 的数据通信来说，如果说设置一个具有 8 位的缓冲寄存器，则 CPU 中断时间和响应时间大约分别为（　　）。

- A、0.8ms，0.8ms　　B、8ms，1ms
- C、0.8ms，0.1ms　　D、0.1ms，0.1ms

7C

CPU 的中断时间为：$8/(9.6\times1024/1000) \approx 0.8$ ms

CPU 的响应时间为：$1/(9.6\times1024/1000) \approx 0.1$ ms

若再增设一个 8 位的缓冲寄存器，响应时间也可放宽到 0.8ms。

课堂练习

8 某操作系统中，采用中断驱动 I/O 控制方式，设中断时 CPU 用 1ms 来处理中断请求，其它时间 CPU 完全用来计算，若系统时钟中断频率为 100HZ，则 CPU 的利用率为（　　）。

- A、60%　　B、70%
- C、80%　　D、90%

课堂练习

8 某操作系统中，采用中断驱动 I/O 控制方式，设中断时 CPU 用 1ms 来处理中断请求，其它时间 CPU 完全用来计算，若系统时钟中断频率为 100HZ，则 CPU 的利用率为（　　）。

- A、60%　　B、70%
- C、80%　　D、90%

8D

两次中断的时间间隔为：1/100 = 0.01s = 10 ms

CPU 的利用率为：1 - 1ms/10ms = 90%

课堂练习

9 下面是一段简单的通道程序，则四个选项中叙述不正确的是（　　）。

操作	P	R	计数	内存地址
WRITE	0	1	90	743
WRITE	0	1	100	250
READ	0	1	230	1200
WRITE	0	0	120	500
WRITE	0	1	120	350
READ	1	1	70	2000

- A、该段通道程序包括六条、两类通道指令
- B、这些指令涉及的数据内存地址有相邻接的地方
- C、该段通道程序共处理了 5 条记录
- D、单记录最大为 230 个字节

课堂练习

9 下面是一段简单的通道程序，则四个选项中叙述不正确的是（　　）。

操作	P	R	计数	内存地址
WRITE	0	1	90	743
WRITE	0	1	100	250
READ	0	1	230	1200
WRITE	0	0	120	500
WRITE	0	1	120	350
READ	1	1	70	2000

- A、该段通道程序包括六条、两类通道指令
- B、这些指令涉及的数据内存地址有相邻接的地方
- C、该段通道程序共处理了 5 条记录
- D、单记录最大为 230 个字节

本章作业

谢 谢

School of Computer & Information Engineering
Henan University
Kaifeng, Henan Province
475001
China