第八章 设备管理

BACK

8.1 I/O系统的组成

8.1.1 I/O系统中各种模块之间的层次结构

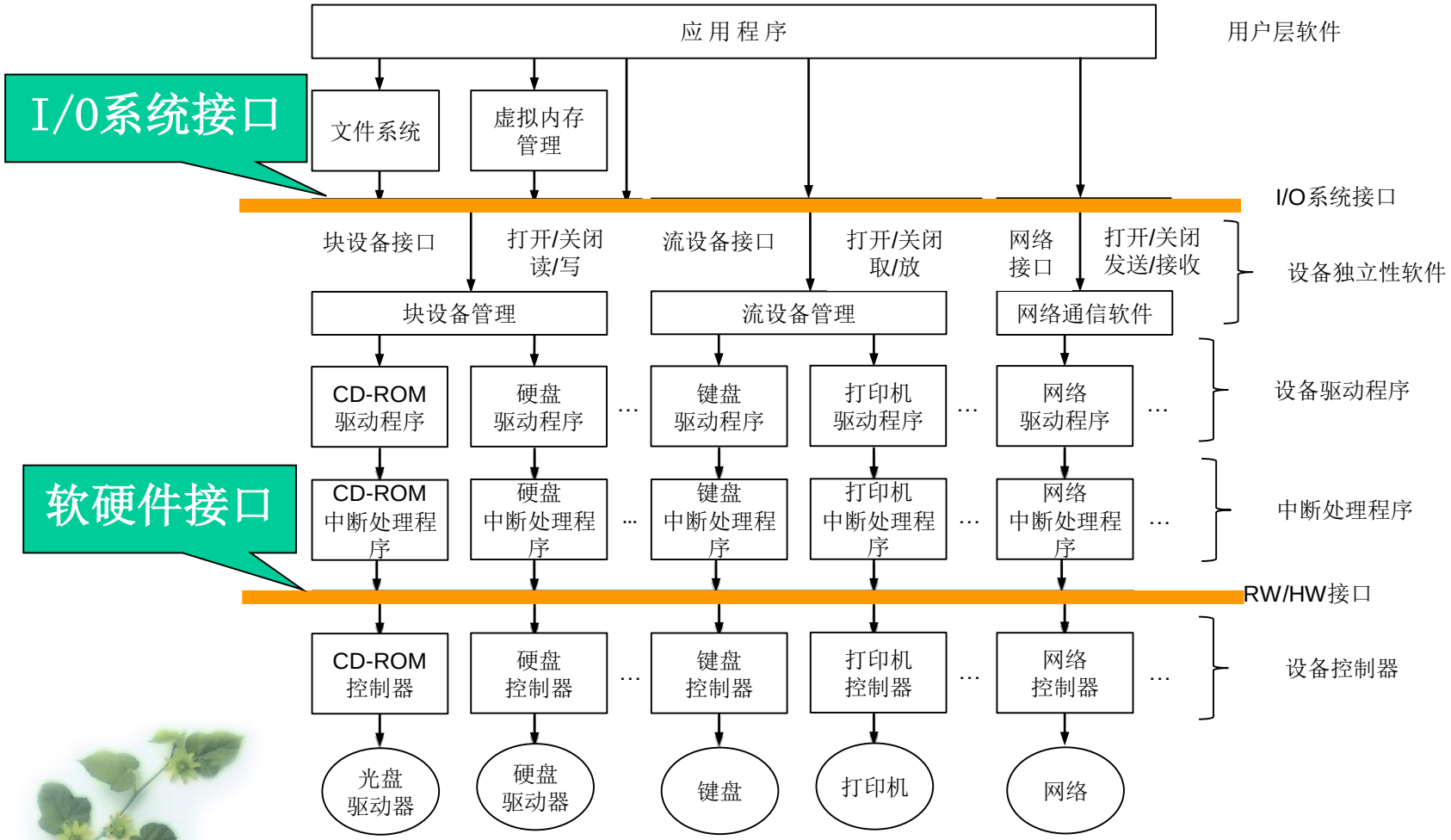


图8-1 I/O系统中各种模块之间的层次结构

8.1.2 I/O设备和设备控制器

1. I/O设备的类型

- 按使用特性分类
  - ①存储设备：也称外存、辅存。
  - ②I/O设备：分为输入设备、输出设备和交互式设备。
- 按传输速率分类
  - ①低速设备：其传输速率仅为每秒钟几个字节至数百个字节的一类设备，如键盘、鼠标器。
  - ②中速设备：传输速率在每秒钟数千个字节至数十万个字节的一类设备，如行式打印机、激光打印机等。
  - ③高速设备：传输速率在数十万字节至千兆字节的一类设备，如磁带机、磁盘机、光盘机等。

8.1.2 I/O设备和设备控制器

1. I/O设备的类型（续）

- 按信息交换的单位分类
  - ①块设备：存储以“块”为单位存储数据的设备，如磁盘，光盘等。
  - ②字符设备：指在I/O传输过程中以字符为单位进行传输的设备，例如键盘，打印机等。
- 按设备的共享属性分类
  - ①独占设备：并发进程要互斥地访问这类设备，系统一旦将该设备分配给某进程，便由该进程独占，直到用完释放。如：打印机。
  - ②共享设备：一段时间内允许多个进程同时访问。如：磁盘。
  - ③虚拟设备：通过虚拟技术将一台独占设备变换为若干逻辑设备。经过虚拟技术处理后的设备，成为虚拟设备。

2. 设备控制器

（1）设备控制器的基本功能

1) 接收和识别来自CPU的各种指令

2) 数据传输：数据寄存器

3) 记录设备的状态：状态寄存器

4) 地址识别：地址译码器

5) 数据缓冲：数据缓冲器

6) 差错控制：差错检测码

（2）设备控制器的组成

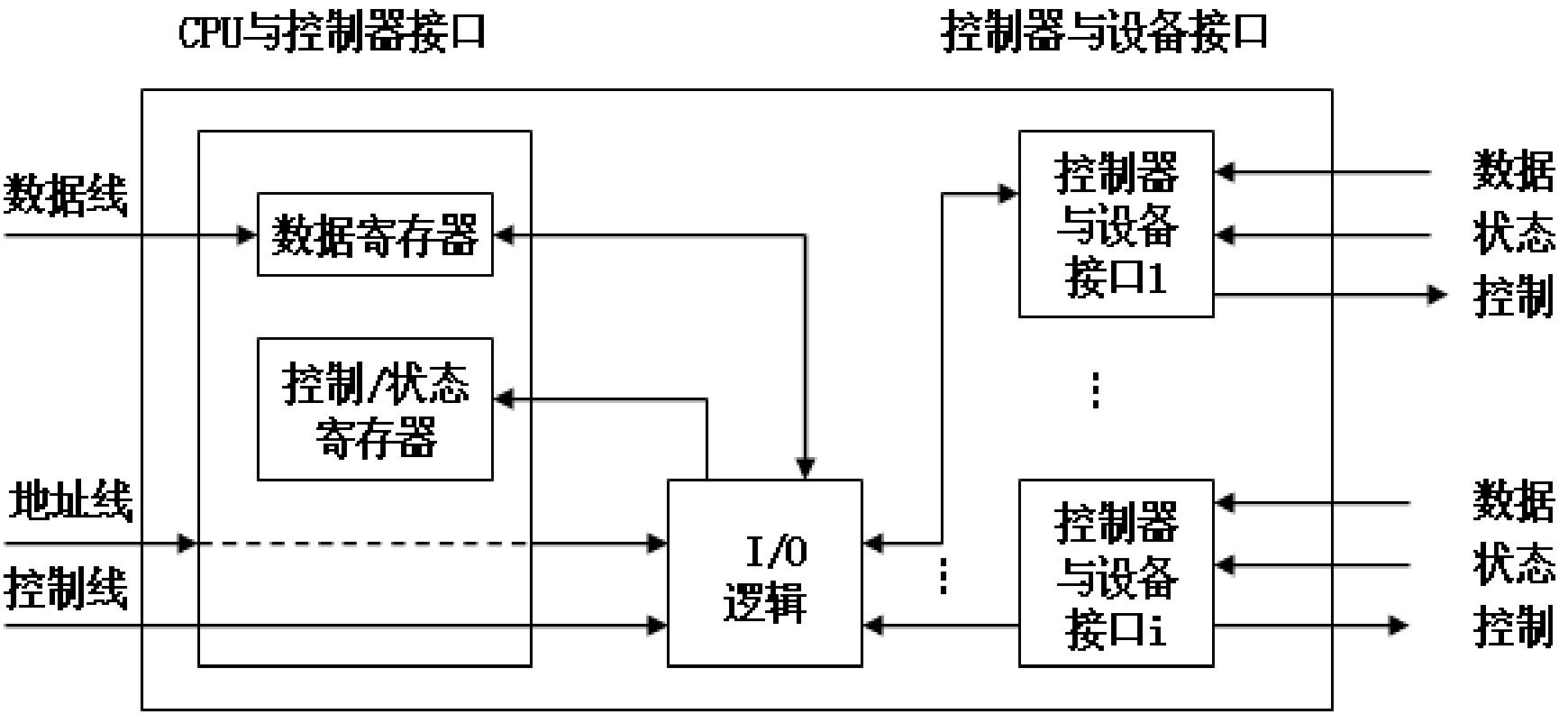


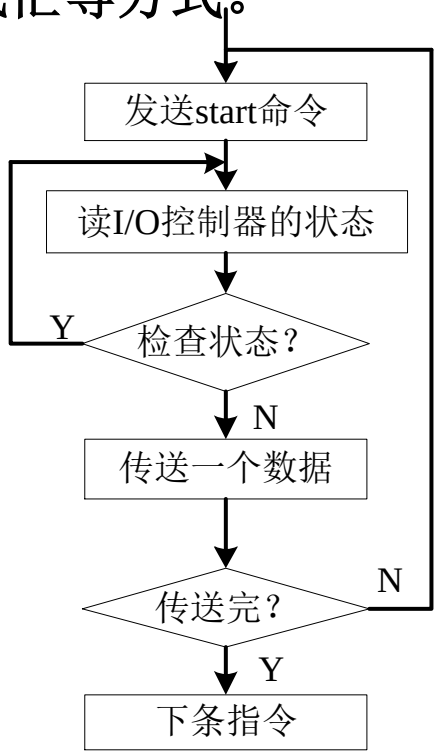
图 8-2 设备控制器的组成

8.2　I/O控制方式

- 按照I/O控制器功能的强弱以及和CPU之间联系方式的不同，可以把I/O设备的控制方式分为四类：直接程序控制方式、中断控制方式、直接存储器访问方式和通道方式。

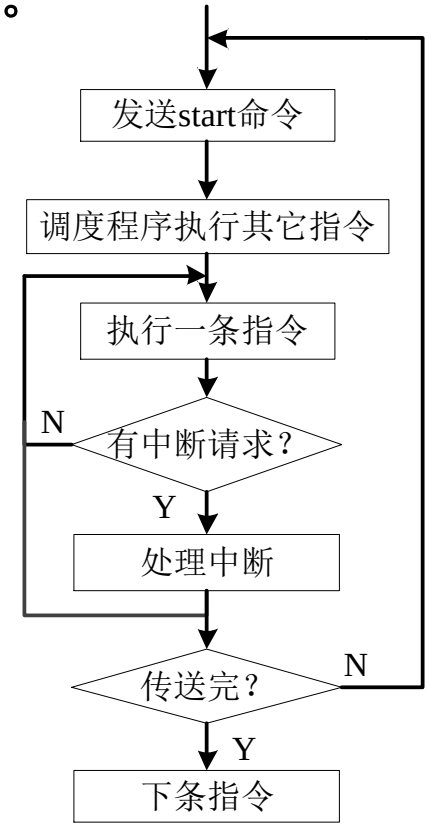
8.2.1 直接程序控制方式

由于CPU的高速性和I/O设备的低速性，使CPU绝大部分时间都处于等待I/O设备完成数据I/O的循环测试中， 造成CPU的极大浪费。在该方式中，CPU之所以不断地测试I/O设备的状态，是因为在CPU中无中断机构，使I/O设备无法向CPU报告它已完成了一个字符的输入操作。 也称为轮询方式或忙等方式。



8.2.2 中断控制方式

在I/O设备输入每个数据的过程中，无须CPU干预，因而可使CPU与I/O设备并行工作。仅当输完一个数据时，才需CPU花费极短的时间去做些中断处理。这样可使CPU和I/O设备都处于忙碌状态，可提高系统的资源利用率及吞吐量。



8.2.3 直接存储器访问（DMA）方式

1. DMA(Direct Memory Access)控制方式的引入

该方式特点：①数据传输的基本单位是数据块；②所传数据是从设备直接送入内存的，或者相反；③仅在传送一个或多个数据块开始和结束时，才需CPU干预，整块数据的传送在控制器的控制下完成。DMA方式进一步提高了CPU与I/O设备的并行操作程度。

2. DMA控制器的组成

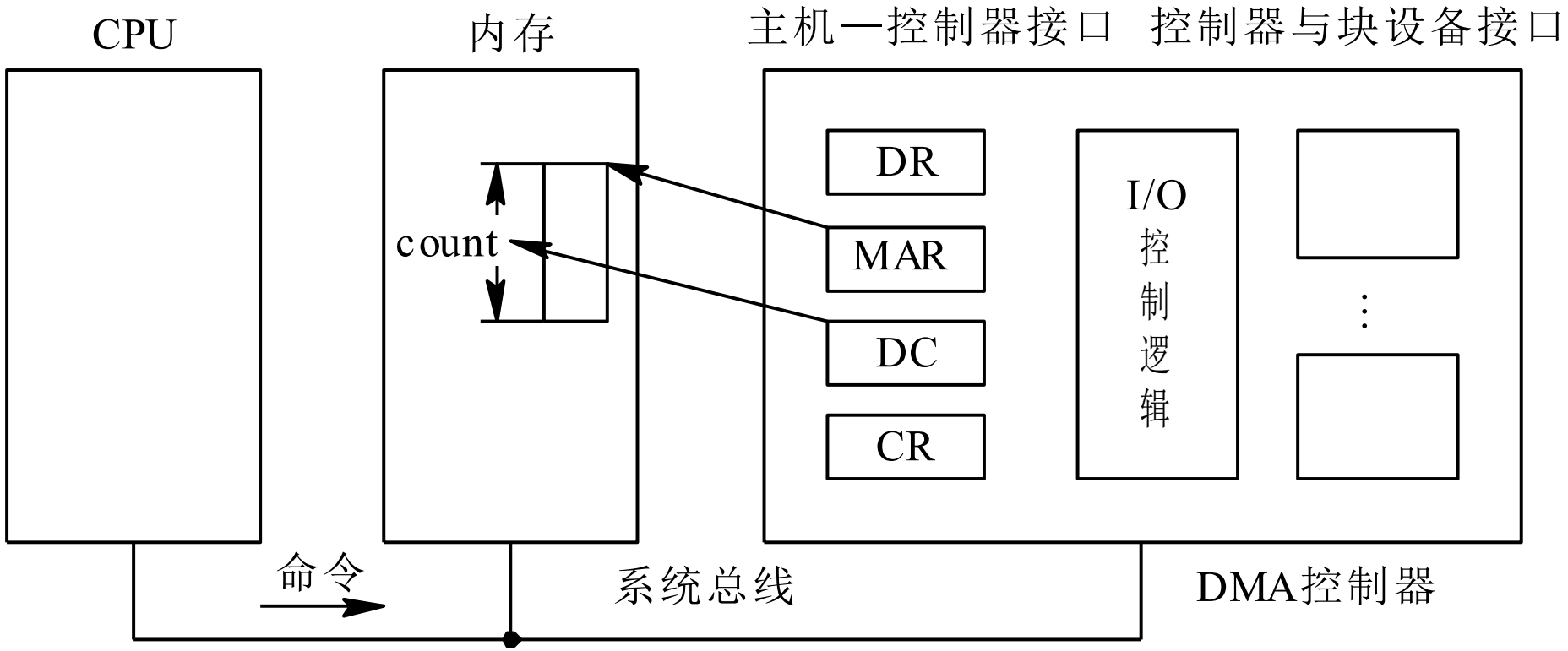


图 8-4 DMA控制器的组成

3. DMA工作过程

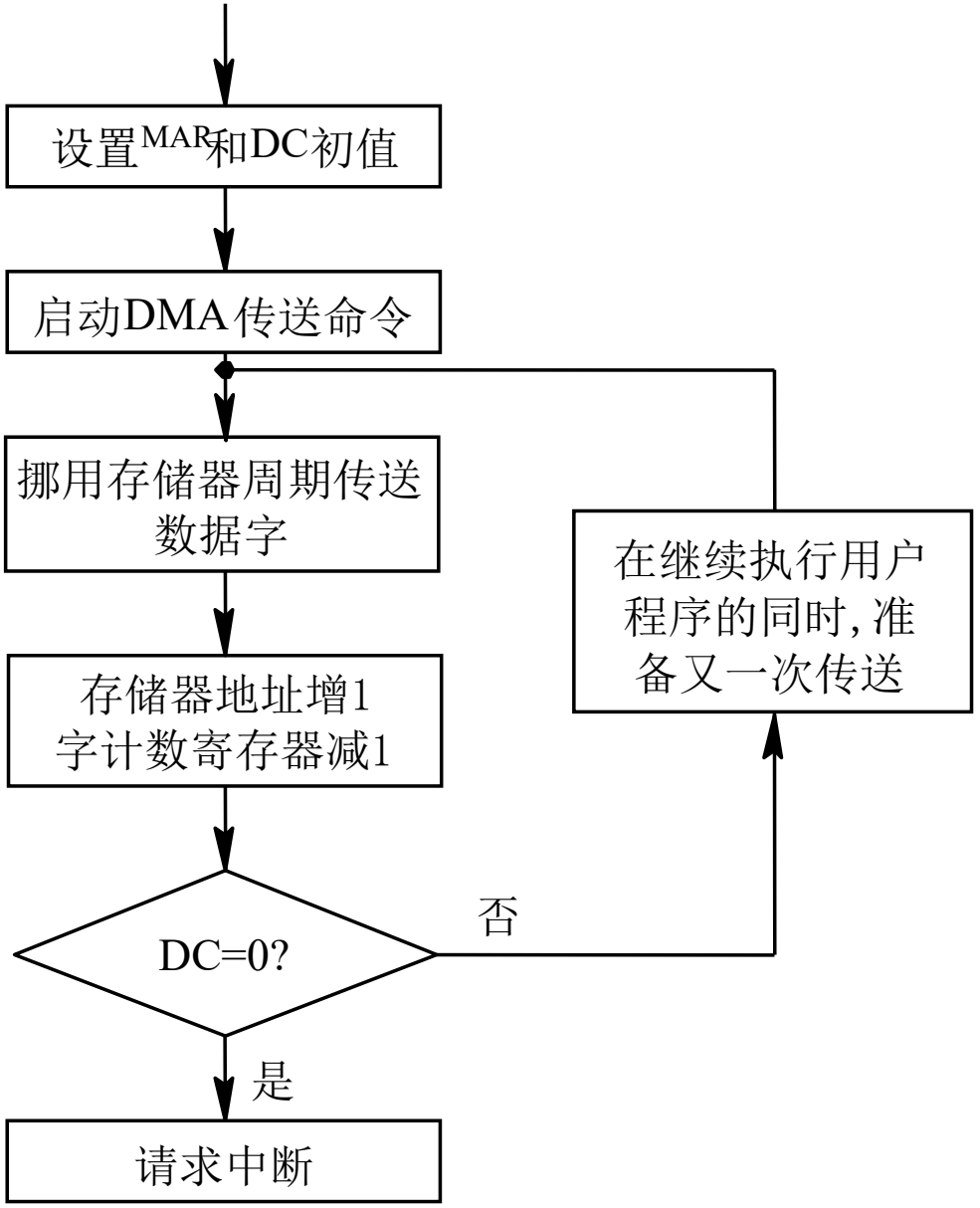


图 DMA方式的工作流程

8.2.4 I/O通道控制方式

1) I/O通道控制方式的引入

可进一步减少CPU的干预，把对一个连续数据块的读(或写)为单位的干预，减少为对一组数据块的读(或写)为单位的干预。可实现CPU、通道和I/O设备三者的并行操作，更有效地提高整个系统的资源利用率。

2) 通道程序 （由一系列的通道指令所构成）

(1) 操作码。

(2) 内存地址。

(3) 计数。

(4) 通道程序结束位P。

(5) 记录结束标志R。

操作	P	R	计数	内存地址
WRITE	0	0	80	813
WRITE	0	0	140	1034
WRITE	0	1	60	5830
WRITE	0	1	300	2000
WRITE	0	0	250	1850
WRITE	1	1	250	720

8.3 缓冲技术

8.3.1 缓冲技术的引入

(1) 缓和CPU与I/O设备间速度不匹配的矛盾，提高CPU和I/O设备之间的并行性。

(2) 减少对CPU的中断频率， 放宽对CPU中断响应时间的限制。

(3) 解决数据粒度不匹配的问题。

8.3.2 单缓冲和双缓冲

1.单缓冲区(Single Buffer)

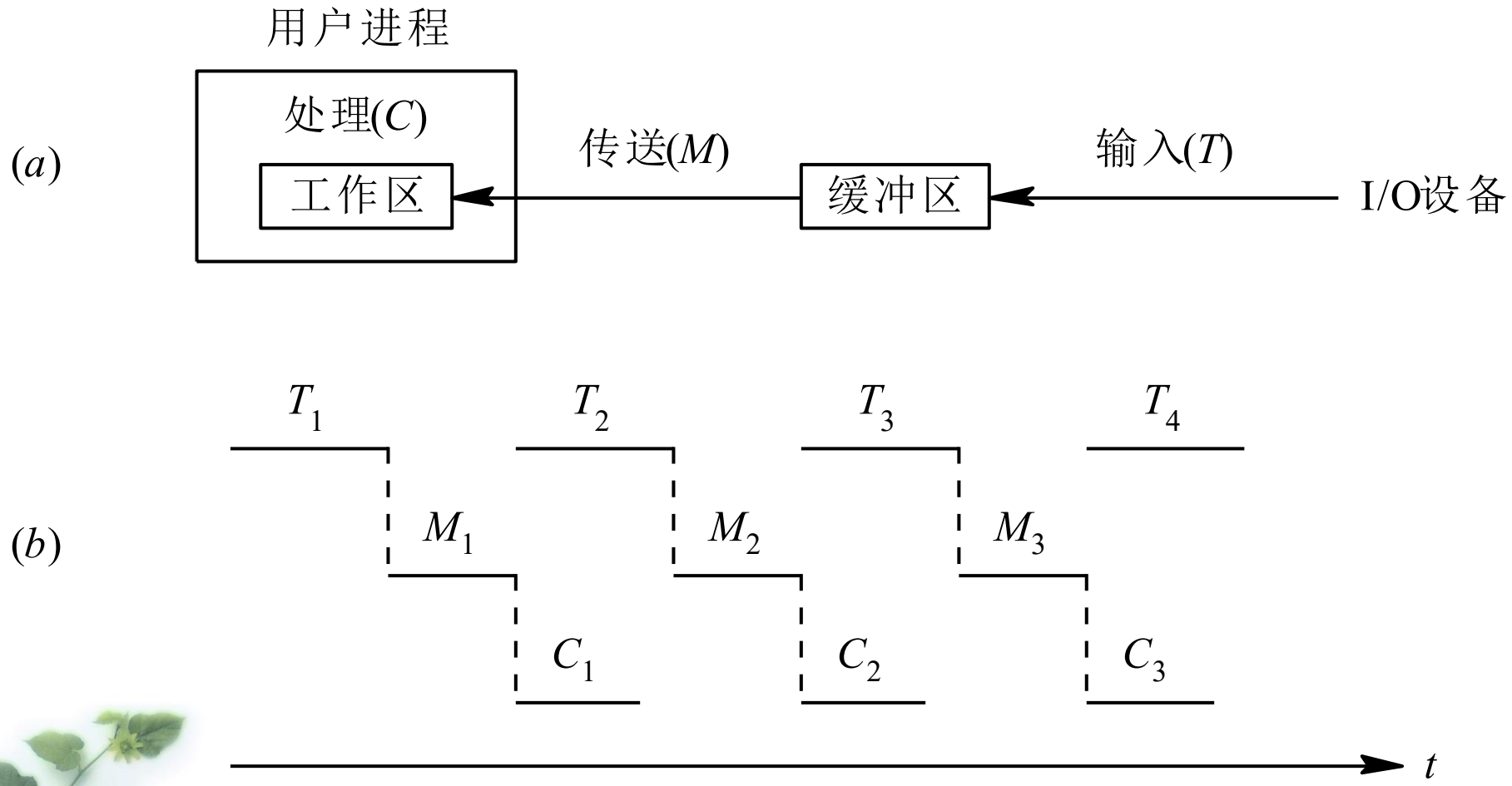


图 单缓冲工作示意图

2. 双缓冲区(Double Buffer)(缓冲对换)

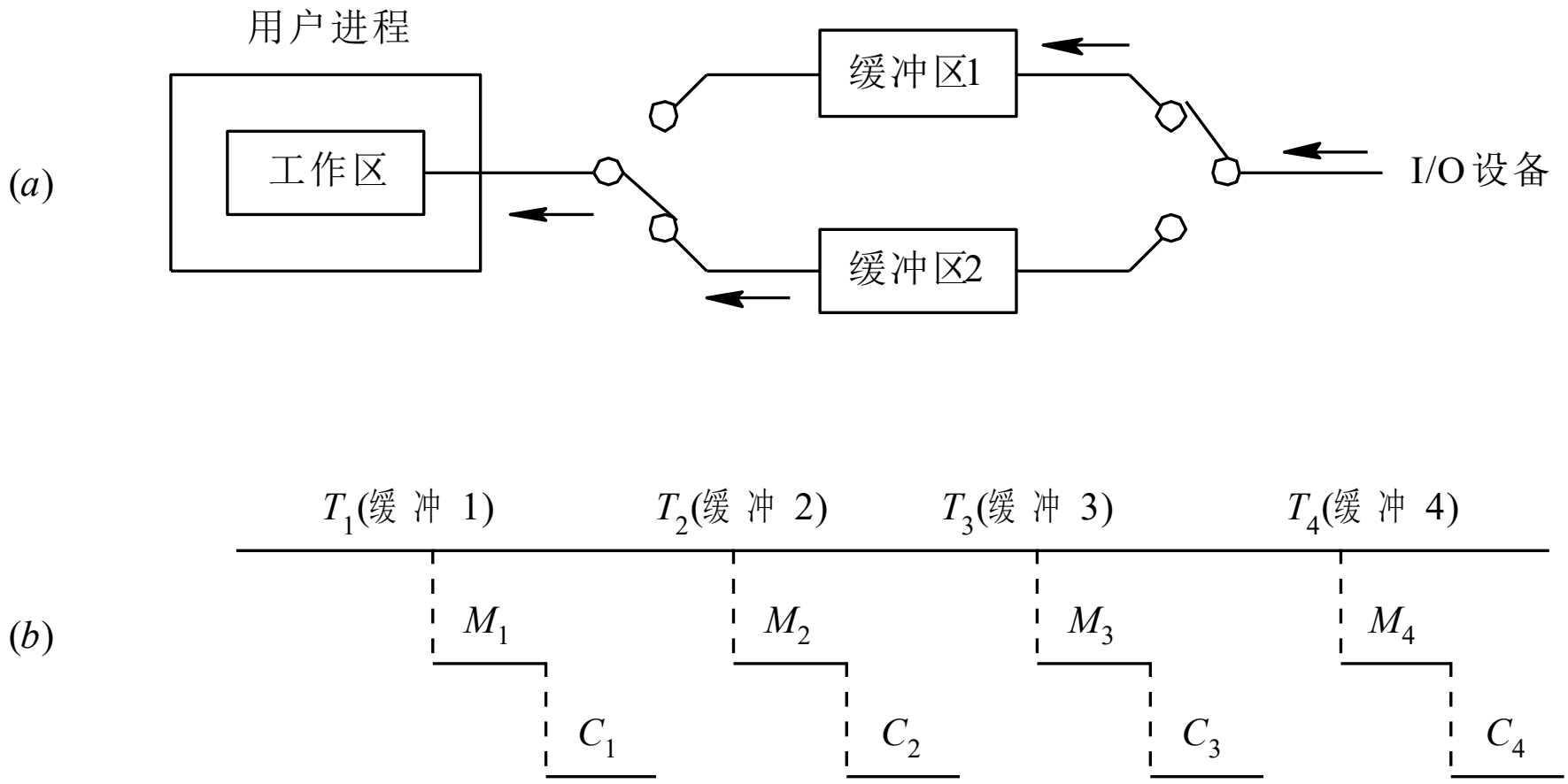


图 8-6 双缓冲工作示意图

8.3.3 环形缓冲区(循环缓冲)

环形缓冲区的组成：多个缓冲区、多个指针

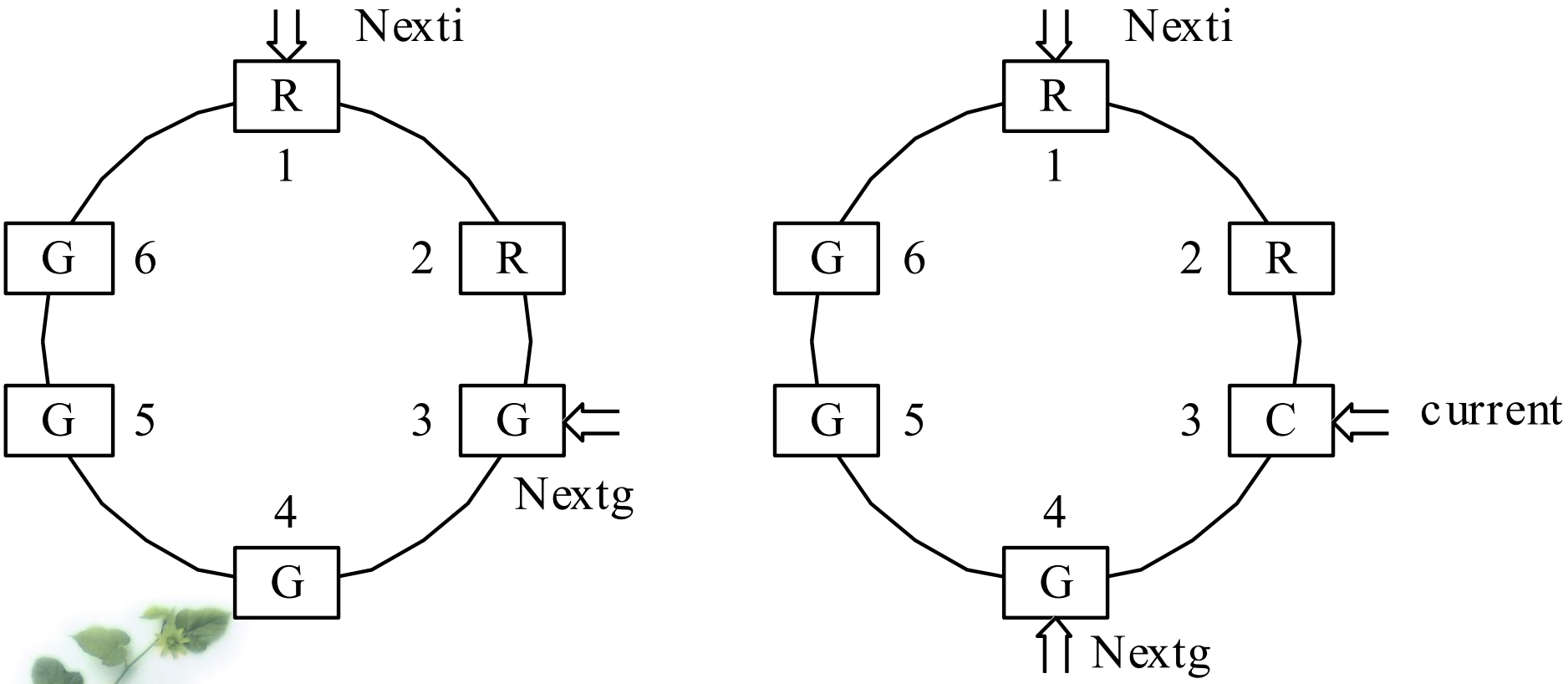


图 环形缓冲区

8.3.4 缓冲池(Buffer Pool)

1.缓冲池的组成

① 空(闲)缓冲区；

② 装满输入数据的缓冲区；

③ 装满输出数据的缓冲区。

将相同类型缓冲区链成一个队列，可形成以下三个队列：

(1) 空白缓冲队列emq。

(2) 输入队列inq。

(3) 输出队列outq。

2. Getbuf过程和Putbuf过程

```
Getbuf(type)
{
    Wait(RS(type));
    Wait(MS(type));
    B(number)=Takebuf(type);
    Signal(MS(type));
}
Putbuf(type, number)
{
   Wait(MS(type));
   Addbuf(type, number);
   Signal(MS(type));
   Signal(RS(type));
}
```

3.缓冲区的工作方式

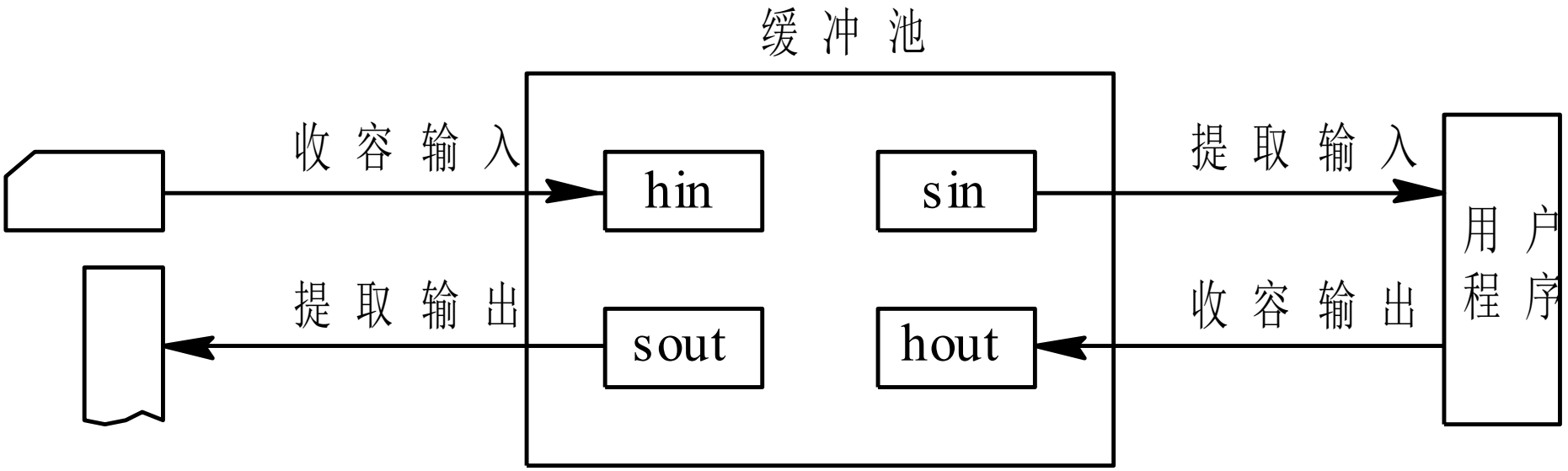


图8-8 缓冲池的工作方式

8.4 设备分配

- 设备分配的基本任务是根据用户的I/O请求，为他们分配所需的设备。如果在I/O设备和CPU之间还存在设备控制器和通道，则还需为分配出去的设备分配相应的控制器和通道。

8.4.1 设备分配中的数据结构

- 为了实现对I/O设备的管理和控制，需要对每台设备、通道、控制器的情况进行登记。设备分配依据的数据结构有设备控制表（DCT）、控制器控制表（COCT）、通道控制表（CHCT）和系统设备表（SDT）。

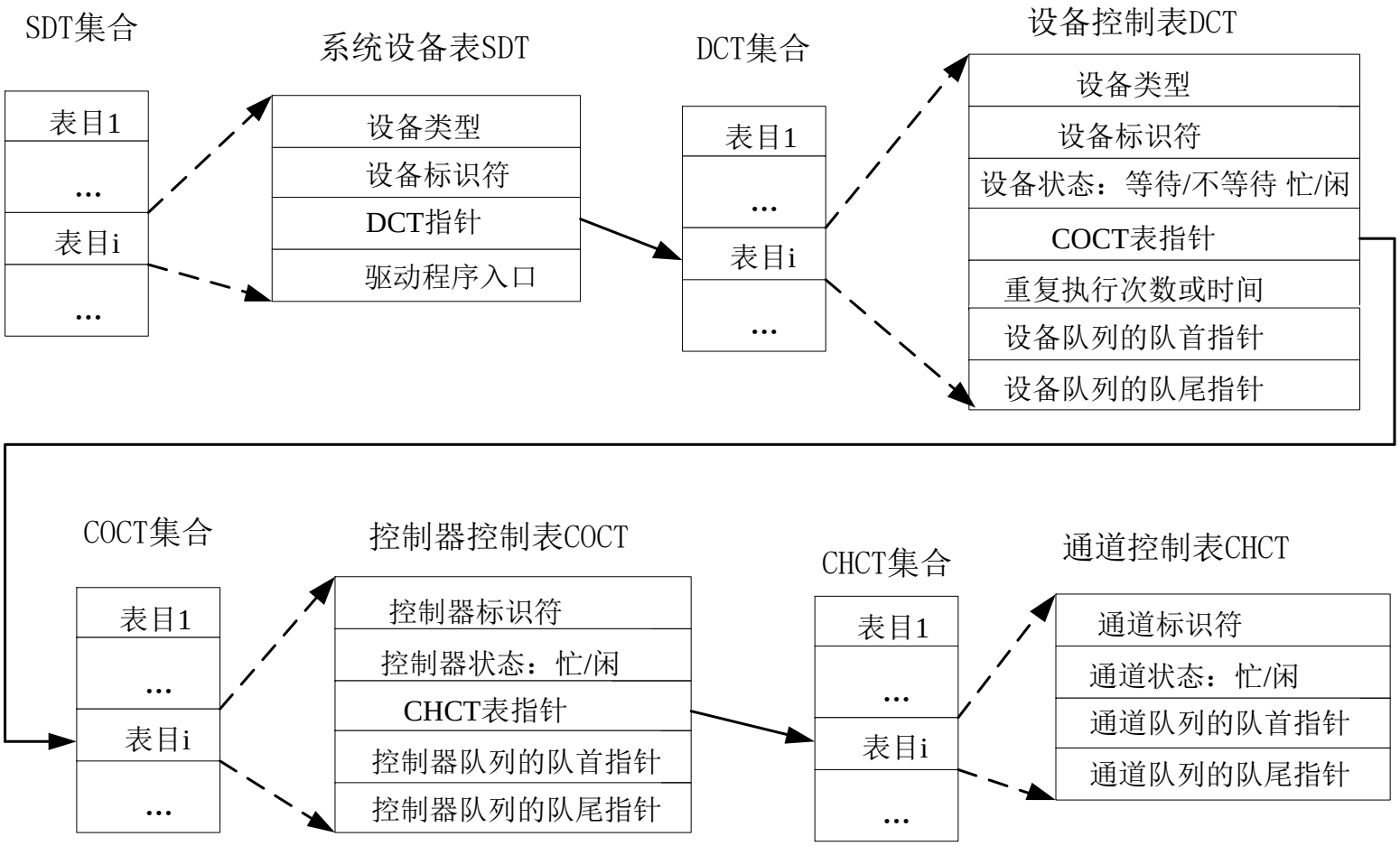


图 8-9 设备分配中的数据结构

8.4.2 设备分配时应考虑的若干因素

- 设备分配的总原则是既要充分发挥设备的使用效率，尽可能的让设备忙，但又要避免由于不合理的分配方法造成进程死锁。另外，还要做到把用户程序和具体物理设备隔离开来。

1. 设备的固有属性：独占/共享/虚拟
2. 设备分配算法
3. 设备分配中的安全性

8.4.5 假脱机（SPOOLing）系统

1. 假脱机技术?

为了缓和CPU的高速性与I/O设备低速性间的矛盾而引入脱机输入、 脱机输出技术。即利用专门的外围控制机，将低速I/O设备上的数据传送到高速磁盘上；或相反。当系统中引入了多道程序技术后，完全可利用其中的一道程序，来模拟脱机输入时的外围控制机功能，把低速I/O设备上的数据传送到高速磁盘上；再用另一道程序来模拟脱机输出时外围控制机的功能，把数据从磁盘传送到低速输出设备上。这样在主机的直接控制下，实现脱机输入、输出功能。此时的外围操作与CPU对数据的处理同时进行，这种在联机情况下实现的同时外围操作的技术称为SPOOLing(Simultaneous Periphernal Operating On-Line)技术，或称为假脱机技术。

2. SPOOLing系统的组成

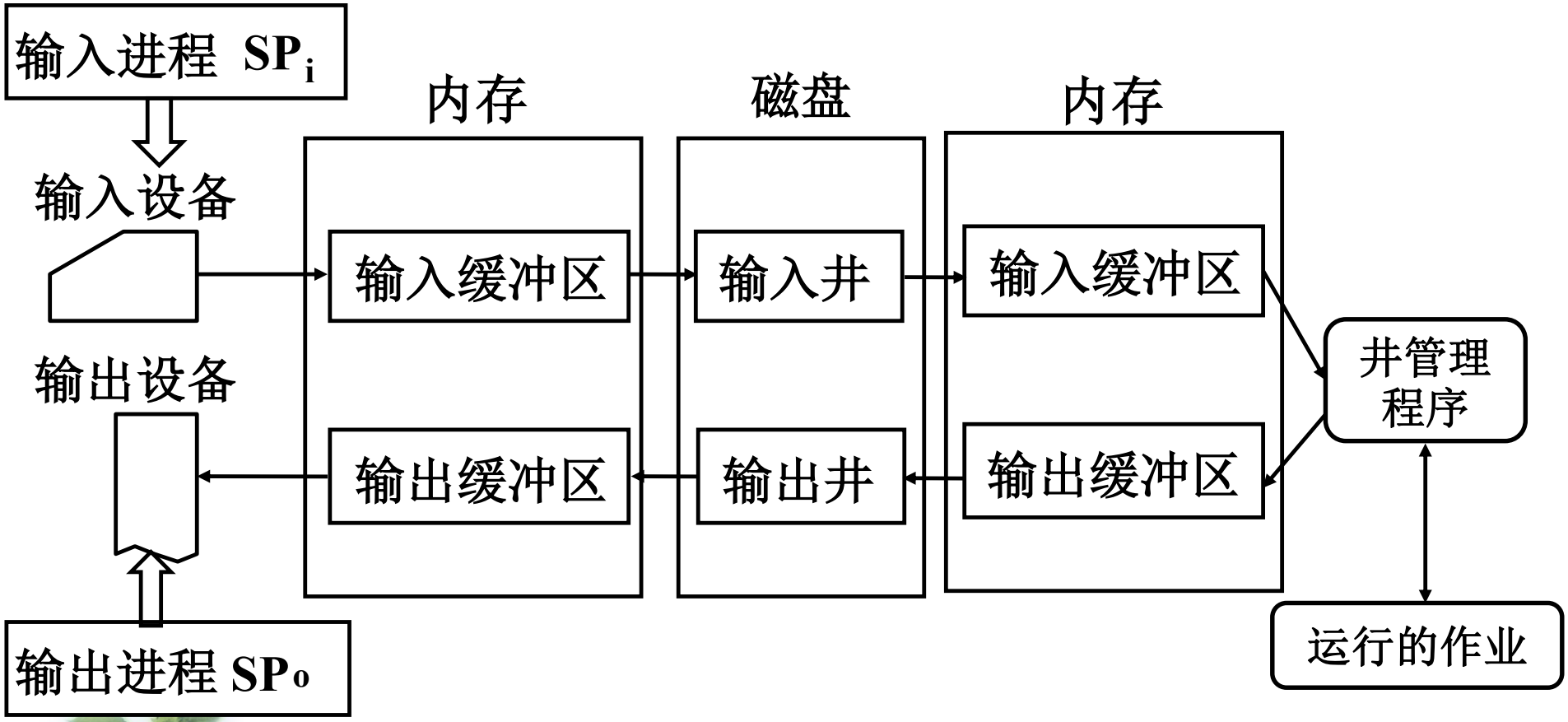


图 6-21 SPOOLing系统的组成

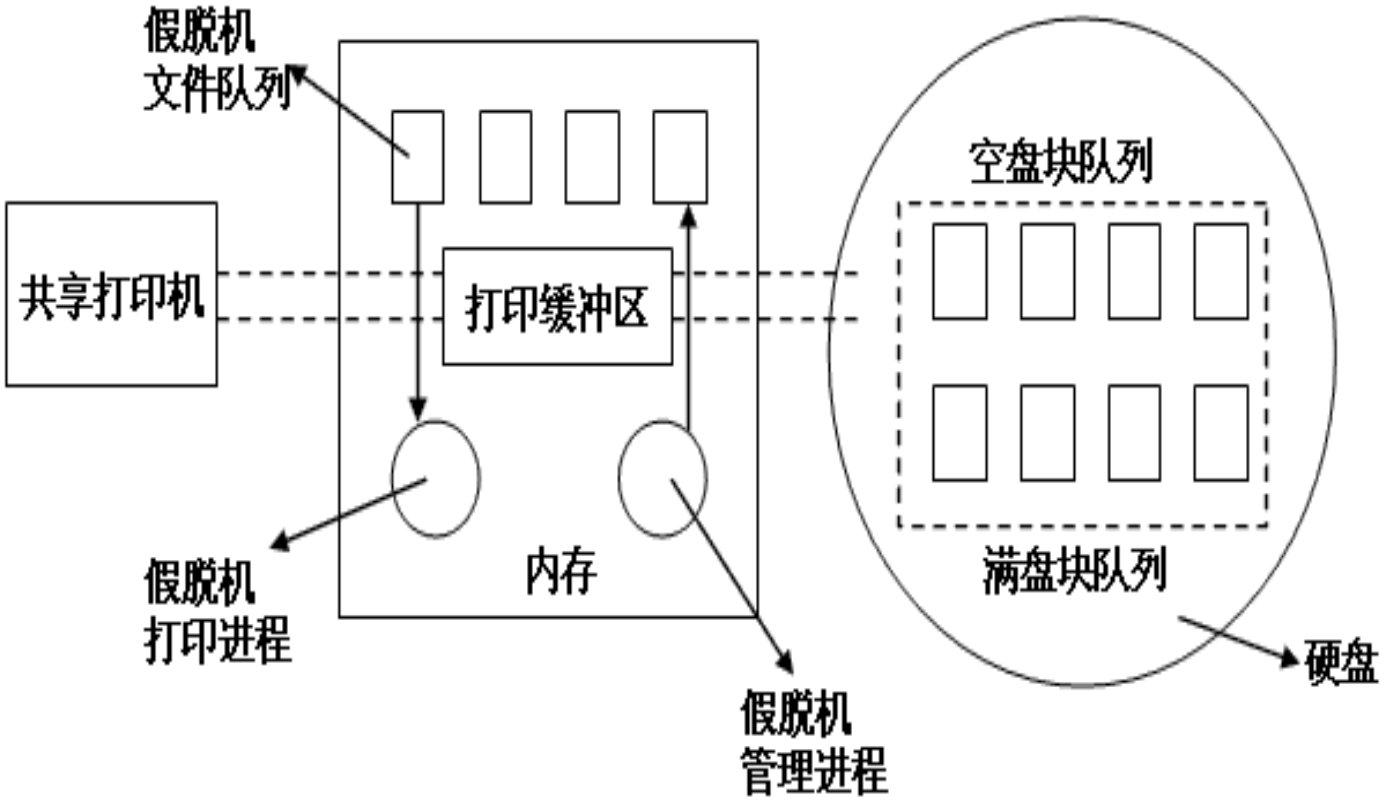
3. SPOOLing系统的特点

(1) 提高了I/O的速度。

(2) 将独占设备改造为共享设备。

(3) 实现了虚拟设备功能。

4. 共享打印机



假脱机打印系统的构成：

(1)磁盘缓冲区。

(2)打印缓冲区。

(3)假脱机管理进程和假脱机打印进程。

SPOOLing技术的使用：

当用户进程请求打印输出时，**SPOOLing**系统同意为它打印输出，但并不真正立即把打印机分配给该用户进程， 而是由假脱机管理进程完成两件事：①在输出井中为之申请一个空闲盘块区，并将要打印的数据送入其中暂存；②为用户进程申请一张空白的用户请求打印表，将用户的打印要求填入其中，再将该表挂到假脱机文件队列上。

如果打印机空闲，由假脱机打印进程从假脱机文件队列的队首取出一张请求打印表，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，再由打印机进行打印。打印完毕后，假脱机打印进程再查看请求打印队列中是否还有等待打印的请求表，直至请求打印队列为空，假脱机打印进程才将自己阻塞起来。

- 结论
  - 对每个用户而言，系统只是即时将数据输出到缓冲区，并没真正被打印，只是让用户感觉系统已为他打印。
  - 其真正的打印操作，是在打印机空闲且该打印任务在等待队列中已排到队首时进行的，并且，打印操作本身也是利用CPU的一个时间片，没有使用专门的外围机。这一过程是用户不可见的。

8.5　I/O软件

- 8.5.1　I/O软件的目标
- 8.5.2　I/O软件的层次结构
- 8.5.3　中断处理程序
- 8.5.4　设备驱动程序
- 8.5.5　设备独立性软件

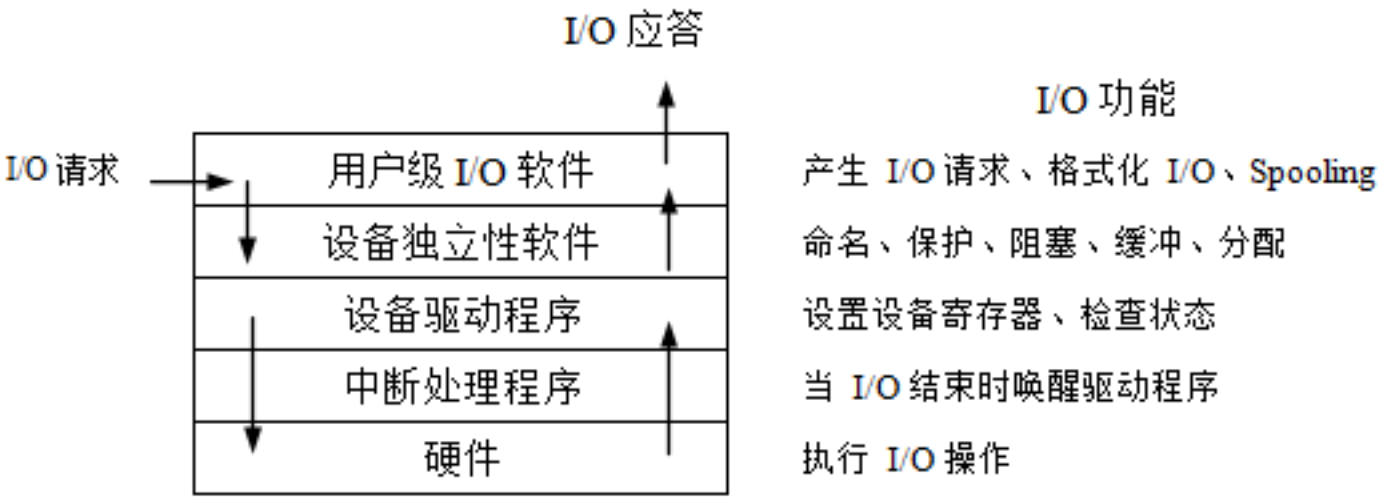
8.5.1　I/O软件的目标

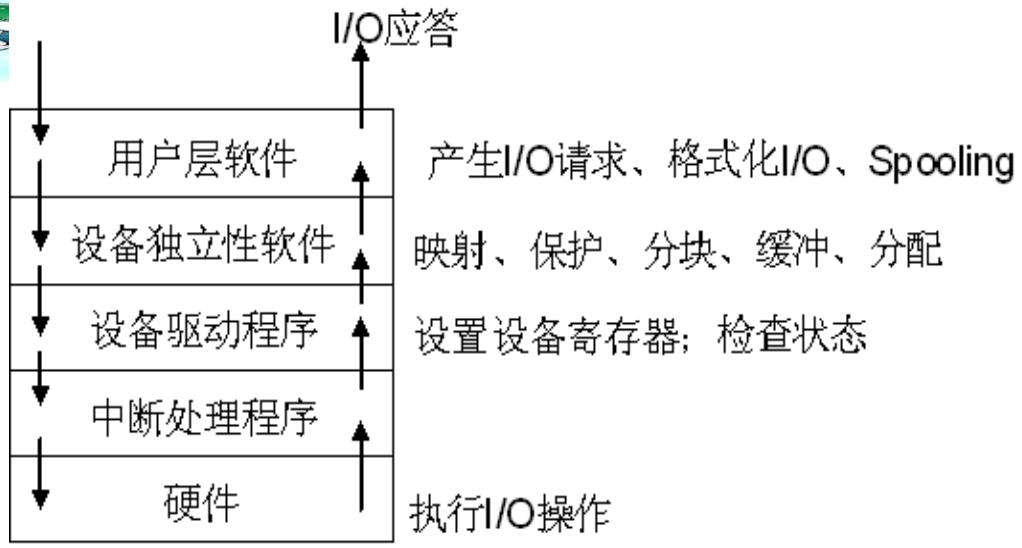
I/O软件是使用I/O设备，与I/O操作相关软件的集合。

1. 设备独立性
2. 统一命名
3. 错误处理
4. 数据传输
5. 缓冲
6. 共享设备和独占设备

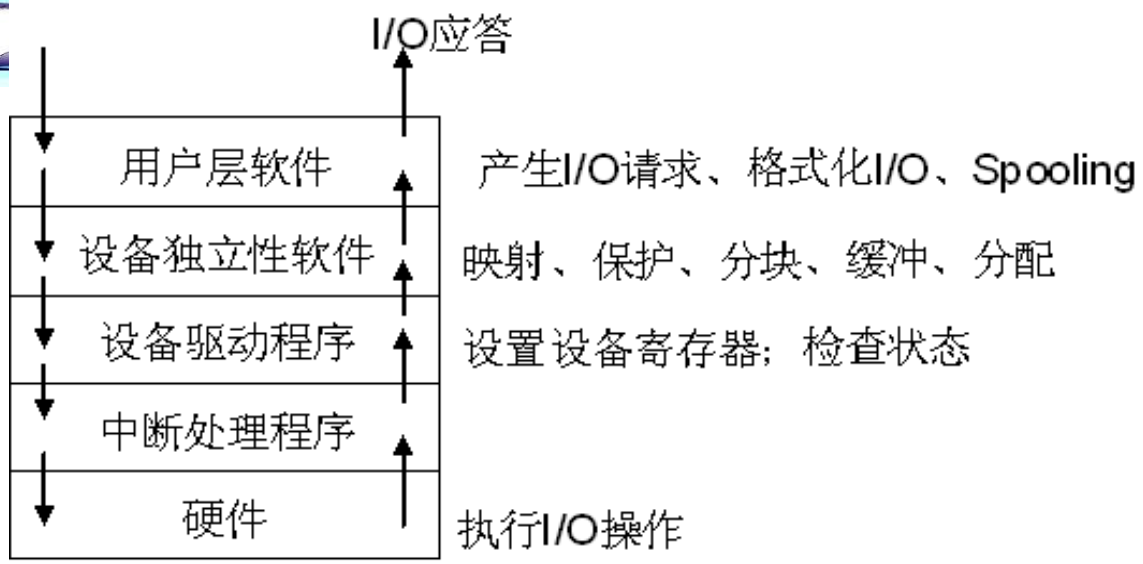
8.5.2 I/O软件的层次结构

- I/O软件通常分为4个层次，从上往下依次是：用户级I/O软件、与设备无关的I/O软件、设备驱动程序、中断处理程序。





- （1）用户层软件
  - 实现与用户交互的接口，用户可直接调用在用户层提供的、与I/O操作有关的库函数，对设备进行操作。
- （2）设备独立软件
  - 用于实现用户程序与设备驱动器的统一接口、设备命名、设备的保护以及设备的分配与释放等，同时为设备管理和数据传送提供必要的存储空间。



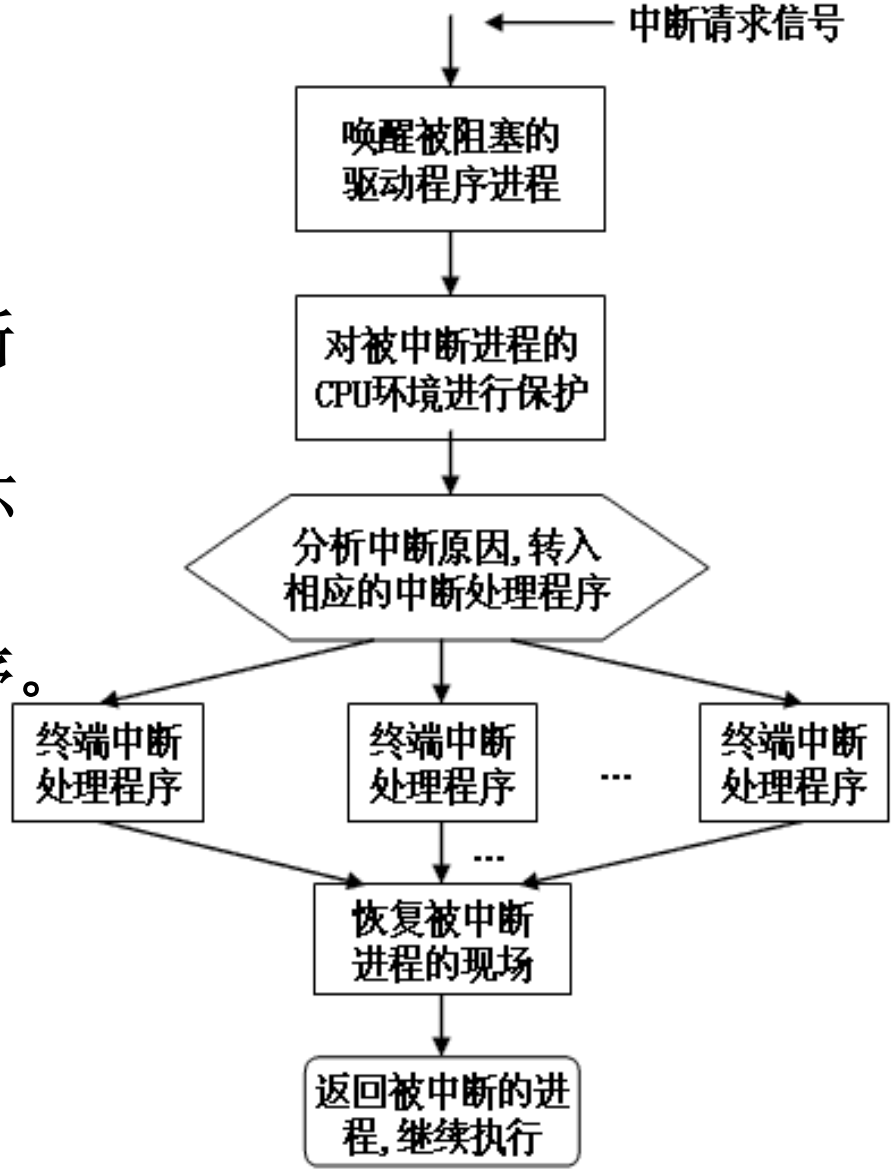
- （3）设备驱动程序
  - 与硬件直接相关，用于具体实现系统对设备发出的操作指令，驱动I/O设备工作的驱动程序。
- （4）中断处理程序
  - 用于保存被中断进程的CPU环境，转入相应的中断处理程序进行处理，处理完后再恢复被中断进程的现场后，返回到被中断进程。

8.5.3 中断处理程序

- 在设备控制器的控制下，I/O设备完成了I/O操作后，控制器向CPU发出一中断请求，CPU响应后便中止当前正在运行的程序，转向处理中断事件的程序段中去执行，这种处理中断的子程序就是中断处理程序，又称为中断服务程序。

8.5.3 中断处理程序

- 中断处理流程
  - 测定是否有未响应的中断信号。
  - 保护被中断进程的CPU环境。
  - 转入相应的设备处理程序。
  - 中断处理。
  - 恢复CPU的现场。



8.5.4 设备驱动程序

- 设备驱动程序与硬件直接相关，负责具体实现系统对设备发出的操作指令，驱动I/O设备工作的驱动程序。
- 通常，每一类设备配置一个设备驱动程序。
- 设备驱动程序向上层用户程序提供一组标准接口，设备具体的差别被设备驱动程序所封装，用于接收上层软件发来的抽象I/O要求，如read和write命令，转换为具体要求后，发送给设备控制器，控制I/O设备工作；它也将由设备控制器发来的信号传送给上层软件。

I/O设备驱动程序的功能

（1）接收由I/O进程发来的命令和参数，并将命令中的抽象要求转换为具体要求，例如，将磁盘块号转换为磁盘的盘面、磁道号及扇区号。

（2）检查用户I/O请求的合法性，了解I/O设备的状态，传递有关参数，设置设备的工作方式。

（3）发出I/O命令。如果设备是空闲的，则立即启动设备去完成指定的I/O操作；如果处于忙碌状态，则将请求者的PCB挂在设备队列上等待。

（4）响应由控制器或通道发来的中断请求，并根据中断类型转到相应的中断处理程序进行处理。

（5）对于设置有通道的计算机系统，I/O操作是由通道执行通道程序来完成的，驱动程序还应能够根据用户的I/O请求，自动地构成通道程序。

8.5.5 设备独立性软件

- 设备独立性也称设备无关性，指的是应用程序独立于具体使用的物理设备。
- 为了实现设备独立性而引入了逻辑设备和物理设备这两个概念。在应用程序中，使用逻辑设备名称来请求使用某类设备；而系统在实际执行时，必须使用物理设备名称。因此，系统须具有将逻辑设备名称转换为某物理设备名称的功能。

设备独立性软件的主要功能

（1）执行所有设备的公有操作。

①独立设备的分配与回收；

②将逻辑设备名映射为物理设备名，进一步可以找到相应物理设备的驱动程序；

③对设备进行保护，禁止用户直接访问设备；

④缓冲管理。

⑤差错控制。由于在I/O操作中的绝大多数错误都与设备有关，故主要由设备驱动程序处理，而设备独立性软件只处理那些设备驱动程序无法处理的错误。

（2）向用户层（或文件层）软件提供统一的接口。

8.6 磁盘调度和管理

8.6.1 磁盘物理特性

1. 数据的组织和格式

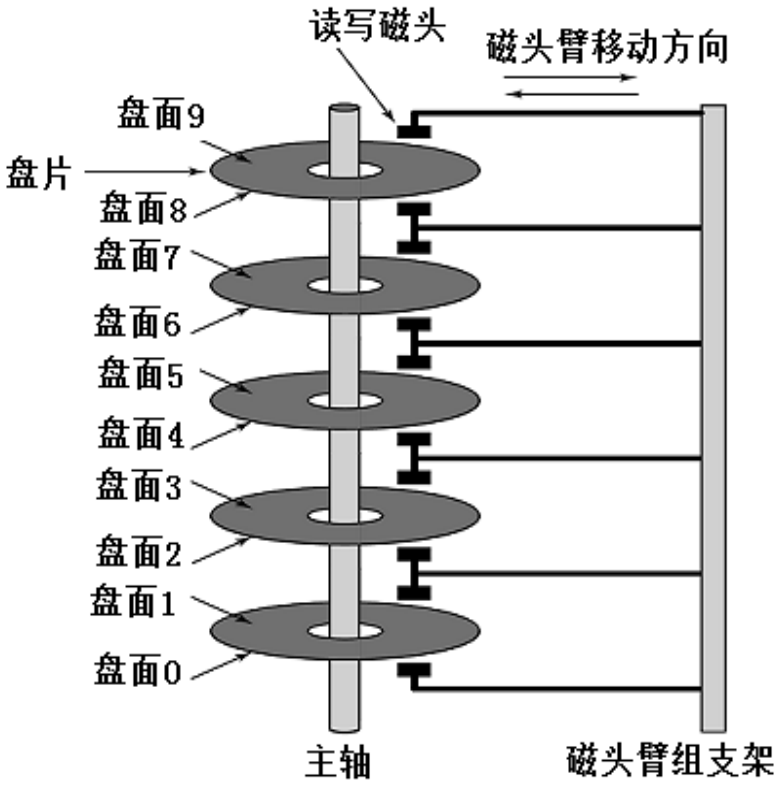


图8-13 磁盘驱动器的结构

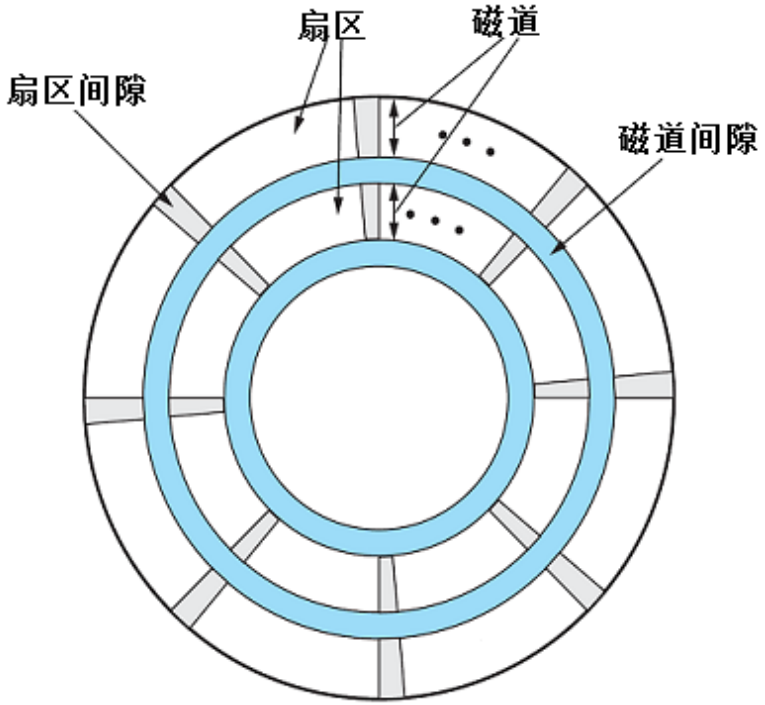


图8-14磁盘的数据布局

1. 数据的组织和格式

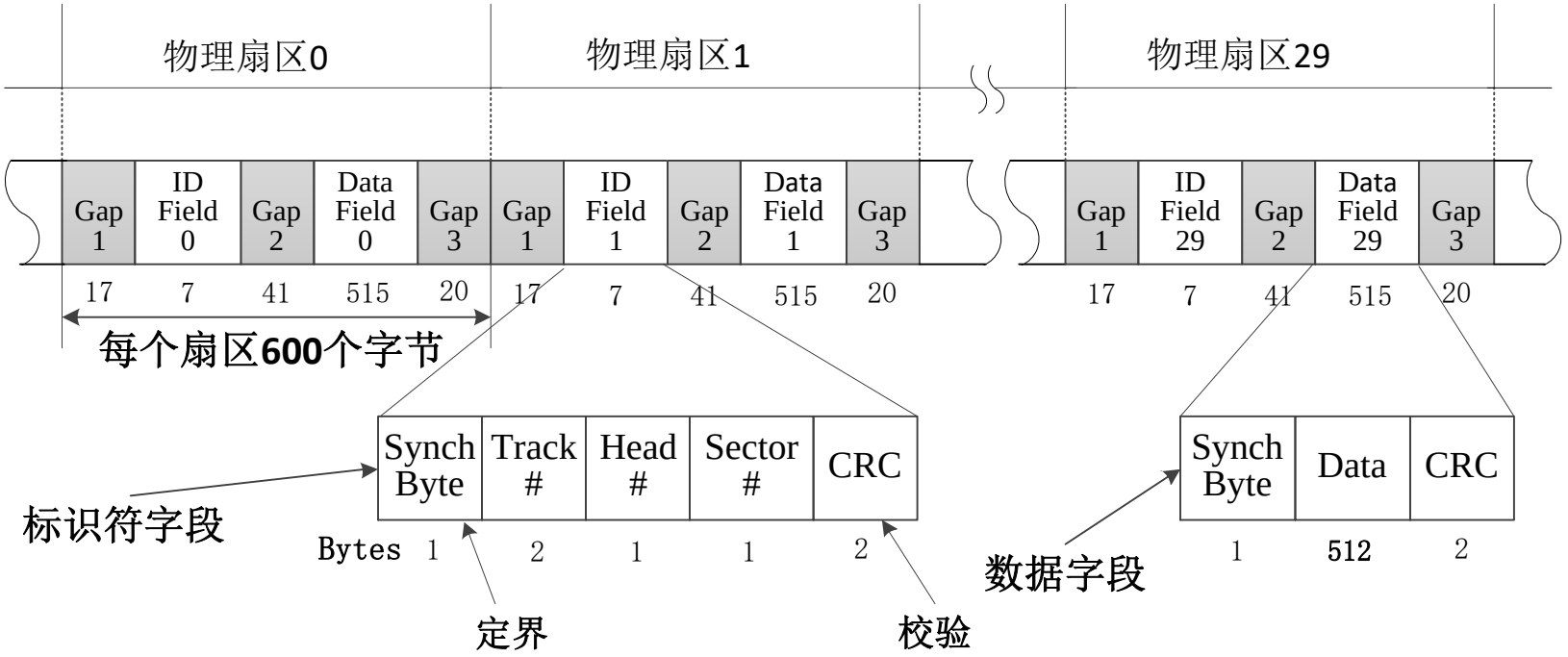


图 8-15 磁盘的格式化

2. 磁盘的类型

1) 固定头磁盘

在每条磁道上都有一读/写磁头，所有的磁头都被装在一刚性磁臂中。通过这些磁头可访问所有磁道，并进行并行读/写，有效提高磁盘的I/O速度。 主要用于大容量磁盘上。

2) 移动头磁盘

每一个盘面仅配一个磁头，也被装入磁臂中。为能访问该盘面上的所有磁道，该磁头必须能移动以进行寻道。移动磁头仅能以串行方式读/写，致使I/O速度较慢；但结构简单，故广泛应用于中小型磁盘设备中。

3. 磁盘访问时间

1) 寻道时间T_s

把磁臂(磁头)移动到指定磁道上所经历的时间。该时间是启动磁臂的时间s与磁头移动n条磁道所花费的时间之和，即

$$T_s=m\times n+s$$

其中，m是常数，与磁盘驱动器的速度有关。磁臂的启动时间约为2 ms。对一般的温盘， 其寻道时间将随寻道距离的增加而增大。

2) 旋转延迟时间T_r

指定扇区移动到磁头下面所经历的时间。

3) 传输时间T_t

把数据从磁盘读出或向磁盘写入数据所经历的时间。T_t的大小与每次所读/写的字节数b和旋转速度有关：

$$T_t = \frac{b}{rN}$$

其中，r为磁盘每秒钟的转数；N为一条磁道上的字节数。

访问时间T_a为：

$$T_a = T_s + T_r + \frac{b}{rN}$$

8.6.2 早期的磁盘调度算法

1. 先来先服务FCFS(First-Come, First Served)

（从 100 号磁道开始）	
被访问的下一个磁道号	移动距离（磁道数）
55	45
58	3
39	19
18	21
90	72
160	70
150	10
38	112
184	146
平均寻道长度：	55.3

图 FCFS调度算法

优点：

公平、简单，且每个进程的请求都能依次处理，不会出现某一进程的请求长期得不到满足的情况。

缺点：

平均寻道时间较长。

适用场合：

请求磁盘I/O的进程数较少。

2. 最短寻道时间优先SSTF(Shortest Seek Time First)

（从 100 号磁道开始）	
被访问的下一个磁道号	移动距离（磁道数）
90	10
58	32
55	3
39	16
38	1
18	20
150	132
160	10
184	24
平均寻道长度：27.5	

图 SSTF调度算法

3. 扫描(SCAN)算法

1) 进程“饥饿”现象

SSTF算法虽能获得较好的寻道性能，但却可能导致某些进程发生“饥饿”(Starvation)现象。因为只要不断有新进程的请求到达，且其所要访问的磁道与磁头当前所在磁道的距离较近，这种新进程的I/O请求必须优先满足。

2) SCAN 算法

电梯调度算法

（从 100# 磁道开始，向磁道号增加方向访问）	
被访问的下一个磁道号	移动距离（磁道数）
150	50
160	10
184	24
90	94
58	32
55	3
39	16
38	1
18	20
平均寻道长度：27.8	

图 6-32 SCAN调度算法示例

4. 循环扫描(CSCAN)算法

（从100#磁道开始，向磁道号增加方向访问）	
被访问的下一个磁道号	移动距离（磁道数）
150	50
160	10
184	24
18	166
38	20
39	1
55	16
58	3
90	32
平均寻道长度： 35.8	

图 CSCAN调度算法示例

5. N-Step-SCAN和FSCAN调度算法

1) N-Step-SCAN算法

在SSTF、 SCAN及CSCAN几种调度算法中， 都可能出现磁臂停留在某处不动的情况， 我们把这一现象称为“磁臂粘着”(Armstickiness)。在高密度磁盘上容易出现此情况。

N步SCAN算法是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。 而每处理一个队列时又是按SCAN算法，对一个队列处理完后，再处理其他队列。 当正在处理某子队列时，如果又出现新的磁盘I/O请求，便将新请求进程放入其他队列，这样就可避免出现粘着现象。 当N值取得很大时，会使N步扫描法的性能接近于SCAN算法的性能； 当N=1时， N步SCAN算法便蜕化为FCFS算法。

N步SCAN算法

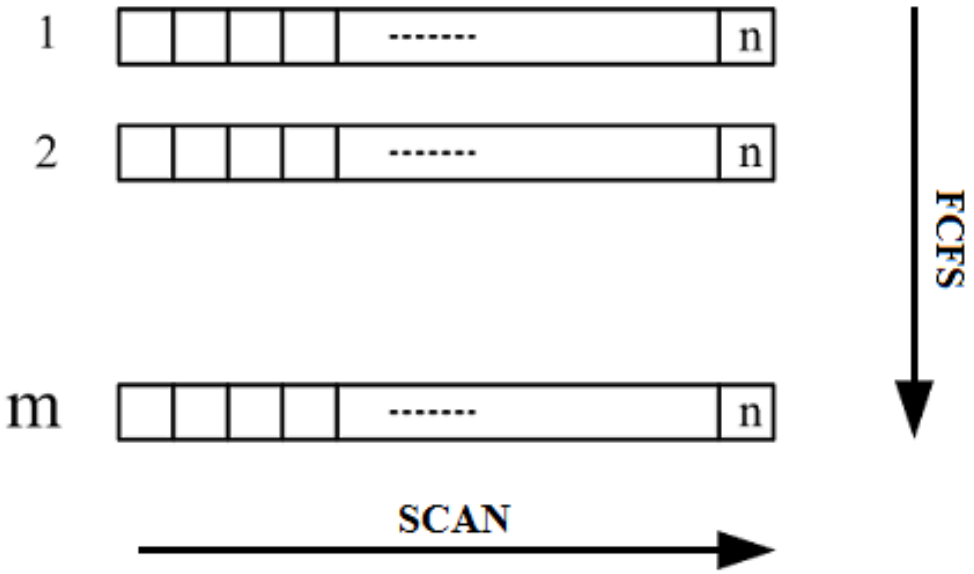


图8-16 N步SCAN算法示意图

2) FSCAN算法

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。

FSCAN算法

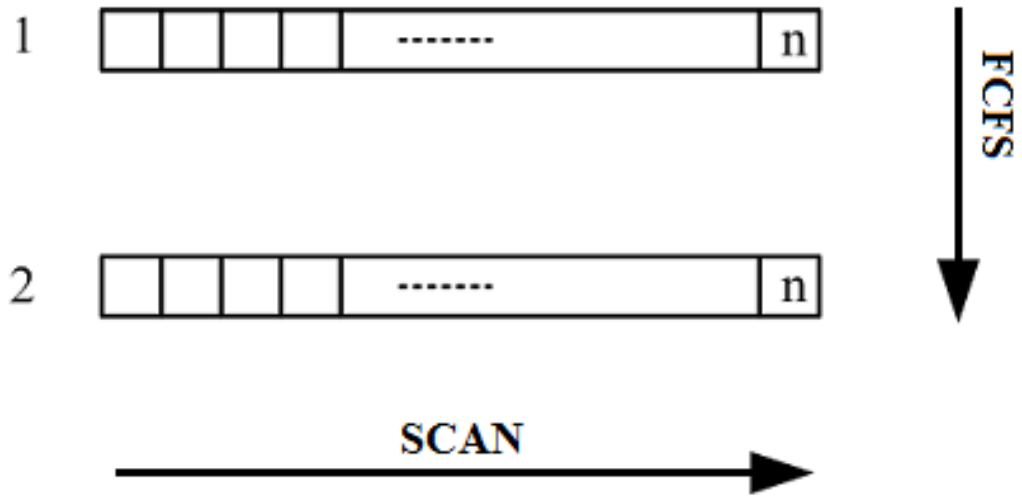


图8-17 FSCAN算法示意图

小　结

I/O系统组成，I/O设备分类；

了解设备控制器和通道；

理解几种I/O控制方式；

缓冲的引入，几种缓冲方式；

设备分配中的数据结构；

了解设备分配时应考虑的若干因素；

设备独立性概念；

理解SPOOLing；

了解设备处理，中断处理程序的处理过程。

小　结

了解磁盘结构、类型；

掌握磁盘调度算法；