第六章 输入输出系统

6.1.1 I/O设备

1. 类型

- 按速度分
  - 低速：键盘
  - 中速：打印机
  - 高速：磁盘
- 按信息交换单位分
  - 块设备：磁盘（可定位）
  - 字符设备：打印机、串口
- 按共享属性分
  - 独占设备：如临界资源
  - 共享设备：如磁盘
  - 虚拟设备：独占 ⇨ 共享

6.1.1 I/O设备

2. 设备与控制器之间的接口

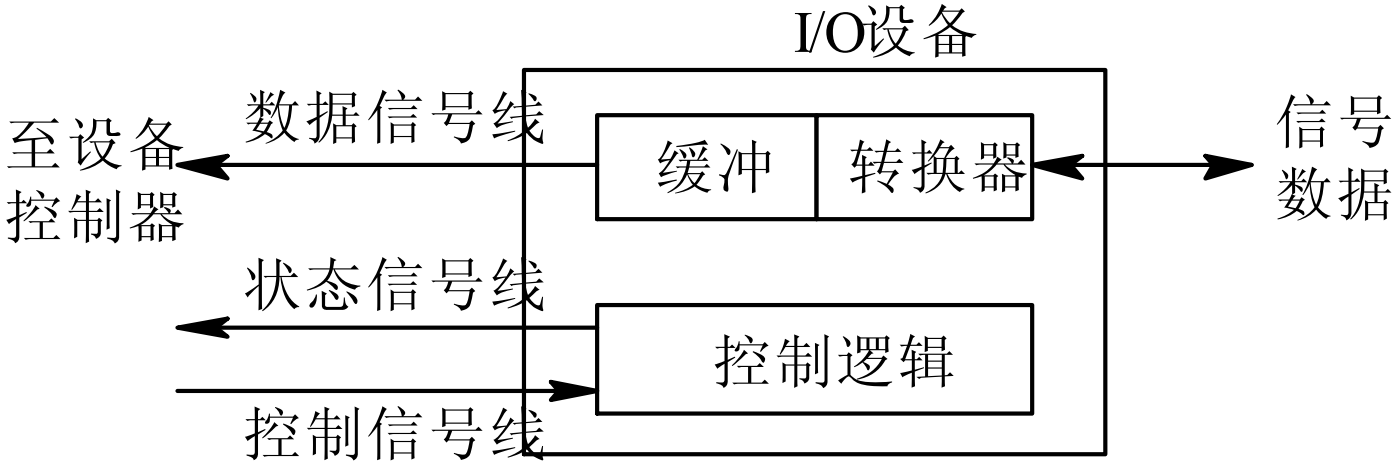


图6-3 设备与控制器之间的接口

CPU———控制器———设备

三种信号：

（1）数据信号：双向，有缓存；

（2）控制信号：控制器发给设备，要求其完成相关操作；

（3）状态信号：设备发给控制器；

6.1.2 设备控制器

1. 设备控制器的基本功能

接收CPU命令，控制I/O设备工作，解放CPU.

① 接收和识别命令

② 数据交换：**数据寄存器**

③ 设备状态的了解和报告：**状态寄存器**

④ 地址识别：CPU通过“地址”与设备通信，设备控制器应能识别它所控制的设备地址以及其各寄存器的地址。

⑤ 数据缓冲

⑥ 差错控制

6.1.2 设备控制器

2. 设备控制器的组成

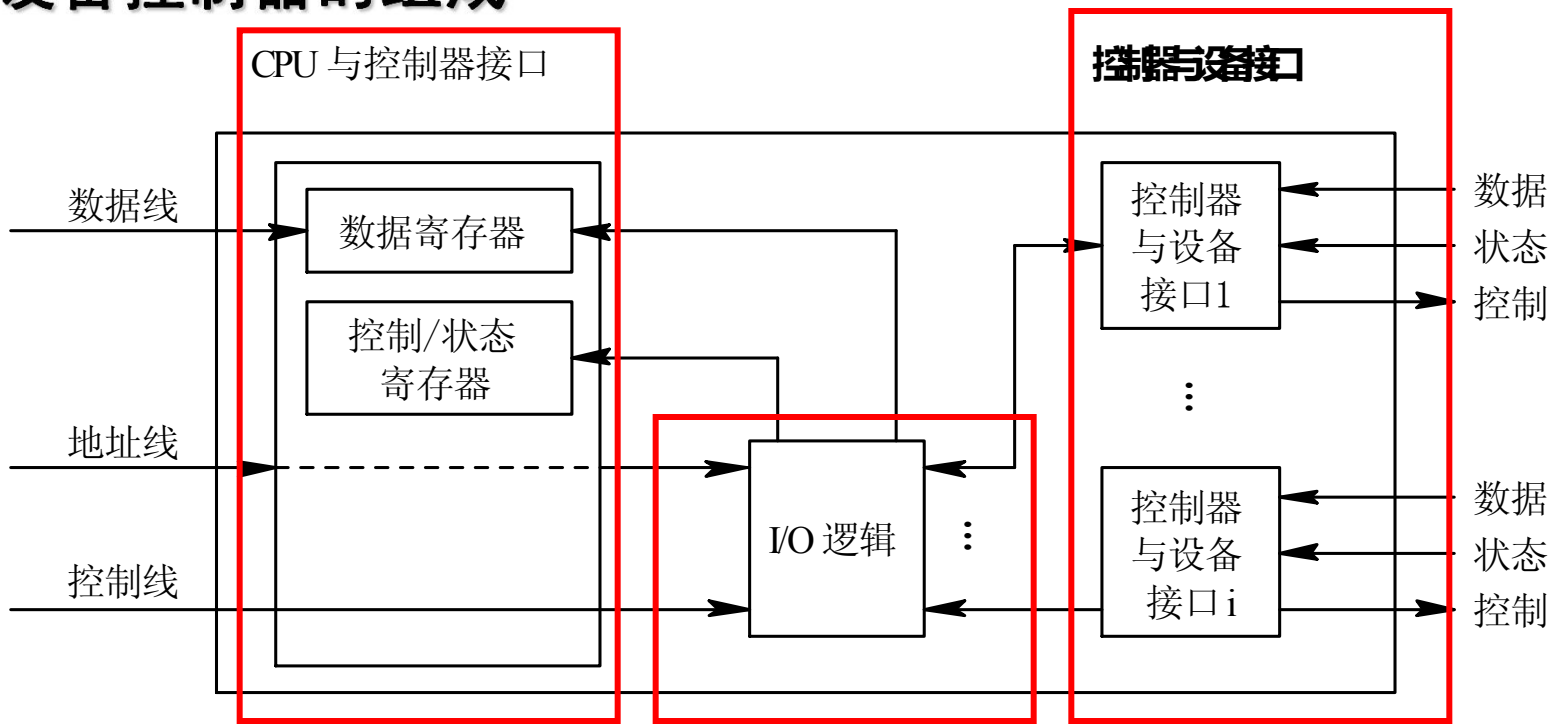


图6-4 设备控制器的组成

- 各类寄存器：数据、状态
- 信号线：数据线、地址线、控制线
- I/O逻辑：在其控制下完成与CPU、设备的通信。

6.1.3 I/O通道

1. 引入

- 引入目的

  解脱CPU对I/O的组织、管理。

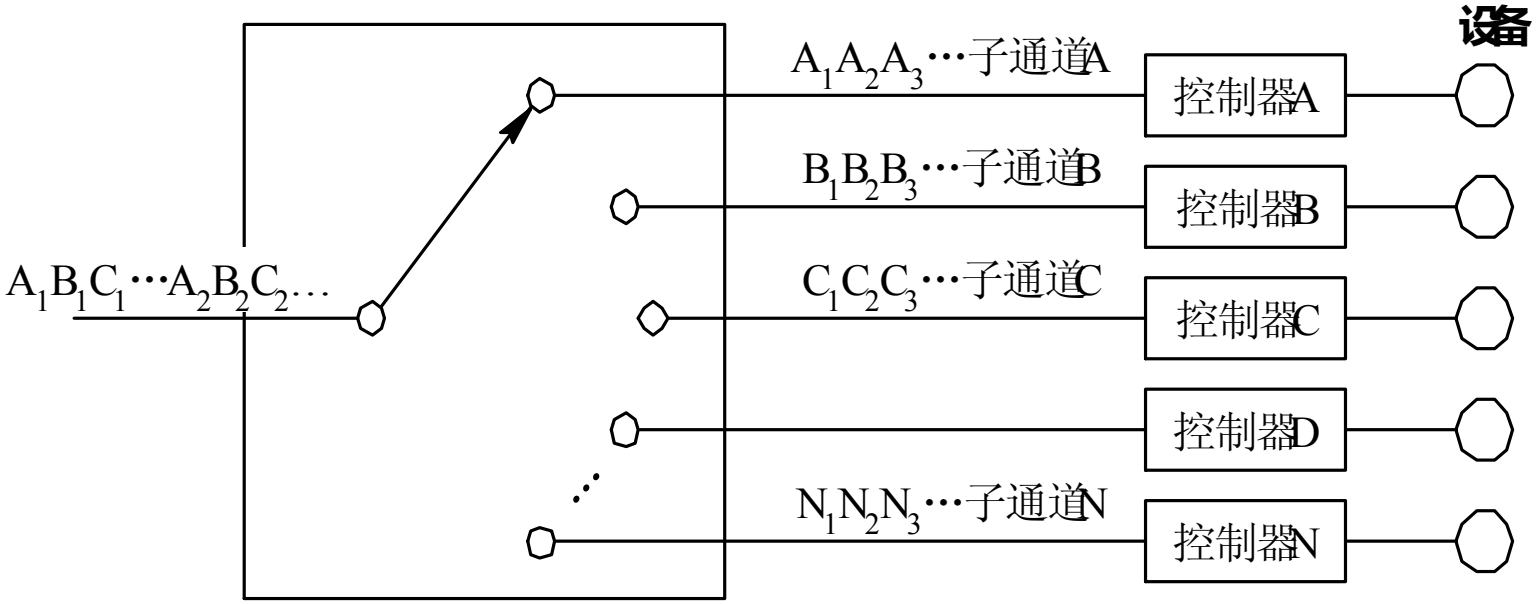
- 通道

  一种特殊的执行I/O指令的处理机，与CPU共享内存，可以有自己的总线。

- CPU只需发送I/O命令给通道，通道建立通道程序，并执行通道程序完成I/O任务。

6.1.3 I/O通道

2. 通道类型



字节多路通道

各子通道以时间片轮转方式共享通道，适用于低、中速设备。

6.1.3 I/O通道

2. 通道类型

数组选择通道

- 无子通道，仅一主通道，某时间由某设备独占，适于高速设备。
- 但通道未共享，利用率低。

数组多路通道

- 多子通道不是以时间片方式，而是“按需分配”，综合了前面2种通道类型的优点。

6.1.3 I/O通道

3.“瓶颈”问题

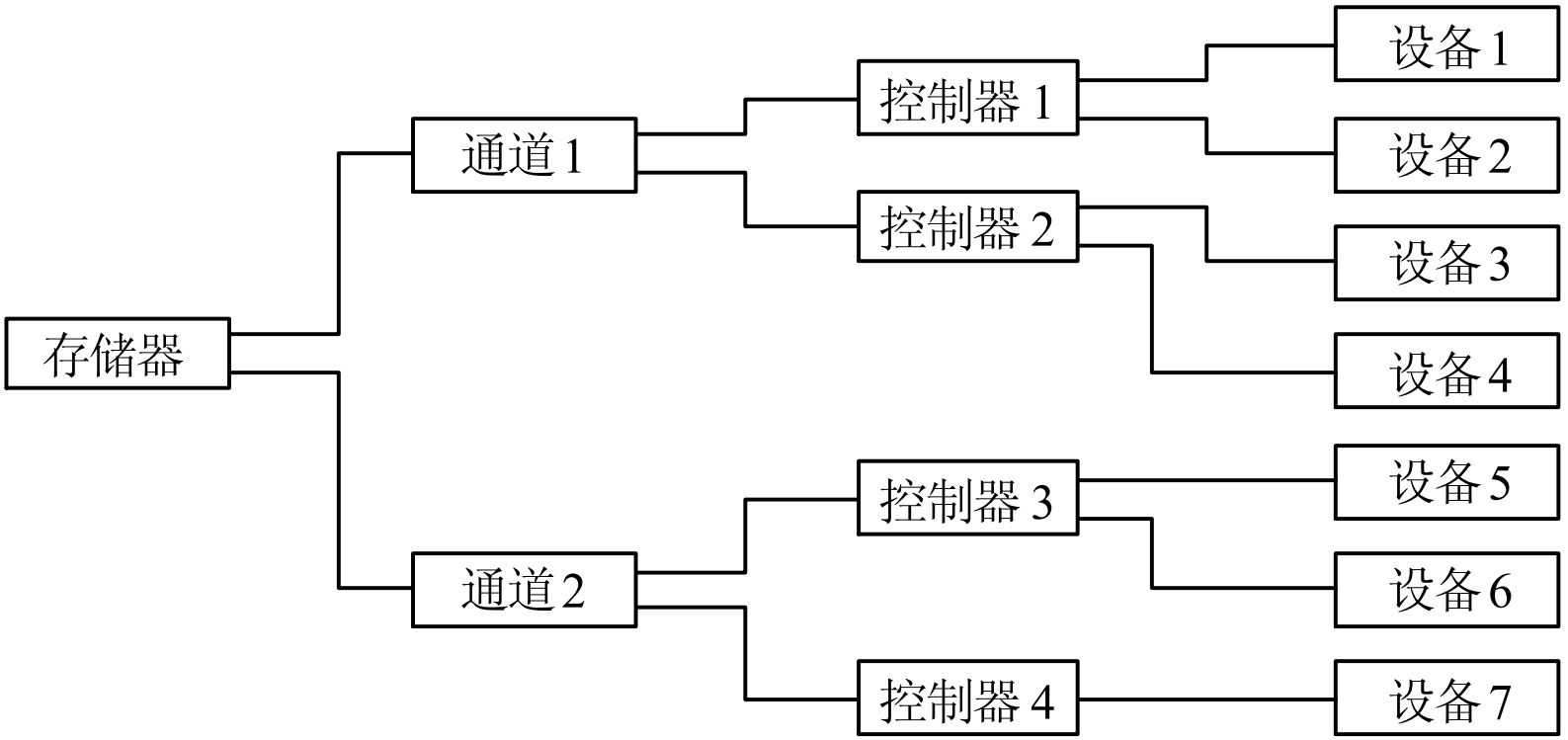


图 6-7 单通路I/O系统

6.1.3 I/O通道

3.“瓶颈”问题----解决：采用复联方式

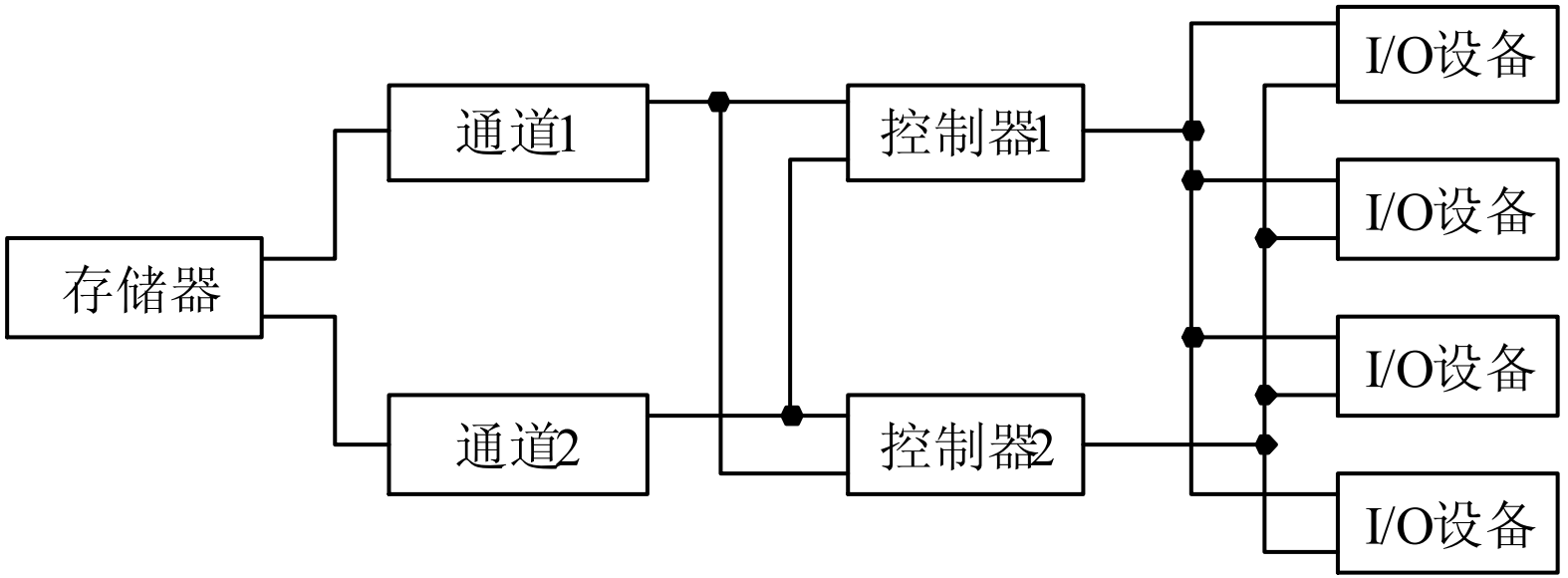
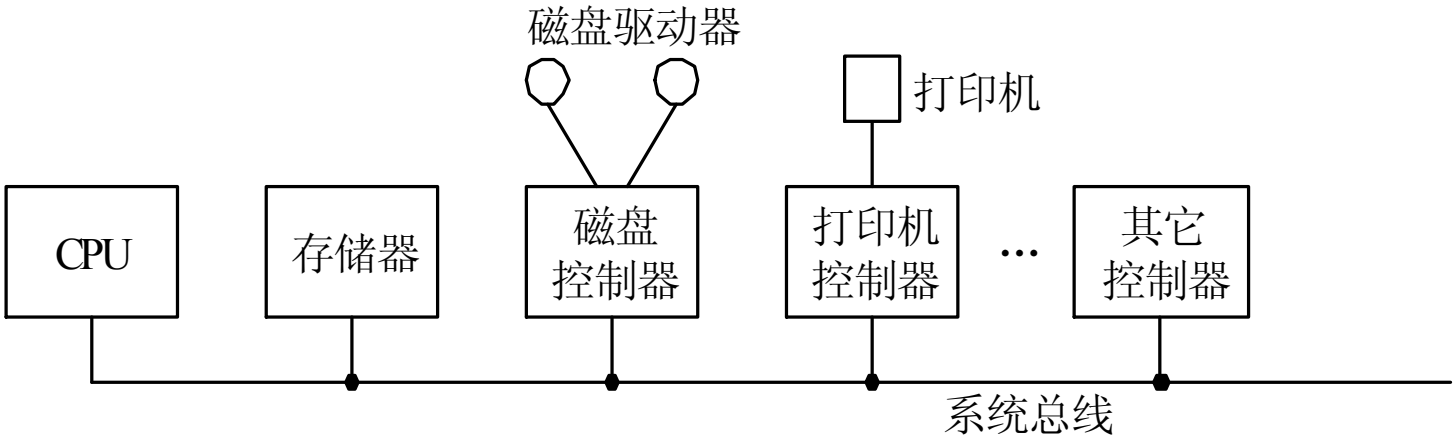


图 6-8 多通路I/O系统

6.1.4 总线系统

1. 微机I/O系统



总线型I/O系统结构图

- 设备控制器与设备是一对多的关系，系统是通过它与设备通信
- 系统——设备控制器—— 设备
- 如：磁盘设备，打印设备
- 缺点：总线瓶颈，CPU瓶颈。

6.1.4 总线系统

2. 主机I/O系统（四级结构）

计算机———I/O通道———I/O控制器———设备

- I/O通道相当于对总线的扩展，即多总线方式，且通道有一定的智能性，能与CPU并行，解决其负担。

- ISA/EISA/LocalBUS/VESA/PCI

6.2 I/O控制方式

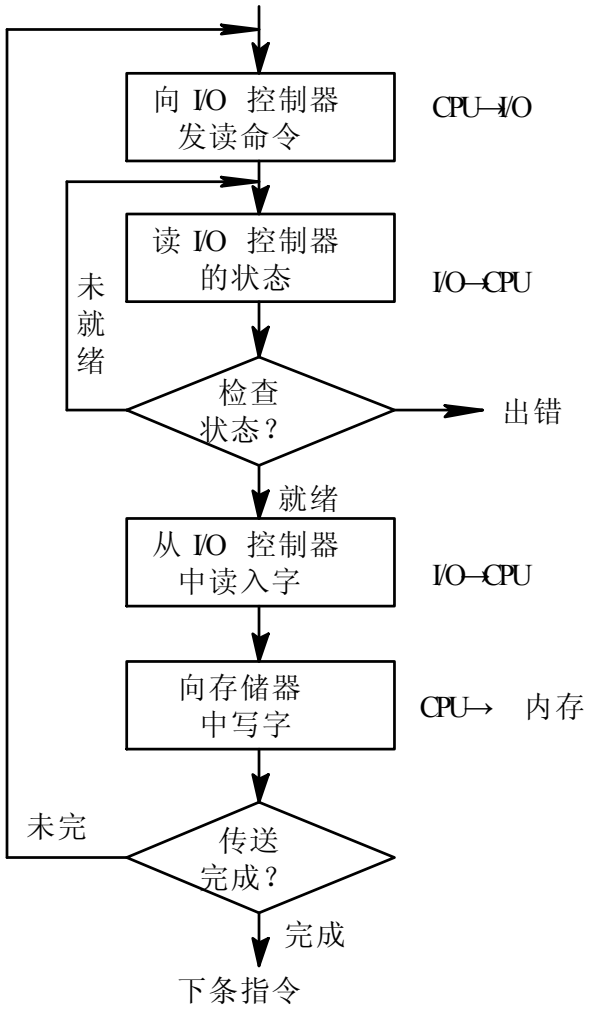
概述

- 四个阶段：

程序I/O——中断I/O——DMA控制——通道控制

- 趋势：减少CPU对I/O操作的干预程度，提高并行度。

6.2.1 程序I/O方式



(a) 程序 I/O 方式

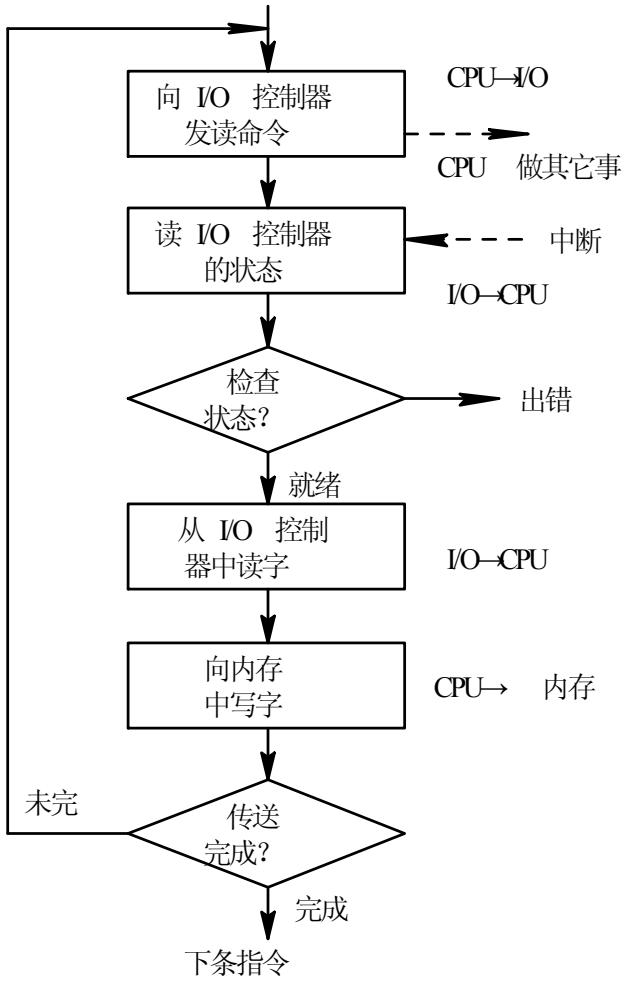
- 查询方式：CPU需花代价不断查询I/O状态
- CPU资源浪费极大
- 例：

  99.9ms+0.1ms=100ms

  在程序I/O方式中99.9在忙等

CPU完全干预I/O

6.2.2 中断I/O方式



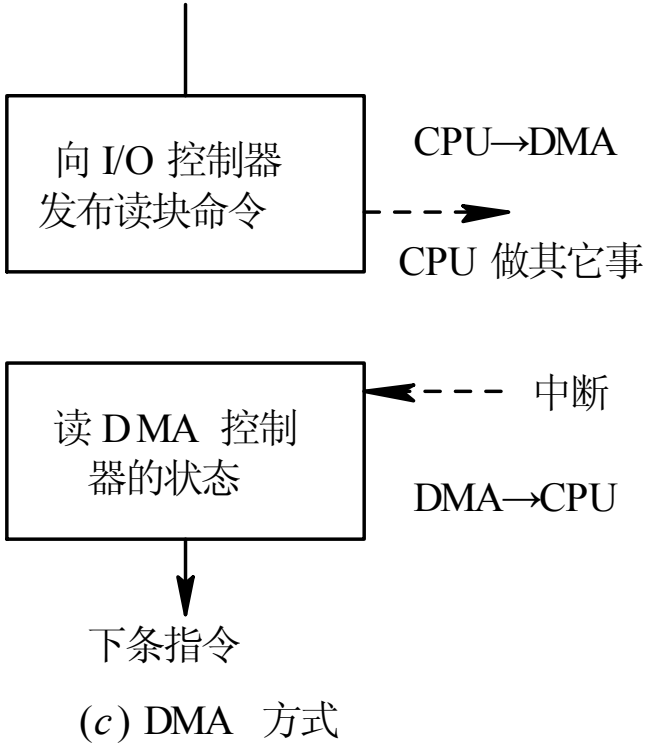
(*b*) 中断驱动方式

6.2 I/O控制方式

- 向I/O发命令——返回——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

CPU以字节为单位干预I/O

6.2.3 DMA方式

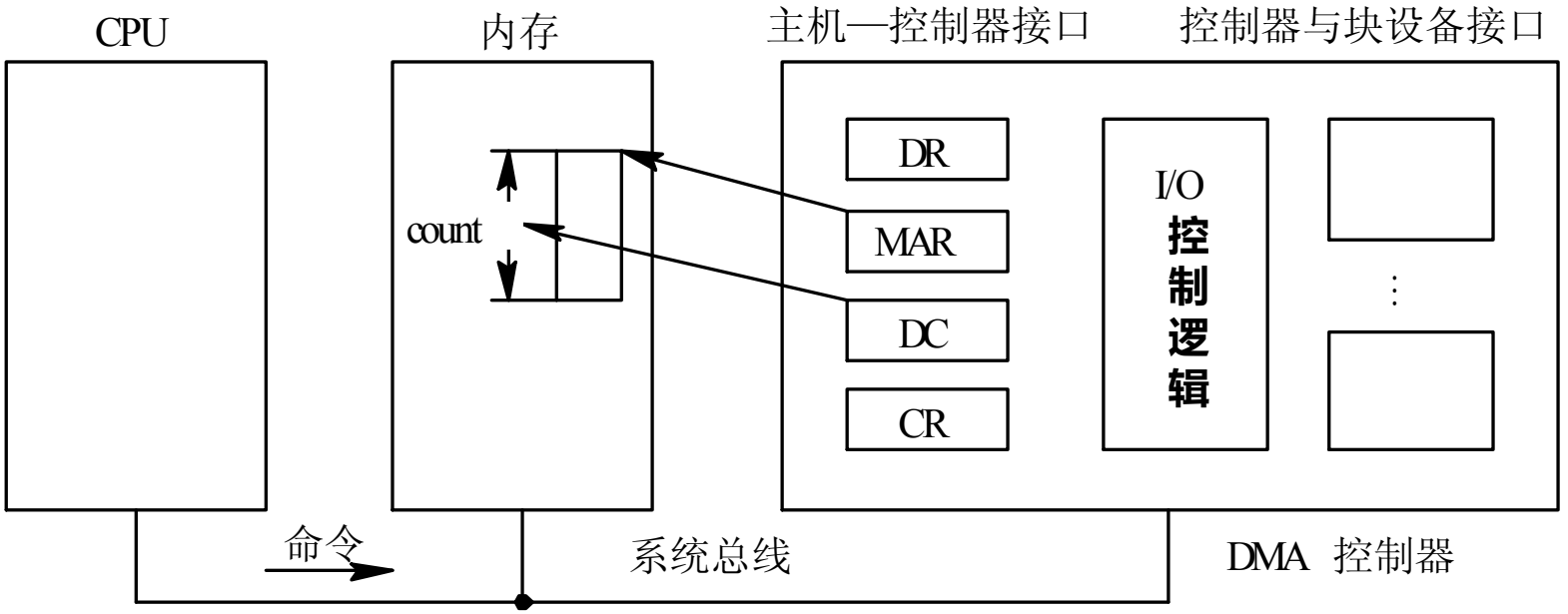


(c) DMA 方式

- 由DMA控制器直接控制总线传递数据块。DMA控制器完成从I/O——内存。
- 主要用在块设备中

CPU以块为单位干预I/O

DMA控制器的组成

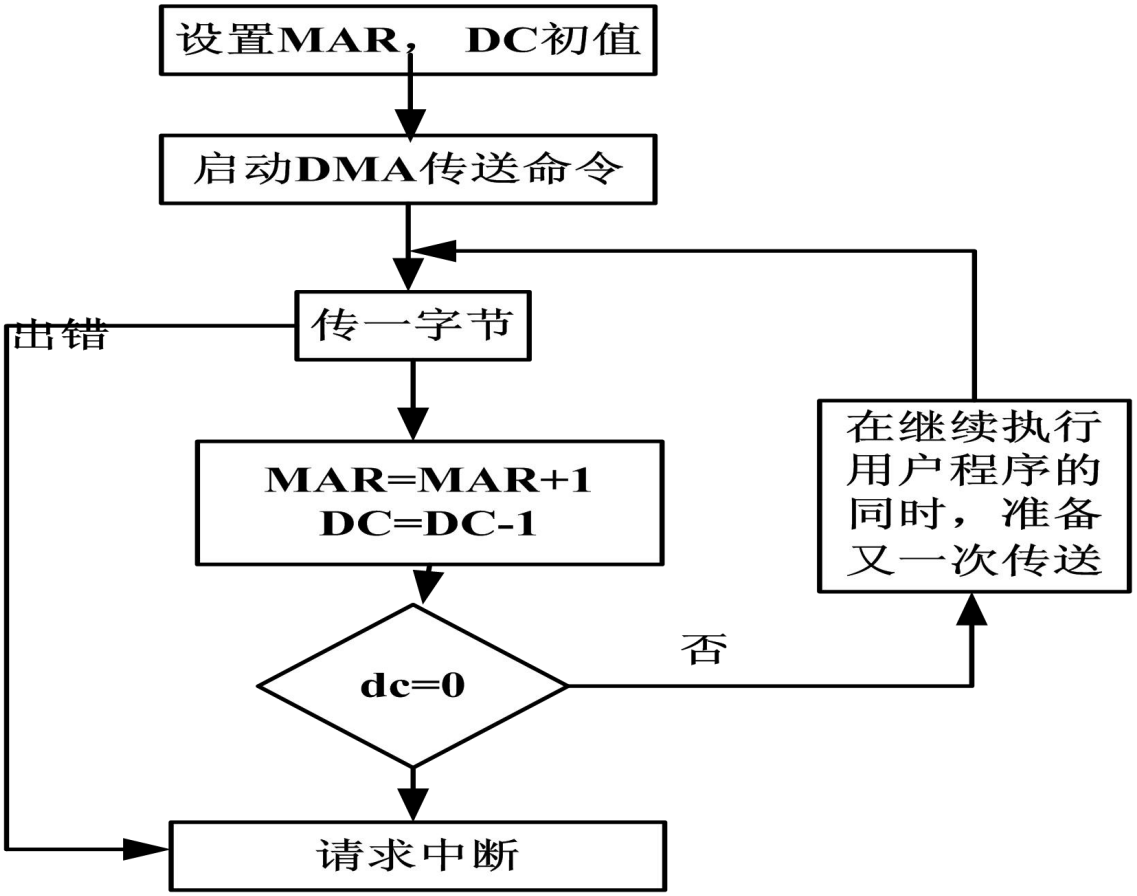


一组寄存器+控制逻辑。

CR（命令/状态）； DR（数据）； MAR（内存地址）；

DC（计数）

DMA的工作过程


设置MAR， DC初值
启动DMA传送命令
传一字节
出错
MAR=MAR+1
DC=DC-1
dc=0
否
在继续执行用户程序的同时，准备又一次传送
请求中断

6.2.4 通道方式

- DMA方式：对需多离散块的读取仍需要多次中断。
- 通道方式：CPU只需给①通道程序首址②要访问的I/O设备后，通道程序就可完成一组块操作。

操作	P	Record	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

通道程序

CPU以一组数据块为单位干预I/O

6.3 缓冲管理

目的：**组织管理、分配、释放buffer。**

6.3.1 引入

1. 缓和CPU和I/O设备间速度不匹配的矛盾。

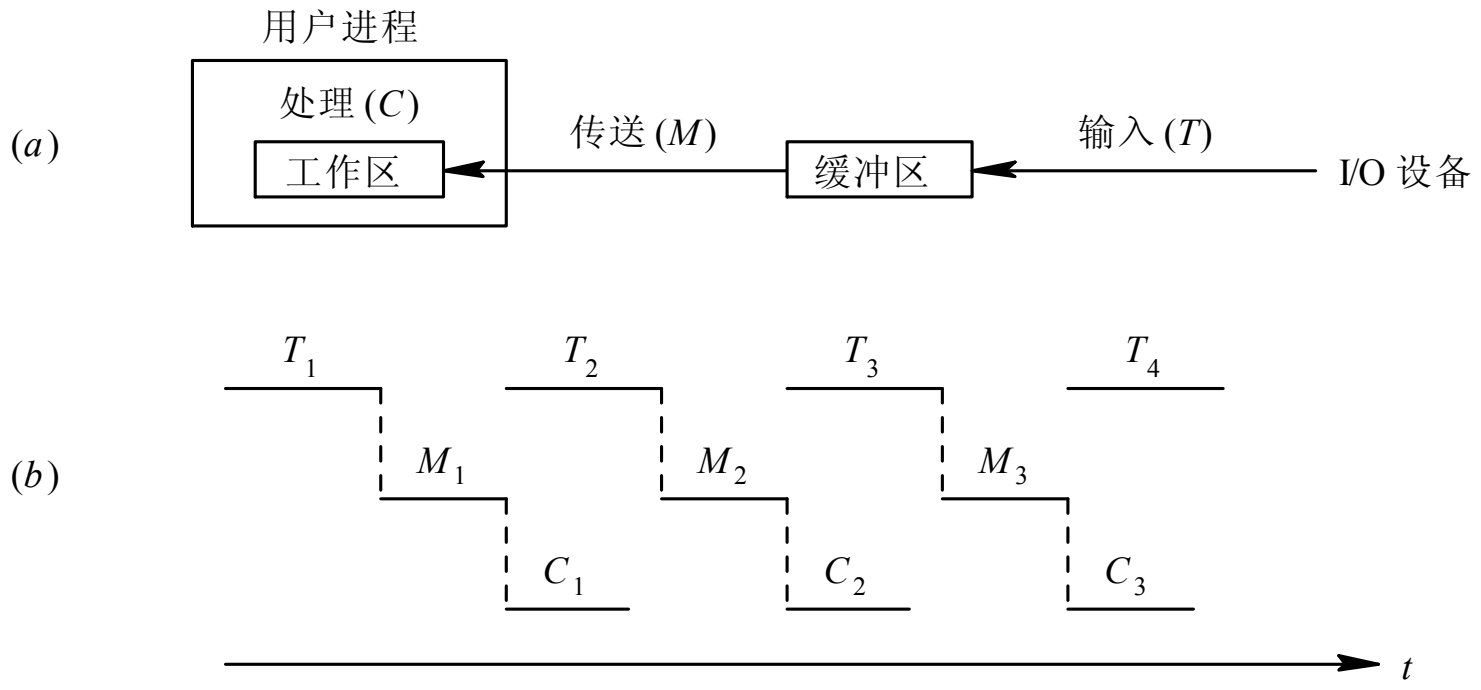
   如：计算——打印buffer——打印

2. 减少对CPU的中断频率

   如：buffer越大，“buffer满”信号发生频率越低。

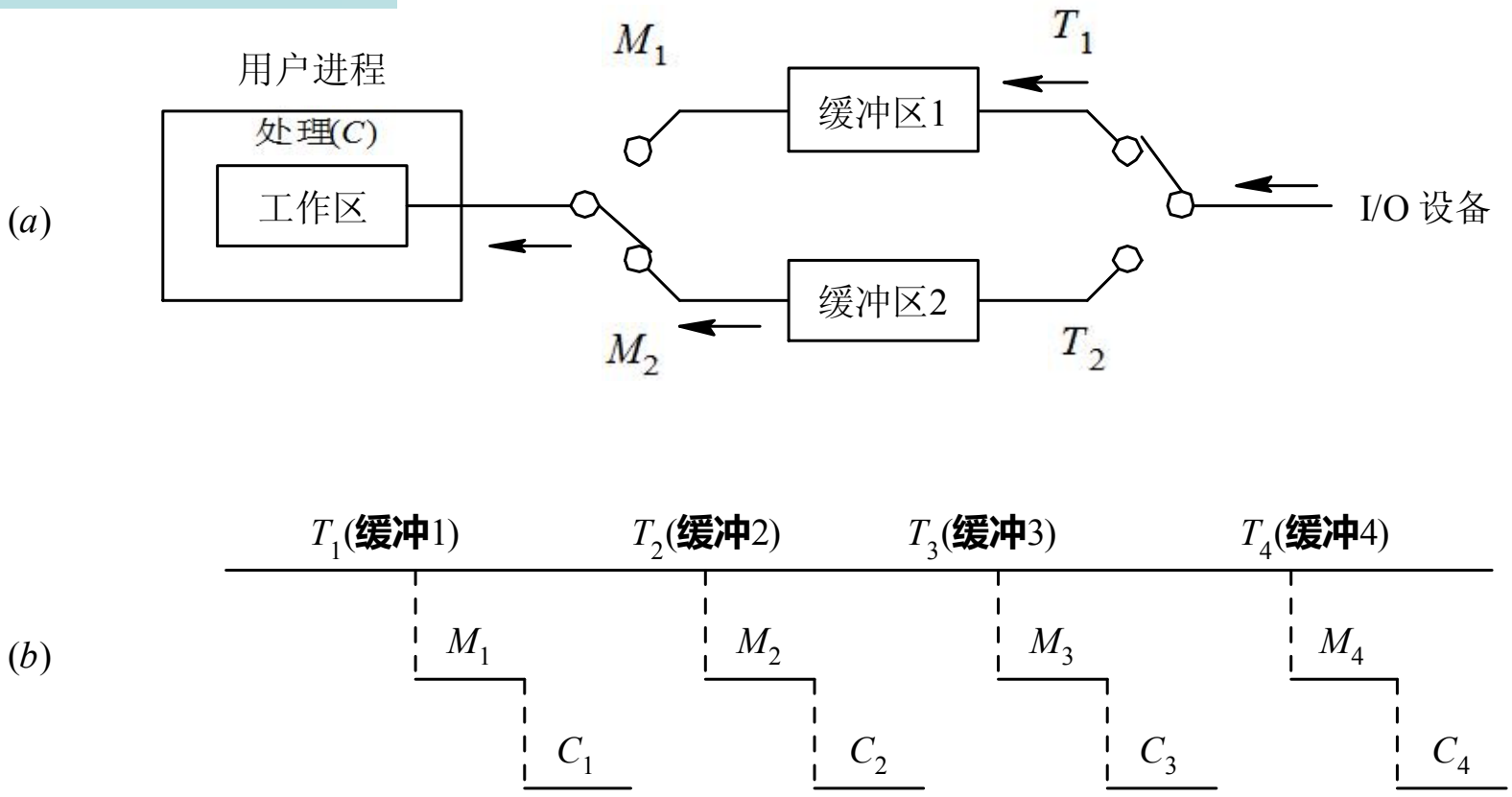
3. 提高CPU和I/O并行性

6.3.2 单缓冲



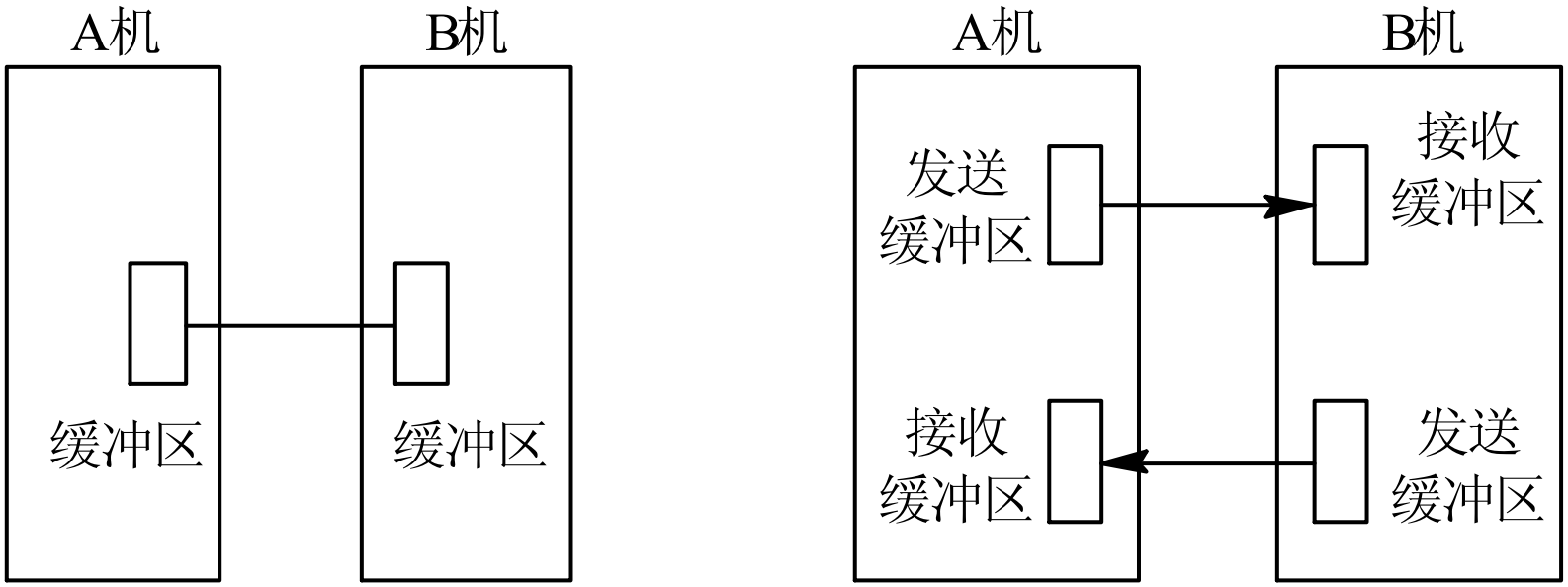
- 由于C和T可并行，M和C或M和T不能并行，因此处理一块数据时间：Max(C,T)+M
- 用户进程何时阻塞？

6.3.3 双缓冲



- 效率有所提高，且进一步平滑了传输峰值。
- 系统处理一块数据的时间约为：MAX(C, T)
- 收发可双向同时传送。

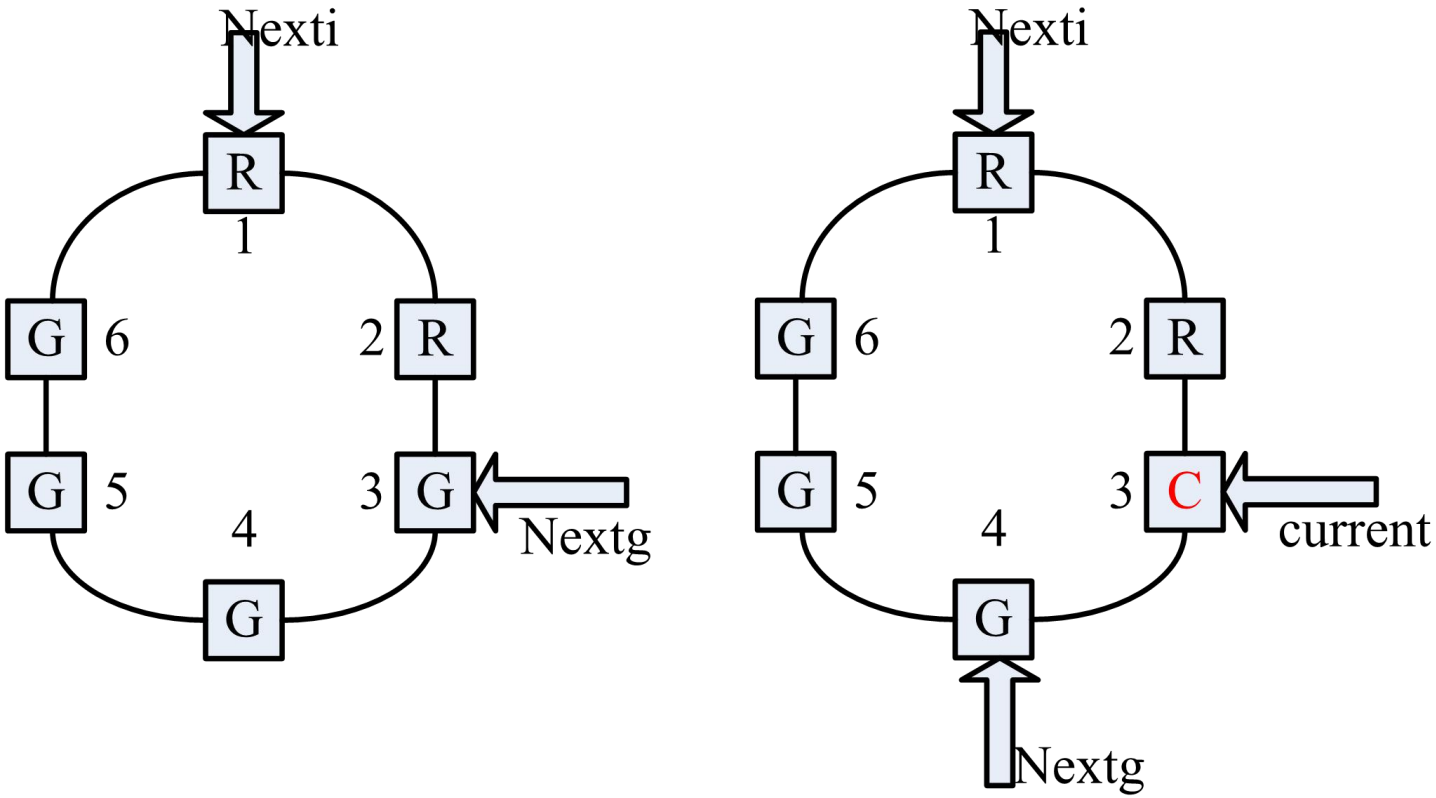
6.3 缓冲管理



(*a*) 单缓冲　　(*b*) 双缓冲

图 5-13 双机通信时缓冲区的设置

6.3.4 循环多缓冲



缓冲区的3种类型：

R: 空缓冲； G: 满缓冲； C: 当前缓冲

6.3.4 循环多缓冲

循环多缓冲的使用

- nextg：指示下一个应取数据的buf
- nexti：指示下一个空buf.
- Getbuf：
  - 取nextg对应缓冲区提供使用，将Nextg置为空，Nextg=（Nextg+1）Mod N
  - 将Nexti对应缓冲区提供使用，将Nexti置为满，Nexti=（Nexti+1）Mod N
- Releasebuf:
  - 若C满，则改为G；
  - 若C空，则改为R；

6.3.4 循环多缓冲

循环多缓冲的同步问题

- Nexti追上Nextg：

  输入速度>输出速度，全部buf满，这时输入进程阻塞。

- Nextg追上Nexti：

  输入速度<输出速度，全部buf空，这时输出进程阻塞。

6.3.5 缓冲池

缓冲池：系统提供的公用缓冲 .

1. 组成：

3个队列：

空缓冲队列emq
输入队列inq
输出队列outq

4个工作缓冲区：

hin： 收容输入数据
sin： 提取输入数据
hout： 收容输出数据
sout： 提取输出数据

2. Getbuf和Putbuf过程

Getbuf(type)

```
Begin
  wait(RS(type));          ①
  wait(MS(type));
  B(number):=takebuf(type);
  signal(MS(type));
End
```

Putbuf(type)

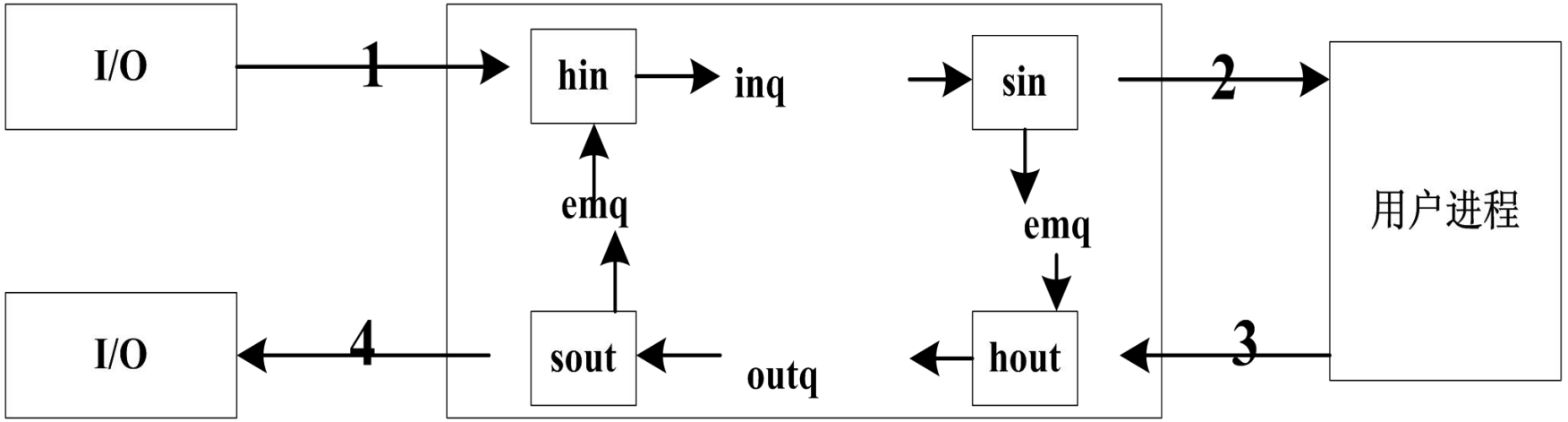
```
Begin
  wait(MS(type));
  addbuf(type,number);
  signal(MS(type));
  signal(RS(type));        ②
End
```

①②出现在不同进程中，实现了进程的同步

RS—缓冲池资源信号量，MS—缓冲池的互斥信号量

3. 四种工作方式

1. 收容输入；2. 提取输入；3. 收容输出；4. 提取输出



1. hin=getbuf(emq)； putbuf(inq,hin)

2. sin=getbuf(inq)； 计算； putbuf(emq,sin)

3. hout=getbuf(emq)； putbuf(outq， hout)

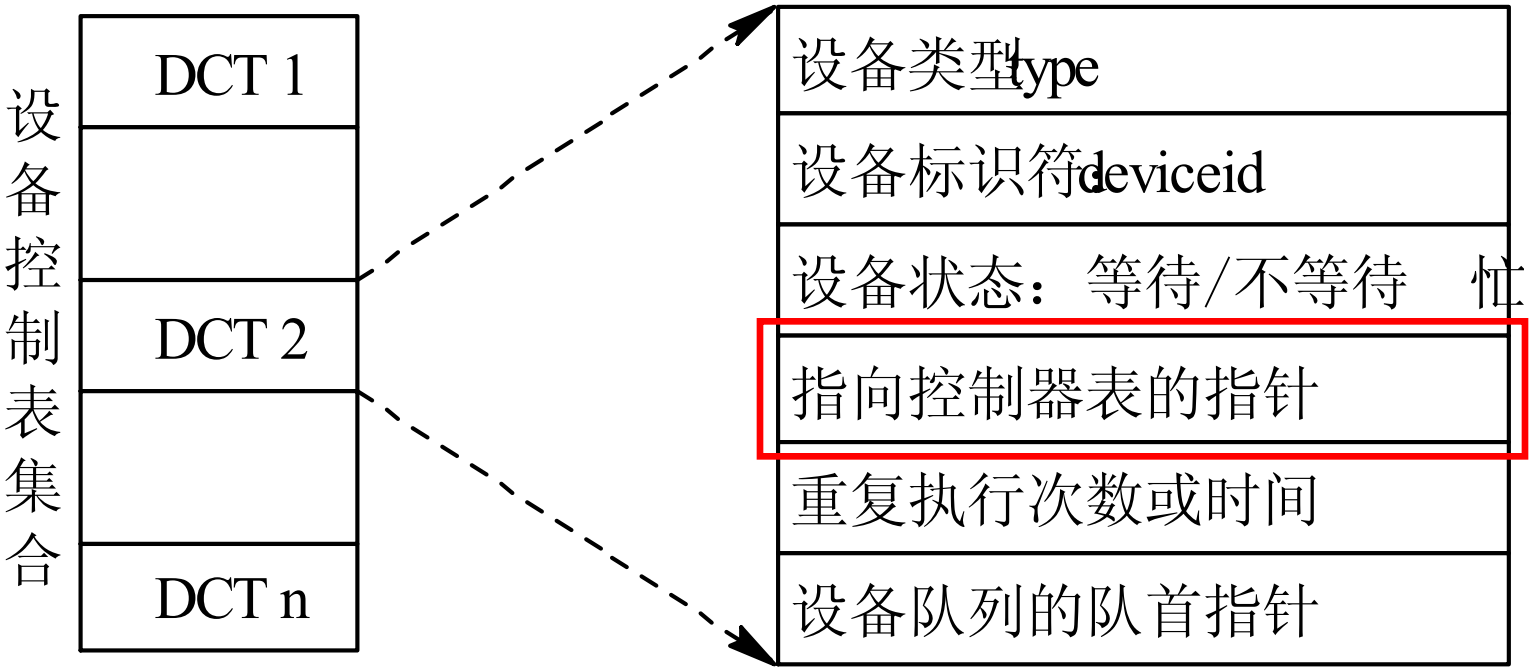
4. sout=getbuf(outq)；输出； putbuf(emq,sout)

6.4 设备分配

包括：对设备、设备控制器、通道的分配

6.4.1 数据结构

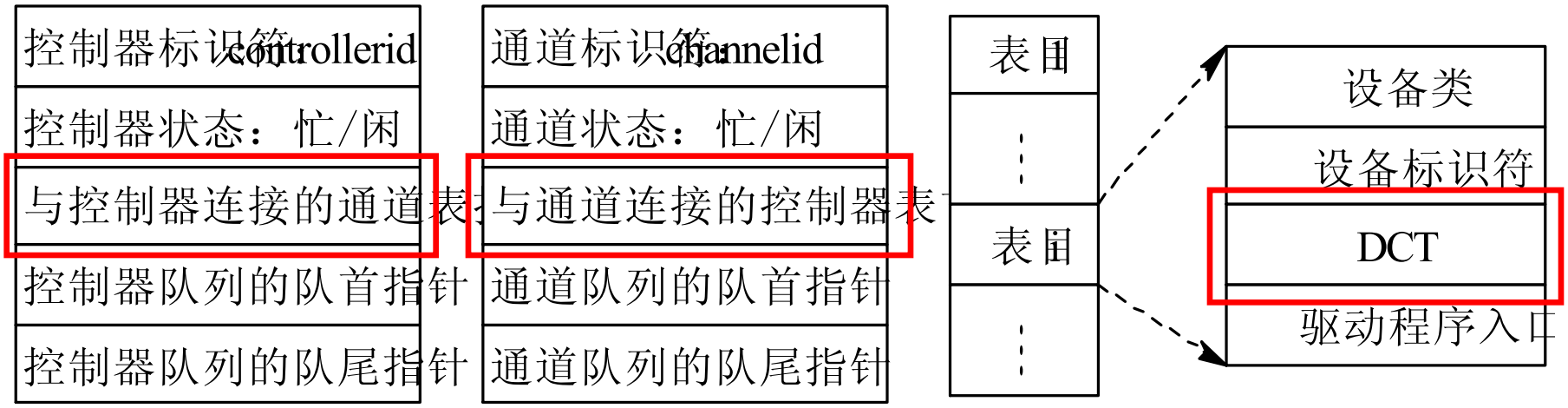
1. 设备控制表DCT：



6.4 设备分配

6.4.1 数据结构

2. 控制器控制表、 通道控制表和系统设备表



SDT:记录了系统中全部设备及其驱动程序地址。

6.4.2 设备分配应考虑的若干因素

1. 设备的固有属性：
   - 共享+虚拟：注意调度的合理性；
   - 独享：排它性分配，控制不好可能死锁。
2. 分配算法：
   - （1）FIFO；（2）优先权。
3. 安全性：
   - 安全分配(同步)：每进程获得一I/O后，即block，直到其I/O完成。
     - 即打破了死锁条件。
     - 缺点：CPU、I/O对该进程是串行，进程进展缓慢。
   - 不安全分配（异步）：需进行安全性检查，进程执行效率高。

6.4.3 设备独立性

1. 概念：

* 即设备无关性，指应用程序独立于具体使用的物理设备。
* 逻辑设备和物理设备
* 逻辑设备表（LUT）：

逻辑设备	物理设备	Driver入口

* 分配流程：进程给出逻辑名——通过LUT得到物理设备及 其driver入口。

6.4.3 设备独立性

❖ **优点：**

- 设备分配更灵活；

  逻辑设备和物理设备间可以是多--多的映射关系。提高了物理设备的共享性，以及使用的灵活性。如：

  - ➢某逻辑名可对应这一类设备，提高均衡性与容错性。
  - ➢几个逻辑名可对应某一个设备，提高共享性。

- 易于实现I/O重定向。

  不变程序，只需改变LUT表的映射关系。

6.4.3 设备独立性

2. 设备独立性软件

- 执行所有设备的公有操作
  - 分配回收
  - 名字映射
  - 保护
  - 缓冲
  - 差错控制
- 向用户层软件提供统一接口
  - read、write

6.4.3 设备独立性

3. 名字映射

* LUT的生成
  - 在用户进程第一次请求设备时完成映射并在LUT中生成相应项
* LUT的配置
  - （1）整个系统一张LUT表：
    - 要求：逻辑名不重复，（一般用于单用户系统）

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

(a)

逻辑设备名	系统设备表指针
/dev/tty	3
/dev/printer	5
⋮	⋮

(b)

6.4.4 独占设备的分配程序

1. 基本的设备分配程序

当某进程提出I/O请求后，设备分配程序按下述步骤进行设备分配：

1）分配设备

2）分配控制器

3）分配通道

2. 设备分配程序的改进

1）增加设备的独立性

2）考虑多通路情况

6.4.5 SPOOLing技术

1. 脱机I/O

在20世纪50年代，为了缓和CPU的高速性与I/O设备低速性间的矛盾，而引入了脱机输入、脱机输出技术。该技术是利用专门的外围控制机，先将低速I/O设备上的数据，传送到高速磁盘上，或者相反。

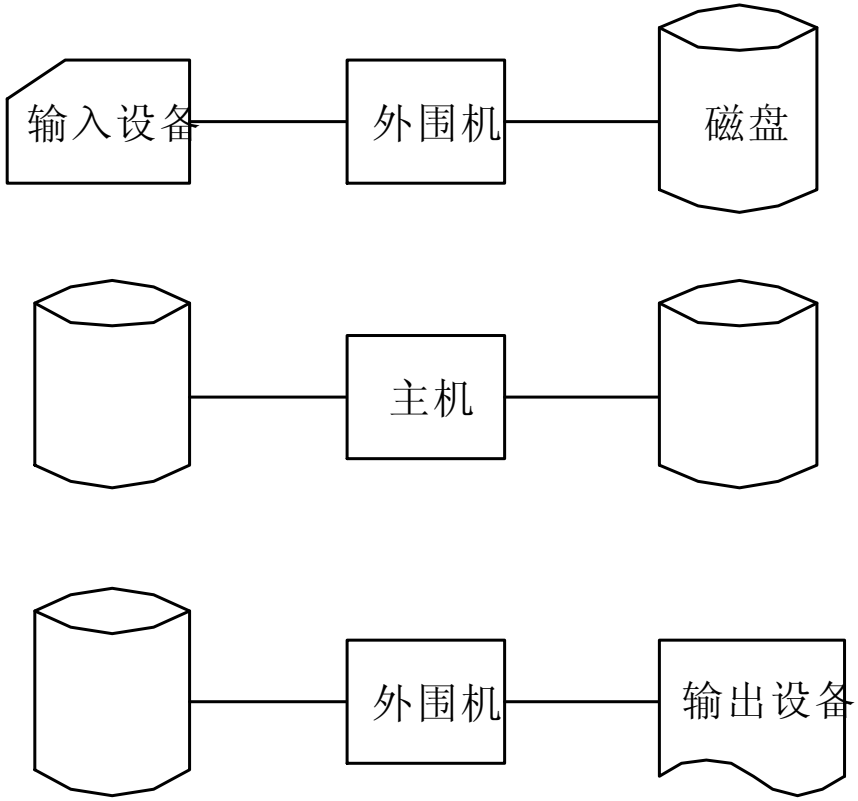


图1-3 脱机I/O示意图

6.4.5 SPOOLing技术

2. 什么是SPOOLing

当系统中引入了多道程序技术后，完全可以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，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，便可在主机的直接控制下，实现以前的脱机输入、输出功能。

此时的外围操作与CPU对数据的处理同时进行，我们把这种在联机情况下实现的同时外围操作的技术，称为SPOOLing技术(Simultaneaus Periphernal Operating OnLine)，或称假脱机技术。

6.4.5 SPOOLing技术

3. SPOOLing系统的组成

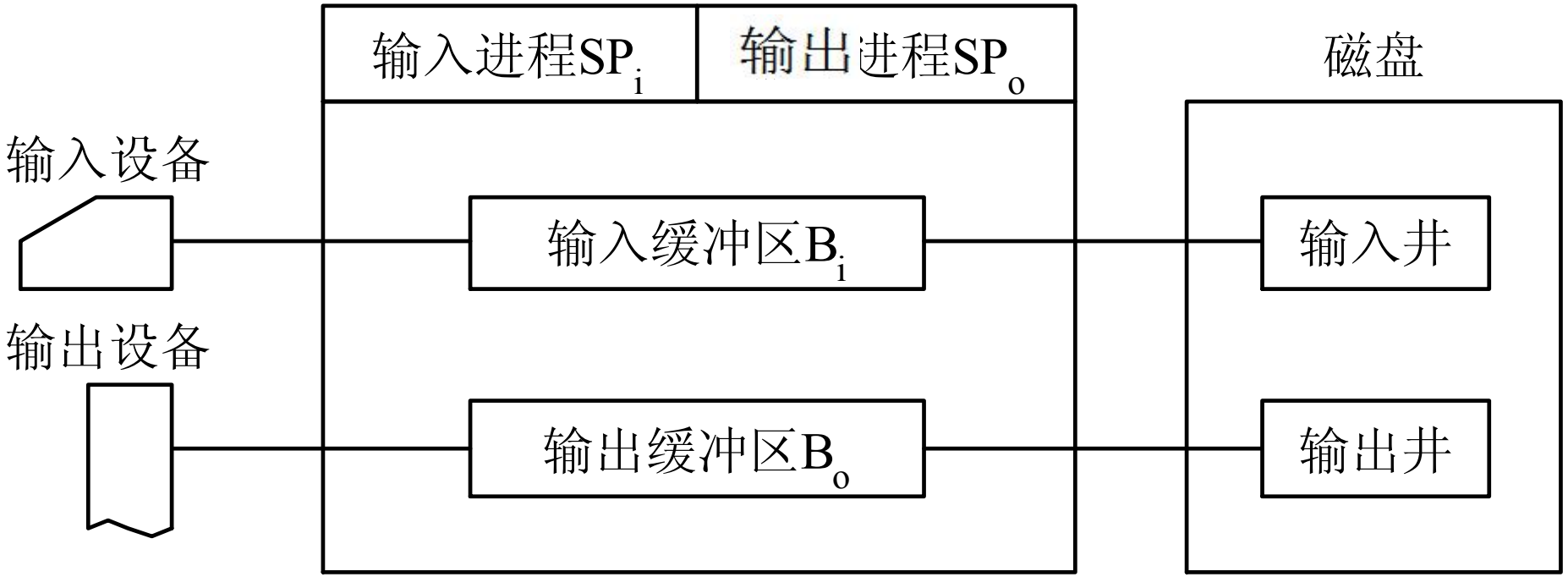


图 6-21 SPOOLing系统的组成

6.4.5 SPOOLing技术

4. 共享打印机

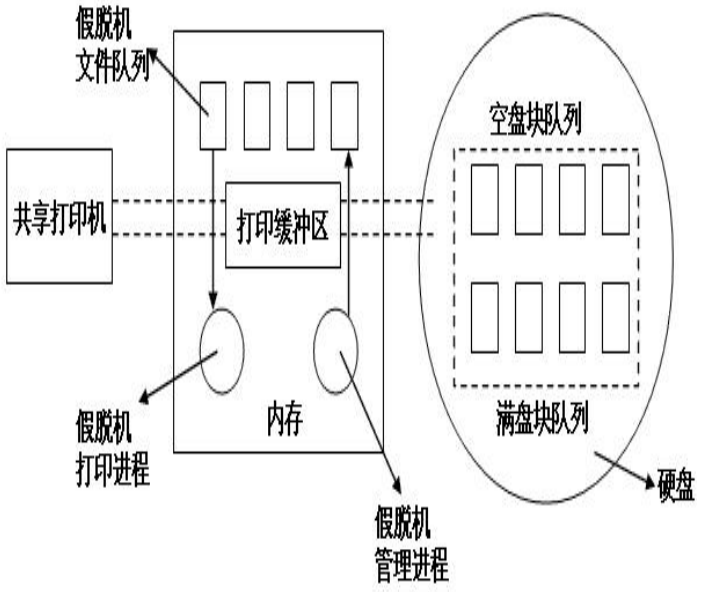
打印机属于独占设备。利用假脱机技术，可将它改造为一台可供多个用户共享的打印设备，从而提高设备的利用率，也方便了用户。

假脱机打印系统的构成

(1)磁盘缓冲区：在磁盘上开辟的一个存储空间，用于暂存用户程序的输出数据；

(2)打印缓冲区：为了缓和CPU和磁盘之间速度不匹配的矛盾，在内存中要有一个打印输出缓冲区，用于暂存从磁盘缓冲区送来的数据；

(3)假脱机管理进程和假脱机打印进程：由假脱机管理进程为每个要求打印的用户数据，建立一个假脱机文件，并把它放入假脱机文件队列中，由假脱机打印进程，依次对队列中的文件进行打印。



6.4.5 SPOOLing技术

5. 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。

6.6.1 设备驱动程序的功能和特点

1. 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，将命令中的抽象要求转换为具体要求，例，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，如果设备空闲，便立即启动I/O设备去完成指定的I/O操作；如果设备处于忙碌状态，则将请求者的请求块挂在设备队列上等待。

(4)及时响应由控制器或通道发来的中断请求，并根据其中断类型调用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，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。

6.6.1 设备驱动程序的功能和特点

2. 设备驱动程序的特点

（1）驱动程序主要是指在请求I/O的进程与设备控制器之间的一个通信和转换程序。

（2）驱动程序与设备控制器和I/O设备的硬件特性紧密相关，因而对不同类型的设备应配置不同的驱动程序。

（3）驱动程序与I/O设备所采用的I/O控制方式紧密相关。

（4）由于驱动程序与硬件紧密相关，因而其中的一部分必须用汇编语言书写。

6.6.1 设备驱动程序的功能和特点

3. 设备处理方式

(1) 为每一类设备设置一个进程，专门用于执行这类设备的I/O操作 .

(2) 在整个系统中设置一个I/O进程，专门用于执行系统中所有各类设备的I/O操作。

(3) 不设置专门的设备处理进程，而只为各类设备设置相应的设备处理程序(模块)， 供用户进程或系统进程调用。

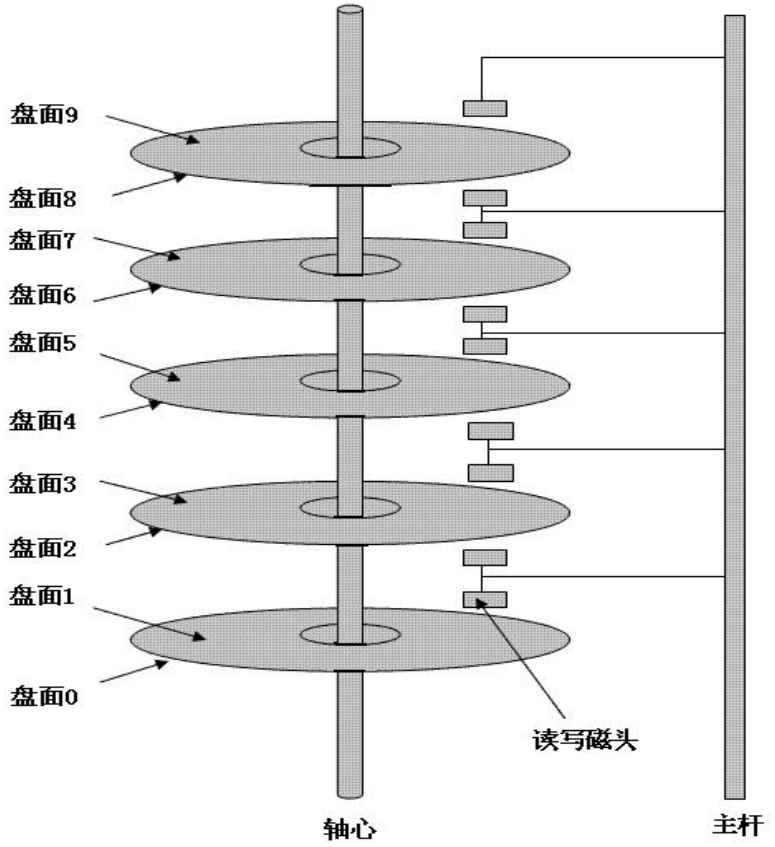
6.6.2 设备驱动程序的处理过程

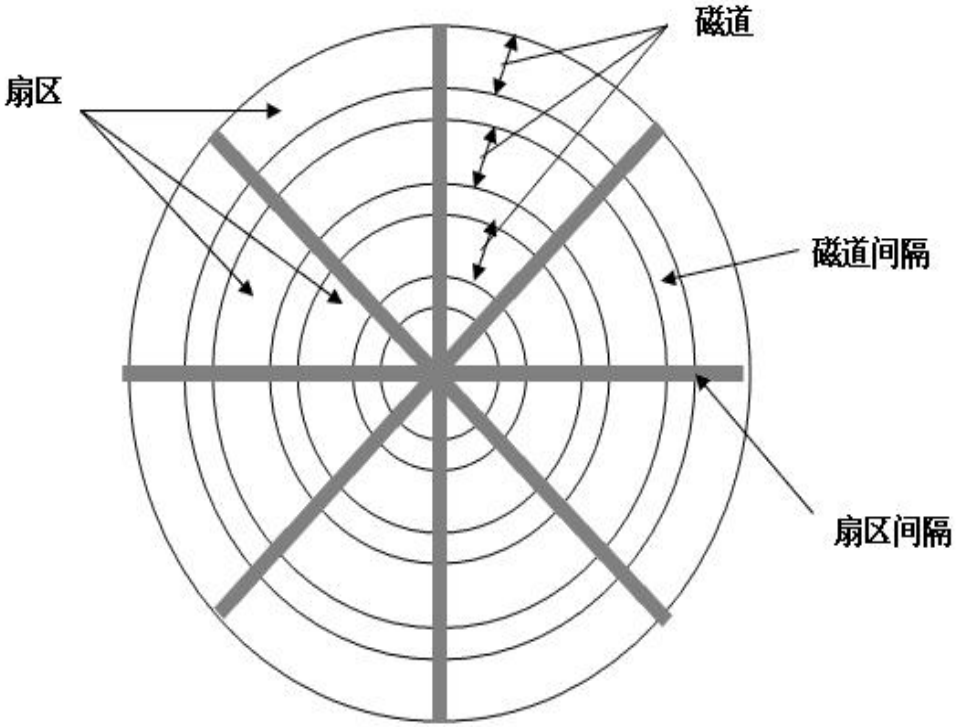
1. 将抽象要求转换为具体要求
2. 检查I/O请求的合法性
3. 读出和检查设备的状态
4. 传送必要的参数
6. 工作方式的设置
6. 启动I/O设备

6.6 磁盘存储器管理

6.6.1 磁盘性能简述

1. 数据的组织和格式





6.6.1 磁盘性能简述

1. 数据的组织和格式

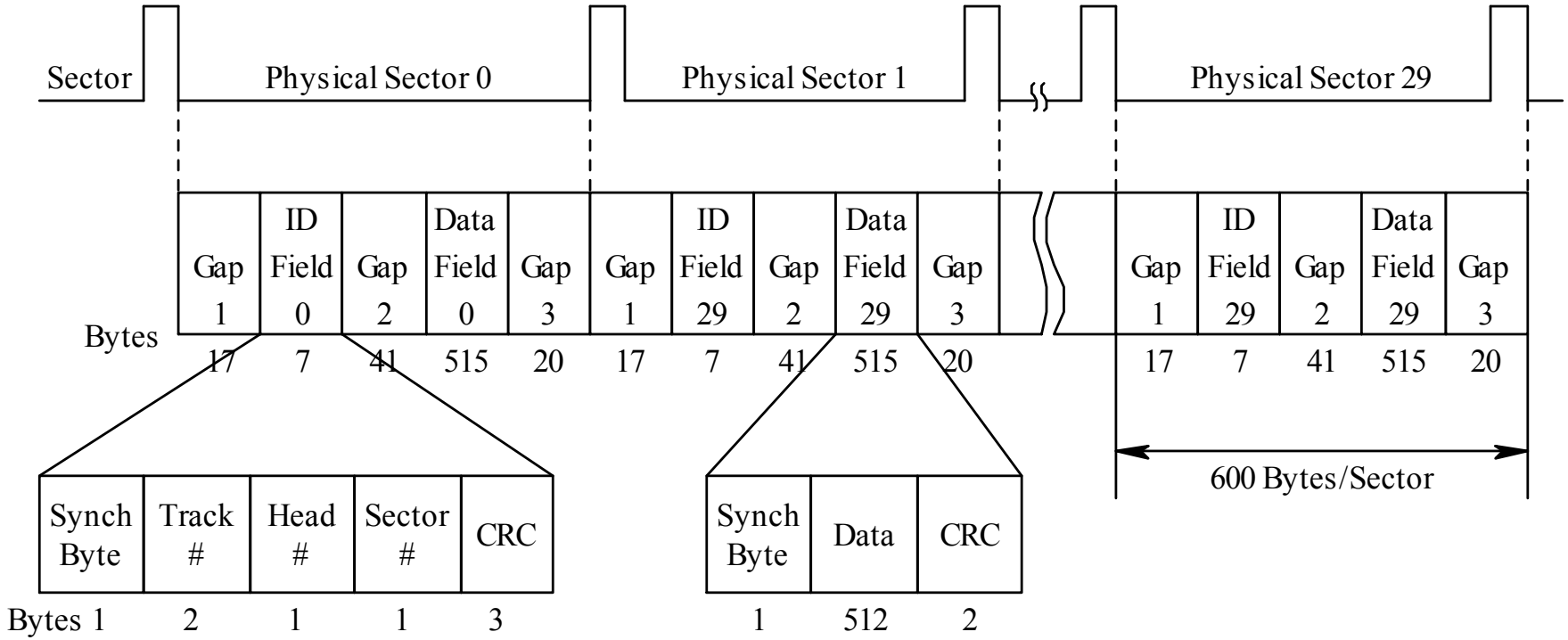


图 5-22 磁盘的格式化

6.6 磁盘存储器管理

6.6.1 磁盘性能简述

2. 磁盘的类型

1） 固定头磁盘

这种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各磁道，并进行并行读/写，有效地提高了磁盘的I/O速度。这种结构的磁盘主要用于大容量磁盘上。

2） 移动头磁盘

每一个盘面仅配有一个磁头，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。可见，移动磁头仅能以串行方式读/写，致使其I/O速度较慢；但由于其结构简单， 故仍广泛应用于中小型磁盘设备中。

6.6 磁盘存储器管理

6.6.1 磁盘性能简述

3. 磁盘的访问时间

1）寻道时间 T_s

这是指把磁臂（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是一常数，与磁盘驱动器的速度有关，对一般磁盘，$m=0.2$；对高速磁盘，$m \leq 0.1$，磁臂的启动时间约为$2ms$。

这样，对一般的温盘，其寻道时间将随寻道距离的增加而增大， 大体上是$5\sim30$ ms。

6.6.1 磁盘性能简述

3. 磁盘的访问时间

2) 旋转延迟时间T_τ

这是指定扇区移动到磁头下面所经历的时间。对于硬盘，典型的旋转速度大多为5400 r/min，每转需时11.1ms，平均旋转延迟时间T_τ为6.55ms；对于软盘，其旋转速度为300 r/min或600 r/min，这样，平均T_τ为50~100 ms。

6.6.1 磁盘性能简述

3. 磁盘的访问时间

3) 传输时间T_t

这是指把**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，当一次读/写的字节数相当于半条磁道上的字节数时，T_t与T_τ相同，因此，可将访问时间T_a表示为：

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

6.6.2 磁盘调度

1.先来先服务FCFS

(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

2.最短寻道时间优先SSTF

(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

6.6.2 磁盘调度

3.扫描（SCAN）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	

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	

防止老进程出现“饥饿”现象

6.6.2 磁盘调度

5. NStepSCAN算法

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。

6. FSCAN算法

FSCAN算法实质上是N步SCAN算法的简化，即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。

6.6.3 磁盘高速缓存(Disk Cache)

1. 磁盘高速缓存的形式

- 利用内存中的存储空间，来暂存从磁盘中读出的一系列盘块中的信息。
- 高速缓存逻辑上属于磁盘， 物理上是内存。
- 高速缓存在内存中可分成两种形式:
  - 第一种是在内存中开辟一个单独的存储空间来作为磁盘高速缓存，其大小是固定的，不会受应用程序多少的影响;
  - 第二种是把所有未利用的内存空间变为一个缓冲池，供请求分页系统和磁盘I/O时(作为磁盘高速缓存)共享。此时高速缓存的大小，显然不再是固定的。

6.6.3 磁盘高速缓存(Disk Cache)

2. 数据交付方式

系统可以采取两种方式， 将数据交付给请求进程：

(1) 数据交付。这是直接将高速缓存中的数据， 传送到请求者进程的内存工作区中。

(2) 指针交付。只将指向高速缓存中某区域的指针， 交付给请求者进程。

后一种方式由于所传送的数据量少，因而节省了数据从磁盘高速缓存存储空间到进程的内存工作区的时间。

6.6.3 磁盘高速缓存(Disk Cache)

3. 置换算法

由于请求分页系统与磁盘高速缓存的工作情况不同，因而在置换算法中所应考虑的问题也有所差异。高速缓存的置换算法除了考虑到最近最久未使用这一原则外， 还考虑了以下几点：

(1) 访问频率。

(2) 可预见性。

(3) 数据的一致性。

6.6.3 磁盘高速缓存(Disk Cache)

4. 周期性地写回磁盘

- UNIX系统

专门增设了一个修改(update)程序，使之在后台运行，该程序周期性地调用一个系统调用SYNC。该调用的主要功能是强制性地将所有在高速缓存中已修改的盘块数据写回磁盘。一般是把两次调用SYNC的时间间隔定为30s。这样，因系统故障所造成的工作损失不会超过30s的劳动量。

- MS-DOS

只要高速缓存中的某盘块数据被修改，便立即将它写回磁盘，并将这种高速缓存称为“写穿透、高速缓存”(write-through cache)。 MS-DOS所采用的写回方式，几乎不会造成数据的丢失， 但须频繁地启动磁盘。