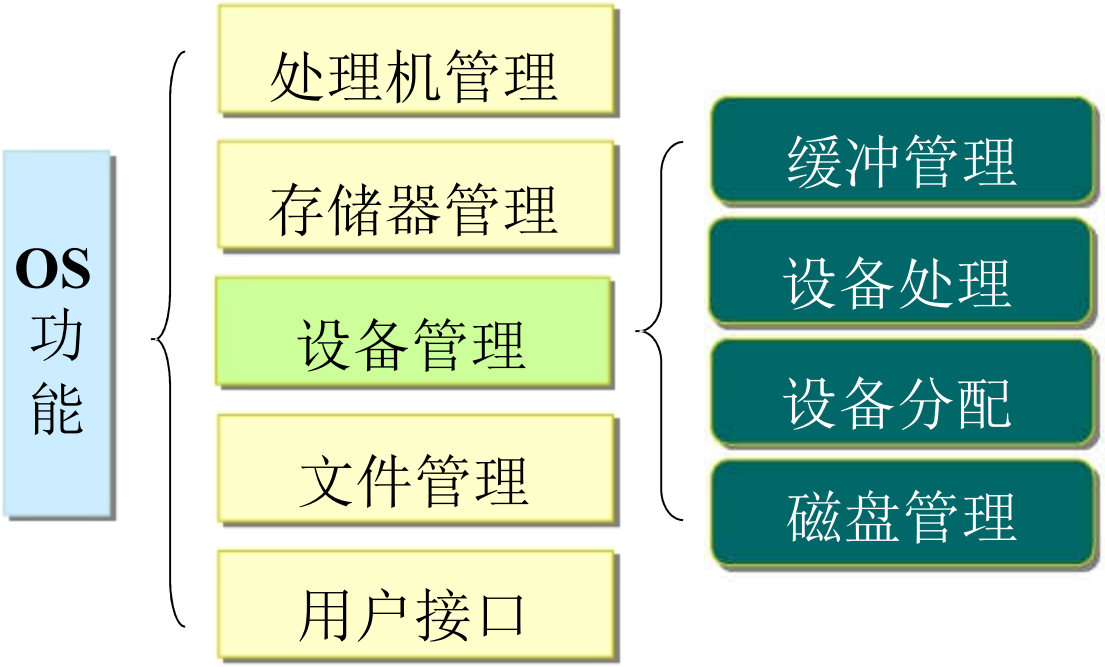
第六章 设备管理

本章内容所处位置



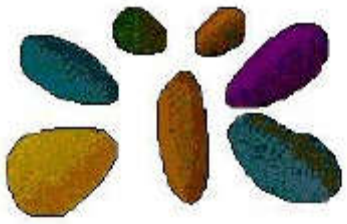
本章主要内容

- 6.1 I/O系统的功能、模型和接口
- 6.2 I/O设备和设备控制器
- 6.3 中断机构和中断处理程序
- 6.4 设备驱动程序
- 6.5 与设备无关的I/O软件
- 6.6 用户层的I/O软件
- 6.7 缓冲区管理
- 6.8 磁盘存储器的性能和调度

6.1 I/O系统的功能、模型和接口

I/O系统是指用于实现数据输入、输出及数据存储的系统。I/O系统是OS中最繁杂且与硬件最紧密相关的部分；其最主要的任务是完成用户提出的I/O请求、提高I/O速率以及提高设备的利用率。

- 6.1.1 I/O系统的功能
- 6.1.2 I/O系统的层次结构
- 6.1.3 I/O系统接口



6.1 I/O系统的功能、模型和接口

❖ 6.1.1 I/O系统的基本功能

1、隐藏物理设备的细节：为用户提供命令

I/O设备类型多、差异大（速度、传输方向、数据表示形式、可靠性、控制指令与参数等）

2、与设备的无关性：为用户提供逻辑设备名

应用逻辑设备名，使得用户在编程序时所使用的设备与实际设备无关，也提高了OS的可移植性和易适应性

3、提高处理机和I/O设备的利用率

引入设备控制器、I/O通道；采用缓冲技术

6.1 I/O系统的功能、模型和接口

4、对I/O设备进行控制：驱动程序的功能

四种控制方式

a.轮询可编程I /O：

--CPU做全部工作

--轮询，忙等待

--适用于嵌入式系统，WHY

```
copy_from_user(buffer, p, count);
for (i = 0; i < count; i++) {
        while (*printer_status_reg != READY) ;
        *printer_data_register = p[i];
}
return_to_user( );
```

```
/* p is the kernel buffer */
/* loop on every character */
/* loop until ready */
/* output one character */
```

6.1 I/O系统的功能、模型和接口

4、对I/O设备进行控制：驱动程序的功能

四种控制方式

b. 中断可编程I/O：传达一次信息需要一次中断；

--阻塞请求I/O的进程

--CPU进行调度直到设备产生中断

--设备处理，I/O中断方式通知CPU

```
copy_from_user(buffer, p, count);
enable_interrupts( );
while (*printer_status_reg != READY) ;
*printer_data_register = p[0];
scheduler( );
```

(a)

```
if (count == 0) {
    unblock_user( );
} else {
    *printer_data_register = p[i];
    count = count – 1;
    i = i + 1;
}
acknowledge_interrupt( );
return_from_interrupt( );
```

(b)

6.1 I/O系统的功能、模型和接口

4、对I/O设备进行控制：驱动程序的功能

四种控制方式

c. 使用DMA的I /O ：

--本质上是程序控制I/O

--减少中断：从每个字符一次到每个缓冲区（数据块）一次

--不适用：DMA太慢或CPU空闲

```
copy_from_user(buffer, p, count);
set_up_DMA_controller( );
scheduler( );
```

(a)

```
acknowledge_interrupt( );
unblock_user( );
return_from_interrupt( );
```

(b)

6.1 I/O系统的功能、模型和接口

4、对I/O设备进行控制：驱动程序的功能

四种控制方式

d. 通道**I /O**：

--通道是简易CPU

--CPU、通道与I/O设备并行工作，I/O无需CPU干预

--减少中断：若干个缓冲区（数据块）的一次存取

--过程：CPU指令--通道（简易CPU）---控制器（接收及解析命令）---设备（执行命令）

6.1 I/O系统的功能、模型和接口

5、确保对设备的正确共享

独占设备：磁带机、打印机

共享设备：磁盘

6、错误处理

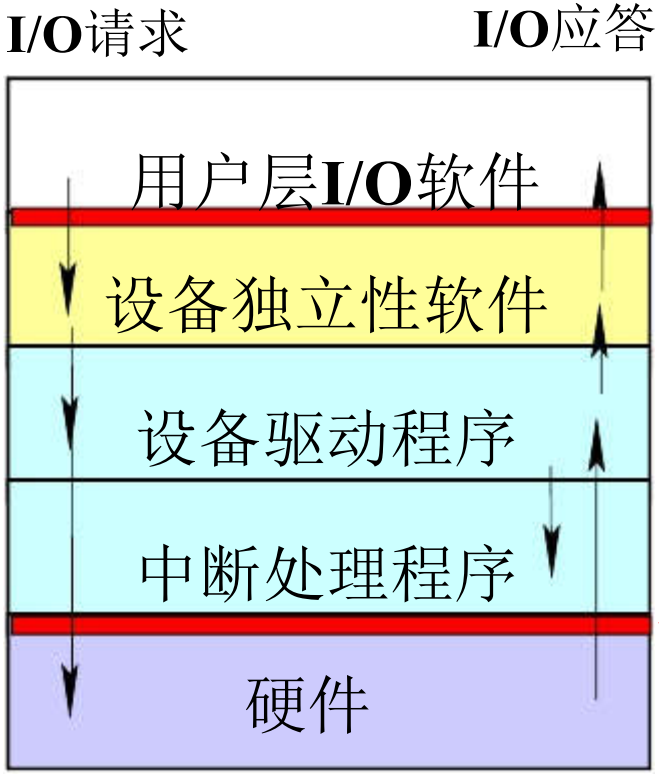
临时性错误：重试来纠正

持久性错误：向上层报告

6.1 I/O系统的功能、模型和接口

❖ 6.1.2 I/O系统的层次结构和模型

1、I/O软件层次结构



(1) 用户交互接口：使用操作命令（如write）

I/O系统接口：向上层（用户、其它软件）提供命令

(2) 提供用户程序与设备的统一接口(如设备命名)

(3) 解析操作命令并存入设备控制器，驱动设备工作

(4) 中断处理

软硬件接口：逻辑指令---物理指令 的转换

(5) 执行I/O操作

图6-1 I/O系统的（软件）层次结构及功能

6.1 I/O系统的功能、模型和接口

如何理解接口？

（1）举例1：螺母相当于接口，连接螺丝（直径尺寸一样）和部件；

（2）举例2：USB接口，可以接不同设备（如相机，手机等），

可以理解为实现规范；

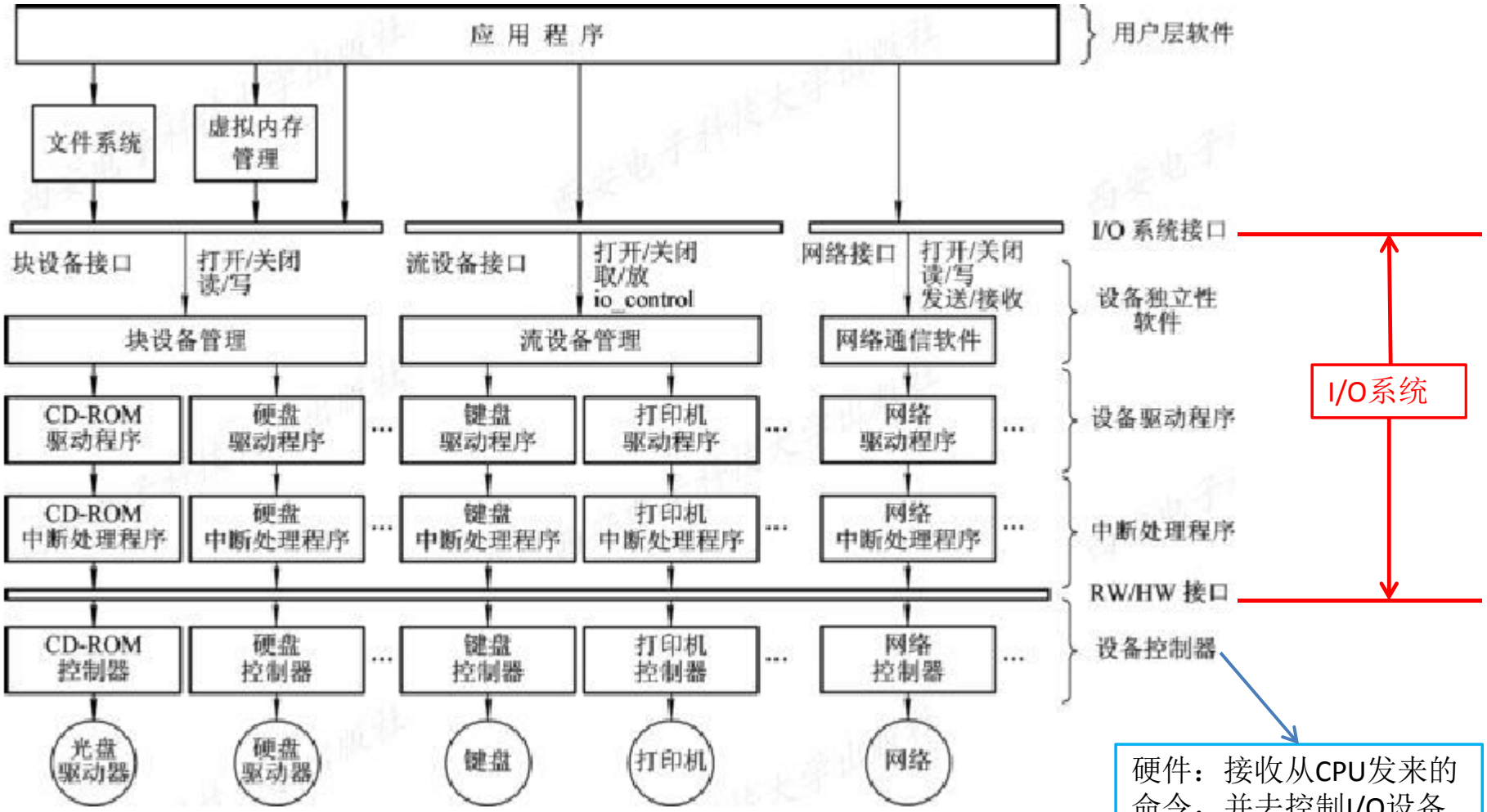
（3）硬件接口：部件之间的电气连接，如USB接口；

（4）功能接口：如打印机的单面打印、双面打印等；

（5）软件接口：控制接口硬件正常工作的软件，如用户界面；

6.1 I/O系统的功能、模型和接口

2、I/O系统中各模块之间的层次视图



6.1 I/O系统的功能、模型和接口

❖ 6.1.3 I/O系统接口

1、块设备接口

块设备：数据的存取与传输以数据块为基本单位，例如磁盘，其I/O控制方式采用DMA方式

2、流（字符）设备接口

字符设备：数据的存取与传输以字符为基本单位，例如键盘、打印机，常采用中断驱动方式

3、网络通信接口

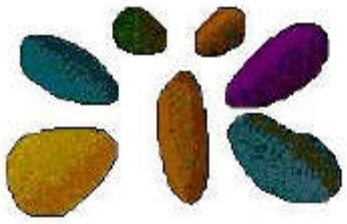
本章主要内容

- **6.1 I/O系统的功能、模型和接口**
- **6.2 I/O设备和设备控制器**
- **6.3 中断机构和中断处理程序**
- **6.4 设备驱动程序**
- **6.5 与设备无关的I/O软件**
- **6.6 用户层的I/O软件**
- **6.7 缓冲区管理**
- **6.8 磁盘存储器的性能和调度**

6.2 I/O设备与设备控制器

I/O设备一般由执行I/O操作的机械部件和执行I/O控制的电子部件组成，前者就是一般的I/O设备，后者称为设备控制器或适配器。

- 6.2.1 I/O设备
- 6.2.2 设备控制器
- 6.2.3 内存映像I/O
- 6.2.4 I/O通道



6.2.1 I/O设备

❖ 1、I/O设备的类型

1> 按设备的使用特性分类

存储设备（外存、后备存储器、辅助存储器）

较内存速度慢、容量大、价格低

输入/输出设备

输入设备

🕔键盘、鼠标、扫描仪、摄像头、传感器等

输出设备

🕔打印机、绘图仪、显示器、投影仪、音响等

交互式设备（上述两类设备的集成）

6.2.1 I/O设备

2> 按设备的传输速率分类

低速设备

每秒几个字节至数百字节

键盘、鼠标、语音输入输出设备等

中速设备

每秒数千至数万字节

行式打印机、激光打印机等

高速设备

每秒数百**K**至数十**M**字节

磁盘机、磁带机、光盘机等

6.2.1 I/O设备

3> 按设备的信息交换单位分类

块设备

信息的存取以块为基本单位，属有结构设备

典型块设备：磁盘

基本特征：传输速率较高、可寻址

字符设备

信息的存取以字符为基本单位，属无结构设备

典型字符设备：交互式终端、打印机

基本特征：传输速率较低、不可寻址、采用中断驱动方式

6.2.1 I/O设备

4> 按设备的共享属性分类

独占设备

一段时间内只允许一个进程访问的设备

例如：打印机

共享设备

一段时间内允许多个进程同时访问的设备

共享设备必须是可寻址的和可随机访问的设备

例如：磁盘

虚拟设备

通过虚拟技术将一台独占设备变换为若干个逻辑设备，供若干个进程同时使用

例如：采用**SPOOLing**技术实现打印机的共享

6.2.1 I/O设备

❖ 2、I/O设备与控制器之间的接口

I/O设备←→设备控制器←→ CPU

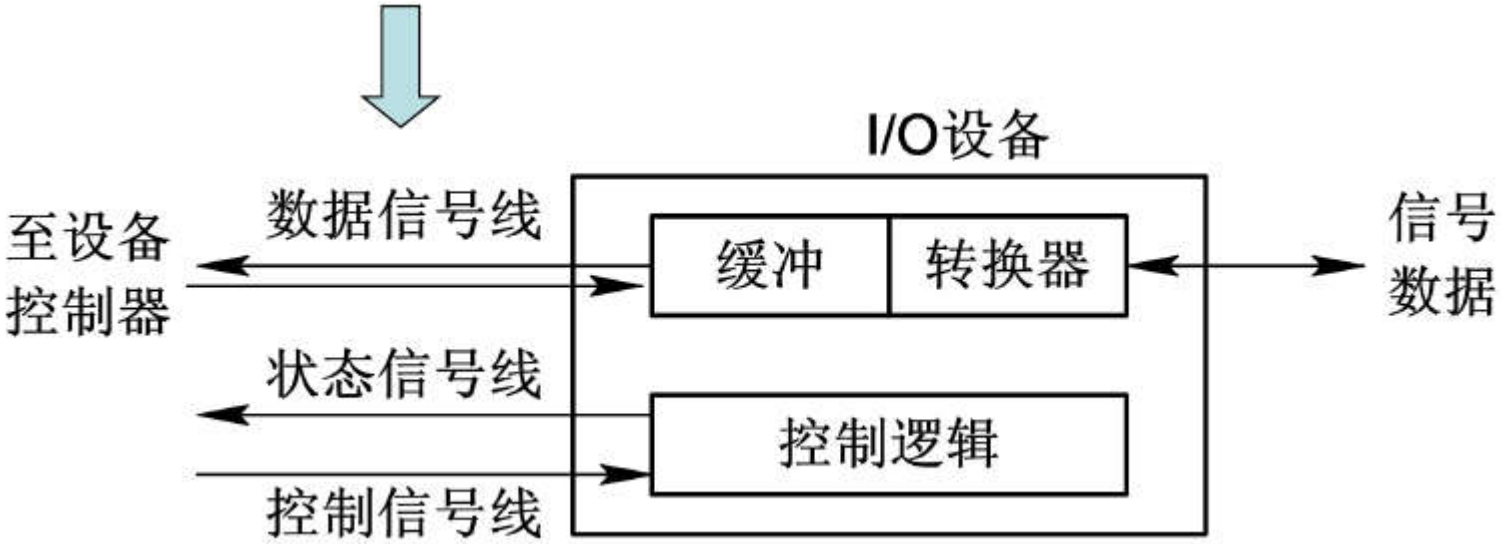


图6-3 I/O设备与控制器间的接口

6.2.1 I/O设备

1> 数据信号线

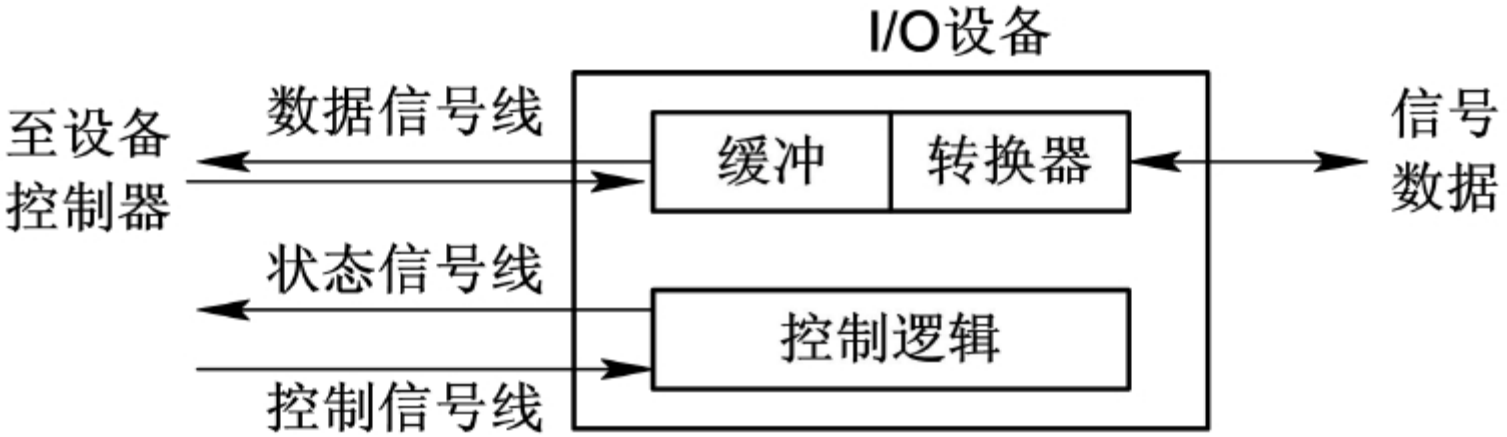
用于I/O设备和设备控制器之间传送数据信号

2> 控制信号线

用于设备控制器向I/O设备发送控制信号

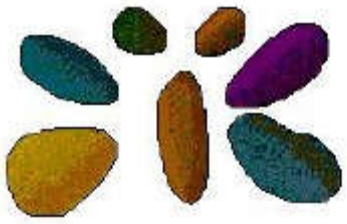
3> 状态信号线

用于I/O设备向设备控制器传送当前状态信号



6.2 I/O设备与设备控制器

- 6.2.1 I/O设备
- 6.2.2 设备控制器
- 6.2.3 内存映像I/O
- 6.2.4 I/O通道



6.2.2 设备控制器

❖ 1、设备控制器基本概念

设备控制器是计算机中的一种硬件实体（常做成印刷电路卡形式，也称接口卡），是**CPU**与**I/O**设备之间的接口，其主要职责是控制一个或多个**I/O**设备，实现**I/O**设备和计算机之间的数据交换，减轻**CPU**的负担。

*接收来自CPU的命令，以此控制I/O工作

设备控制器分类

字符型设备控制器

块设备控制器

6.2.2 设备控制器

- 2、设备控制器基本功能

(1) 接收和识别CPU命令

(2) 实现CPU与控制器、控制器与I/O设备数据交换

(3) 标识和报告I/O设备的状态

(4) 保存和识别I/O设备的地址

(5) 输入、输出数据缓冲

(6) 差错检测控制

6.2.2 设备控制器

❖ 3、设备控制器基本组成

CPU与设备控制器的接口

实现**CPU**与设备控制器之间的通信

三类信号线：数据线、地址线、控制线

两类寄存器：数据寄存器、控制/状态寄存器

控制器与设备的接口

用于连接一个或多个设备

三类信号线：数据线、控制线、状态线

I/O逻辑（译码器）

对**CPU**命令、地址译码，实现对**I/O**设备的控制

6.2.2 设备控制器

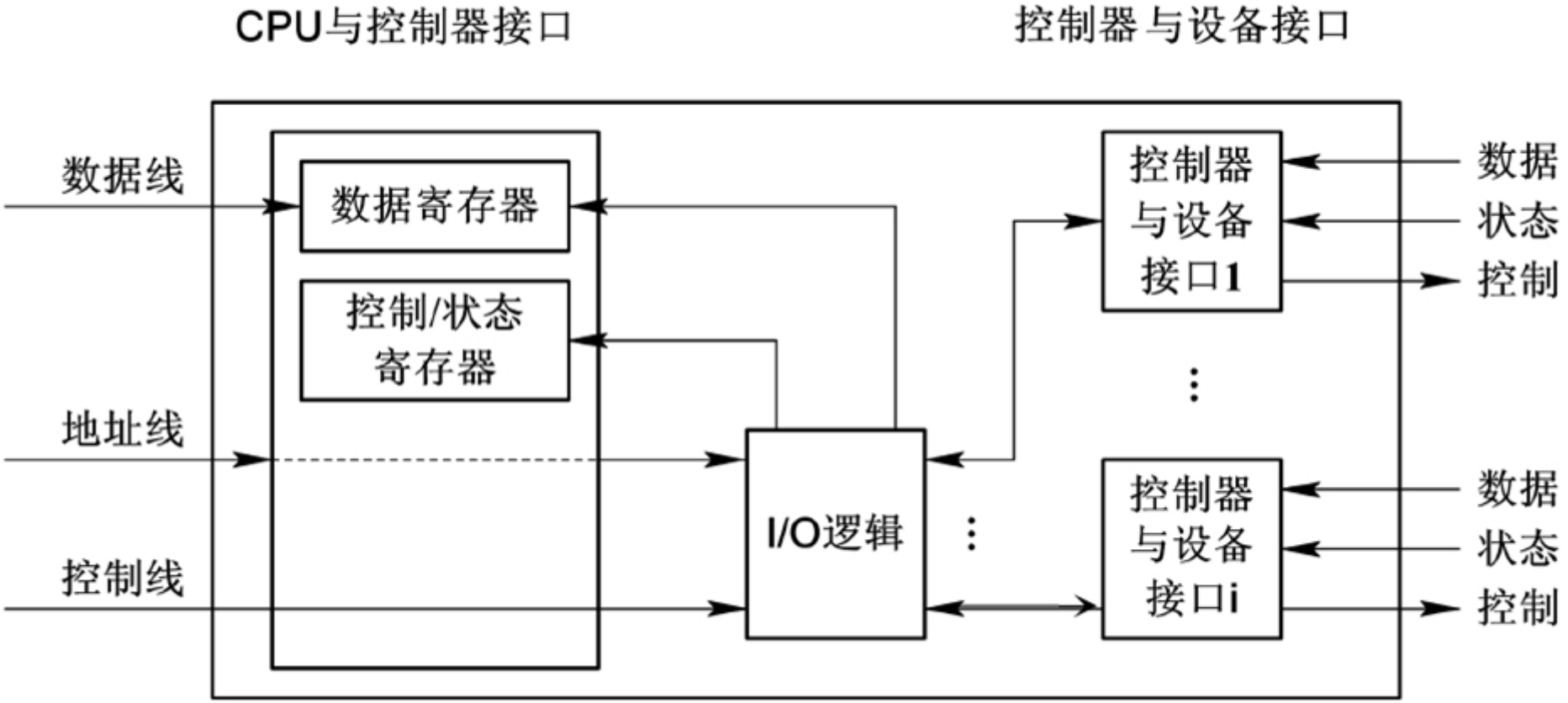
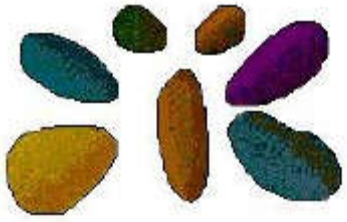


图 6-4 设备控制器的组成

6.2 I/O设备与设备控制器

- 6.2.1 I/O设备
- 6.2.2 设备控制器
- 6.2.3 内存映像I/O
- 6.2.4 I/O通道



6.2.3 内存映像I/O

问题引入：在**CPU**与设备控制器通信过程中，需将**CPU**寄存器中的内容复制到设备控制器的寄存器中，如何完成？

- **1、利用特定指令完成**

  例：**io-store cpu-reg, dev-no, dev-reg**

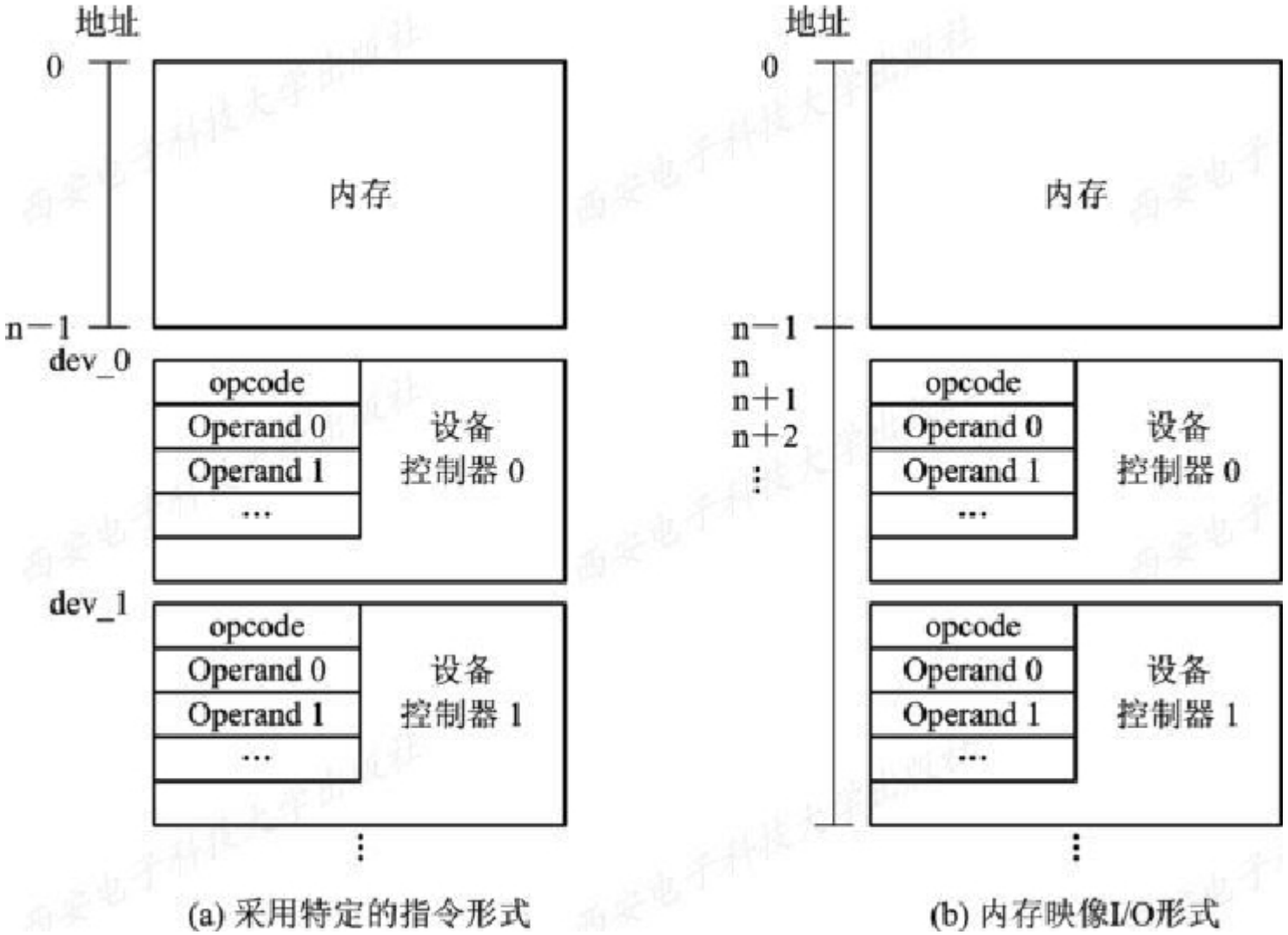
  缺点：访问设备与访问内存需要两种不同指令，不利于**I/O**编程

- **2、利用统一指令完成（内存映像I/O）**

  基本思想：将设备控制器的寄存器(或I/O端口)与内存统一编址，采用与内存访问相统一的指令访问

  例：**store cpu-reg, n**

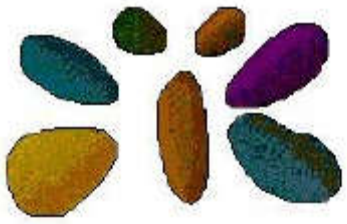
6.2.3 内存映像I/O



(a) 采用特定的指令形式　　(b) 内存映像I/O形式

6.2 I/O设备与设备控制器

- 6.2.1 I/O设备
- 6.2.2 设备控制器
- 6.2.3 内存映像I/O
- 6.2.4 I/O通道



6.2.4 I/O通道

❖ 1、I/O通道（I/O Channel）设备的引入

I/O设备←→设备控制器←→I/O通道←→ CPU

I/O通道是一种特殊的处理机，专门负责I/O操作，有自己简单的指令系统（只有数据传送指令和I/O设备控制指令）和执行I/O指令的能力，通过执行通道（I/O）程序完成I/O操作。

引入I/O通道的主要目的：建立独立的I/O操作，使有关对I/O操作的组织、管理及其结束处理尽量独立于CPU，减轻CPU负担。

I/O通道与一般处理机的区别

指令类型单一，局限于与I/O操作有关的指令。

没有独立的内存，I/O通道与CPU共享内存。

6.2.4 I/O通道

❖ 2、I/O通道设备的类型

1> 字节多路通道（Byte Multiplexor Channel）

原理：一个主通道连接多个子通道，以时间片轮转方式（字节交叉方式）共享主通道，每个子通道每次只传送一个字节，适于连接多台中、低速设备。

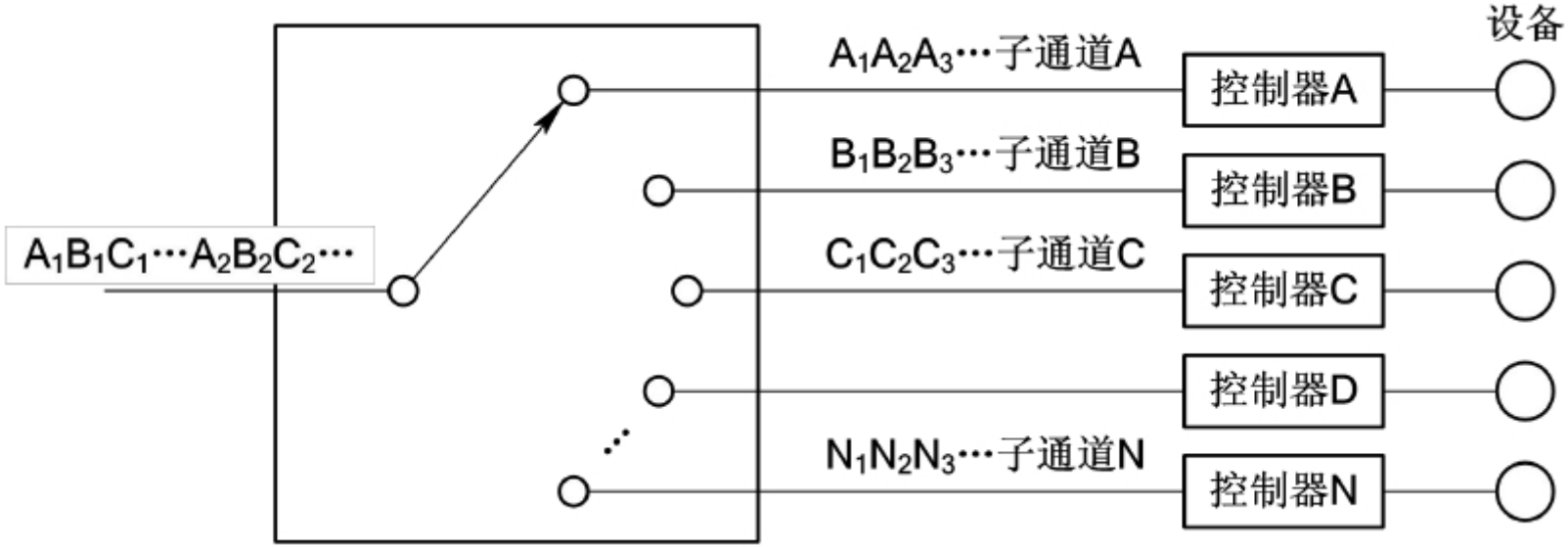


图 6-6 字节多路通道的工作原理

6.2.4 I/O通道

2> 数组选择通道（Block Selector Channel）

原理：虽可连接多台I/O设备，但无子通道，仅一主通道，某时间由某一I/O设备独占，以数据块为单位传递数据，适于连接高速I/O设备。缺点是设备申请使用通道的等待时间较长，利用率低。

3> 数组多路通道（Block Multiplexor Channel）

原理：综合了前面2种通道类型的优点，多子通道以分时方式（块交叉方式）并以数据块为单位传送数据。

数组多路通道在实际中应用较多。

6.2.4 I/O通道

❖ 3、I/O通道设备的“瓶颈”问题

“瓶颈”问题：通道不足

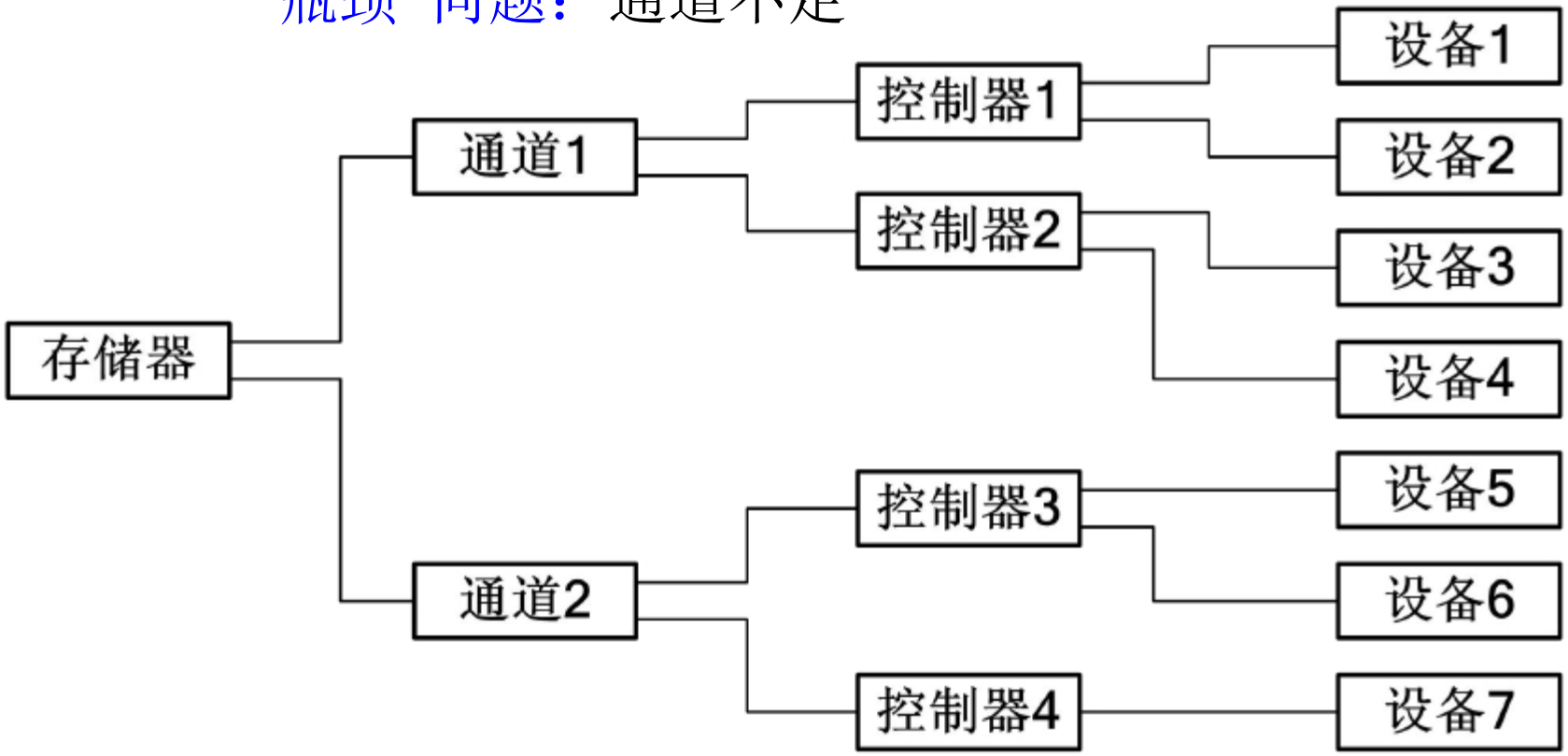


图 6-7单通路I/O系统

6.2.4 I/O通道

解决办法：采用复联方式，增加通路

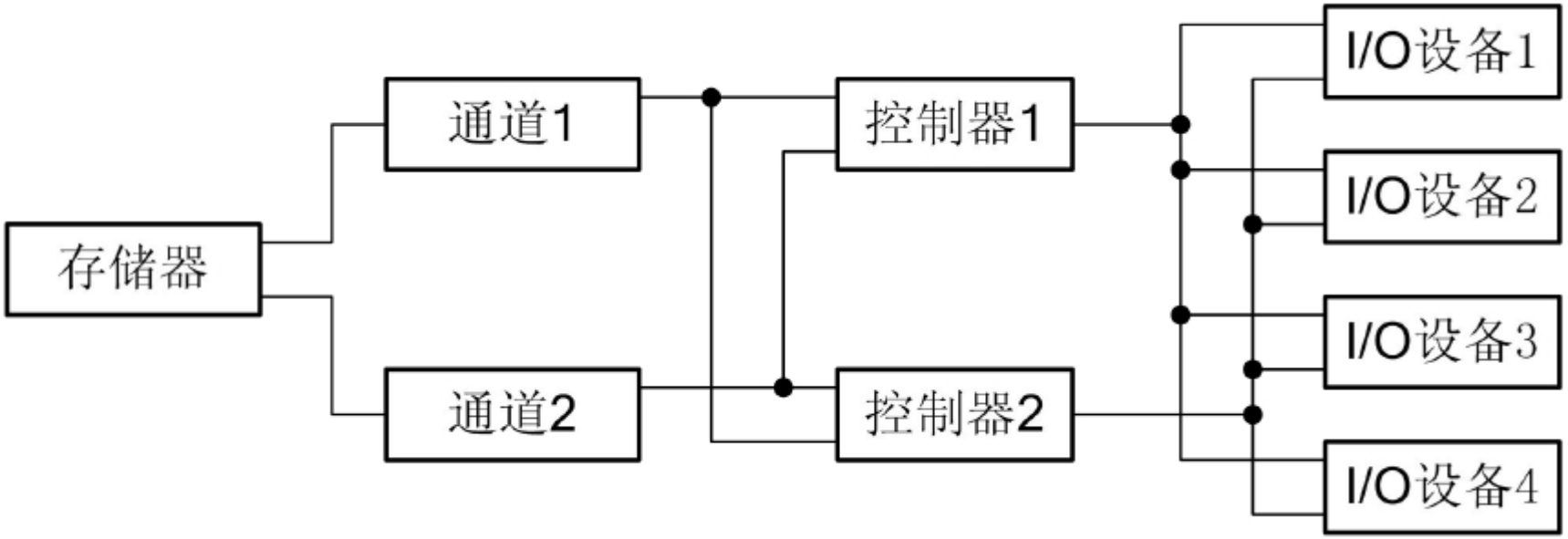


图6-8 多通路I/O系统

本章主要内容

- 6.1 I/O系统的功能、模型和接口
- 6.2 I/O设备和设备控制器
- 6.3 中断机构和中断处理程序
  - 6.3.1 中断简介
  - 6.3.2 中断处理程序
- 6.4 设备驱动程序
- 6.5 与设备无关的I/O软件
- 6.6 用户层的I/O软件
- 6.7 缓冲区管理
- 6.8 磁盘存储器的性能和调度

6.3.1 中断简介

- 1、中断的产生

  I/O请求→ ……→设备启动→**I/O**完成→设备控制器向**CPU**发送中断请求→ **CPU**调用中断处理程序

- 2、中断与陷入

  中断：是指**CPU**对**I/O**设备发来的中断信号的一种响应。是由**CPU**外的设备引起的，又称外中断。

  陷入**(trap)**：是由**CPU**内部事件引起的一种中断，也称内中断。

- 3、中断向量表和中断优先级

  中断向量表：表中每一条记录由一个中断号（代表具体的中断请求）及相应的中断处理程序的入口地址组成。

  中断优先级：给不同的中断信号源赋予不同的优先级

6.3.1 中断简介

❖ 4、对多中断源的处理方式

1> 屏蔽(禁止)中断: 处理一个中断时, 屏蔽其余中断，按顺序处理中断

简单，但不能用于实时性要求高的中断请求

2> 嵌套中断

优先响应最高优先级的中断请求

高优先级的中断请求抢占低优先级的中断请求

6.3.2 中断处理程序

❖ 1、中断处理过程

1> 唤醒被阻塞的驱动程序进程

2> 保护被中断进程CPU环境（图6-10）

将被中断进程的CPU现场信息压入中断栈

3> 转入相应的设备处理程序

CPU对各中断源测试，确定引起本次中断的I/O设备，并发送一应答信号给发出中断请求的设备，驱动程序进程调用相应的I/O设备中断处理程序。

6.3.2 中断处理程序

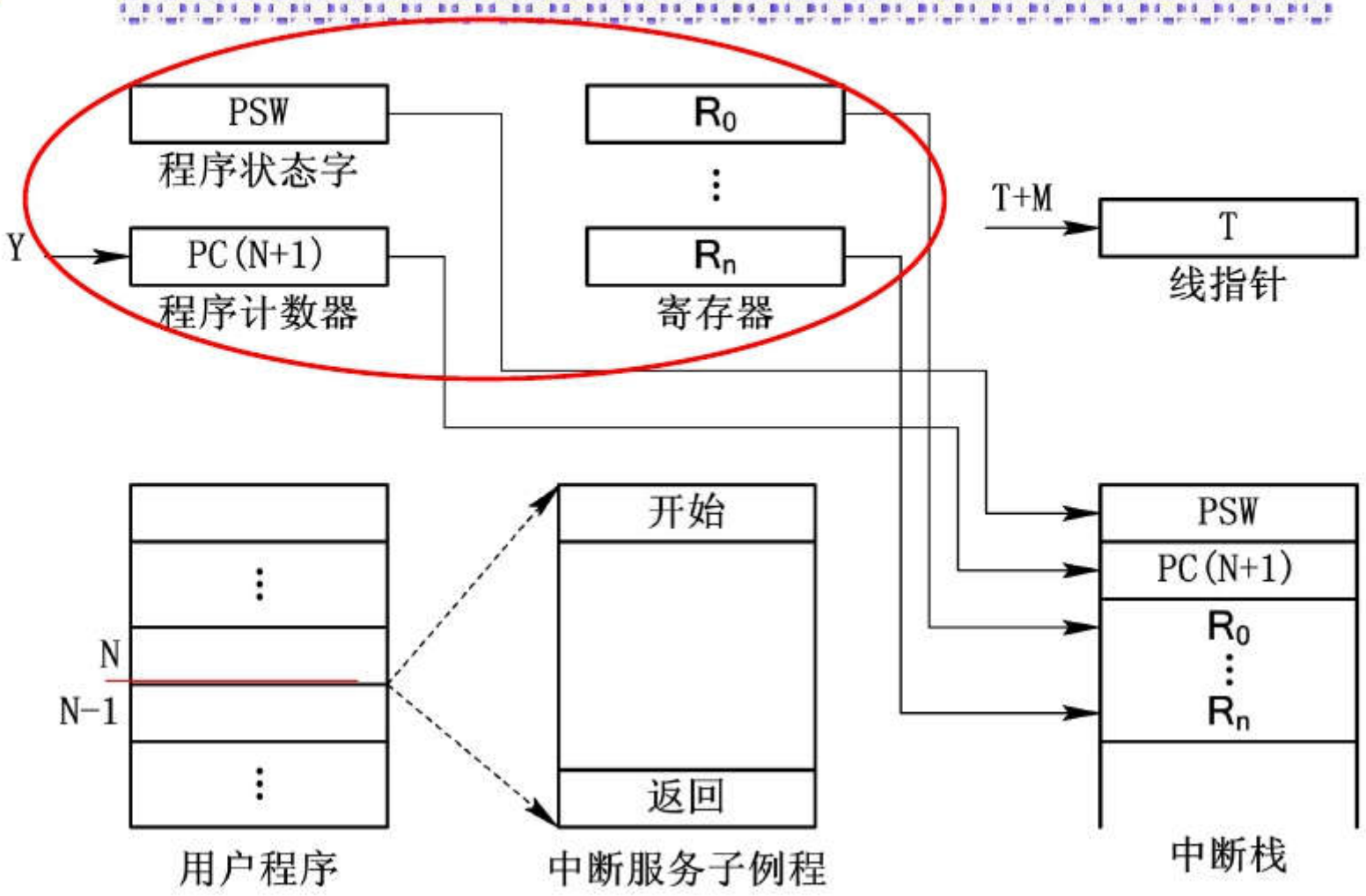


图 6-10 中断现场保护示意图

6.3.2 中断处理程序

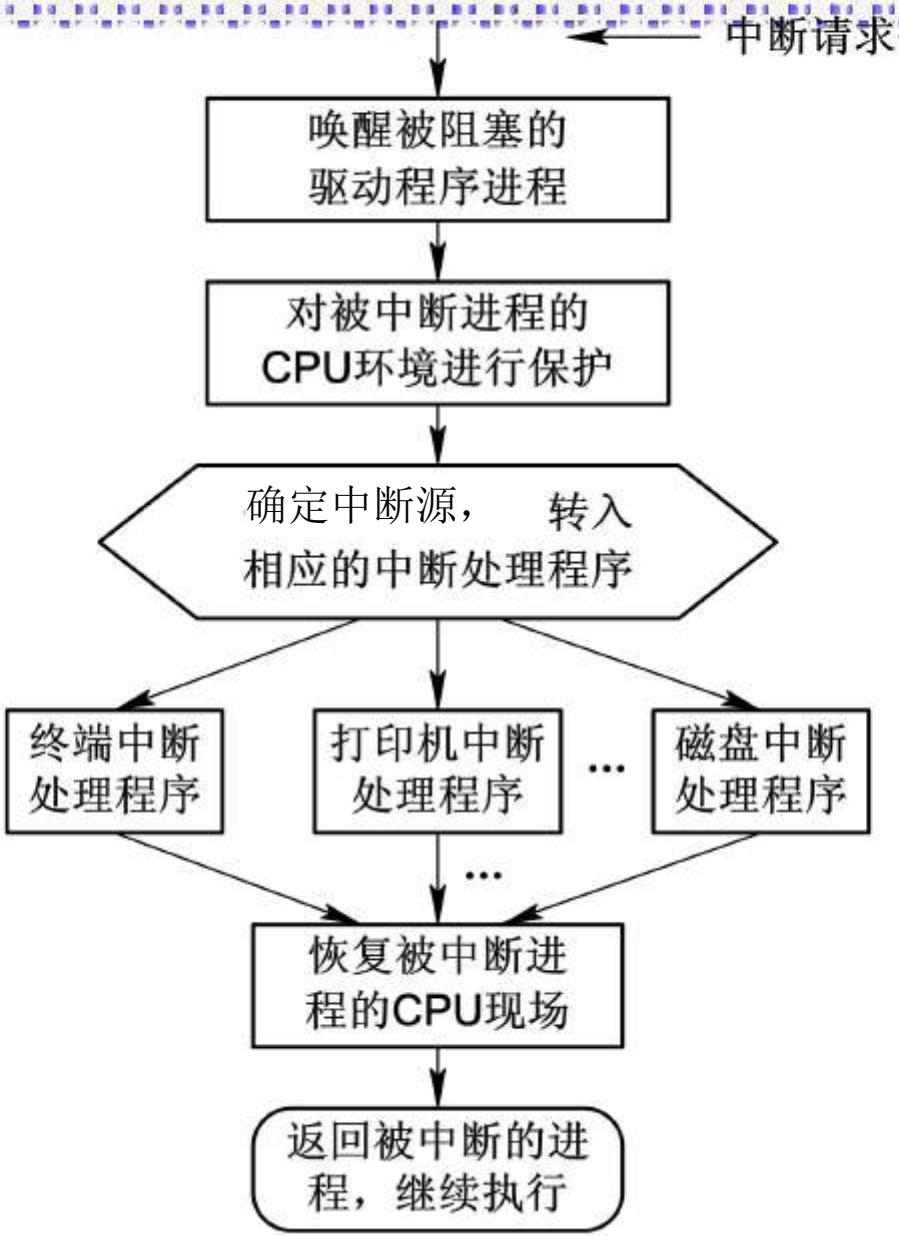
4> 中断处理

中断处理程序从设备控制器读出设备状态，以判别本次中断是正常结束中断，还是异常结束中断，若为后者，则根据异常原因做进一步处理。

5> 恢复被中断进程的现场

如果没有其它的中断请求待处理：从中断栈中取出被中断进程的**CPU**现场信息，装入相应的寄存器中，恢复被中断进程运行；如果还有其它的中断请求需求要处理，则接着处理。

图6-11　中断处理流程


中断请求信号
唤醒被阻塞的
驱动程序进程
对被中断进程的
CPU环境进行保护
确定中断源，　转入
相应的中断处理程序
终端中断
处理程序
打印机中断
处理程序
...
磁盘中断
处理程序
...
恢复被中断进
程的CPU现场
返回被中断的进
程，继续执行

本章主要内容

- 6.1 I/O系统的功能、模型和接口
- 6.2 I/O设备和设备控制器
- 6.3 中断机构和中断处理程序
- 6.4 设备驱动程序
- 6.5 与设备无关的I/O软件
- 6.6 用户层的I/O软件
- 6.7 缓冲区管理
- 6.8 磁盘存储器的性能和调度

6.4.1 设备驱动程序概述

❖ 1、概念

设备驱动程序是与硬件直接相关，负责具体实现系统对设备发出的操作指令，驱动**I/O**设备工作的程序。

设备驱动程序又称设备处理程序，是**I/O**进程与设备控制器之间的通信和转换程序，常以进程的形式存在，它接收来自上层**I/O**软件的、抽象的**I/O**命令，再把它换成具体的命令发送给设备控制器，来启动设备进行**I/O**操作。

6.4.1 设备驱动程序概述

❖ 2、设备驱动程序的功能

（1）接收进程的**I/O**抽象命令，并转换成具体命令；

（2）检查**I/O**请求的合法性、检查设备的状态、设置设备的工作方式；

（3）发布**I/O**具体命令，驱动**I/O**操作；

（4）响应设备中断，调用相应中断处理程序处理；

如果系统中有通道的话，根据**I/O**抽象命令，生成通道程序。

6.4.1 设备驱动程序概述

❖ 3、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，不同控制方式的驱动程序存在较大差异。

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。目前有很多驱动程序的基本部分，已经固化在**ROM**中。

6.4.1 设备驱动程序概述

5> 驱动程序应允许可重入，一个正在运行的驱动程序常会在一次调用完成前被再次调用。

6.4.2 设备驱动程序的处理过程

❖ 1、设备驱动程序的处理过程

两大过程

1> 启动设备过程

2> 中断处理过程（即6.3.2节内容）

启动设备过程

1> 将抽象命令转化为具体命令

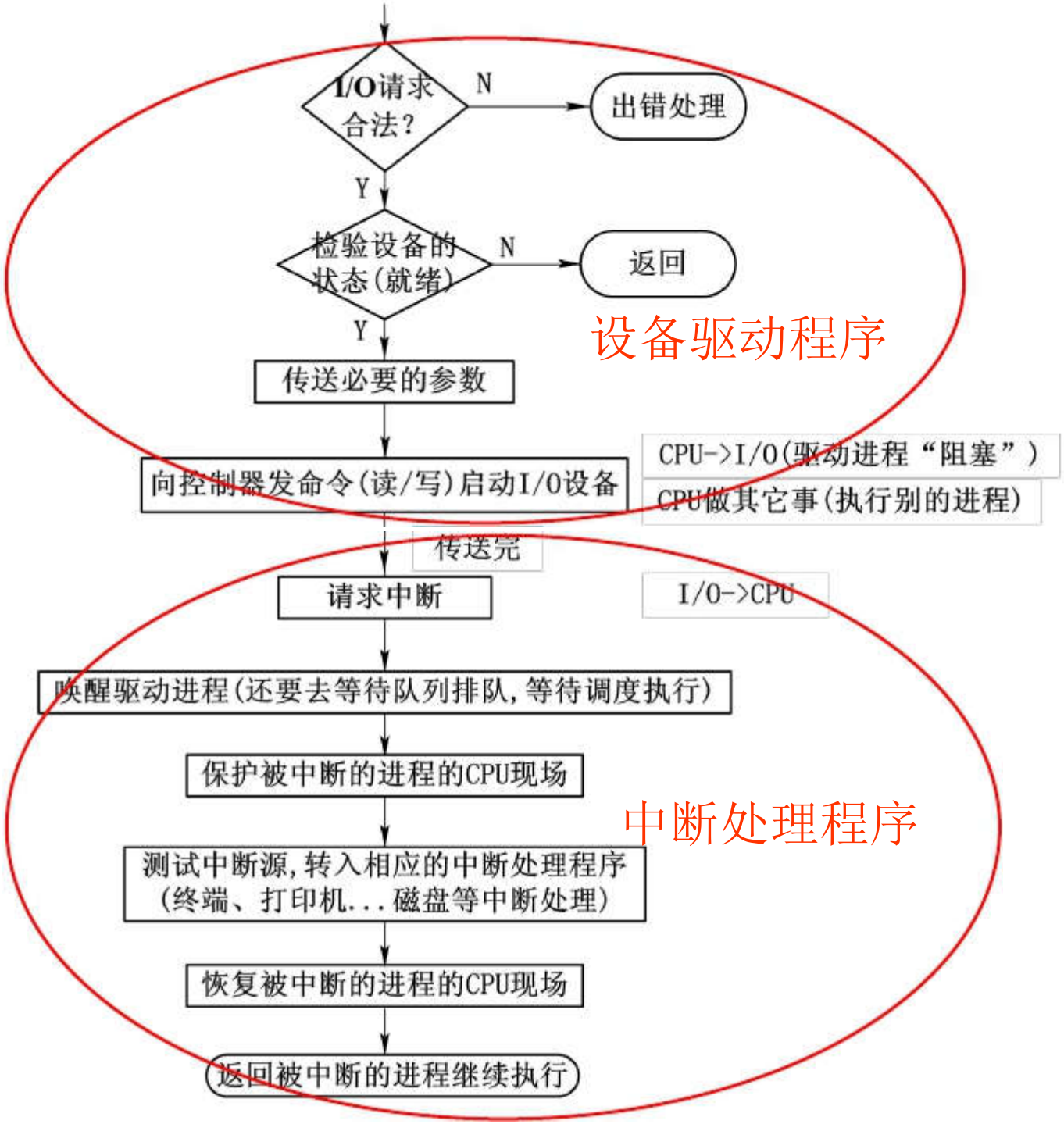
2> 检查I/O请求合法性

3> 读出和检查设备状态

4> 传送必要的参数

5> 启动I/O设备

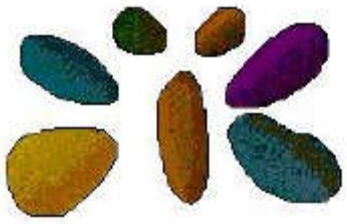
图　设备驱动程序工作流程



6.4.3 对I/O设备的控制方式

- 1、 使用轮询的可编程I/O方式
- 2 、使用中断的可编程I/O方式
- 3 、直接存储器访问（DMA）方式
- 4、 I/O通道控制方式

发展趋势：提高CPU与I/O设备的并行度，尽量减少CPU的干预



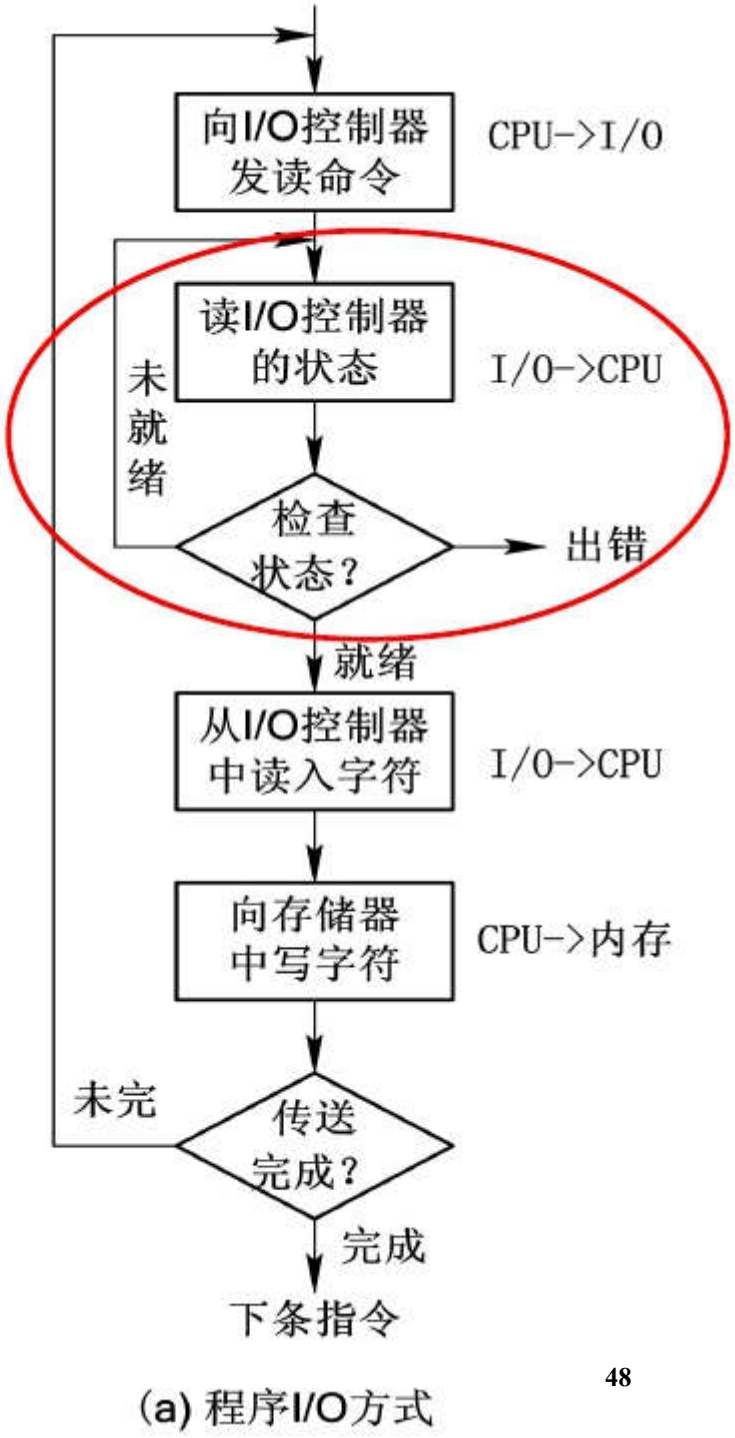
1、使用轮询的可编程I/O方式

- 基本原理

这种方式也称为“忙-等”方式。 **CPU**向设备控制器发出**I/O**命令后，由于计算机系统中无中断机构，**CPU**必须不断循环检测**I/O**设备的状态，看数据是否已由 **I/O**设备输入或输出完毕。

每次传送一个字符。

CPU资源浪费严重。



(a) 程序I/O方式

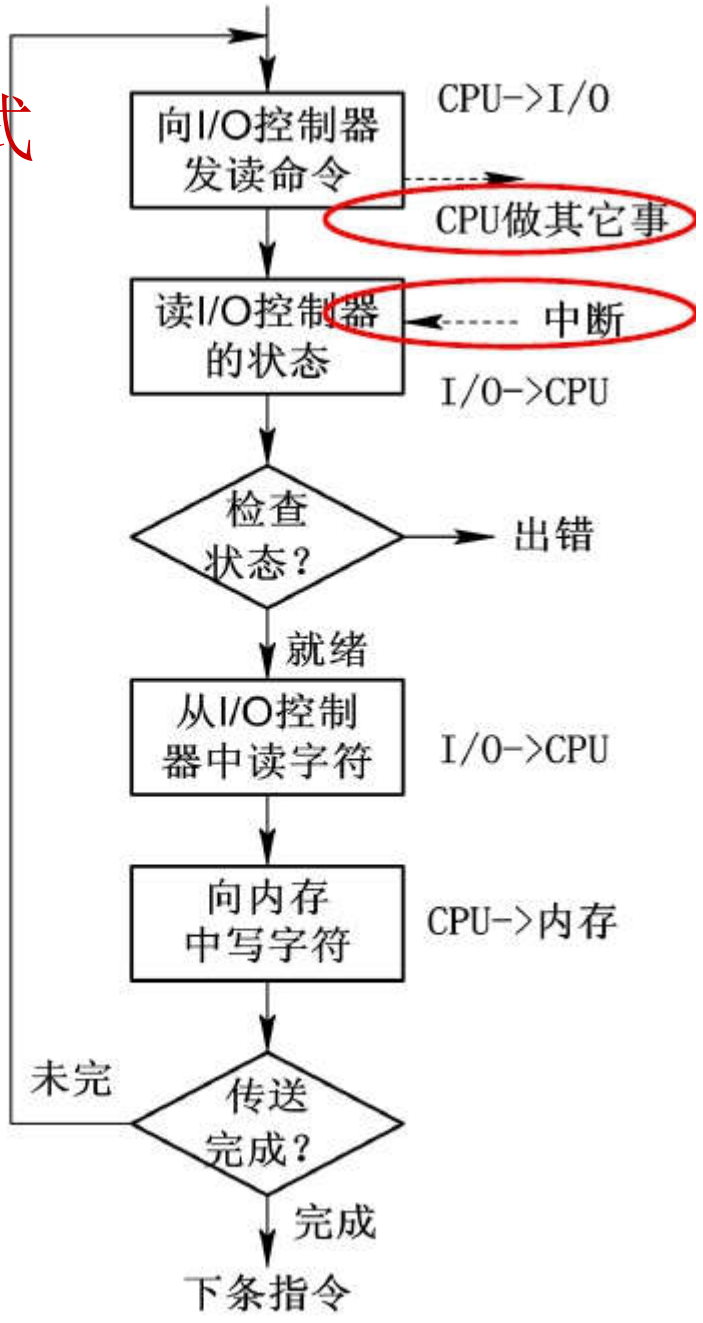
2、使用中断驱动的可编程I/O方式

- 基本原理

引入中断机构，**CPU**与**I/O**设备并行操作：**CPU**向设备控制器发出**I/O**命令后立即返回继续执行原来或者别的任务；设备控制器按**I/O**命令控制指定**I/O**设备进行**I/O**，完成后，再通过中断机构向**CPU**发送一中断信号。

每次传送一个字符。

较轮询**I/O**控制方式，**CPU**的利用率大幅提高。



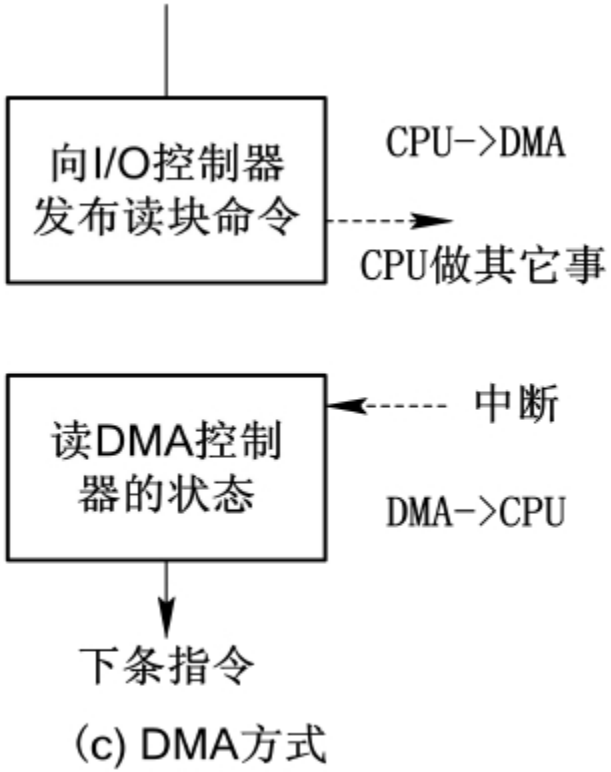
(b) 中断驱动方式

3、直接存储器访问（DMA）I/O控制方式

❖ 1、DMA I/O控制方式的引入

中断驱动I/O 存在的问题

数据传输的基本单位为字节，每传输一个字节，设备控制器就要向CPU请求一次中断，效率低下。

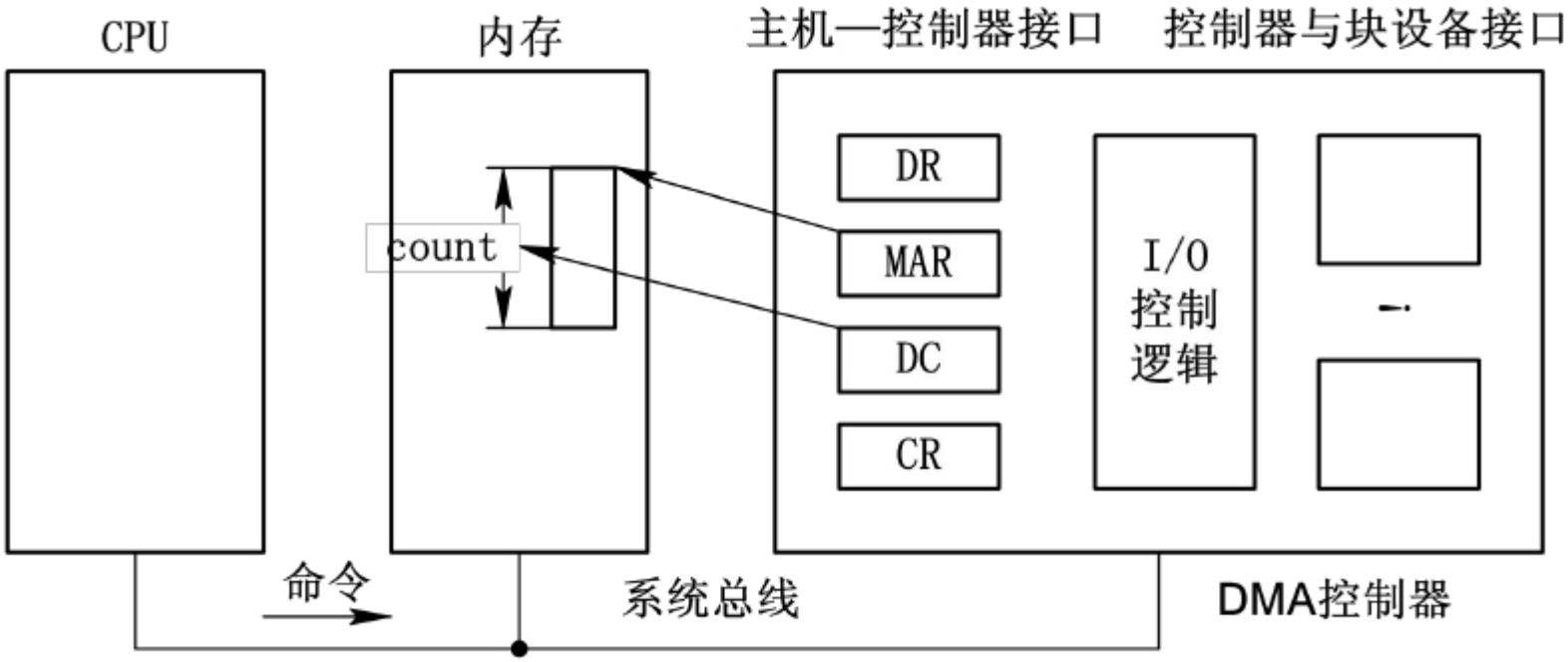


DMA I/O控制方式

DMA I/O控制方式是指由DMA控制器实现I/O设备与内存之间数据块的直接传送，在数据块传送期间无需CPU干预，仅当在传送一个或多个连续的数据块开始和结束时才需要CPU干预。

3、直接存储器访问（DMA）I/O控制方式

❖ 2、 DMA控制器的组成

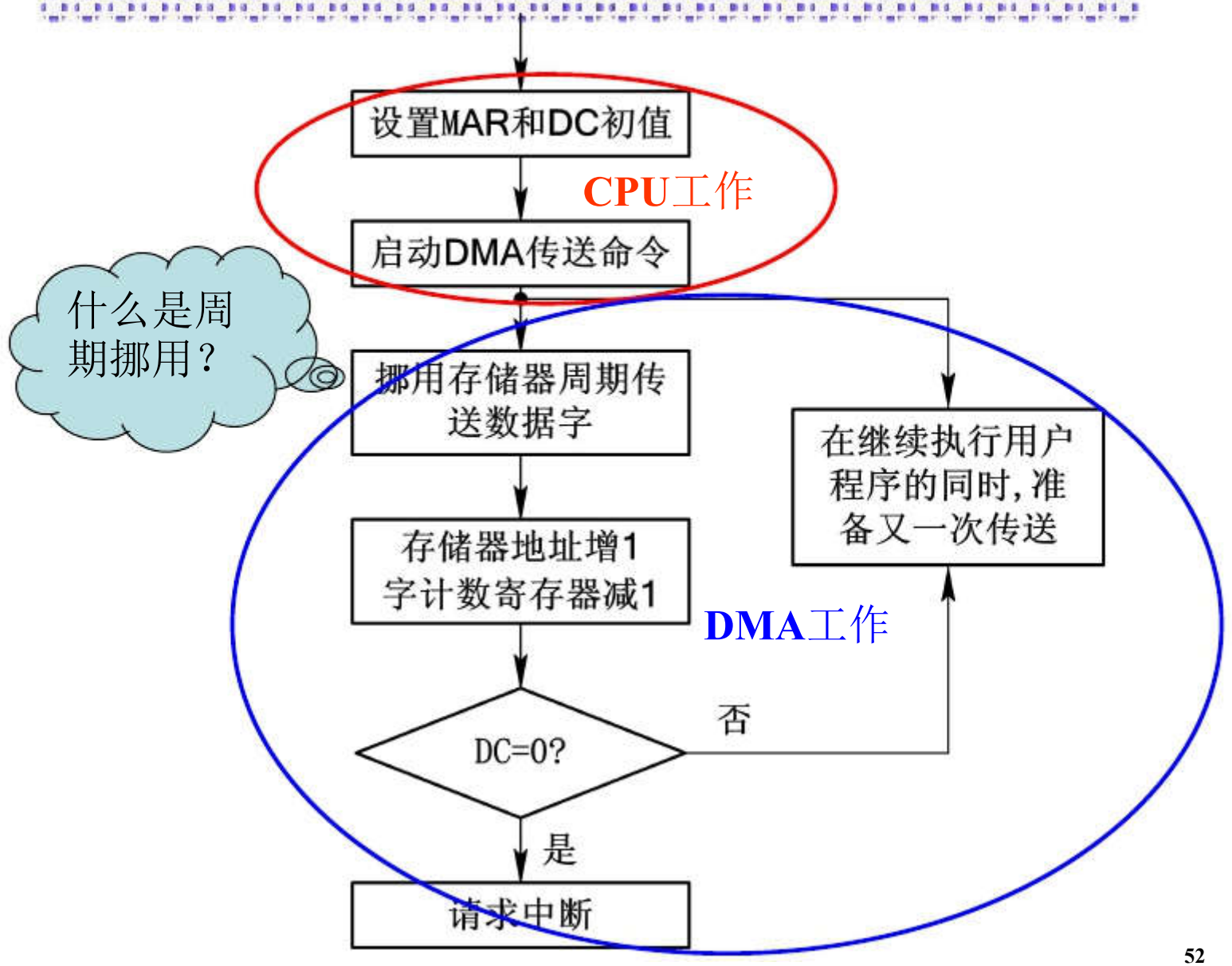


命令/状态寄存器（**CR**）　　内存地址寄存器（**MAR**）

数据寄存器（**DR**）　　数据计数器（**DC**）

3、直接存储器访问（DMA）I/O控制方式

DMA 工作过程



4、I/O通道控制方式

❖ 1、I/O通道控制方式的引入

DMA I/O 存在的问题

DMA I/O对多个离散块存取仍需多次中断。

I/O通道控制方式

通道控制方式同DMA控制方式一样，也是一种以内存为中心，是设备与内存直接交换数据的控制方式。CPU只需向通道发出一条I/O指令，给出其所要执行的通道程序的首址和要访问的I/O设备，通道接到指令后，通过执行通道程序便可完成一组数据块的I/O操作。

通道控制方式可实现CPU、通道和I/O设备三者的并行操作，从而更有效地提高整个系统的资源利用率。

4、I/O通道控制方式

❖ 2、I/O通道程序

通道程序

通道程序由一系列通道指令构成。通道通过执行通道程序，并与设备控制器共同实现对I/O设备的控制。

通道指令包含的信息

1> 操作码

规定指令所执行的操作，如读、写、控制等

2> 内存地址

标明字符送入内存(读)或从内存取出(写)的内存首址

3> 计数

本条指令所要读/写的字节数

4、I/O通道控制方式

4> 通道程序结束位P

表示通道程序是否结束，**P=1**表示结束。

5> 记录结束标志R

R=0表示本指令与下一指令处理同一个记录；
R=1表示处理某记录的最后一条指令。

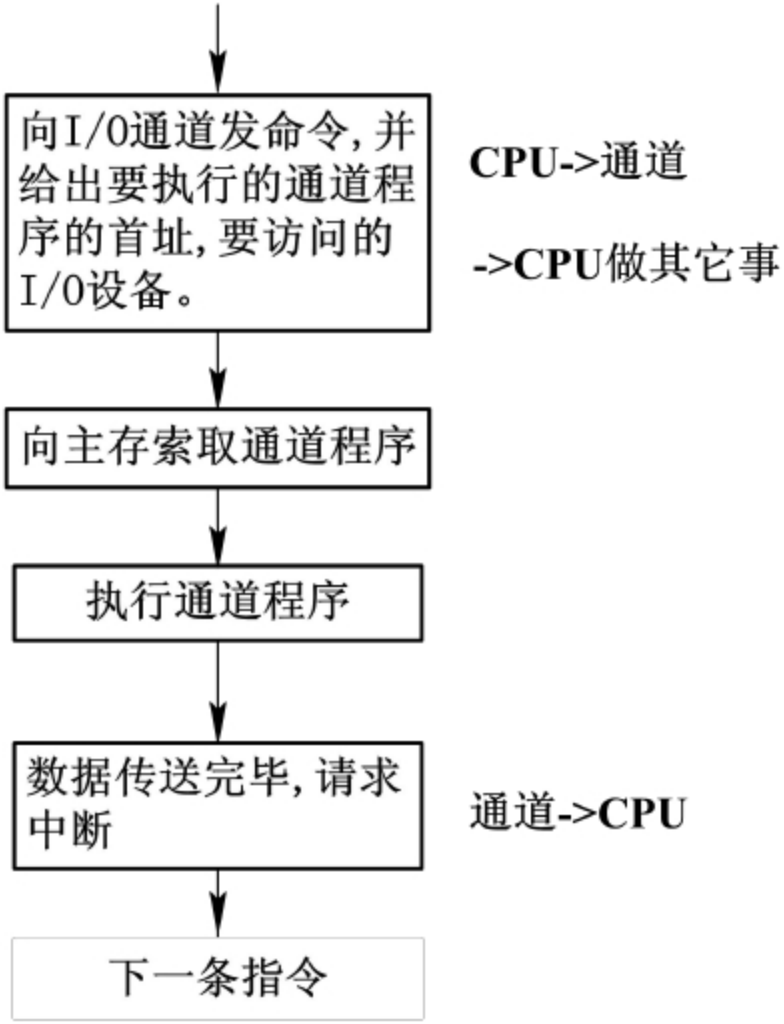
4、I/O通道控制方式

表 含有六条通道指令的通道程序示例（不讲）

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

4、I/O通道控制方式

❖ 3、I/O通道控制方式工作流程

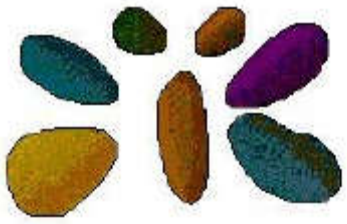


本章主要内容

- **6.1 I/O系统的功能、模型和接口**
- **6.2 I/O设备和设备控制器**
- **6.3 中断机构和中断处理程序**
- **6.4 设备驱动程序**
- **6.5 与设备无关的I/O软件（设备独立性软件）**
- **6.6 用户层的I/O软件**
- **6.7 缓冲区管理**
- **6.8 磁盘存储器的性能和调度**

6.5 设备独立性软件

- 6.5.1 基本概念
- 6.5.2 设备独立性软件的主要功能
- 6.5.3 设备分配
- 6.5.4 逻辑设备名到物理设备名映射的实现



6.5.1 基本概念

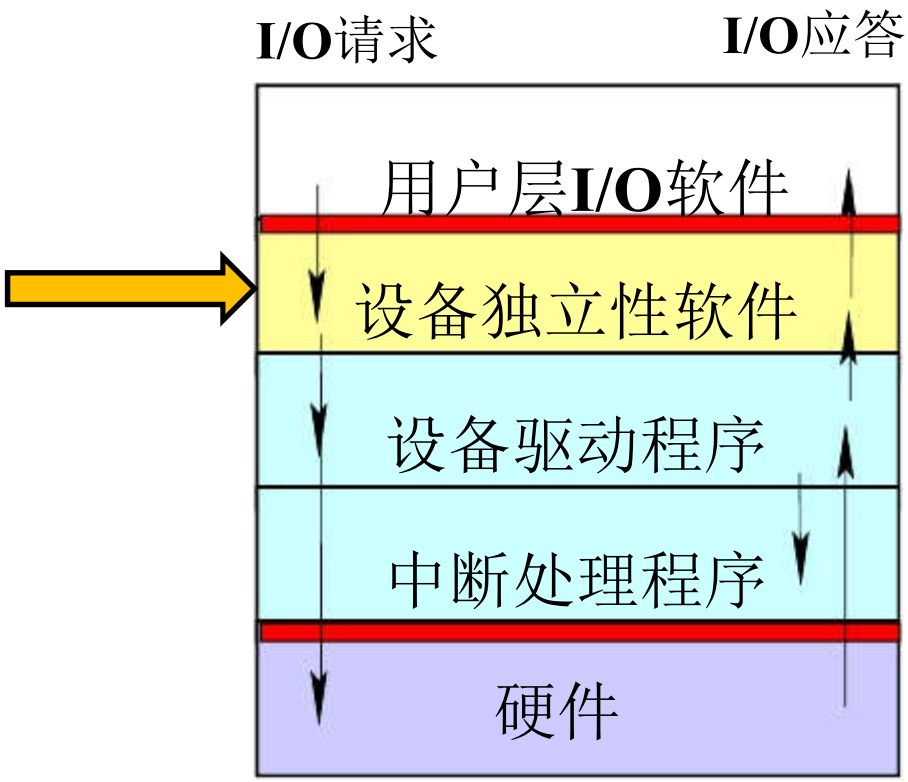
❖ 1、设备独立性的概念

基本含义

设备独立性**(Device Independence)**，也称为设备无关性，即应用程序独立于具体使用的物理设备，也就是说用户在编程序时所使用的设备与实际设备无关。

实现办法

在应用程序中，使用逻辑设备名称来请求使用某类设备；而系统实际执行时，使用物理设备名称启动**I/O**设备。**OS**通过设置逻辑设备表完成逻辑设备名称和物理设备名称的映射转换。



应用程序独立于物理设备

1、设备驱动程序：与硬件紧密相关

2、为了实现设备独立性，加入的软件称之为“设备独立性软件”

6.5.1 基本概念

设备独立性的好处

1> 增加了设备分配的灵活性

逻辑设备名称和物理设备名称之间可以是多对多的映射关系，提高了物理设备的共享性，以及使用的灵活性。如：

- 一个逻辑名称可对应一类设备，提高均衡性与容错性。
- 几个逻辑名称对应某一个设备，提高共享性。

2> 易于实现I/O重定向

I/O重定向： 通过 改变逻辑设备表的映射关系来实现**I/O**设备的更换（即重定向），而不必修改应用程序。

6.5.2 设备独立性软件的主要功能

- 设备独立性软件主要功能

  1> 向用户层软件提供统一接口

  提供统一接口，统一命令等；

  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。

  2> 缓冲管理，以提高I/O的效率 (详参6.7节内容)

  3> 差错控制，处理那些设备驱动程序无法处理的错误（与设备无关的I/O操作错误）。

  4> 对各种设备进行分配与回收。

  5> 提供独立于设备的逻辑数据块。

6.5.3 设备分配

❖ 1、设备分配中的数据结构

1> 设备控制表（DCT）

系统为每一个设备都配置了一张设备控制表，用于记录该设备的使用情况。

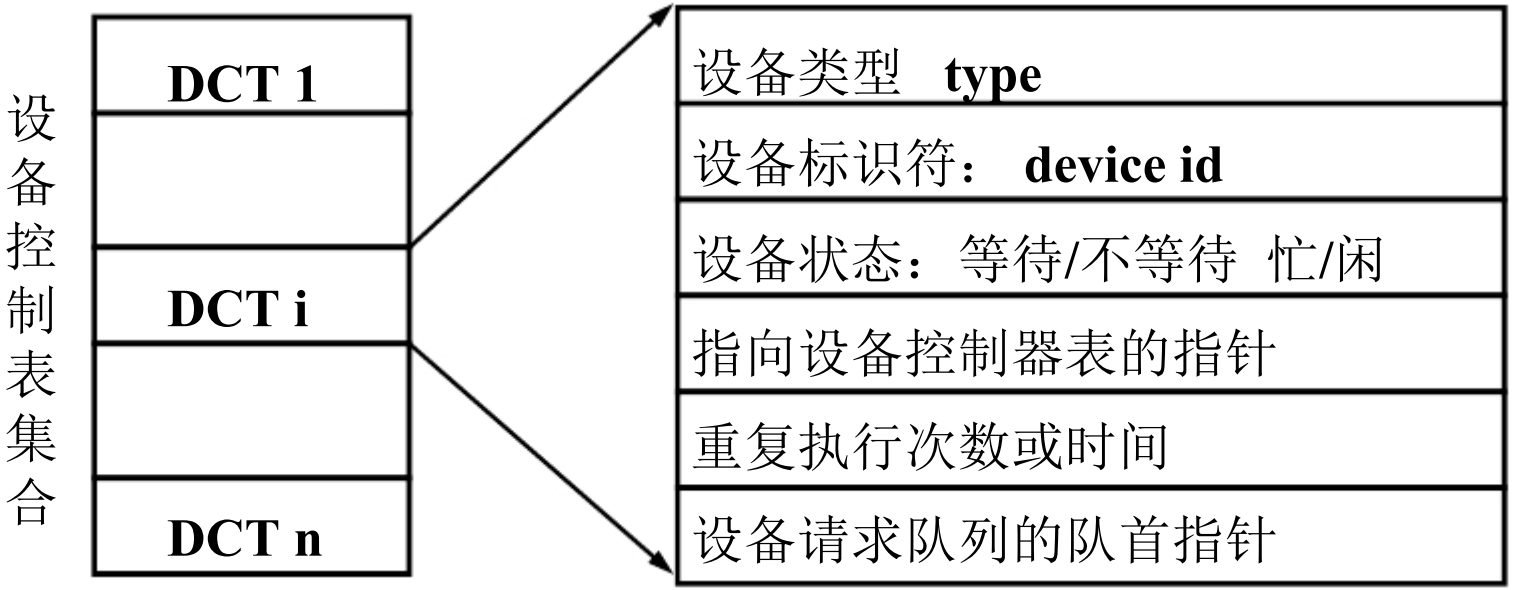


图6-17 设备控制表

6.5.3 设备分配

2> 控制器控制表（COCT）

系统为每一个控制器都配置了一张设备控制器控制表，用于记录该控制器的使用情况。

控制器标识符：controllerid
控制器状态：忙/闲
与控制器连接的通道表指针
控制器队列的队首指针
控制器队列的队尾指针

图6-18 (a) 控制器控制表

6.5.3 设备分配

3> 通道控制表（CHCT）

系统为每一个通道都配置了一张通道控制表，用于记录该通道的使用情况。

通道标识符：channelid
通道状态：忙/闲
与通道连接的控制器表首址
通道队列的队首指针
通道队列的队尾指针

图6-18 (b) 通道控制表

6.5.3 设备分配

4> 系统设备表（SDT）

记录系统中的全部设备的使用情况，每个物理设备占一个表目。整个系统配置一张。

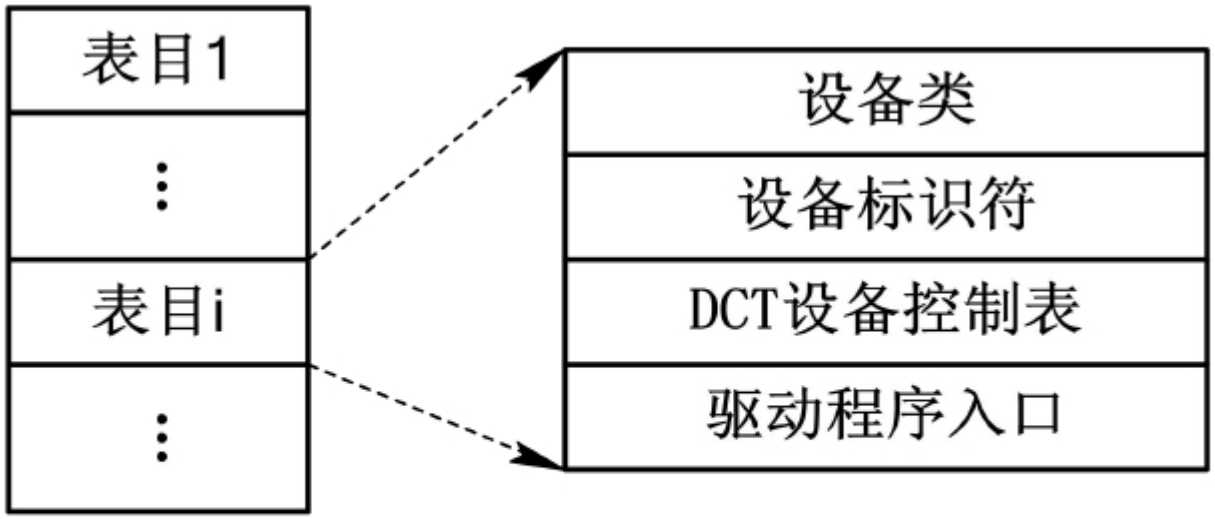


图6-18 (c) 系统设备表

6.5.3 设备分配

❖ 2、设备分配时应考虑的因素

1> 设备的固有属性

① 独占性（独占设备）

采用独享分配策略，独占设备分配给某进程后便由该进程独占，直至其用完或释放该设备。

② 共享性（共享设备）

允许多个进程同时共享该类设备。

③ 虚拟性（虚拟设备）

虚拟设备是利用虚拟技术把独占设备改造成可由多个进程共用的设备。

虚拟设备允许多个进程同时共享。

6.5.3 设备分配

2> 设备分配算法

① 先来先服务

按进程对设备请求的先后次序，将进程排成一个设备请求队列，设备分配程序总是把设备首先分配给队首进程。

② 优先级高者优先

优先权高的进程排在设备队列前面，而对于优先级相同的**I/O**请求，则按先来先服务原则排队。

6.5.3 设备分配

3> 设备分配中的安全性

① 安全分配方式

每当进程发出**I/O**请求后，便进入阻塞状态，直到其**I/O**操作全部完成后唤醒。

优点：摒弃了“请求和保持条件”，不会产生死锁。

缺点：进程进展缓慢，即**CPU**与**I/O**设备串行工作。

② 不安全分配方式

进程发出**I/O**请求后仍继续运行。

优点：可操作多个设备，推进迅速。

缺点：可能导致“死锁”，需增加预测死锁的安全性计算，这在一定程度上增加了程序的复杂性。

6.5.4 逻辑设备名到物理设备名映射的实现

❖ 1、逻辑设备表(Lgical Unit Table，LUT)

作用：将应用程序中使用的逻辑设备名映射为物理设备名。

生成：在用户进程第一次请求设备分配时完成映射，并在**LUT**中生成相应表项。

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)

图6-19 逻辑设备表

6.5.4 逻辑设备名到物理设备名映射的实现

❖ 2、逻辑设备表的设置

整个系统设置一张**LUT**

要求：要求所有用户使用的逻辑设备名称不能重复，一般用于单用户系统。

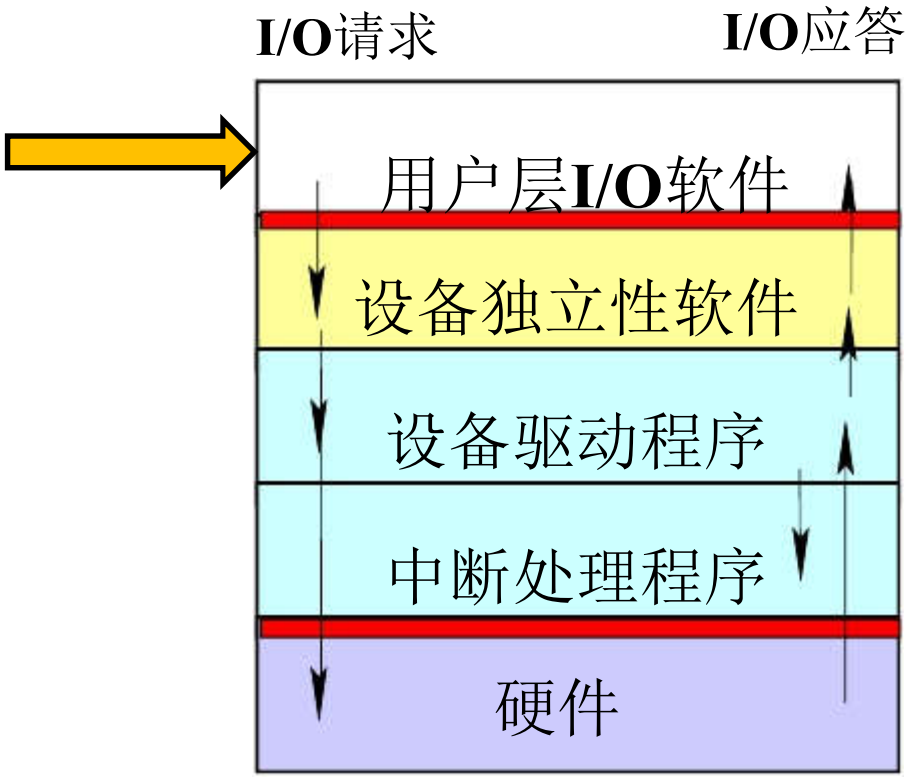
每个用户设置一张**LUT**

可重名/可限制用户对某些设备的使用。

本章主要内容

- **6.1 I/O系统的功能、模型和接口**
- **6.2 I/O设备和设备控制器**
- **6.3 中断机构和中断处理程序**
- **6.4 设备驱动程序**
- **6.5 与设备无关的I/O软件**
- **6.6 用户层的I/O软件**
- **6.7 缓冲区管理**
- **6.8 磁盘存储器的性能和调度**

6.6 用户层的I/O软件



用户层I/O软件是指运行在操作系统用户层中的软件，包括：

- 6.6.1 系统调用与库函数
- 6.6.2 假脱机(SPOOLing)系统

6.6.1 系统调用与库函数

❖ 1、系统调用与库函数

系统调用：是应用程序请求**OS**内核完成某功能时的一种过程调用（详参教材**P294**）。

库函数：现代**OS**中对系统调用采用**C**语言编写，以函数形式提供的一种系统调用。

6.6.2 假脱机(SPOOLing)系统

❖ 1、什么是SPOOLing

脱机技术：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，缓和CPU的高速性与I/O设备低速性间的矛盾。

假脱机技术：在主机控制下，用程序模拟脱机I/O操作，使外围I/O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称为**SPOOLing**（**Simultaneous Periphernal Operating OnLine**）。

假脱机技术的作用：通过缓冲方式，将独占设备改造为共享设备。

目的：缓和CPU和I/O设备速度不匹配

6.6.2 假脱机(SPOOLing)系统

❖ 2、SPOOLing系统的组成

1> 输入井和输出井（外存上）

输入井：用来模拟脱机输入磁盘设备，用来暂存I/O输入设备输入的数据；

输出井：用来模拟脱机输出磁盘设备，用来暂存用户程序输出的数据。

2> 输入缓冲区和输出缓冲区（内存中）

输入设备→输入缓冲区→输入井→用户区

用户区→输出井→输出缓冲区→输出设备

3> 输入进程和输出进程

用来模拟脱机I/O时的外围控制机。

6.6.2 假脱机(SPOOLing)系统

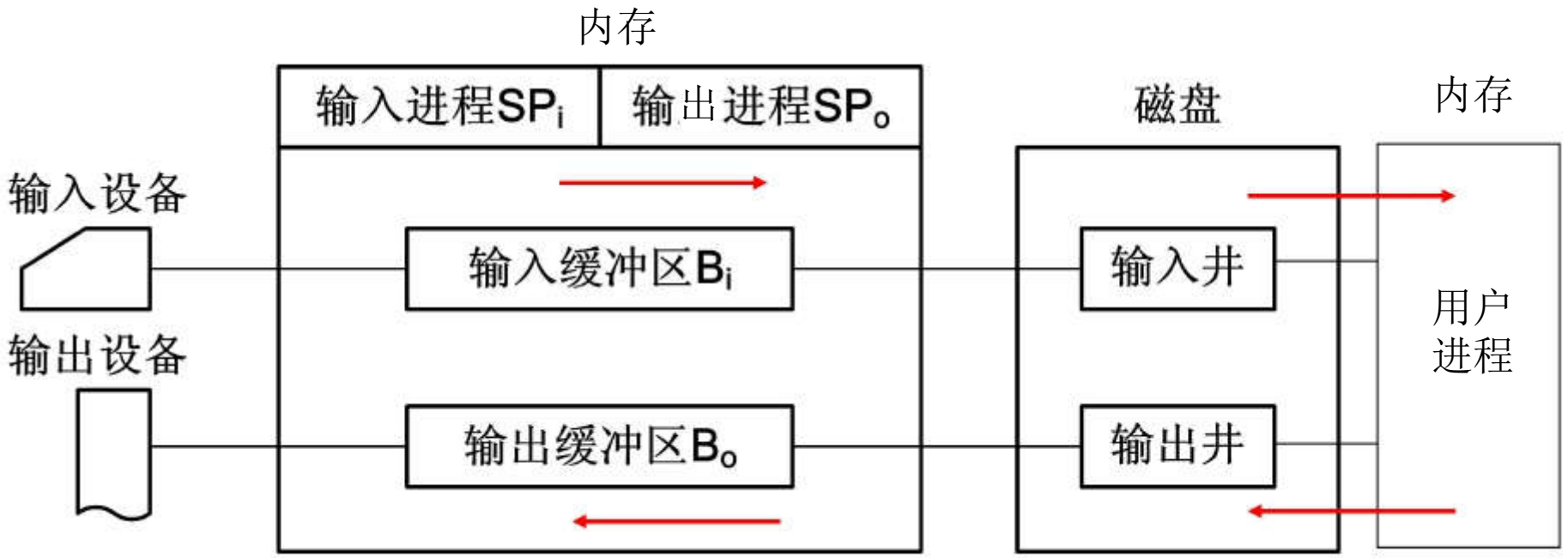


图6-21 SPOOLing系统的组成

6.6.2 假脱机(SPOOLing)系统

❖ 3、假脱机打印机系统

打印机为独占设备，利用**SPOOLing**技术，可将之改造为共享设备。

用户请求打印时，**SPOOLing**系统处理如下：

- ✓ **1>** 由假脱机管理进程在输出井中为之申请一个空闲磁盘块区， 并将要打印的数据送入其中;
- ✓ **2>** 假脱机管理进程再为用户进程申请一张空白的用户请求打印表，并将用户的打印要求填入其中，再将该表挂到请求打印队列上;
- ✓ **3>** 如果打印机空闲，假脱机打印进程将从请求打印队列的队首取出一张请求打印表，按表中要求将要打印的数据从输出井送到输出缓冲区，再由打印机打印。

6.6.2 假脱机(SPOOLing)系统

❖ 4、SPOOLing系统的特点

1> 提高**I/O**速度

对低速设备操作变为对输入/输出井操作

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

3> 实现了虚拟设备功能

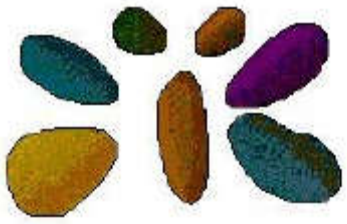
将独占设备转换为若干台逻辑设备

本章主要内容

- 6.1 I/O系统的功能、模型和接口
- 6.2 I/O设备和设备控制器
- 6.3 中断机构和中断处理程序
- 6.4 设备驱动程序
- 6.5 与设备无关的I/O软件
- 6.6 用户层的I/O软件
- 6.7 缓冲区管理
- 6.8 磁盘存储器的性能和调度

6.7 缓冲区管理

- 6.7.1 缓冲的引入
- 6.7.2 单缓冲和双缓冲
- 6.7.3 循环缓冲
- 6.7.4 缓冲池



6.7.1 缓冲的引入

- 引入缓冲区的原因

  1> 缓和CPU与I/O设备间速度不匹配的矛盾。

  例如： CPU与打印机、CPU与输入设备

  2> 减少对CPU的中断频率， 放宽对CPU中断响应时间的限制。 （参见教材P209图6-22）

  3> 解决数据粒度不匹配的问题。

  4> 提高CPU和I/O设备之间的并行性，以提高系统的吞吐量和设备的利用率。

6.7.1 缓冲的引入

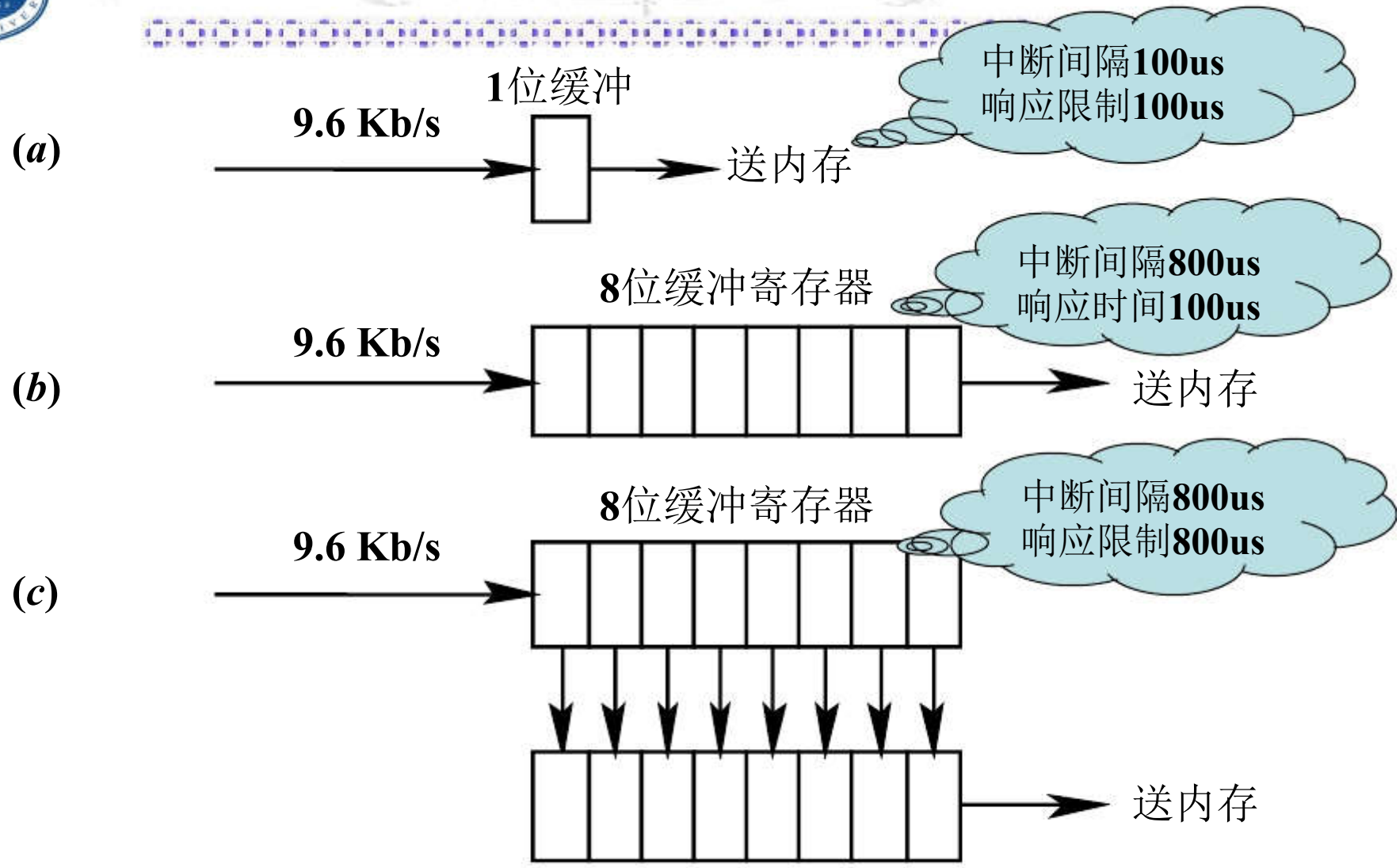


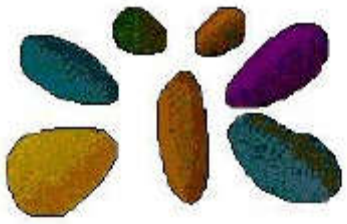
图 6-22 利用缓冲寄存器实现缓冲

6.7 缓冲区管理（课下自学）

- 6.7.1 缓冲的引入
- 6.7.2 单缓冲和双缓冲
- 6.7.3 循环缓冲
- 6.7.4 缓冲池

6.7.2、6.7.3 —— 特定进程专用缓冲

6.7.4 —— 多个进程公用缓冲



6.7.2 单缓冲和双缓冲

❖ 1、单缓冲（Signal Buffer）

Max(C,T)+M /块(行)

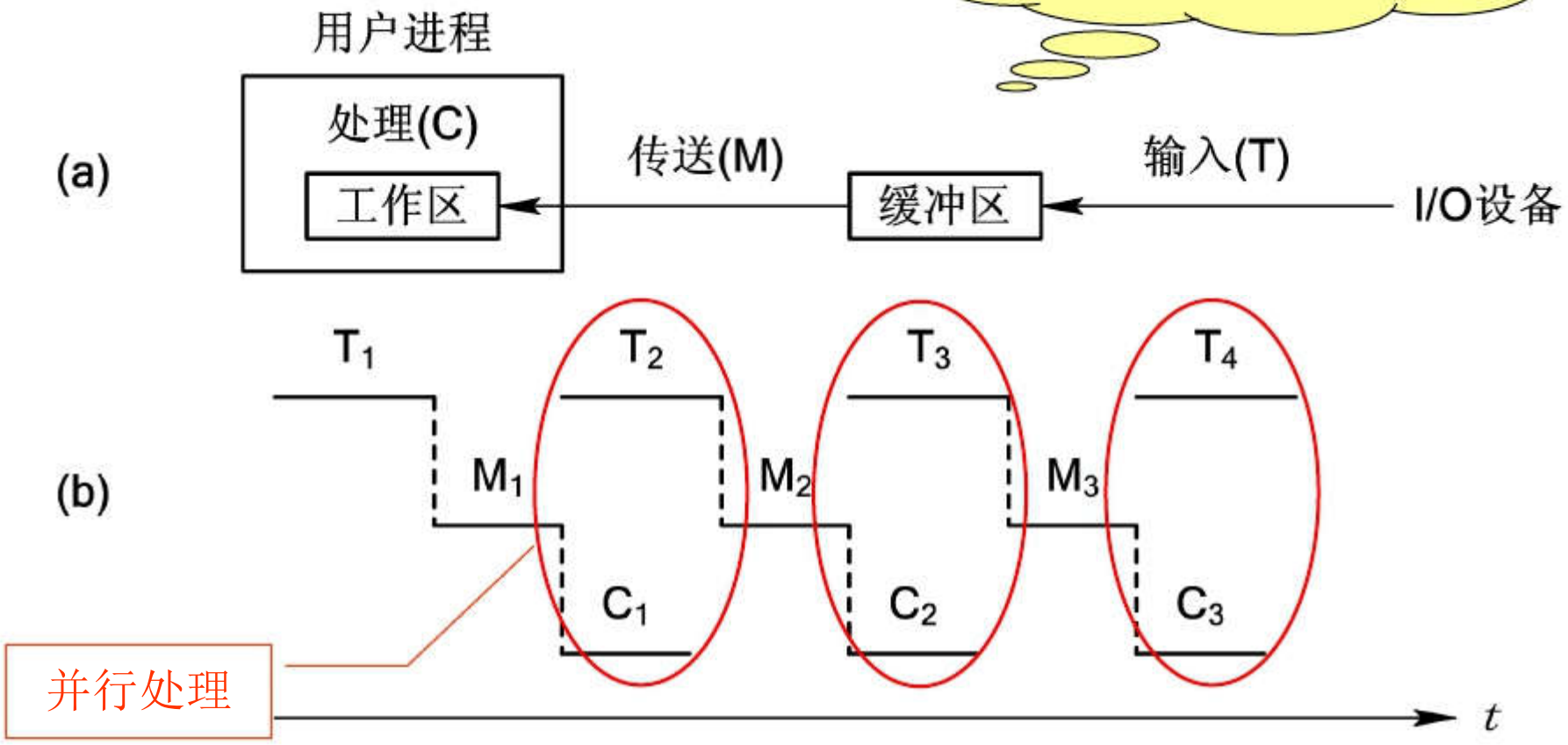


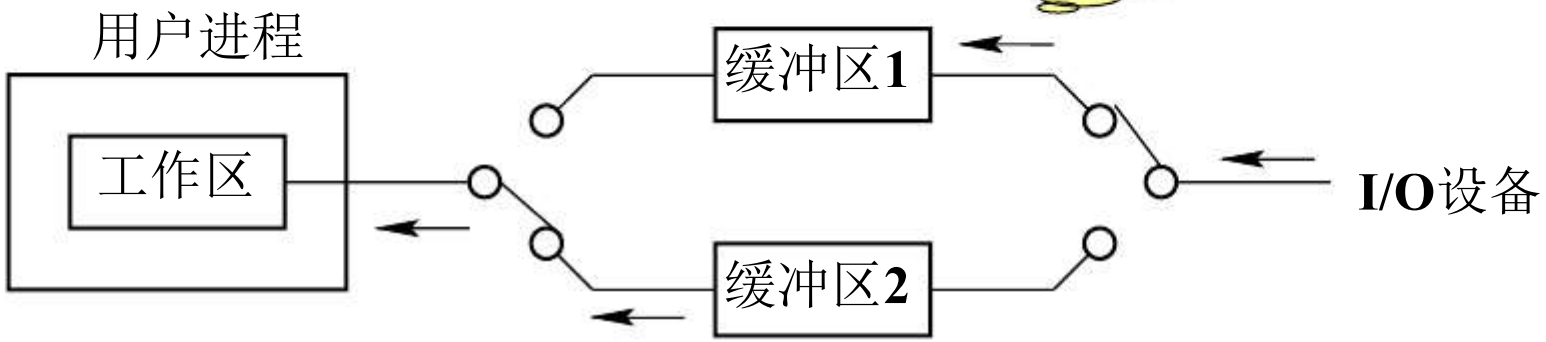
图 6-23　单缓冲工作示意图

6.7.2 单缓冲和双缓冲

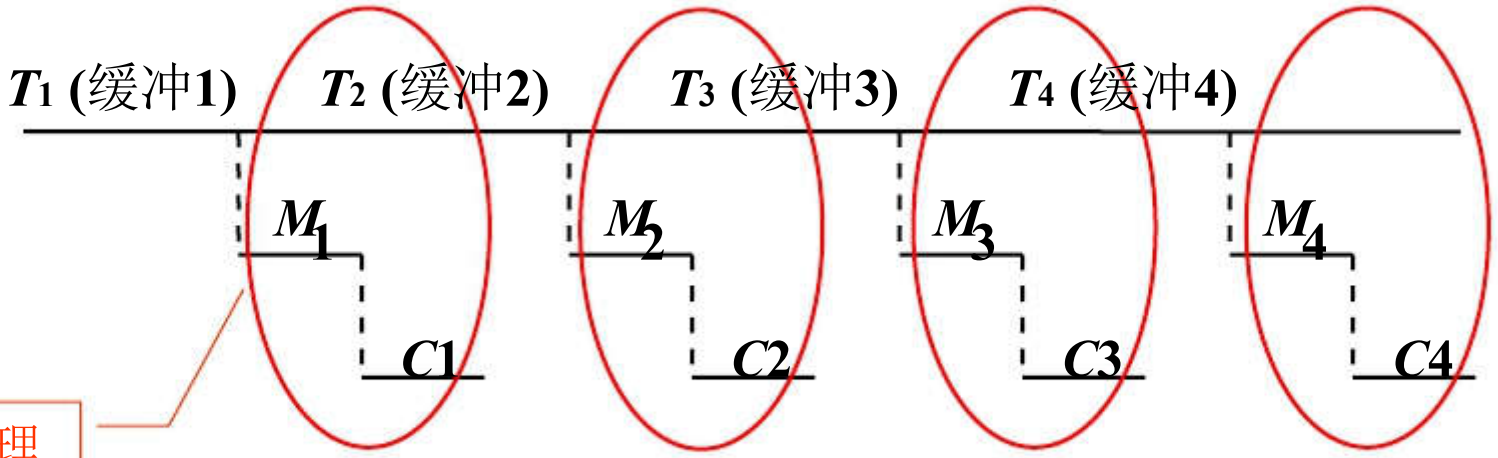
❖ 2、双缓冲（Double Buffer）

Max(M+C,T)　/块(行)

(a)



(b)



并行处理

图6-24　双缓冲工作示意图

6.7.2 单缓冲和双缓冲

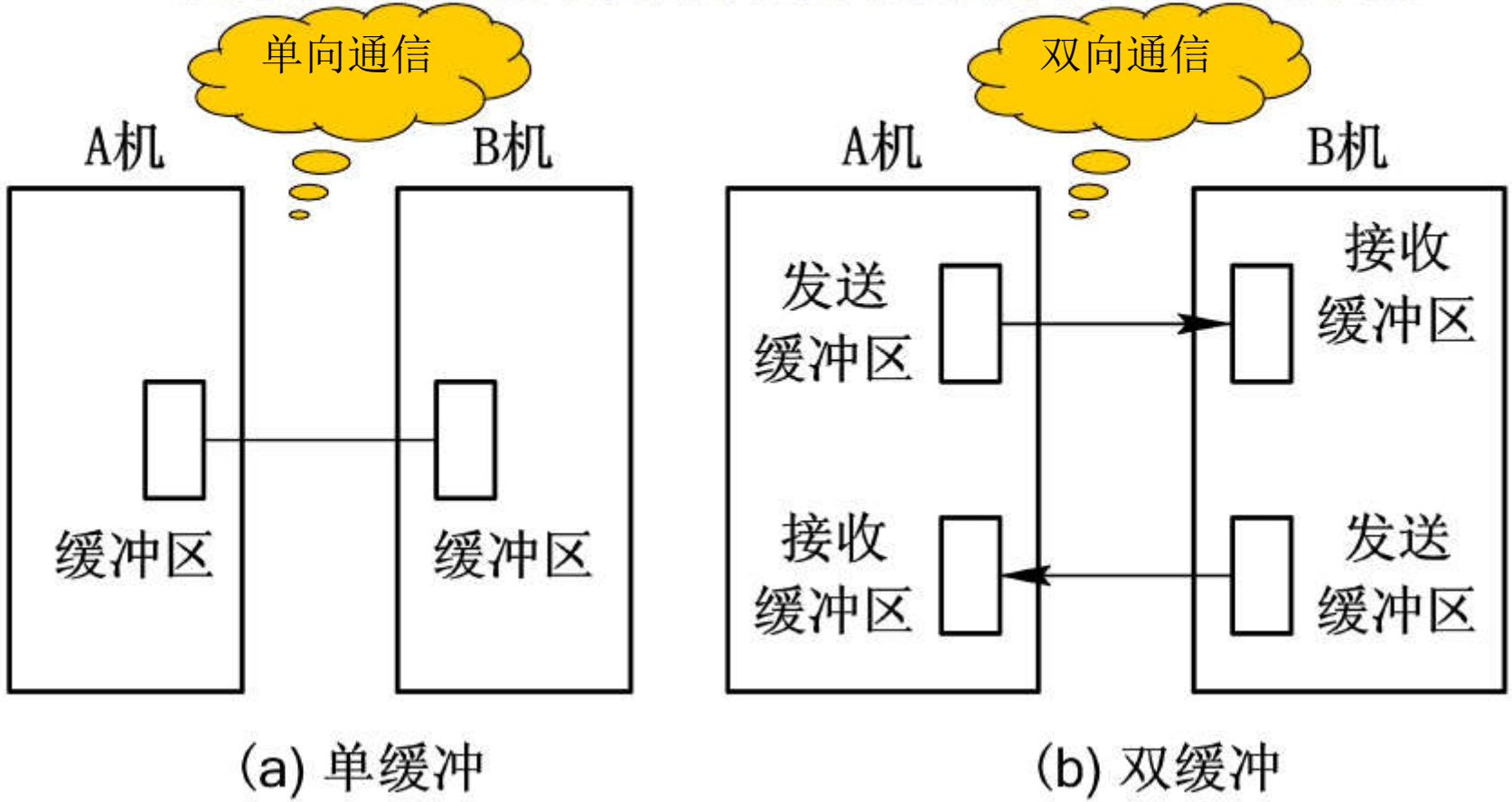


图6-25　双机通信时缓冲区的设置

6.7.2 单缓冲和双缓冲

- 【补充示例1】现有假设T是从磁盘输入一块数据的时间，C是CPU对一块数据进行处理的时间，而M是将一块数据从缓冲区传送到工作区的时间。当用户进程按顺序访问的方式处理大量的数据时，请问在单缓冲和双缓冲的情况下，系统对一块数据的处理时间分别是多少？

6.7.2 单缓冲和双缓冲

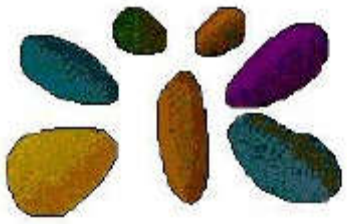
❖ 【解】

1> 单缓冲情形：数据从**I/O**控制器到缓冲区、数据由缓冲区到工作区和**CPU**从工作区取出数据进行处理，必须串行操作。但**CPU**在处理一块数据时，**I/O**控制器可输入下一块数据。因此，系统对一块数据处理的时间为**Max**（**C**，**T**）**+M**。

2> 双缓冲情形： 数据由**I/O**控制器到双缓冲，以及数据由双缓冲到工作区，可以并行工作，因此系统对一块数据处理的时间为**Max**（**T**，**M+C**）。由于**M<<T**，则此时系统对一块数据处理的时间约为**Max**（**T** ，**C**）。

6.7 缓冲区管理

- 6.7.1 缓冲的引入
- 6.7.2 单缓冲和双缓冲
- 6.7.3 循环缓冲
- 6.7.4 缓冲池



6.7.3 循环缓冲

❖ 1、循环缓冲的引入

当**CPU**处理数据的速度快于数据**I/O**的速度时，双缓冲也难以满足需要，用户进程会经常因等待**I/O**操作而阻塞；可引入多个缓冲，组织成循环缓冲的形式。

❖ 2、循环缓冲的组成

三类缓冲区

用于装输入数据的空缓冲区**R**

已装满数据的满缓冲区**G**

计算进程正在使用的现行工作缓冲区**C**

三类指针

指示计算进程下一可用缓冲区的指针**Nextg**

指示输入进程下一可用空缓冲区的指针**Nexti**

指示计算进程正在使用的缓冲区的指针**Current**

6.7.3 循环缓冲

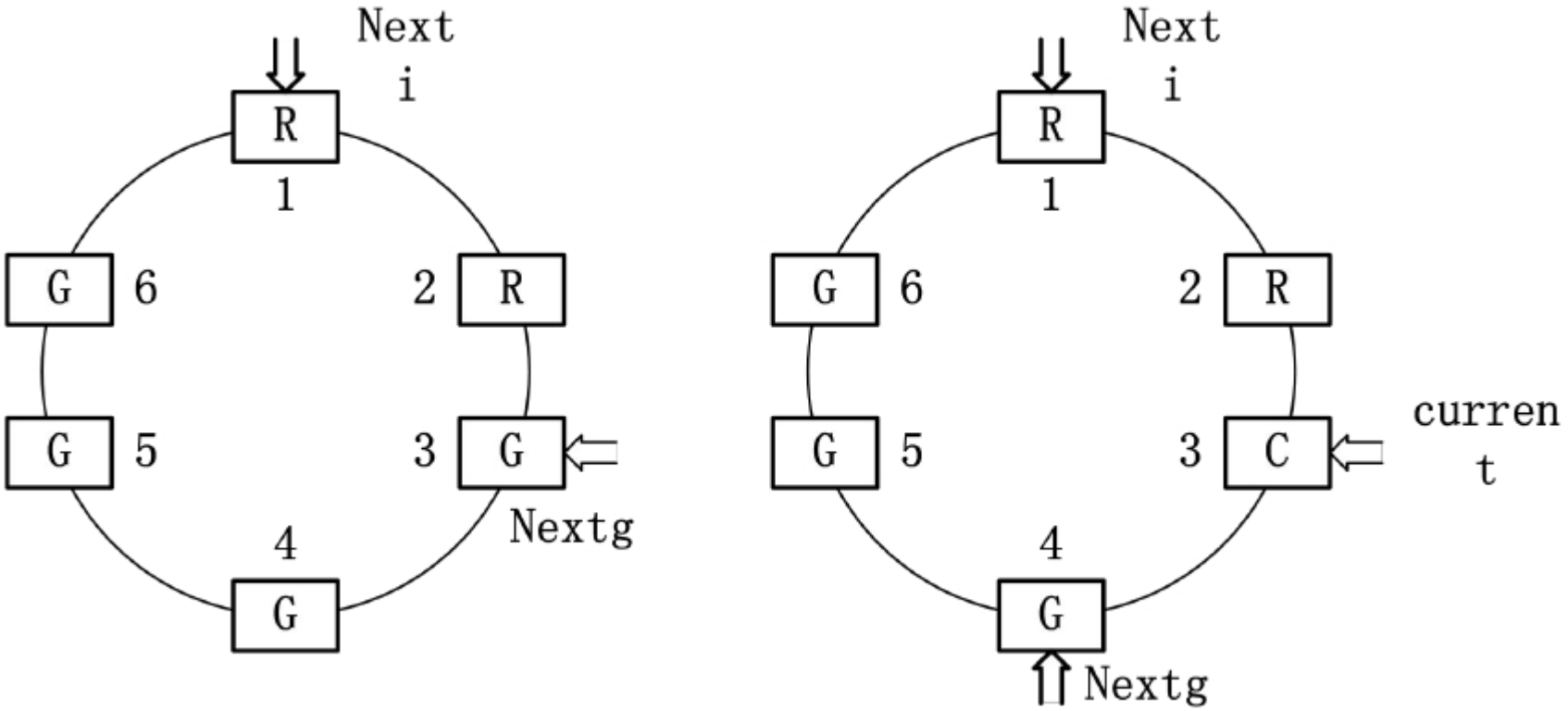


图 6-26　循环缓冲

6.7.3 循环缓冲

❖ 3、循环缓冲的使用（了解）

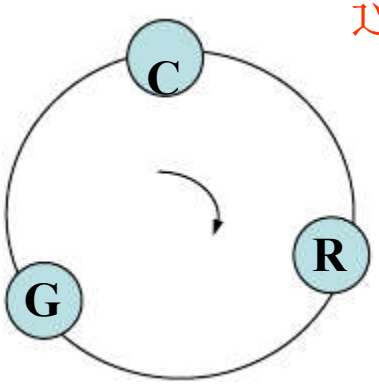
1> Getbuf过程

为计算进程和输入进程提供缓冲区，并移动指针

2> Releasebuf过程

计算进程提取完一个**C**类缓冲区的数据，就调用**Releasebuf**过程将该缓冲区释放；

输入进程装满一个**R**类缓冲区，就调用**Releasebuf**过程将该缓冲区改为**G**类缓冲区。



6.7.3 循环缓冲

❖ 4、进程同步（了解）

1> 系统受计算限制

Nexti指针追赶上**Next**g指针，输入进程速度大于计算进程，最终将导致全部空缓冲区装满，输入进程阻塞。

2> 系统受**I/O**限制

Nextg指针追赶上**Next**i指针，计算进程速度大于输入进程，最终将导致全部满缓冲区提取完，计算进程阻塞。

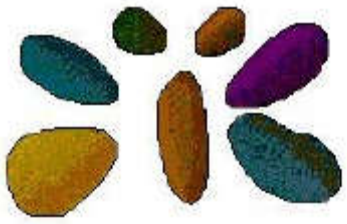
6.7 缓冲区管理

- 6.7.1 缓冲的引入
- 6.7.2 单缓冲和双缓冲
- 6.7.3 循环缓冲

特定进程专用缓冲

- 6.7.4 缓冲池

多个进程公用缓冲



6.7.4 缓冲池

❖ 1、缓冲池的基本概念

缓冲池：是系统提供的公用缓冲，即把系统内的缓冲区统一管理起来，变专用为通用。

当某进程需要使用缓冲区时，提出申请，由管理程序从缓冲池提取一个缓冲区分配给它，用完后释放缓冲区，这样可用少量的缓冲区为更多的进程服务。

6.7.4 缓冲池

❖ 2、缓冲池的组成

1> 缓冲区的类型

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

2> 缓冲区的组织

① 空缓冲队列**emq**

② 输入队列**inq**

③ 输出队列**outq**

6.7.4 缓冲池

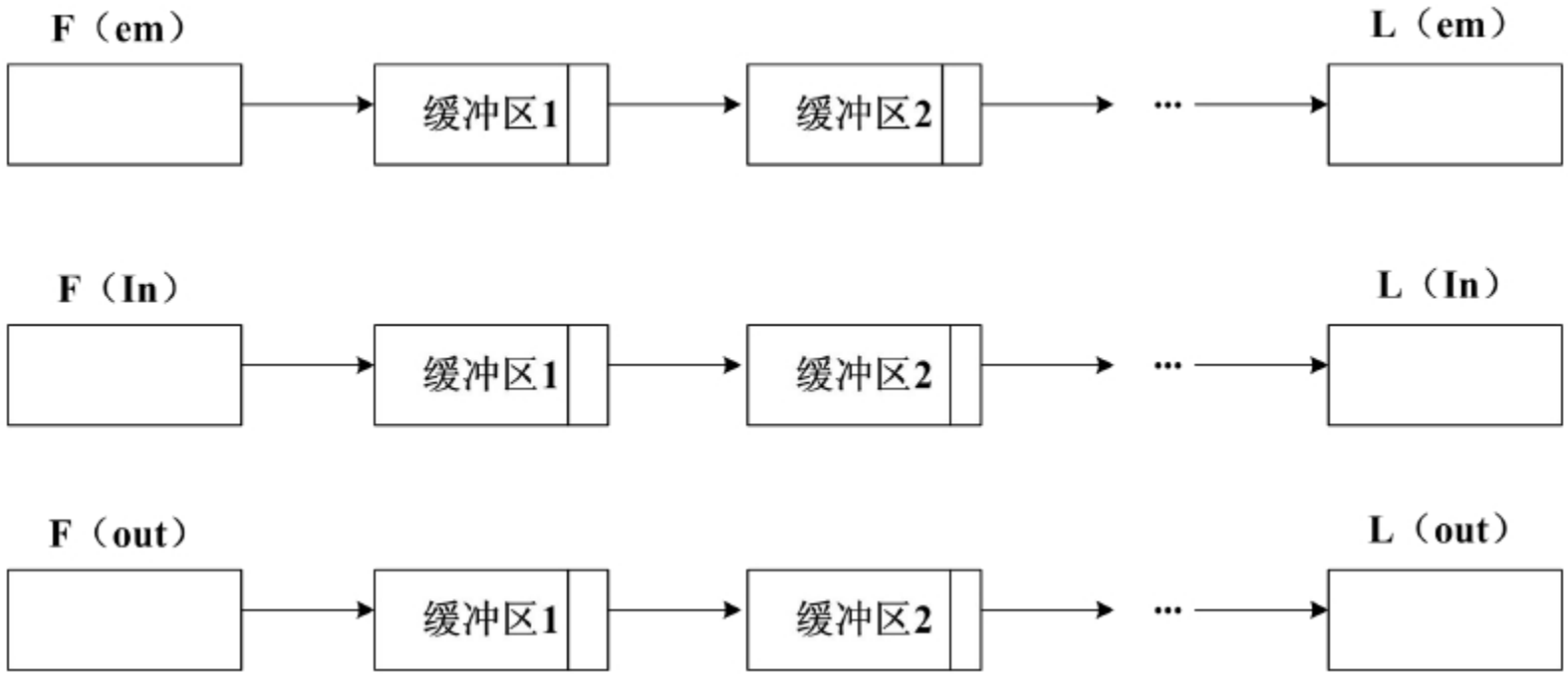


图　三种缓冲区的组织

6.7.4 缓冲池

❖ 3、 Getbuf 过程和Putbuf过程（了解）

缓冲池中的队列是临界资源，因此多个进程访问同一个缓冲区队列必须互斥访问，不同队列之间的访问应该同步。

```
Getbuf(type)
Begin
  wait(RS(type));
  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));
end
```

6.7.4 缓冲池

❖ 4、缓冲池的工作方式（了解）

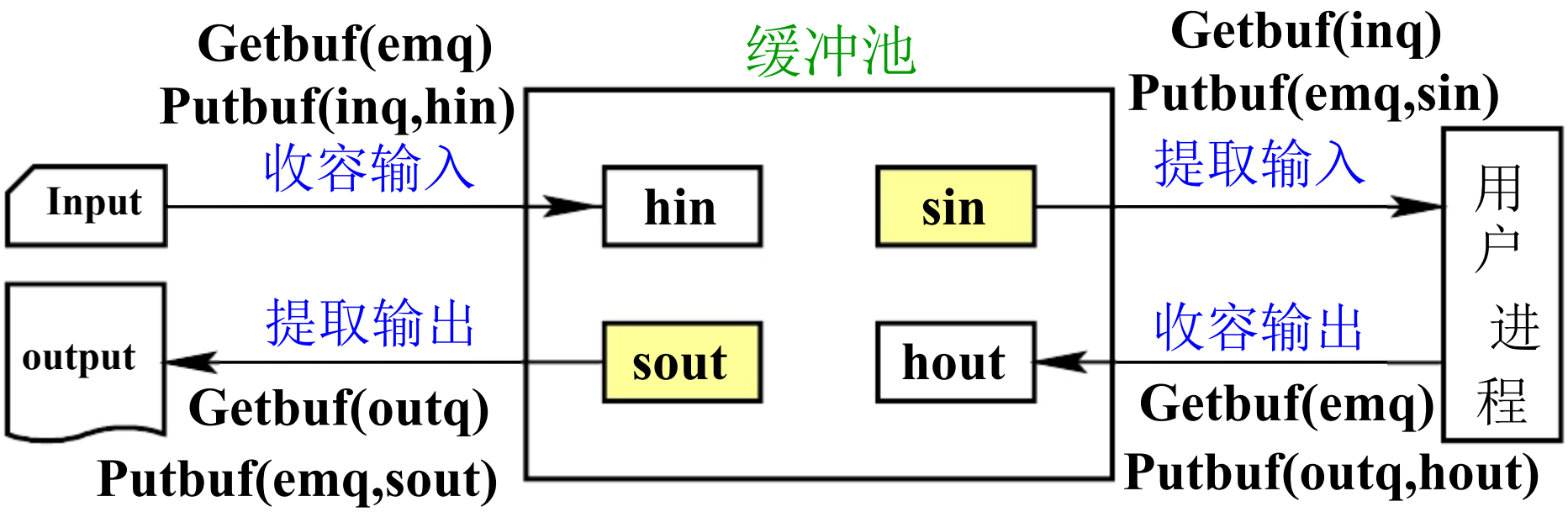


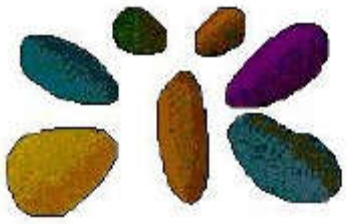
图6-27 缓冲池的工作方式

本章主要内容

- **6.1 I/O系统的功能、模型和接口**
- **6.2 I/O设备和设备控制器**
- **6.3 中断机构和中断处理程序**
- **6.4 设备驱动程序**
- **6.5 与设备无关的I/O软件**
- **6.6 用户层的I/O软件**
- **6.7 缓冲区管理**
- **6.8 磁盘存储器的性能和调度**

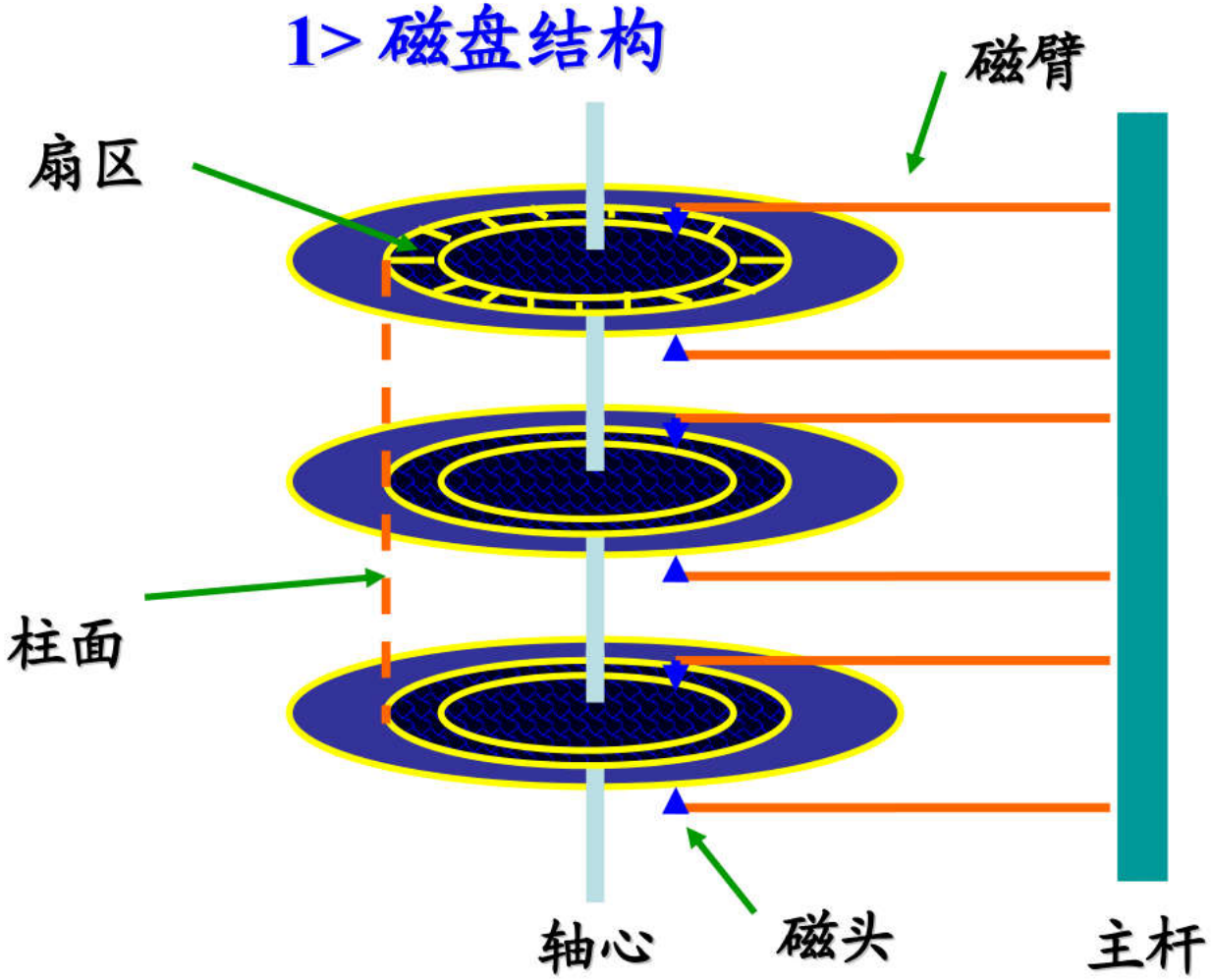
6.8 磁盘存储器的性能和调度

- 6.8.1 磁盘性能简述
- 6.8.2 磁盘调度



6.8.1 磁盘性能简述

❖ 1、数据的组织和格式



磁盘记录块总数=
盘面数*磁道数*扇区数

图6-28 磁盘驱动器结构

6.8.1 磁盘性能简述

2> 磁盘格式化

格式化后每个扇区包含两个字段：**1>** 标识符字段；**2>** 数据字段 。

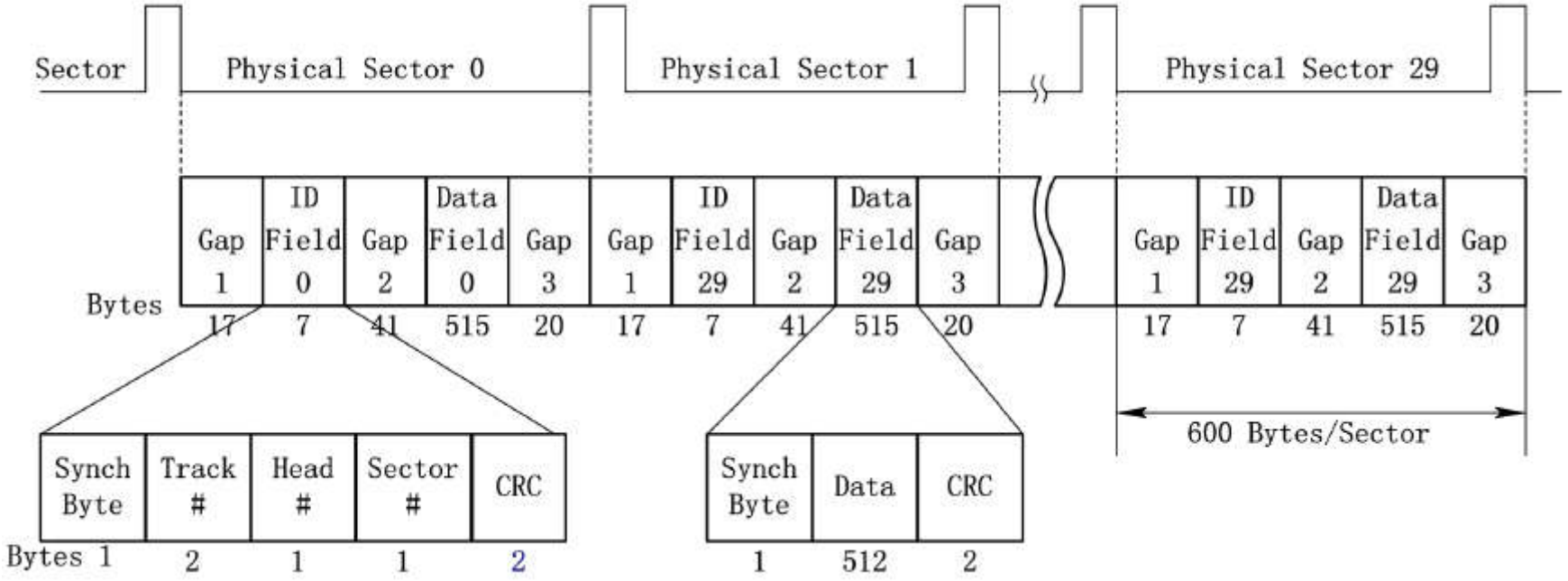


图6-29 磁盘的格式化

6.8.1 磁盘性能简述

❖ 2、磁盘的类型

1> 固定磁头磁盘

磁盘每条磁道上都有一读/写磁头，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。

2> 移动磁头磁盘

磁盘每一个盘面仅配有一个磁头，为能访问该盘面上所有磁道，该磁头必须能移动以进行寻道，因而I/O速度较慢。

Ts、Tr与所读写数据无关，是Ta的主要组成部分，因而数据集中传输可减少寻道次数，提高传输效率

❖ 3、磁盘访问时间

访问时间 **Ta=Ts+1/2r+b/rN**

1> 寻道时间 **Ts=m*n+s** **5-30ms**

寻道时间是指磁头移动到磁道上所需时间

m-常量，**n-**移动磁道数，**s-**磁臂启动时间

2> 旋转延迟时间 **T_r=1/2r**（均值） **2-3ms**

旋转延迟时间是指扇区旋转到磁头下所需时间

r-磁盘每秒转数

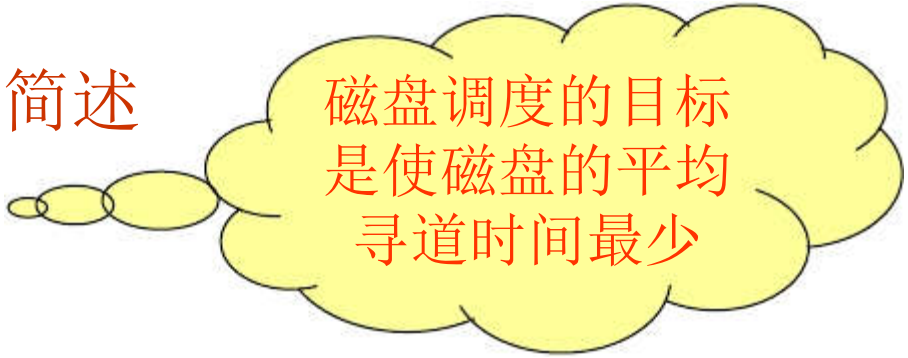
3> 数据传输时间 **T_t＝b/(rN)**

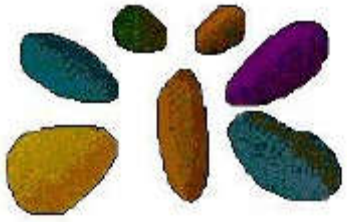
数据传输时间是指从磁盘读/写数据所需时间

b-读/写字节数，**N-**每个磁道上的字节数

6.8 磁盘存储器的性能和调度

- 6.8.1 磁盘性能简述
- 6.8.2 磁盘调度





6.8.2 磁盘调度

❖ 1、先来先服务（FCFS）

原理：根据进程请求访问磁盘的先后次序进行调度。

优点：简单、貌似公平，每个进程得到满足。

缺点：平均寻道距离较大，仅适用于请求磁盘I/O的进程数目较少的场合。

（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6-30 FCFS调度算法

6.8.2 磁盘调度

例：图6-30从100号磁道开始，进程对磁道的先后请求次序为：55、58、39、18、90、160、150、38、184，采用FCFS算法调度次序为：

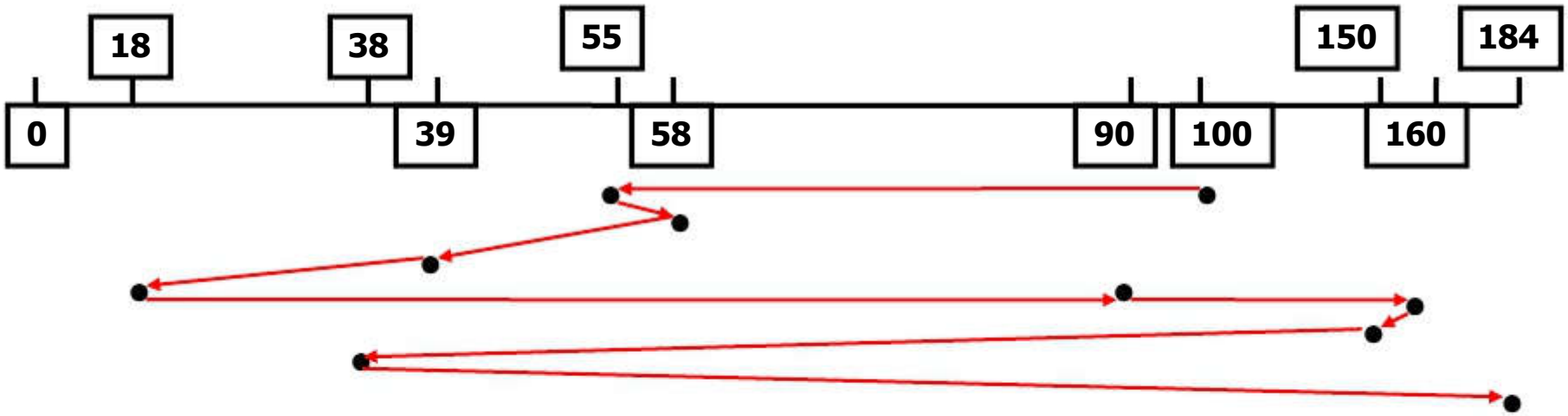


图6-30 FCFS调度算法

6.8.2 磁盘调度

❖ 2、最短寻道时间优先（SSTF）

原理：每次选择离当前磁道距离最近的磁道进行调度（贪心算法）。

优点：比**FCFS**有更好的寻道性能。

缺点：可能导致某些进程出现“饥饿”现象，在很长时间内其磁盘请求得不到满足。

（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6-31 SSTF调度算法

6.8.2 磁盘调度

例：假设当前磁道为**100**，进程请求磁道队列为：**55**、**58**、**39**、**18**、**90**、**160**、**150**、**38**、**184**，则采用**SSTF**算法调度次序为：

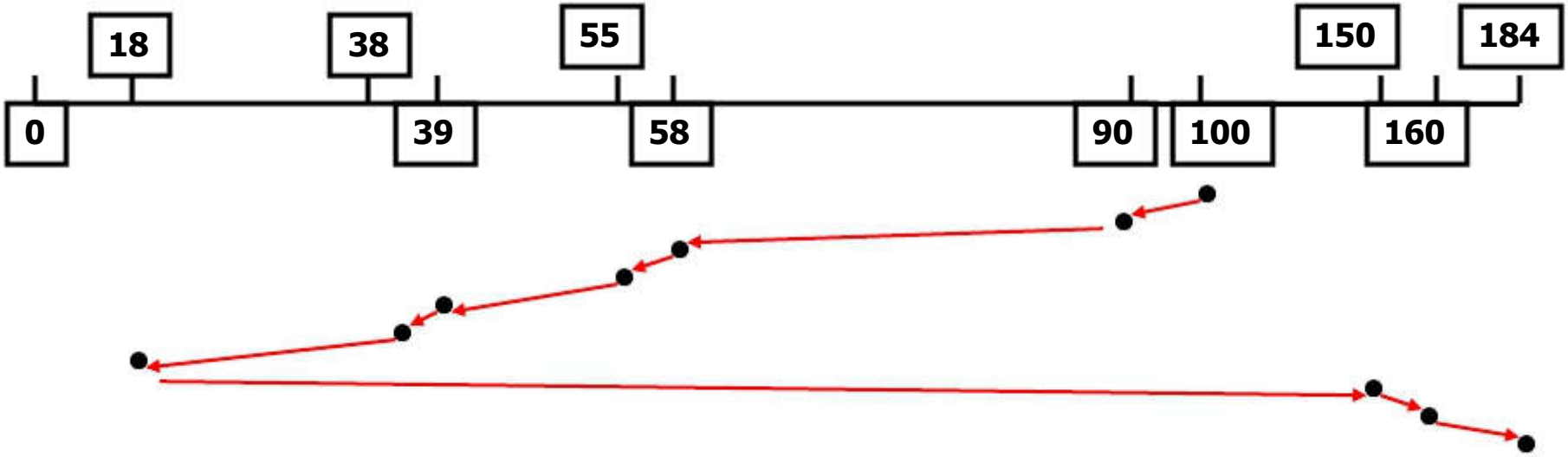


图6-31 SSTF调度算法

6.8.2 磁盘调度

❖ 3、扫描算法（SCAN）

原理：磁头像电梯一样往返移动，在移动过程中再采用**SSTF**算法；也称电梯算法。

优点：性能较好，可以避免饥饿现象的出现。

缺点：某些进程请求存在等待延迟现象。

（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

6.8.2 磁盘调度

例：假设当前磁道为**100**，向磁道号增加方向访问，进程请求磁道队列为：**55、58、39、18、90、160、150、38、184**，则采用**SCAN**算法调度次序为：

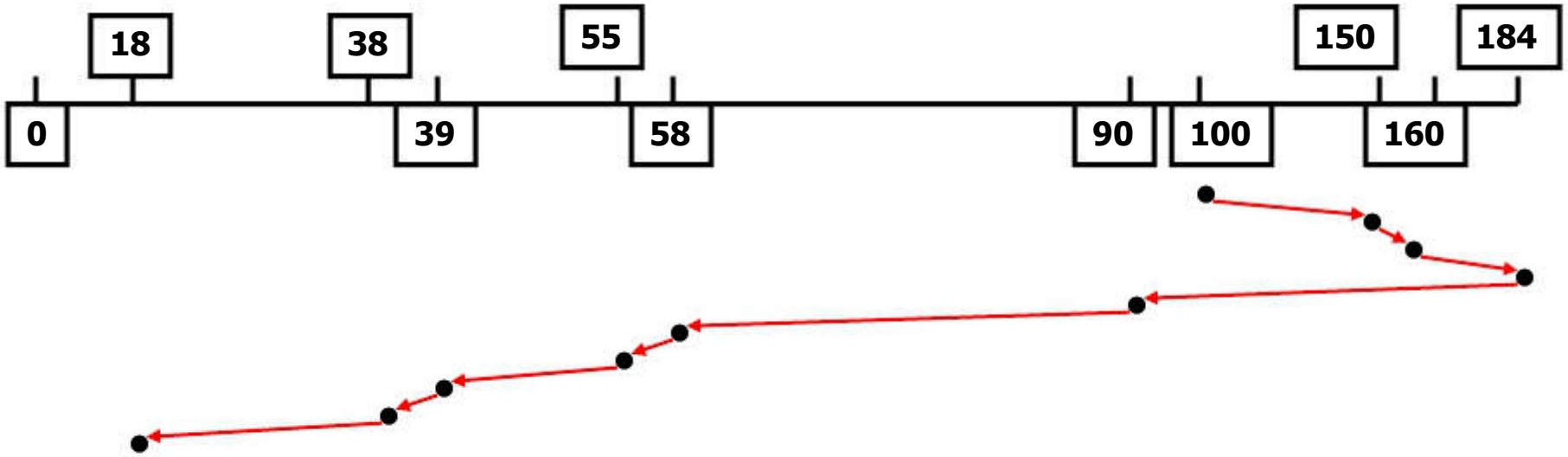


图6-32 SCAN调度算法

6.8.2 磁盘调度

❖ 4、循环扫描算法（CSCAN）

原理：磁头循环单向移动，在移动过程中再采用**SSTF**算法。

优点：能改善某些进程的请求等待延迟。

（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：36.8	

图6-33 CSCAN调度算法

6.8.2 磁盘调度

例：假设当前磁道为**100**，向磁道号增加方向访问，进程请求磁道队列为：**55、90、58、39、18、90、160、150、38、184**，则采用**CSCAN**算法调度次序为：

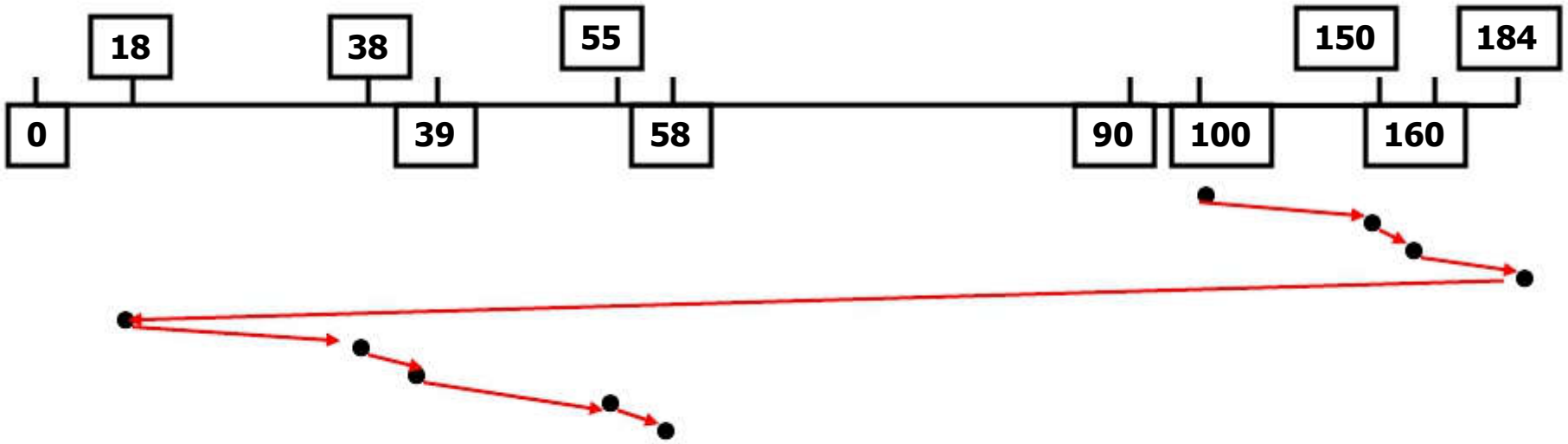


图6-33 CSCAN调度算法

6.8.2 磁盘调度

❖ 5、N-Step-SCAN和FSCAN算法

1> SSTF、SCAN、CSCAN算法存在的问题

a) 这三种算法对于先来和后来的磁盘请求同等处理，不合理。

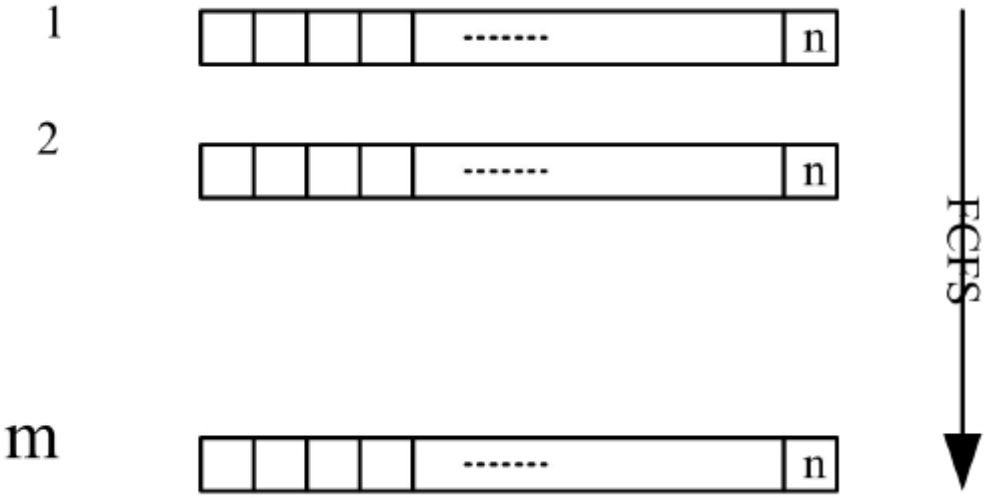
b) 容易出现“磁臂粘着”现象，即一个或多个进程对某一磁道有较高的访问频率，从而不停对该磁道进行**I/O**操作，使得其它磁盘请求在很长时间内得不到处理。

6.8.2 磁盘调度

2> N-Step-SCAN算法

原理：根据磁盘请求到来的先后次序将其组织成若干个长度为N的请求队列，然后依次对每一队列采用SCAN 算法进行处理。

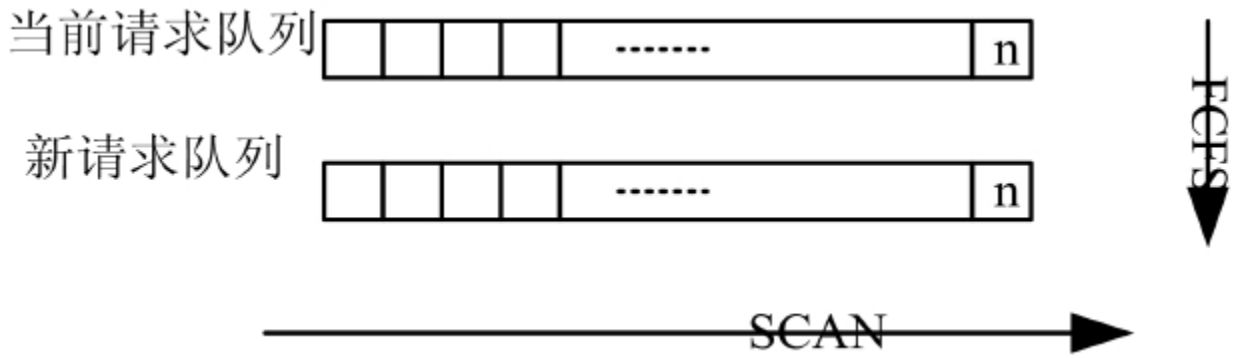
性能：当N=1时，算法蜕化为FCFS；当N很大时，算法性能接近SCAN算法。



6.8.2 磁盘调度

3> FSCAN算法

原理： **FSCAN是N步SCAN算法的简化，即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，采用SCAN算法进行处理；另一个列队是在扫描期间新出现的磁盘请求组成的队列。**



6.8.2 磁盘调度

- 【补充示例2】 假定磁盘有**200** 个柱面，编号**0~199**，当前存取臂的位置在**143** 号柱面上，并刚刚完成了**125** 号柱面的服务请求，如果请求队列的先后顺序是：**86**，**147**，**91**，**177**，**94**，**150**，**102**，**175**，**130**；试问：为完成上述请求，下列算法存取臂移动的总量是多少？并给出存取臂移动的顺序。

(1) 先来先服务算法**FCFS**；
(2) 最短寻道时间优先算法**SSTF**；
(3) 扫描算法**SCAN**；
(4) CSCAN算法。

6.8.2 磁盘调度

❖ 【解】

1> 先来先服务　　磁头移动顺序为：

143→86→147→91→177→94→150→102→175→130，
磁头移动共**565**柱面。

2> 最短寻道时间优先（SSTF）　磁头移动顺序为：

143→147→150→130→102→94→91→86→175→177，
磁头移动共**162**柱面。

3> SCAN算法　磁头移动顺序为：

143→147→150→175→177→130→102→94→91→86，
磁头移动共**125**柱面。

4> CSCAN算法　　磁头移动顺序为：

143→147→150→175→177→86→91→94→102→130，
磁头移动共**169**柱面。

本章小结

- **I/O设备（设备、控制器、通道）**
- **I/O系统的五个层次**
  - 设备独立性软件*
  - 设备驱动程序*
  - **I/O中断处理程序***
- 四种**I/O**控制方式*
- 四种缓冲技术*
- **SPOOLing技术****
- 磁盘调度算法**
  - **FCFS、SSTF、SCAN、CSCAN、NStepSCAN、FSCAN****

本章小结

- 重要概念

  设备控制器、通道、中断、收容输入、提取输入、收容输出、提取输出、设备独立性、**SPOOLing**等

本章作业

- 要求：

  一定要做在作业本上

- 交作业日期：

  两周后

- 作业内容：

  操作系统第**6**章网络在线测试

  补充题**1**道

本章作业

- 补充题1：假设磁盘有**200** 个磁道，磁盘请求队列中是一些随机请求，它们按照到达的次序分别处于**98**、**183**、**37**、**122**、**14**、**124**、**65**、**67**号磁道上，当前磁头在**53**号磁道上，并向磁道号减小的方向上移动。请给出按先来先服务算法（**FCFS**）、最短寻道时间优先算法**(SSTF)**、扫描算法（**SCAN**）及循环扫描算法（**CSCAN**）算法进行磁盘调度时满足请求的次序，并算出它们的平均寻道长度？

本章课程结束！谢谢大家！