西安电子科技大学
XIDIAN UNIVERSITY

计算机科学与技术学院
SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY
国家示范性软件学院
NATIONAL PILOT SCHOOL OF SOFTWARE ENGINEERING

操作系统原理

第七章 设备管理

主讲：黄伯虎

第六章 小结

基本内容及重点	1. 文件的基本概念 2. 文件的结构(逻辑结构；物理结构) 3. 文件的存取方式(顺序存取；随机存取) 4. 文件目录、目录结构和目录的维护(打开文件机构) 5. 文件共享(硬链接，软连接) 6. 外存空间管理(空闲区表；位示图；空闲块链；UNIX成组链接法) 7. 文件性能(磁盘调度，旋转延迟，信息的优化分布)
理解重点	• 文件的物理结构(UNIX三级索引结构) • 文件目录结构(UNIX树形目录)/打开文件机构 • UNIX外存空闲空间管理：UNIX成组链接法 • 磁盘调度、旋转调度问题

一、基本概念

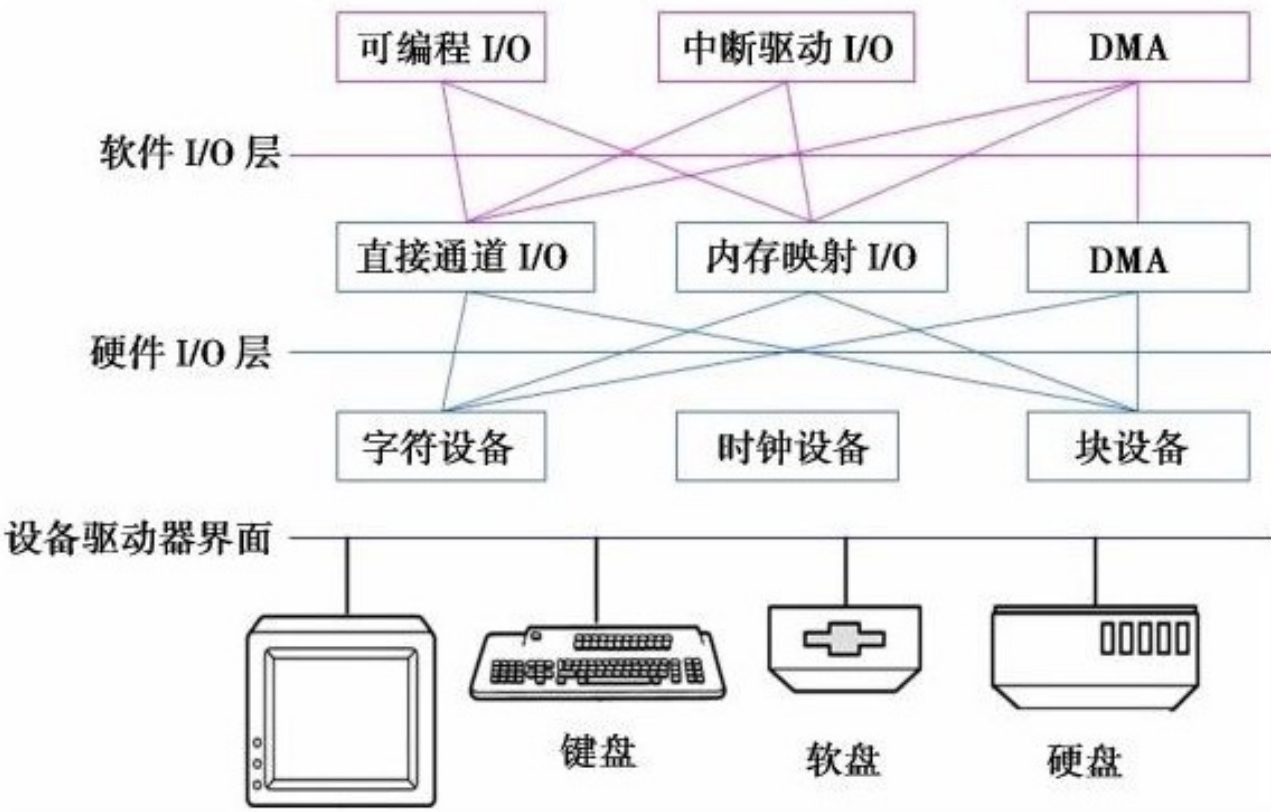
设备的概念

- 一个计算机系统就是由大量的设备构成的，例如：CPU，磁盘，显卡、显示器、鼠标、键盘等。这些设备的特点和功能各不相同。在这些设备中，有一类是作为计算机系统与外界交互的工具使用的，它具体负责计算机与外部的输入输出(I/O)工作，我们称这类设备为外部设备，简称为外设(I/O设备)，本章重点研究的就是操作系统中对这类设备的管理策略。

设备管理的目标

- 提高设备的利用率
  - 就是提高CPU与I/O设备之间的并行操作程度。
- 为用户提供方便、统一的界面
  - 方便：屏蔽外部设备的差异；
  - 统一：是指对不同的设备尽量使用统一的操作方式。

一、基本概念

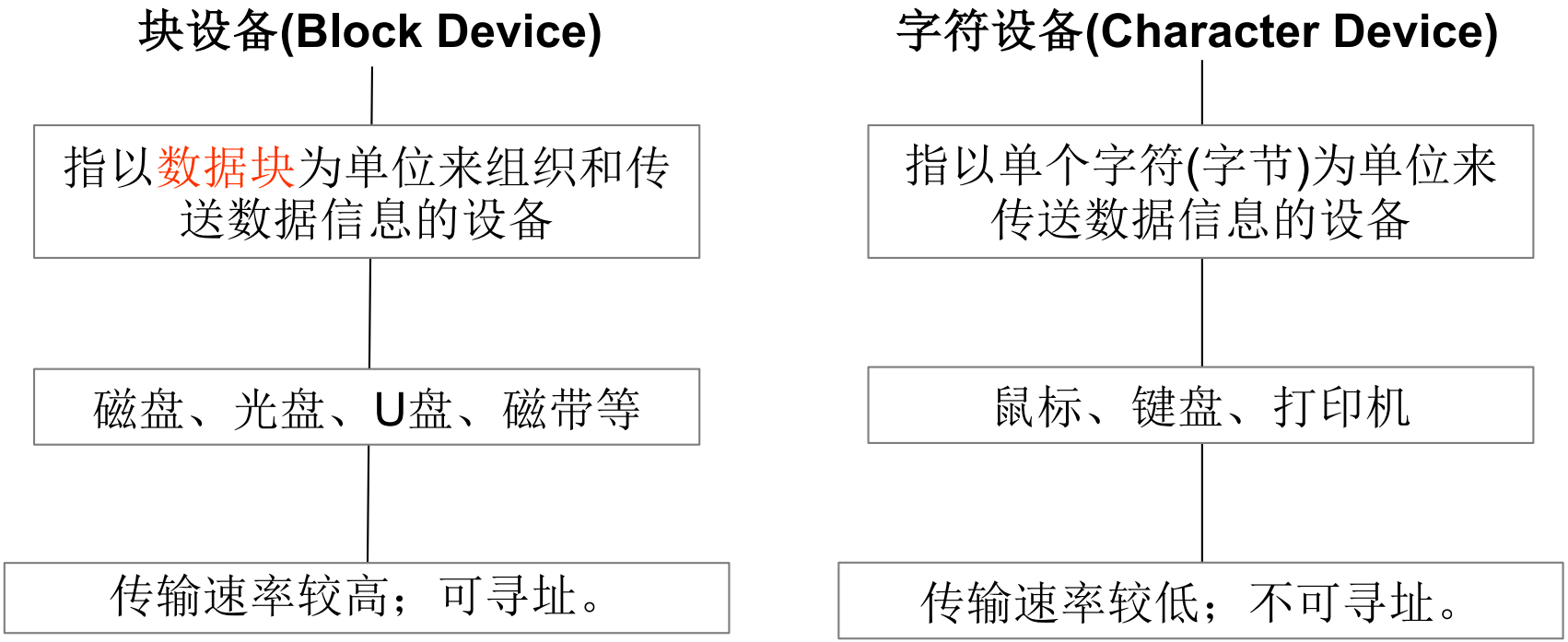


操作系统中的外设管理其实是对外部设备的以一种抽象！

二、外设的分类

I/O设备的分类

❖ 按数据组织(存储/传输方式)分类



二、外设的分类

❖ 按数据传输率分类

- 低速设备：指传输速率为每秒钟几个字节到数百个字节的设备。典型的设备有键盘、鼠标、语音的输入等；
- 中速设备：指传输速率在每秒钟数千个字节至数十千个字节的设备。典型的设备有行式打印机、激光打印机等；
- 高速设备：指传输速率在数百千个字节至数兆字节的设备。典型的设备有磁带设备、磁盘设备、光盘设备等。

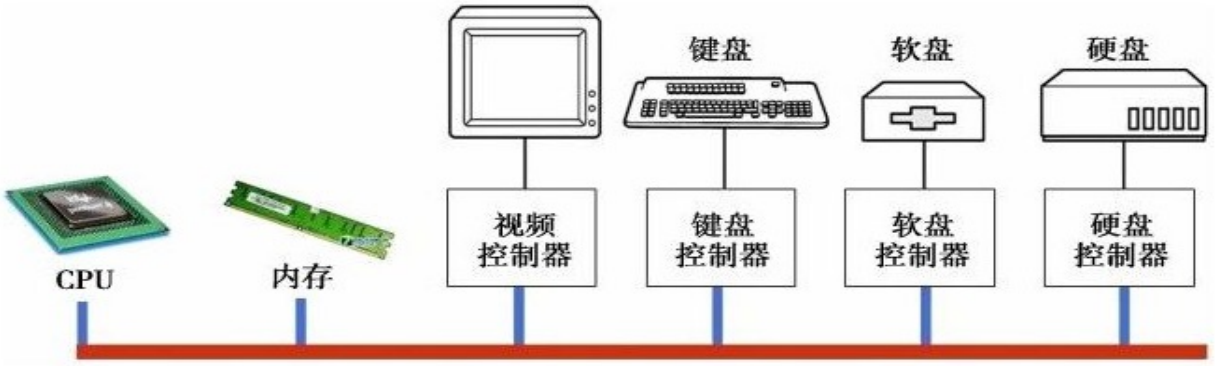
二、外设的分类

不同设备的数据传输速率

输入输出设备	每秒数据传输速率	输入输出设备	每秒数据传输速率
键盘	10B	快速以太网	12.5MB
鼠标	100B	ISA 总线和 EIDE 磁盘	16.7MB
56K 调制解调器	7KB	火线	50MB
双向 ISDN 线路	16KB	XGA 监视器	60MB
激光打印机	100KB	SONET OC－12	78MB
扫描仪	400KB	SCSI Ultra 2 磁盘	80MB
以太网	1.25MB	千兆级以太网	125MB
USB	1.5MB	Ultrium 磁带	320MB
IDE 磁盘	5MB	PCI 总线	528MB
40X CD-ROM	6MB	Sun 千兆平面 XB backplane	20GB

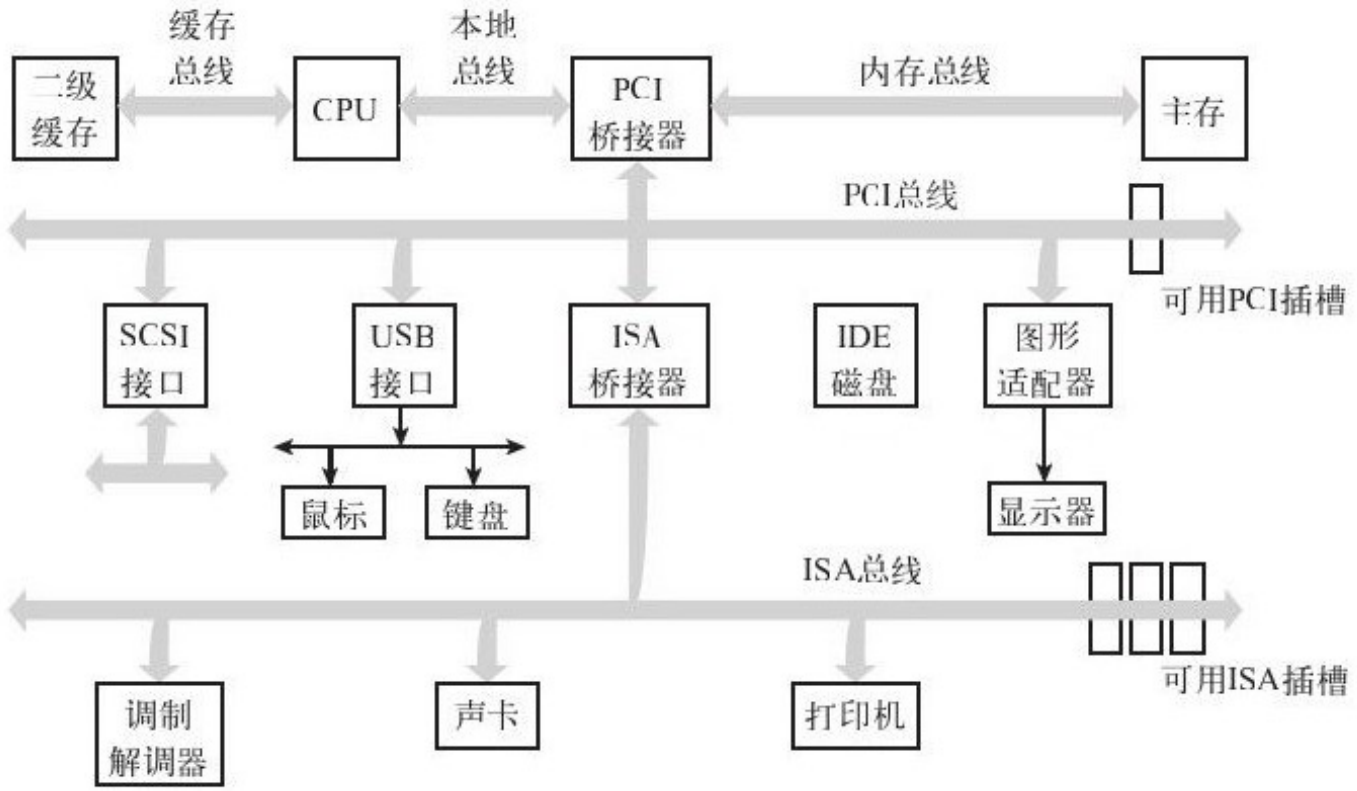
三、I/O系统及结构

- I/O系统的定义
  - 计算机中负责管理I/O的机构称为I/O系统（硬件和软件的组合）。
- I/O系统的结构
  - 单总线结构



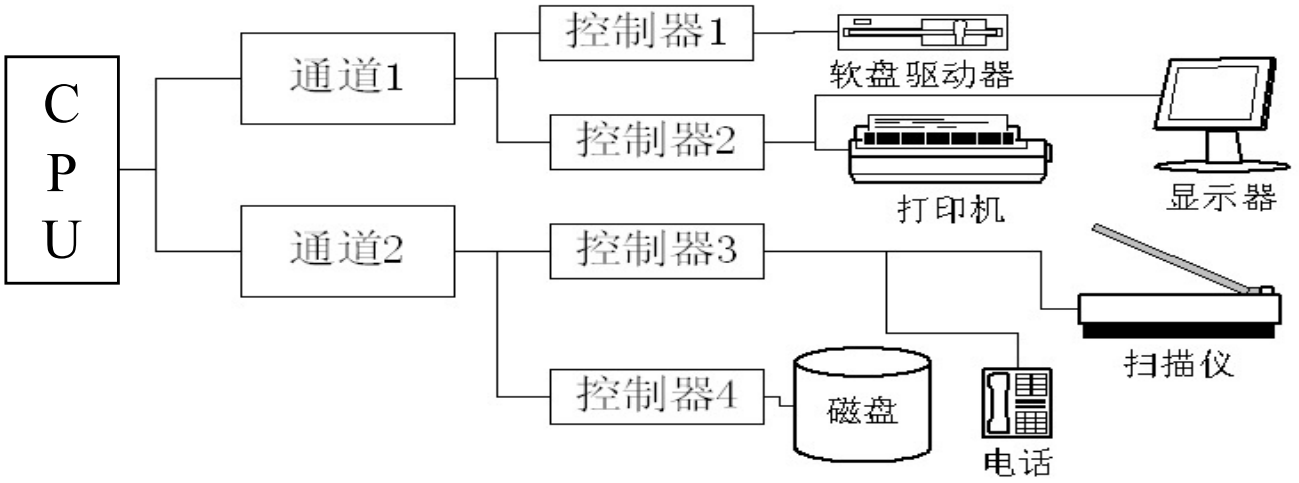
三、I/O系统及结构

❖ 多总线结构



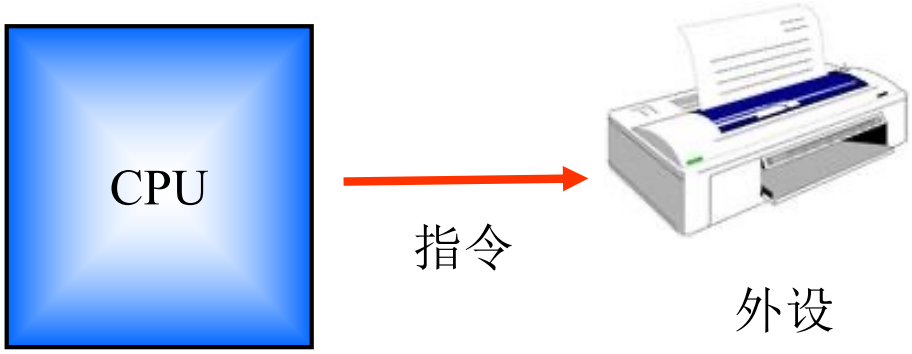
三、I/O系统及结构

❖ 通道系统



四、I/O系统控制方式(逻辑I/O模式)

程序控制I/O（直接控制方式/可编程I/O模式）



优点：简单。

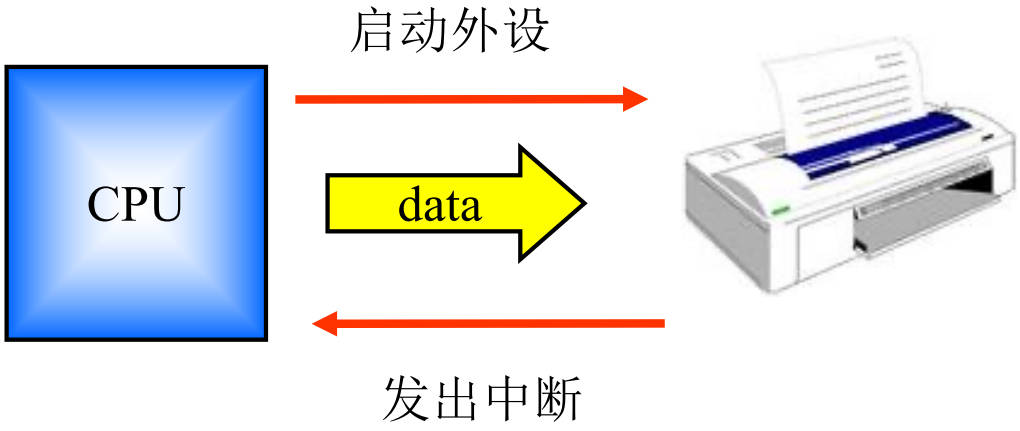
缺点：CPU的大部分时间都用于对硬件进行测试，效率低下。

```
copy_from_user( buffer, p, count)                    /* p 为内核缓冲区 */
    For (i = 0; i < count; i + + ) {
        while( * printer_status_reg! = READY);
        * printer_data_register = p[i];              /* 输出一个字符 */
    }
    Return_to_user( );
```

打印一串字符

四、I/O系统控制方式

中断驱动I/O



优点：在外设进行数据处理时，CPU不必等待，可以继续执行该程序或其他程序。提高了CPU的利用率。中断技术使得CPU和外设之间的并行工作成为可能。

缺点：数据仍然需要通过CPU进行传输，由于CPU每次处理的数据量少，因此这种方式只适于数据传输率较低的设备。

打印代码

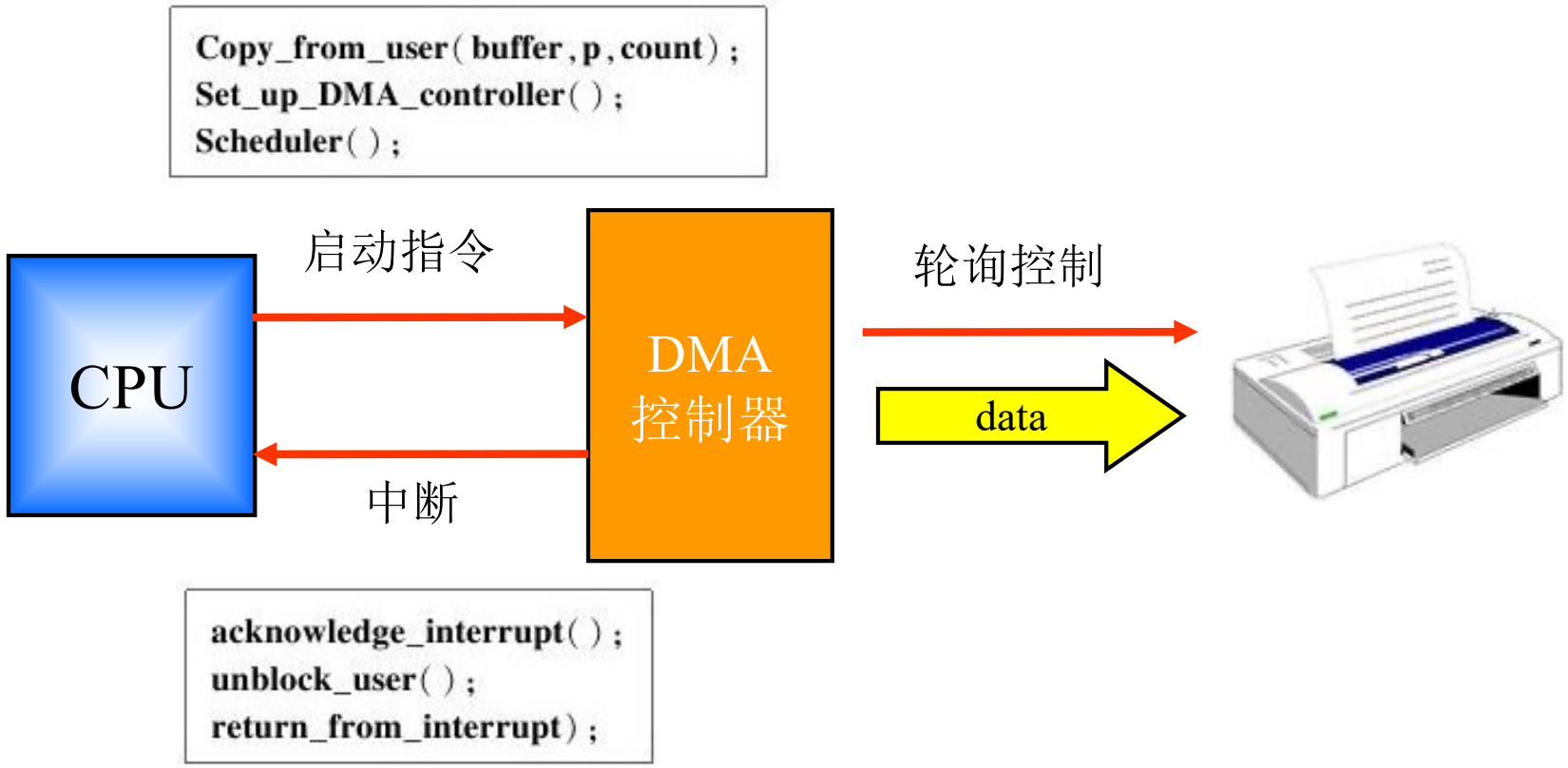
```
Copy_from_user( buffer, p, count)
Enable_interrupts( );
while( * printer_status_reg!  = READY);
 * printer_data_register = p[0];
Scheduler( );
```

```
If( count = =0){
   unblock_user( );
} else {
    * printer_data_register = p[i];
   Count = count - 1;
   i = i + 1;
}
acknowledge interrupt( );
return_from_intertupt( );
```

中断处理代码

四、I/O系统控制方式

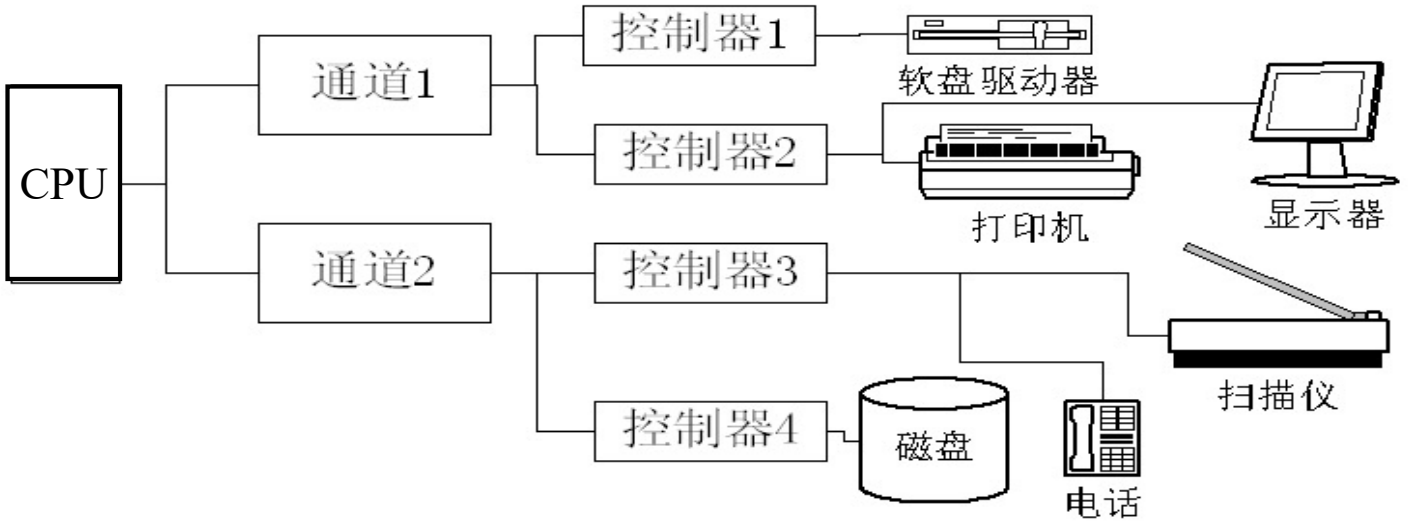
直接存储访问I/O(DMA, Direct Memory Access)



优点：CPU只需干预I/O操作的开始和结束，而其中的数据读写无需CPU控制，适于高速设备。

四、I/O系统控制方式

通道控制方式I/O



一个CPU可以连接若干个通道，一个通道可以连接若干个控制器，一个控制器可以连接若干个设备。

优点：解决了I/O操作的独立性和各部件工作的并行性。通道把中央处理机从繁琐的输入输出操作中解放出来。采用通道技术后，不仅能实现CPU和通道的并行操作，而且通道与通道之间也能实现并行操作，各通道上的外围设备也能实现并行操作，从而可达到提高整个系统的效率的根本目的。

四、I/O系统控制方式

通道的种类

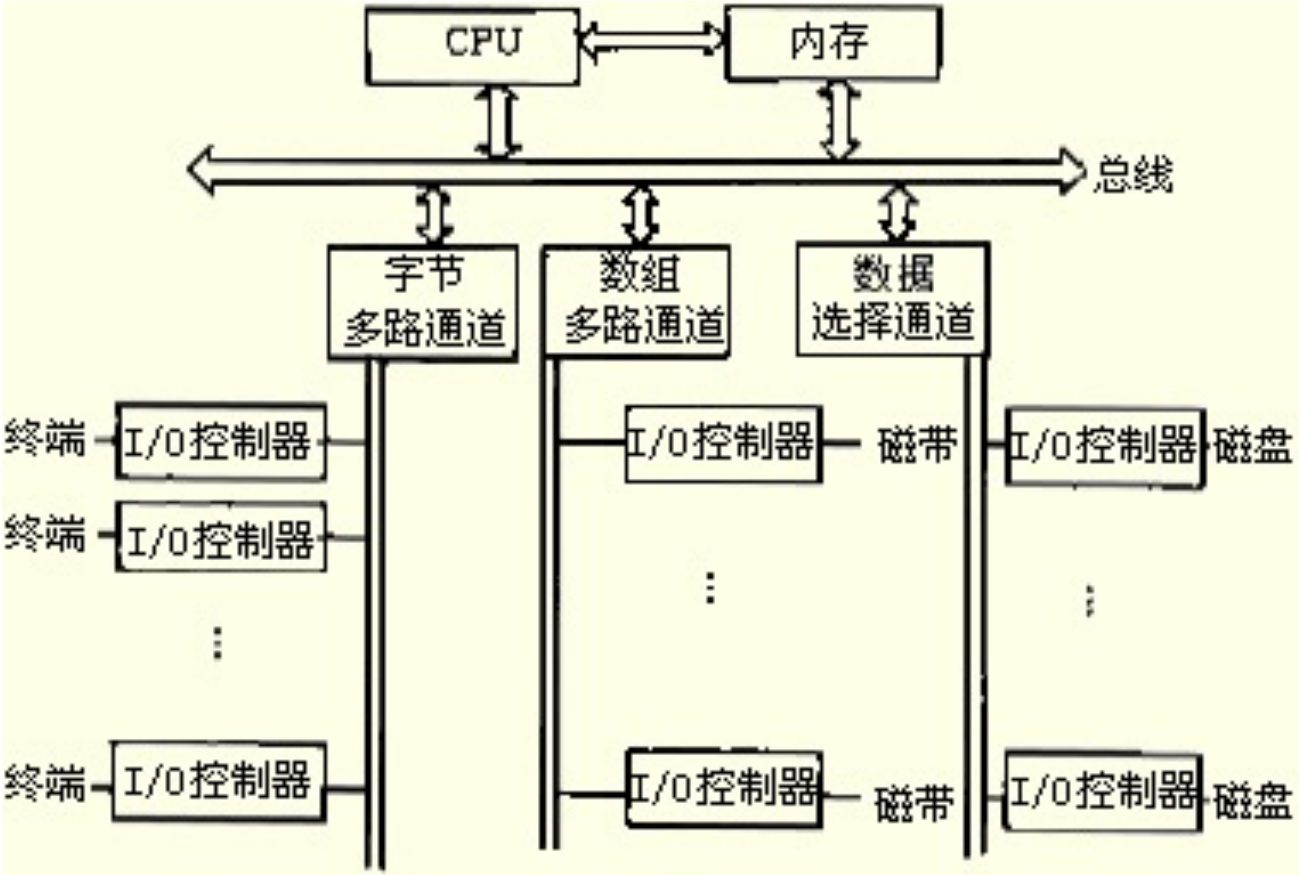


图 5-6 通道方式的数据传送结构

五、I/O软件

I/O软件的组成部分

- I/O交通管制程序；
  - 负责各I/O设备之间的协调工作；
- I/O调度程序；
  - 负责设备的分配和调度；
- I/O设备处理程序；
  - 负责每类设备的具体操作。

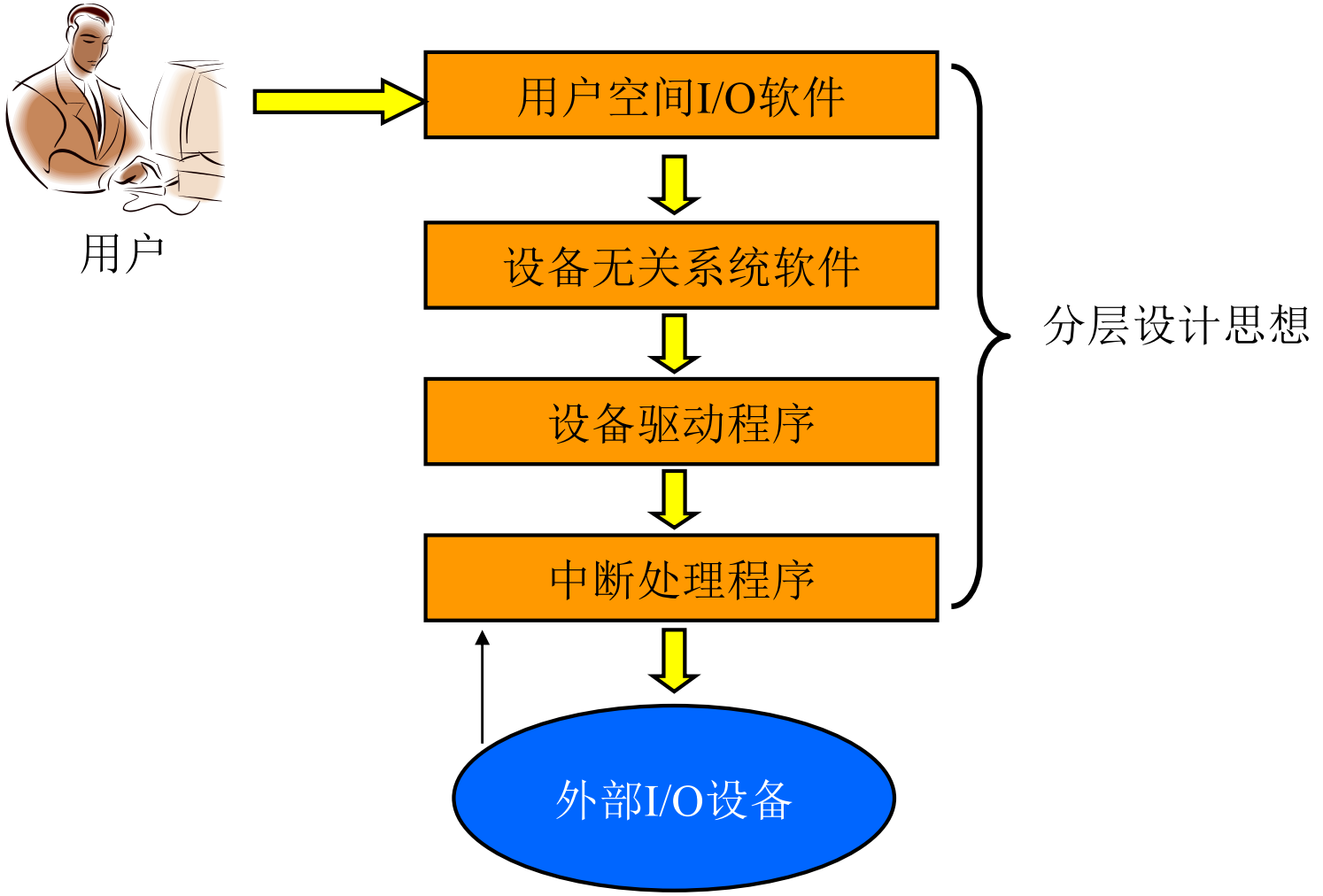
五、I/O软件

I/O软件的设计目标

- 设备独立
- 统一命名
- 错误处理
- 数据传输
- 缓冲管理
- 设备共享

……

I/O软件的结构



设备驱动程序层

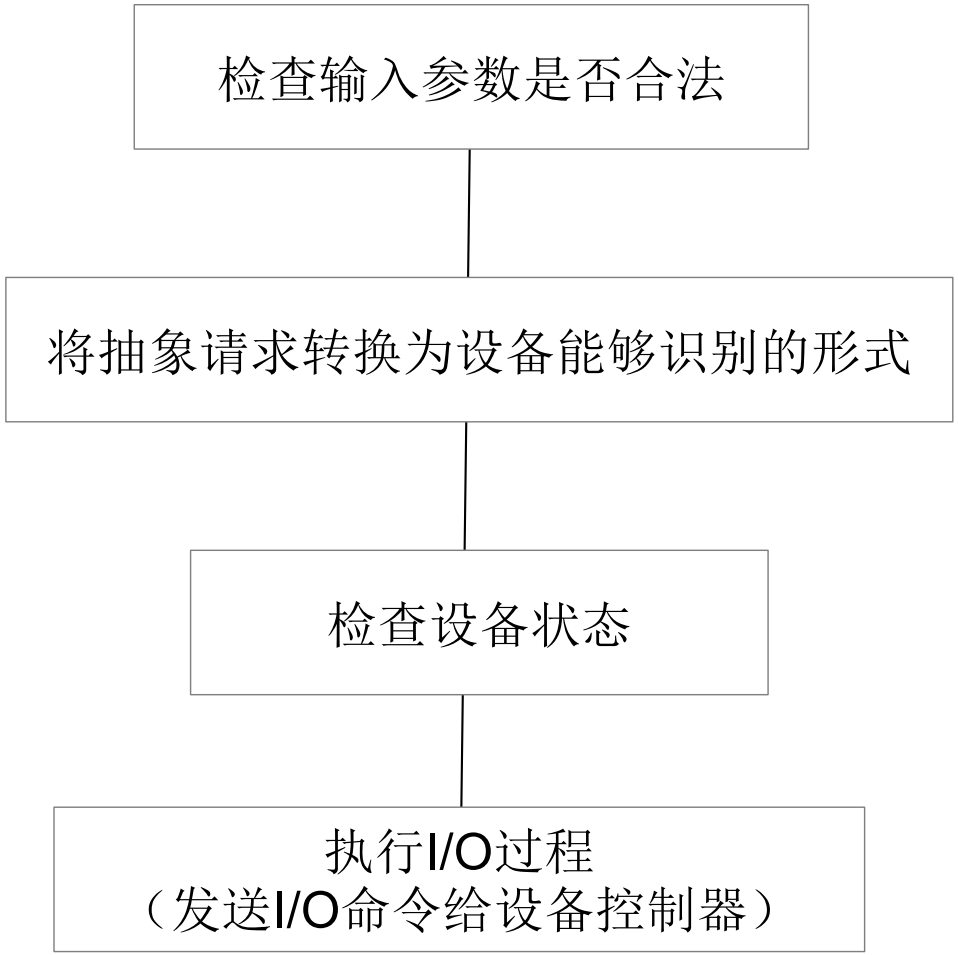
- 设备驱动程序是直接同硬件(控制器)打交道的内核软件模块。它是整个操作系统中唯一知道(关注)具体硬件信息的模块。
- 一般而言，设备驱动程序的任务为：
  - 向上：接受来自与设备无关的上层软件的抽象请求；
  - 向下：进行与设备相关的操作。

五、I/O软件

- 功能主要有：
  - 从上层接收抽象的读写请求；
  - 控制和监督各I/O控制器的正确执行，并确保读写操作完成；
  - 初始化I/O设备，开/关设备；
  - 缓冲区管理；

    ……

❖ 驱动程序工作过程


检查输入参数是否合法
将抽象请求转换为设备能够识别的形式
检查设备状态
执行I/O过程
（发送I/O命令给设备控制器）

五、I/O软件

❖ 设备驱动程序的特点

- 与I/O设备的硬件结构密切联系，是操作系统底层中唯一知道各种输入输出设备的控制器细节及其用途的部分。
- 驱动程序属于操作系统内核的组成部分，因此也是操作系统安全的巨大隐患。

五、I/O软件

- 设备无关系统软件层
  - 负责实现对所有设备都具有共性的功能，并向上提供一个统一的接口。
  - 例如：缓冲管理，错误报告，分配与释放独享设备...
- 主要功能：
  - 统一界面：通过对下层(驱动程序层)规定一个标准化的功能清单，让所有的设备看上去都一样或者相似。
  - 缓冲管理：缓冲区是外设的常用组成部分，设置缓冲的目的是为了桥接不同速度的设备并提供灵活的安全机制。

五、I/O软件

- 错误处理：要负责处理两类错误——程序错误和I/O错误。
- 提供与设备无关的逻辑块：屏蔽底层各种I/O设备空间大小、处理速度和传输速率的差异，只向上层提供大小统一的逻辑块尺寸。
- 存储设备的块分配。
- 独占设备的分配与释放。

五、I/O软件

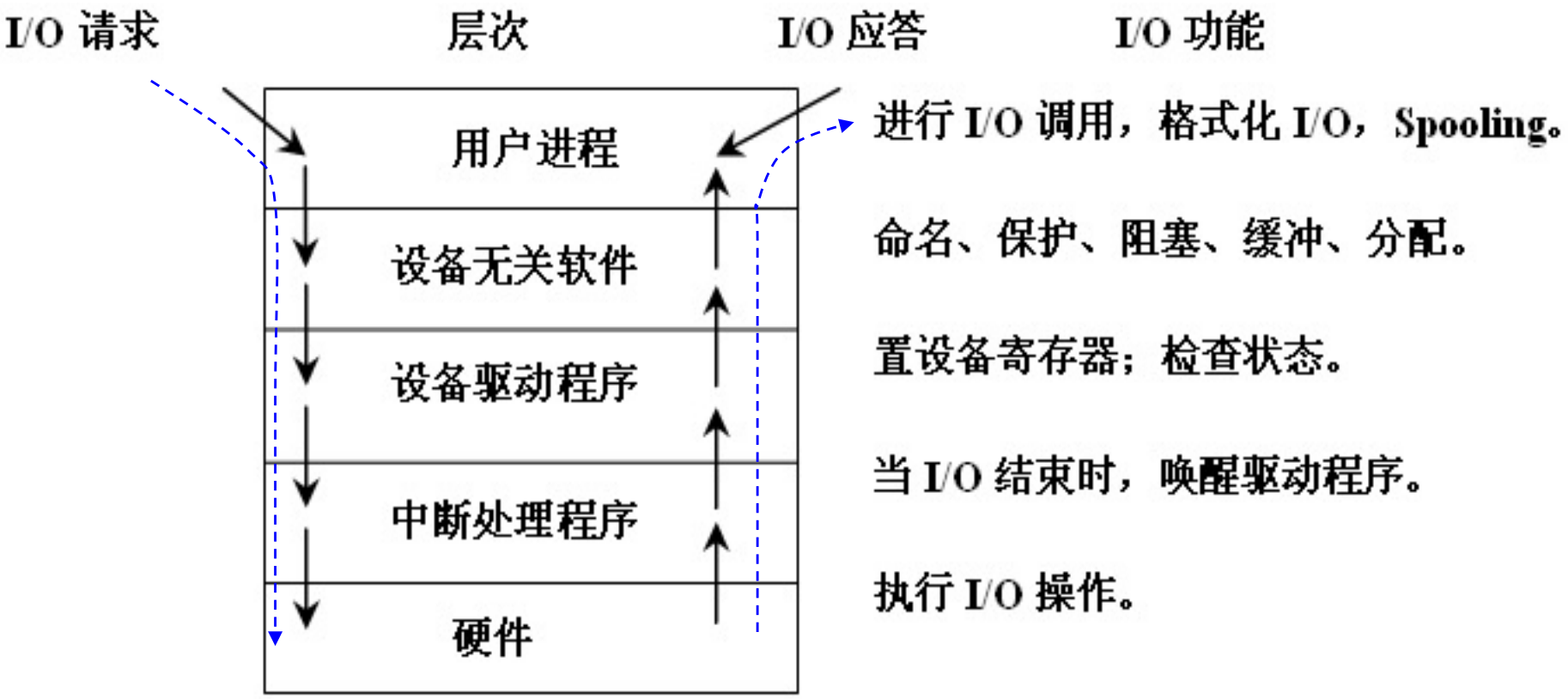
用户空间I/O软件

- 就是具有I/O功能但在用户态下运行的软件(函数)或功能模块。
- 例如：

```
write(fd, buffer, nbytes);
open(…);
printf(…);
gets(…);
```

SPOOLing系统

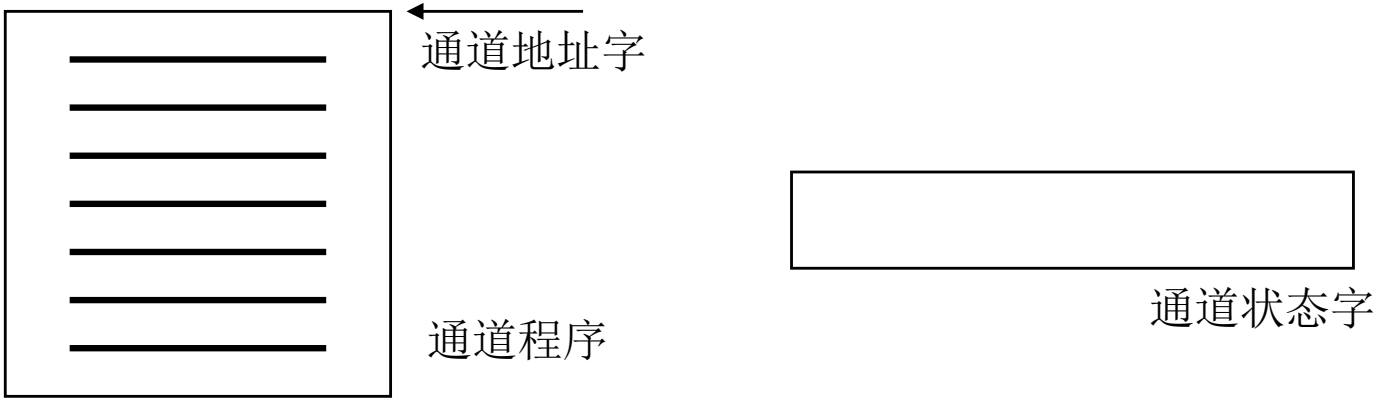
I/O系统的层次结构与每层的主要功能



六、具有通道的设备管理

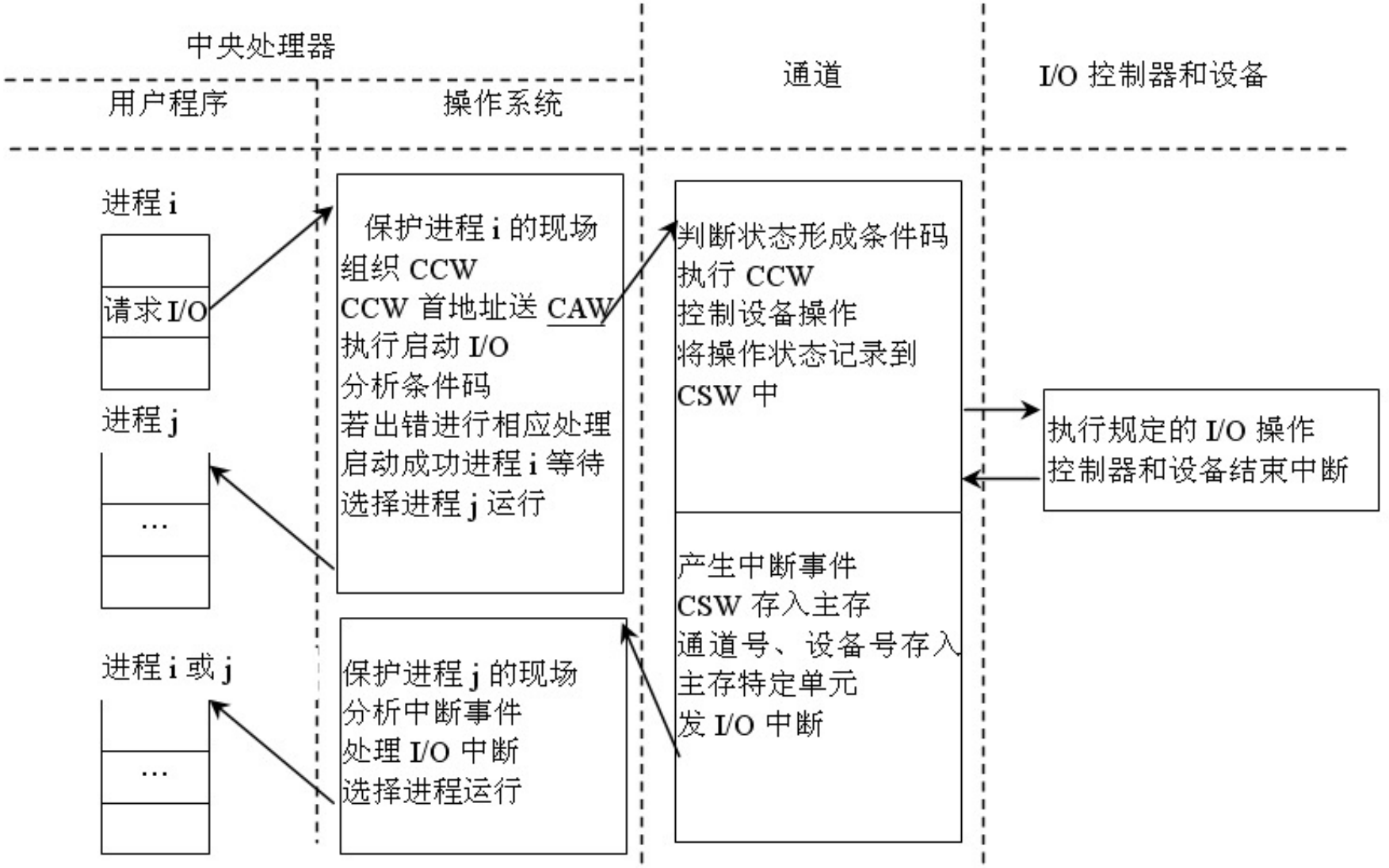
术语：

- 通道命令(Channel Command Word, CCW)：通道又称为I/O处理机，具有自己的指令系统，常常把I/O处理机的指令称通道命令。
- 通道程序：用通道命令编写的程序称通道程序，通道通过执行通道程序控制I/O设备运行。
- 通道地址字(Channel Address Word, CAW)：用来存放通道程序首地址的内存单元称通道地址字。
- 通道状态字(Channel Status Word, CSW)：是通道向操作系统报告工作情况的状态汇集。



六、具有通道的设备管理

通道的工作原理



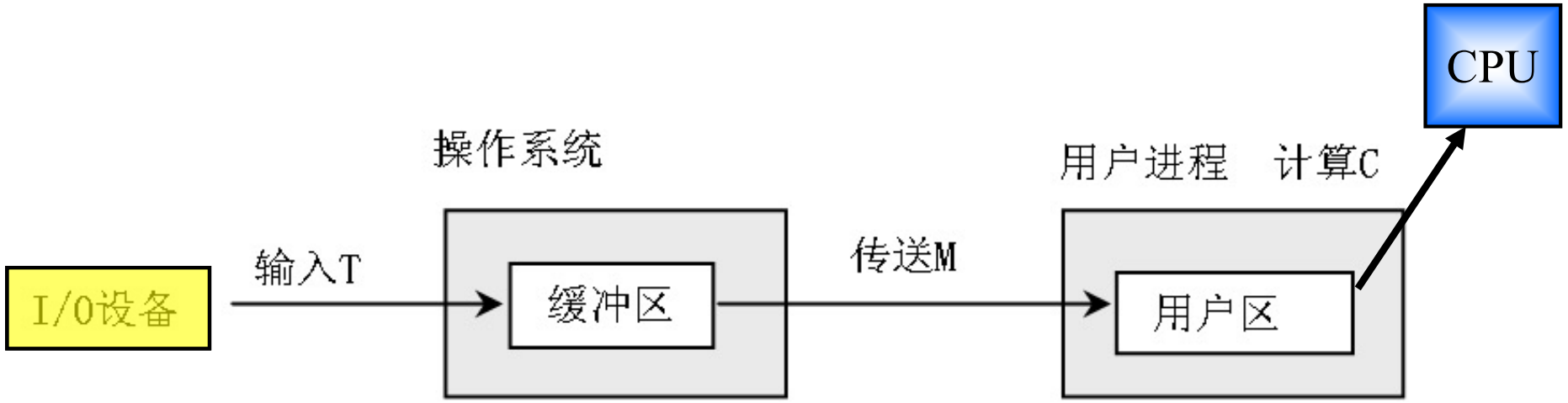
七、缓冲技术

- 缓冲区(buffer)是一种在设备和设备之间或设备与程序之间交换数据的区域。
- 设置缓冲区的目的：
  - 缓解缓冲两端设备(程序)速度的差异
  - 协调数据大小的不一致性
  - 实现应用程序I/O的语义拷贝
- (几乎)所有的设备之间，都会使用缓冲区来交换数据，因此缓冲区的合理管理十分重要。

缓冲技术的分类

单缓冲技术(single buffer)

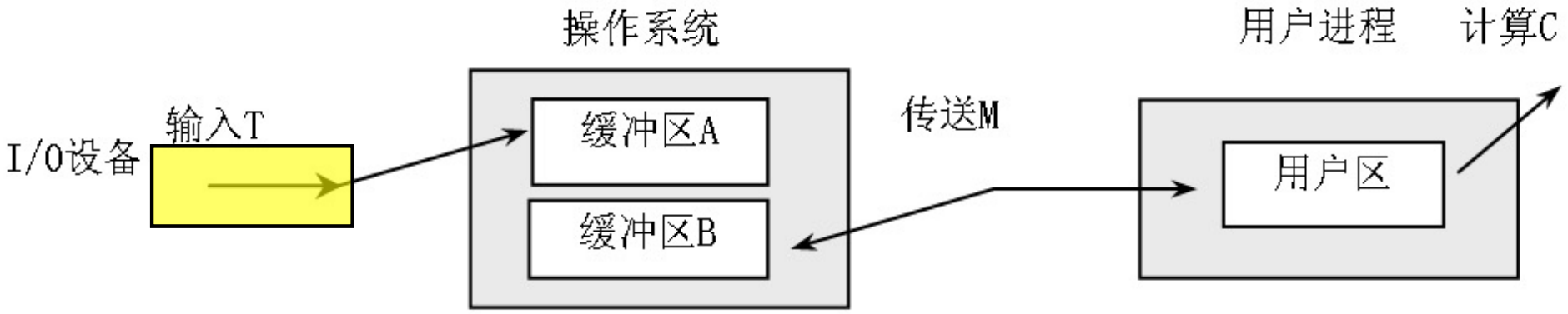
- 例如：



计算过程C和数据输入缓冲的过程T可以并行。

七、缓冲技术

❖ 双缓冲(double buffer)

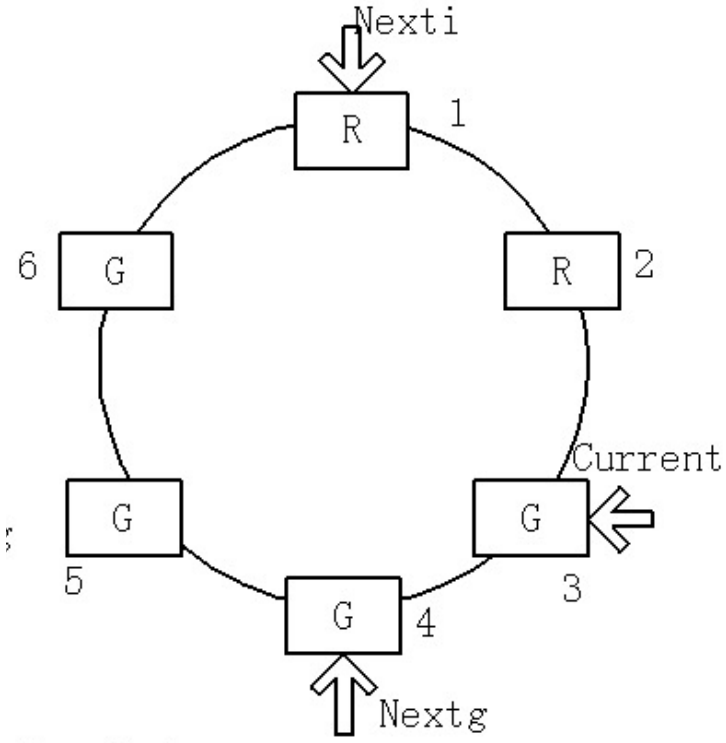


计算过程C、数据输入缓冲的过程T、数据传送的过程M均可以并行。

七、缓冲技术

❖ 环形缓冲

➢ 结构:



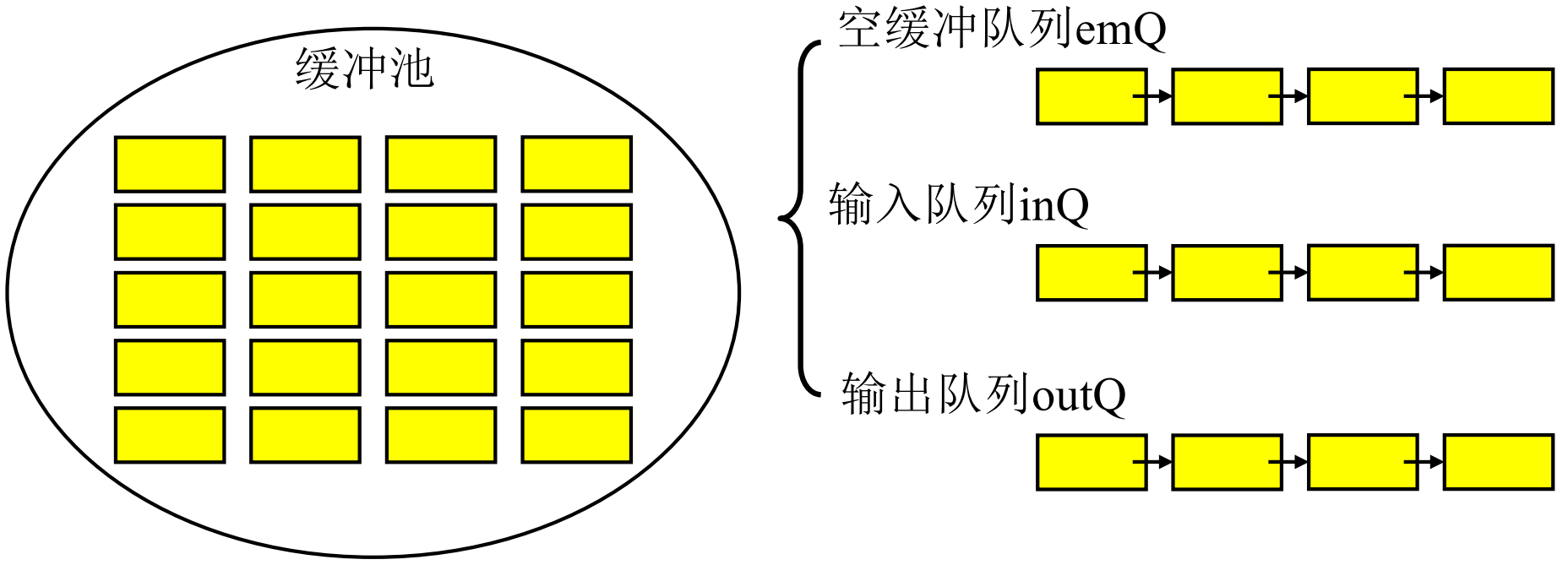
环形缓冲的两种现象

系统受限计算：Nexti追上Nextg

系统受限I/O：Nextg追上Nexti

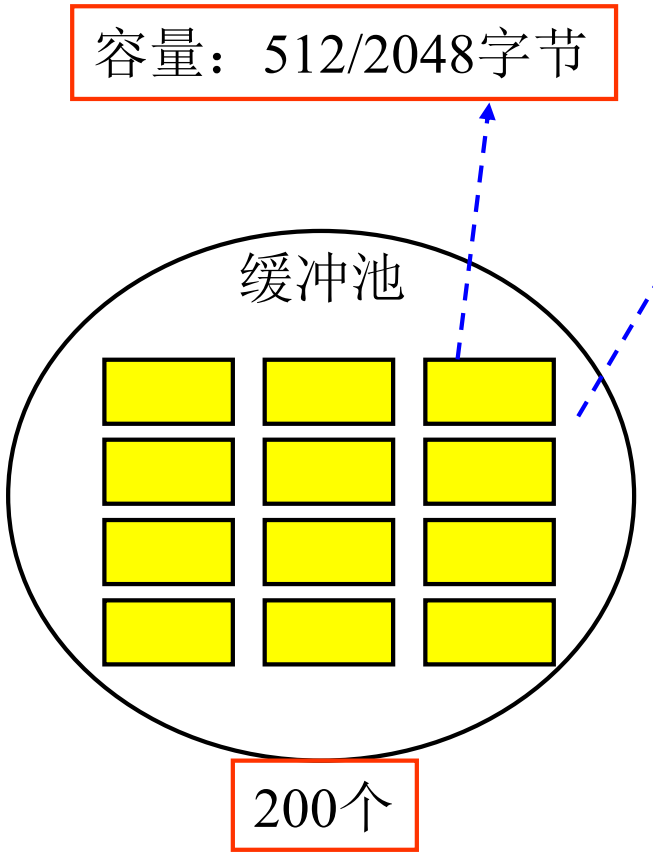
七、缓冲技术

- 缓冲池(buffer pool)
  - 可供多个进程共享的双向缓冲技术。



UNIX设备缓存管理

管理用到的主要数据结构



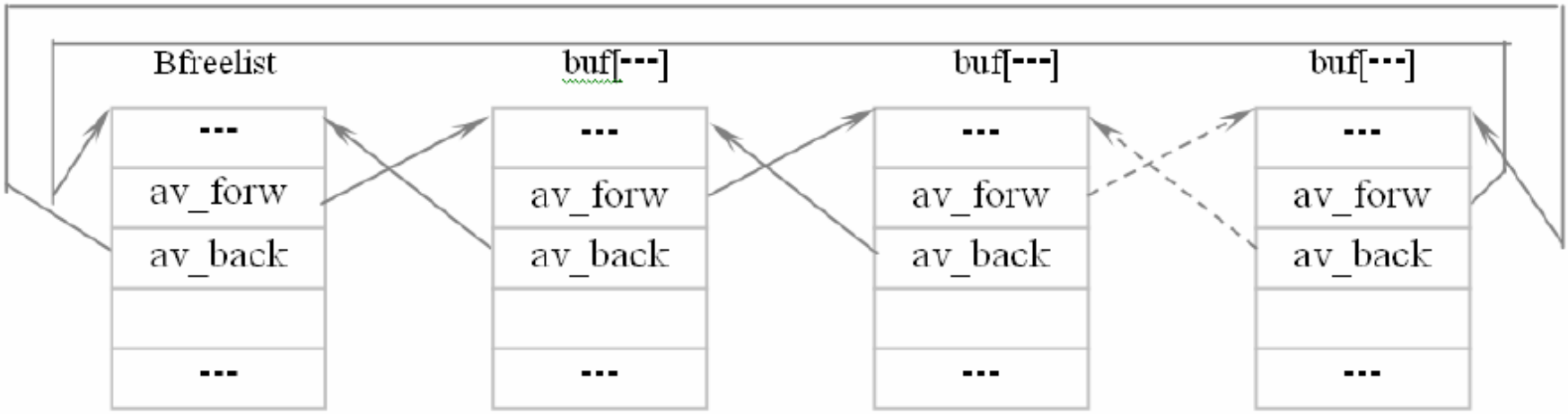

```
struct buf    //缓存控制块
  {
    int    b_flage;              /*缓冲区标志*/
    struct buf *b_forw;          /*设备队列前向指针*/
    struct buf *b_back;          /*设备队列后向指针*/
    struct buf *av_forw;         /*自由队列前向指针*/
    struct buf *av_back;         /*自由队列后向指针*/
    dev_tb_dev;                  /*逻辑设备号*/
    unsigned b_bcount;           /*传送数据字节数*/
    union{
       caddr_t    b_addr;       /*缓冲区内存首地址*/
       int    b_words;          /*要刷新的起始地址*/
       struct filsys *b_filsys;  /*超级块*/
       struct dinode *b_dino;    /*磁盘inode表*/
       daddr_t    *b_daddr;     /*间接块*/
    }b_un;
    daddr_t    b_blkno;          /*磁盘上数据的块号*/
    char  b_error;               /*返回给调用者的出错信息*/
    unsigned int b_resid;        /*因出错而未被传送的数据字节数*/
    time_t b_start;              /*I/O请求起始时间*/
    struct proc *b_proc;         /*执行物理或兑换I/O的进程*/
  } buf[NBUF];
```

缓冲管理队列

- 为方便对缓冲区进行管理，UNIX系统设置了三种队列。
  - 自由buf队列
  - 设备buf队列
  - NODEV设备队列
- 由于buf记录了与缓冲存储区有关的各种管理信息，所以缓冲区管理队列实际上就是对缓存控制块buf队列的管理。

自由buf队列

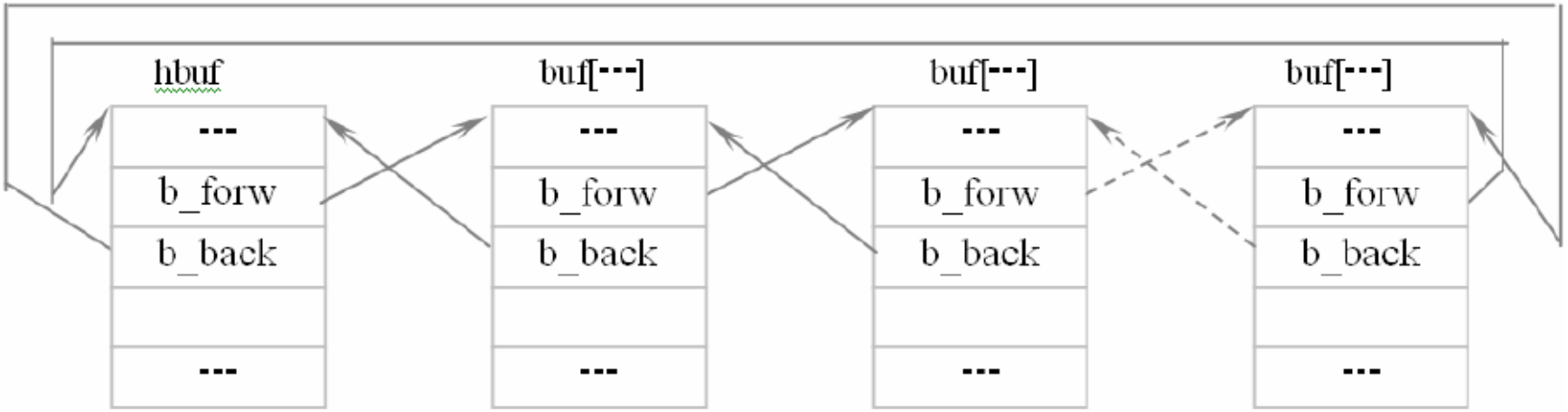
- 在Unix系统中，一个可被分配作它用的缓冲存储区，其相应的buf位于自由buf队列中。



自由buf队列采用FIFO管理算法。一个缓存被释放时，其相应的buf被送入自由buf队列的队尾；当要求分配一个缓存时，从队首取出一个buf，它所管理的缓存就可被“移作它用”。

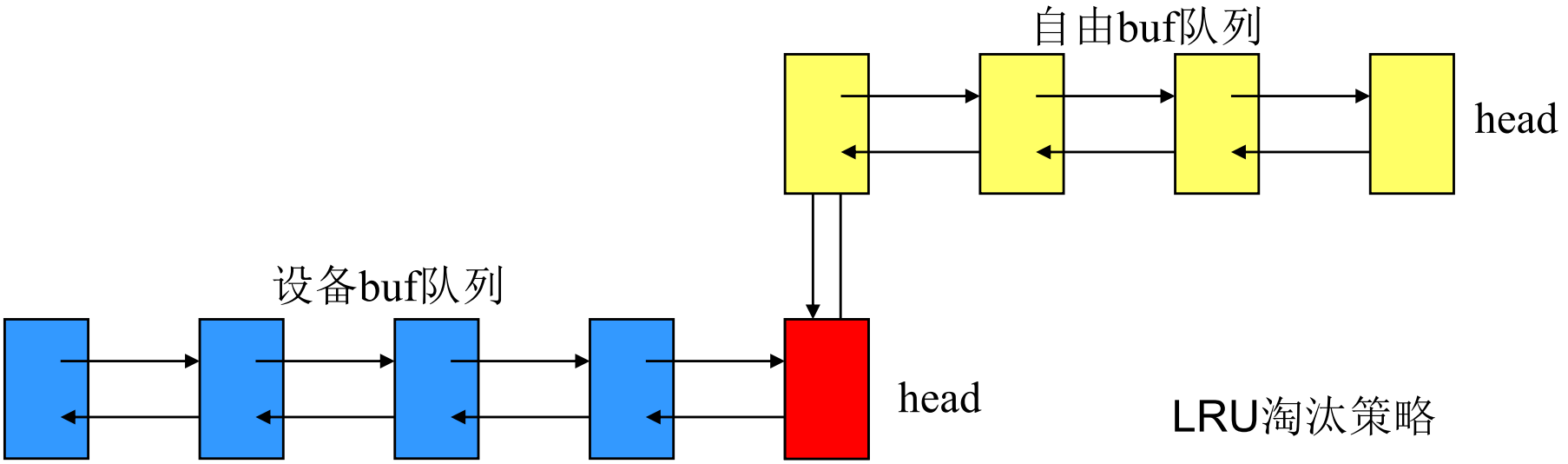
设备buf队列

- 在Unix系统中，每类设备都有一个buf队列。



一个缓存被分配用于读、写某类块设备上的某一字符块时，其相应的buf就进入该类设备的buf队列，除非被“移作它用”，否则一直保留在该队列中。

为什么缓冲控制块buf中要设置两对指针？

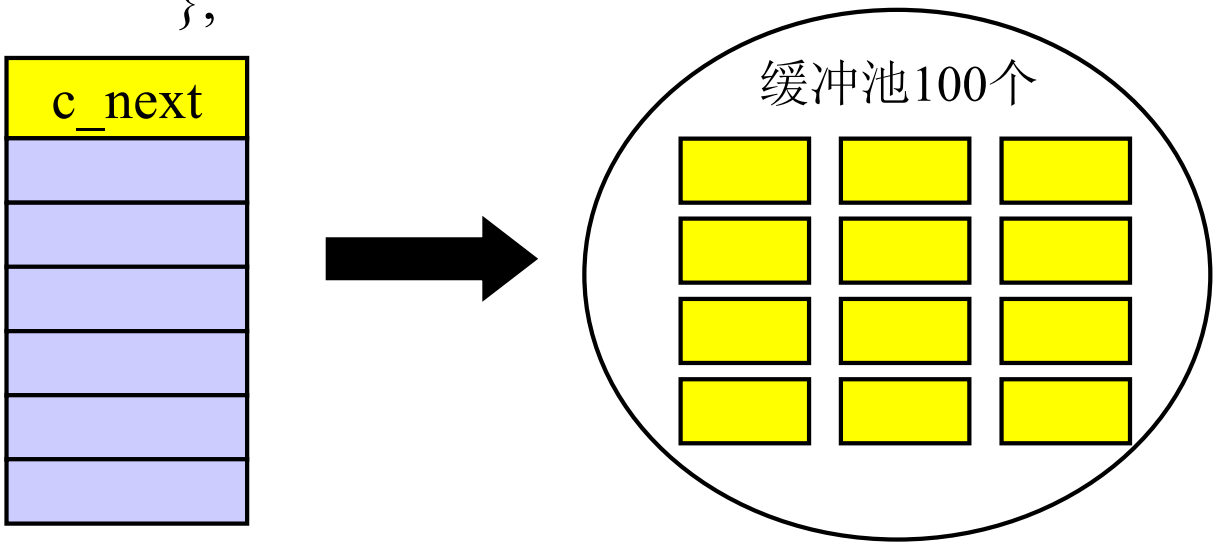


- 可以使得一个缓冲区同时处于两个队列中。
- 一个设备buf队列中的缓冲区当使用完后，便同时处于原设备buf队列中和自由buf队列中。
- 避免了重复、费时的I/O操作过程，从而提高了系统的I/O效率。

字符设备的缓存管理

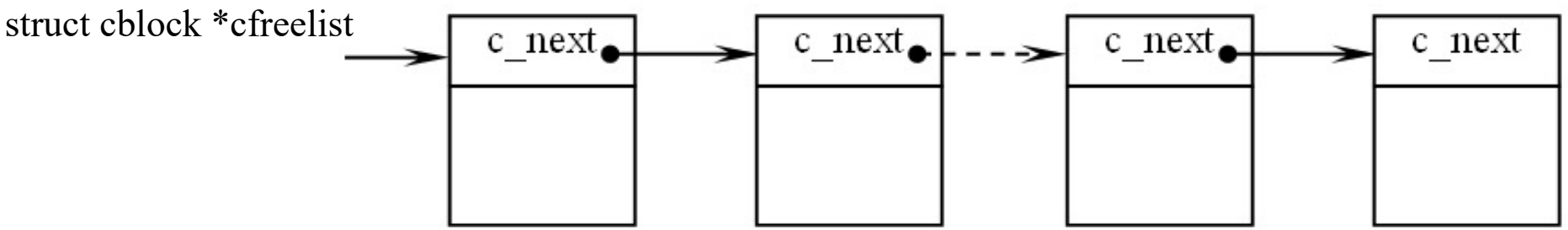
- 字符缓存用于解决CPU与字符设备间速度不匹配的矛盾，由于字符缓存很短，所以UNIX在实现上没有设置专门的缓存控制块，其字符缓存的结构如下：

```
struct cblock
{
    struct cblock *c_next;      /*字符缓存指针*/
    char info[6];               /*字符缓存信息区*/
};
```



❖ 自由字符缓存队列

- 由空闲的字符缓存构成自由队列。



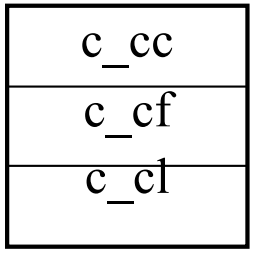
字符缓存的分配和释放都是在队首进行。

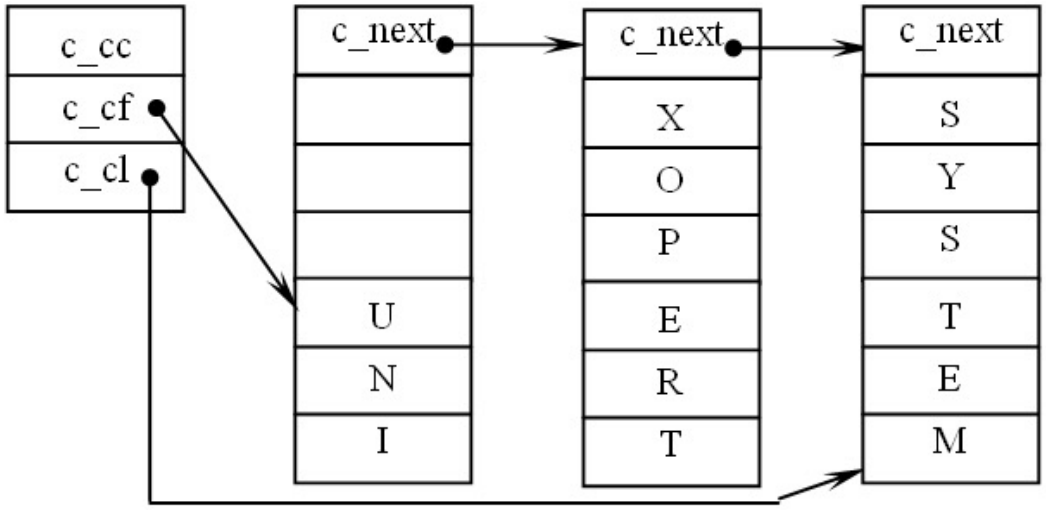
七、缓冲技术

❖ I/O字符缓存队列

➢ 字符设备通过字符缓存进行输入或输出。各个正被使用的字符缓存按照它们的不同用途形成多个I/O队列，每个队列设置一个控制块，其结构如下:

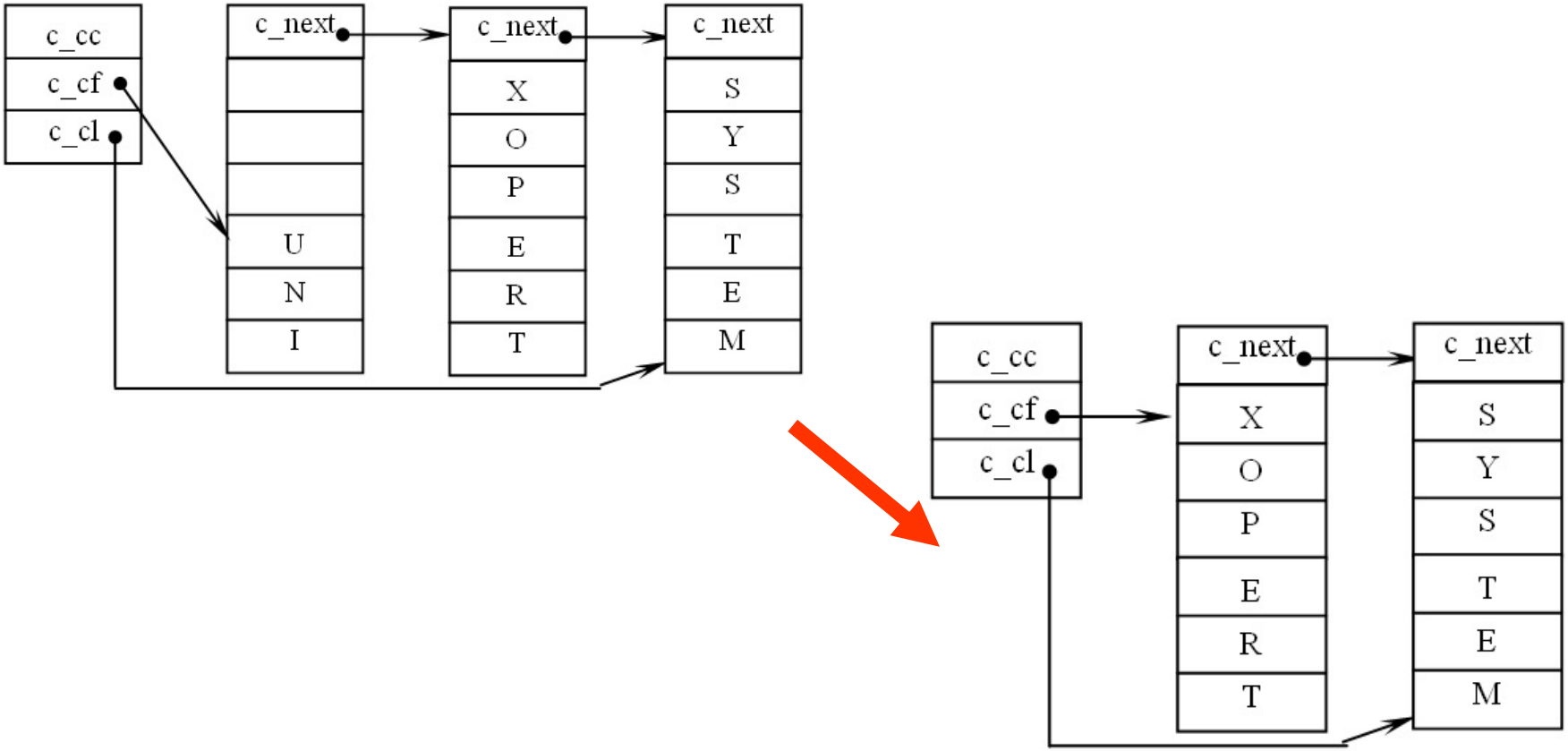
```
struct list
  {
    int c_cc;       /*字符计数器*/
    int c_cf;       /*缓存队列首指针*/
    int c_cl;       /*缓存队列尾指针*/
  };
```





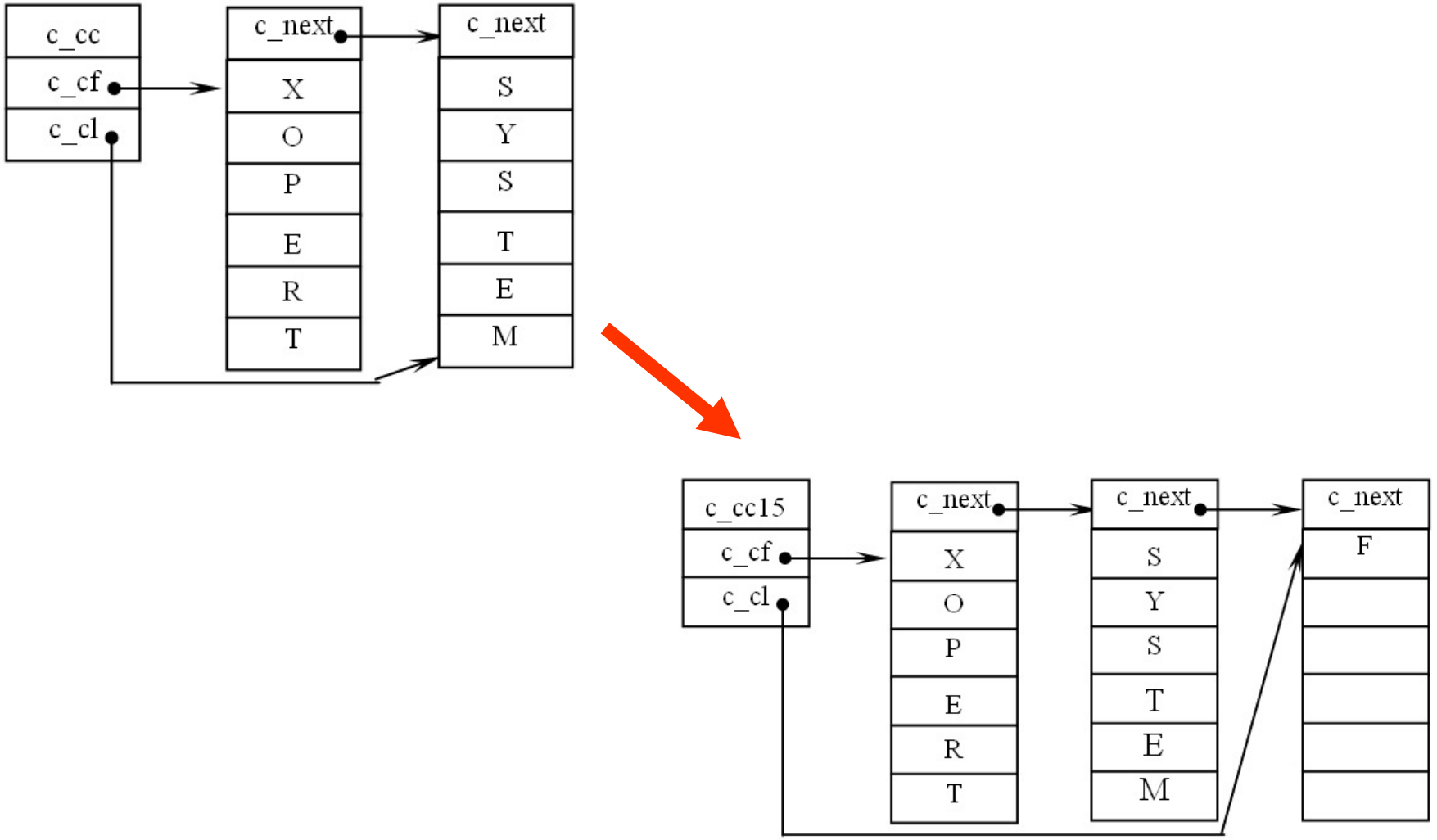
字符缓存管理

❖ 取字符和释放字符缓存



七、缓冲技术

❖ 送字符和分配字符缓存



八、设备管理数据结构

操作系统中的设备分配

- 在多道程序环境下，系统中的设备供所有进程共享。
- 为防止诸进程对系统资源的无序竞争，特规定系统设备不允许用户自行使用，必须由系统统一分配。每当进程向系统提出I/O请求时，只要是可能和安全的，设备分配程序便按照一定的策略，把设备分配给请求用户（进程）。

八、设备管理数据结构

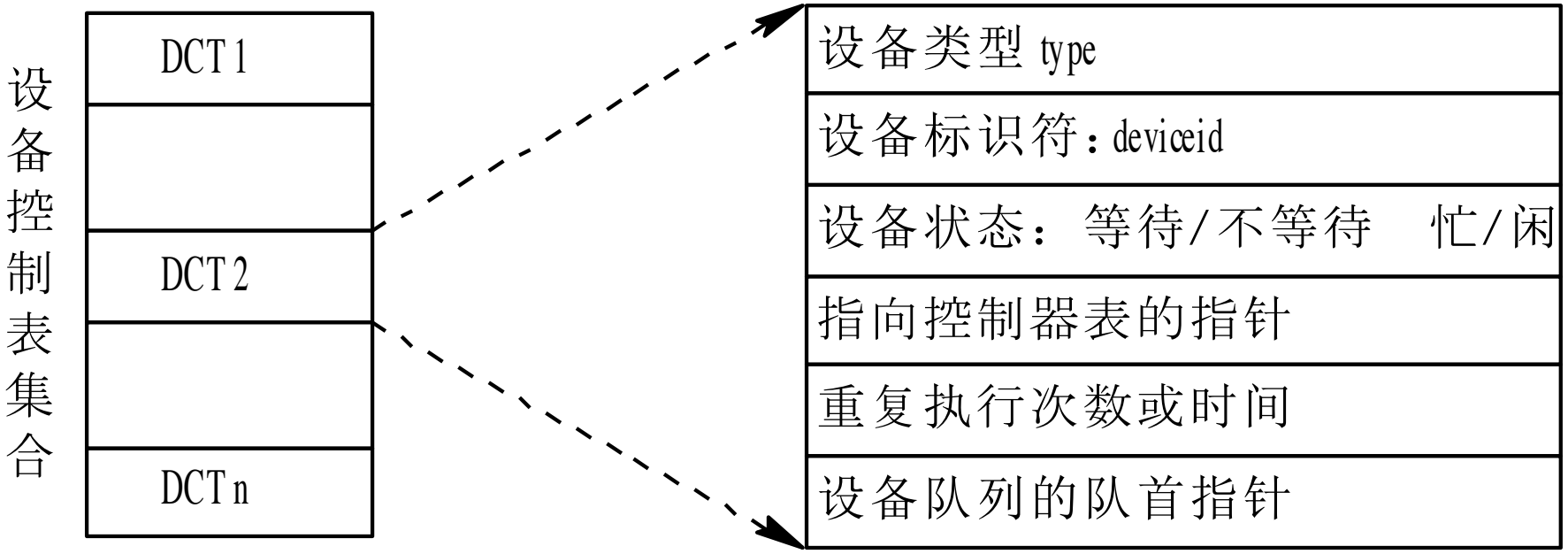
设备管理数据结构

- 在进行设备分配时，通常都需要借助于一些表格的帮助。在表格中记录了相应设备或控制器的状态及对设备或控制器进行控制所需的信息。
- 在进行设备分配时所需的数据结构（表格）有：
  - 设备控制表（DCT）
  - 控制器控制表（COCT）
  - 通道控制表（CHCT）
  - 系统设备表（SDT）等

八、设备管理数据结构

设备控制表(Device Control Table, DCT)

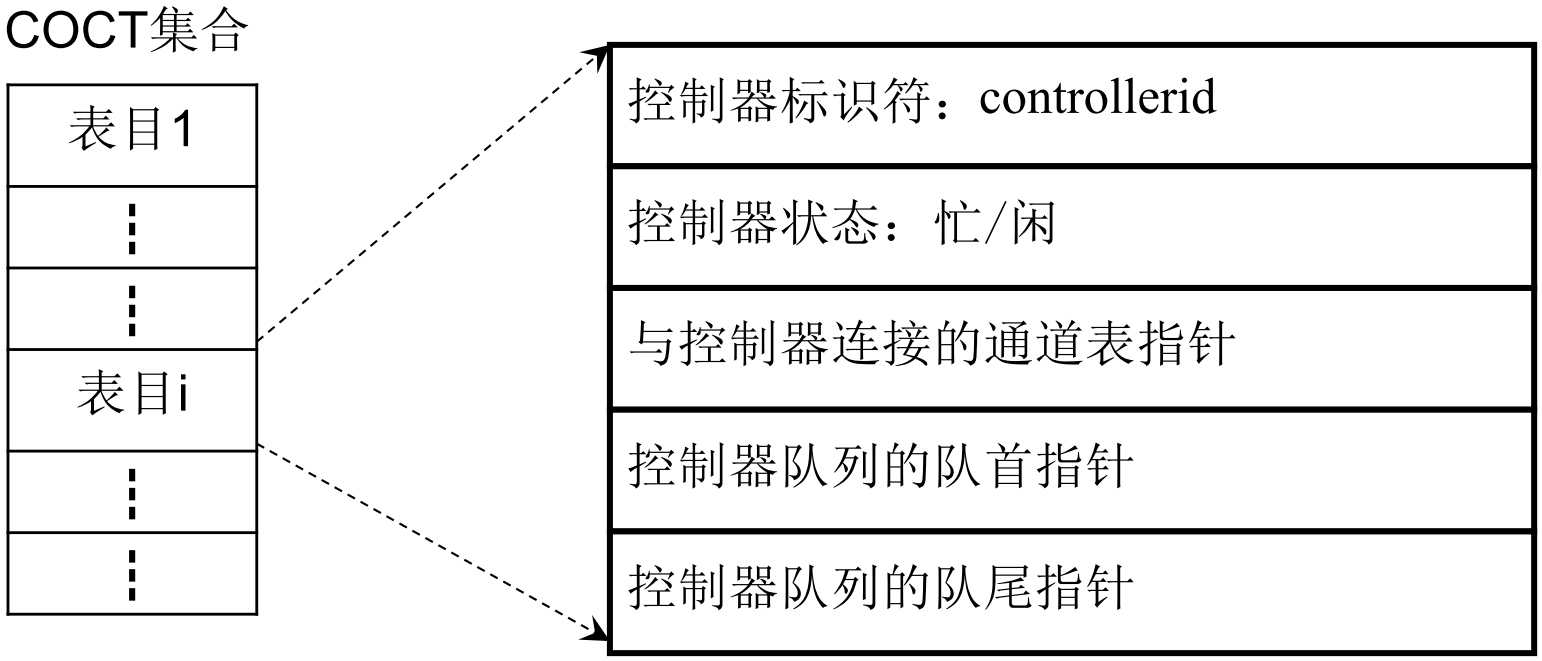
- 系统为每个设备配置一张DCT，描述设备的特性和状态，设备与控制器的连接情况等。



八、设备管理数据结构

控制器控制表(COntroller Control Table, COCT)

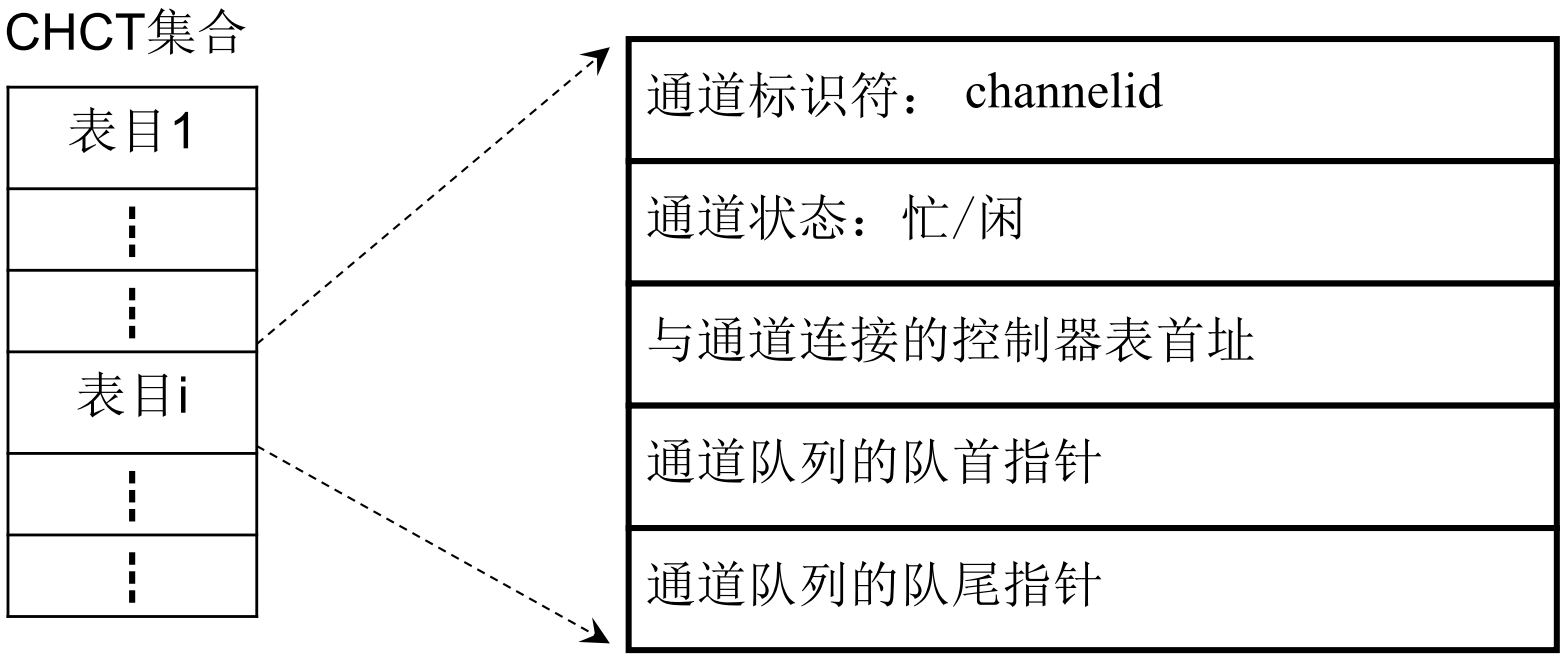
- 每个设备控制器一张，描述设备控制器的配置和状态。



八、设备管理数据结构

通道控制表(CHannel Control Table, CHCT)

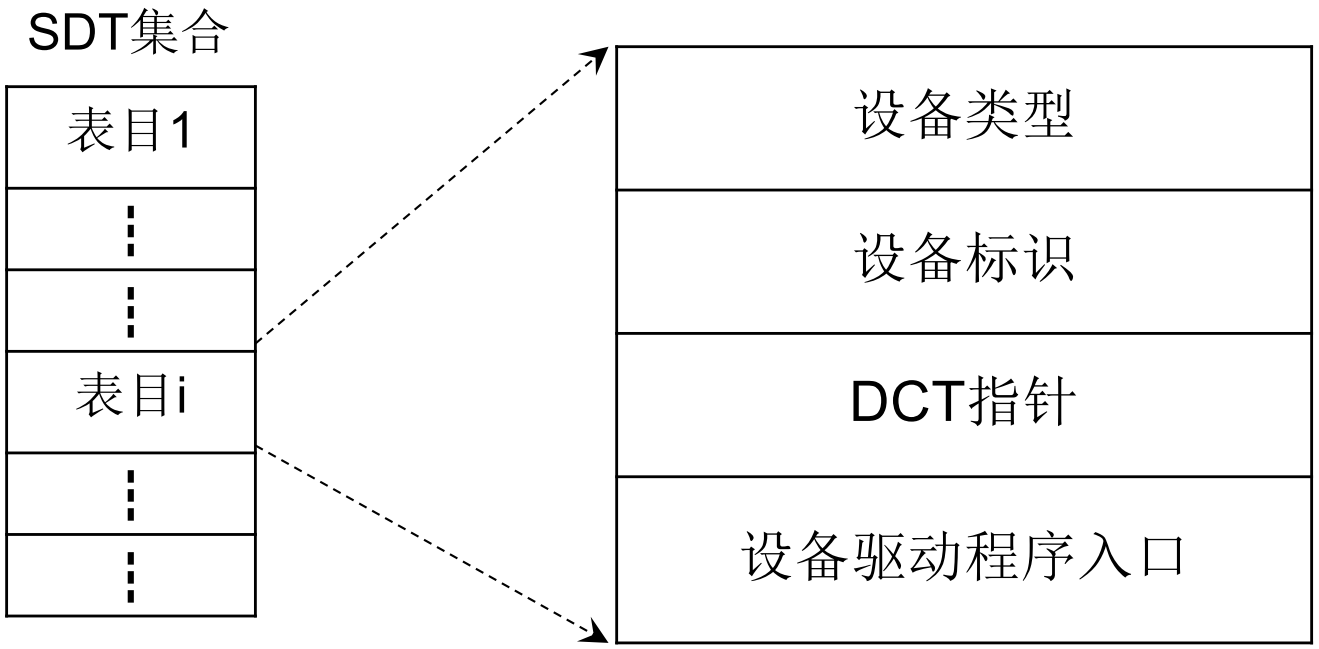
- 每个通道一张，描述通道工作状态。



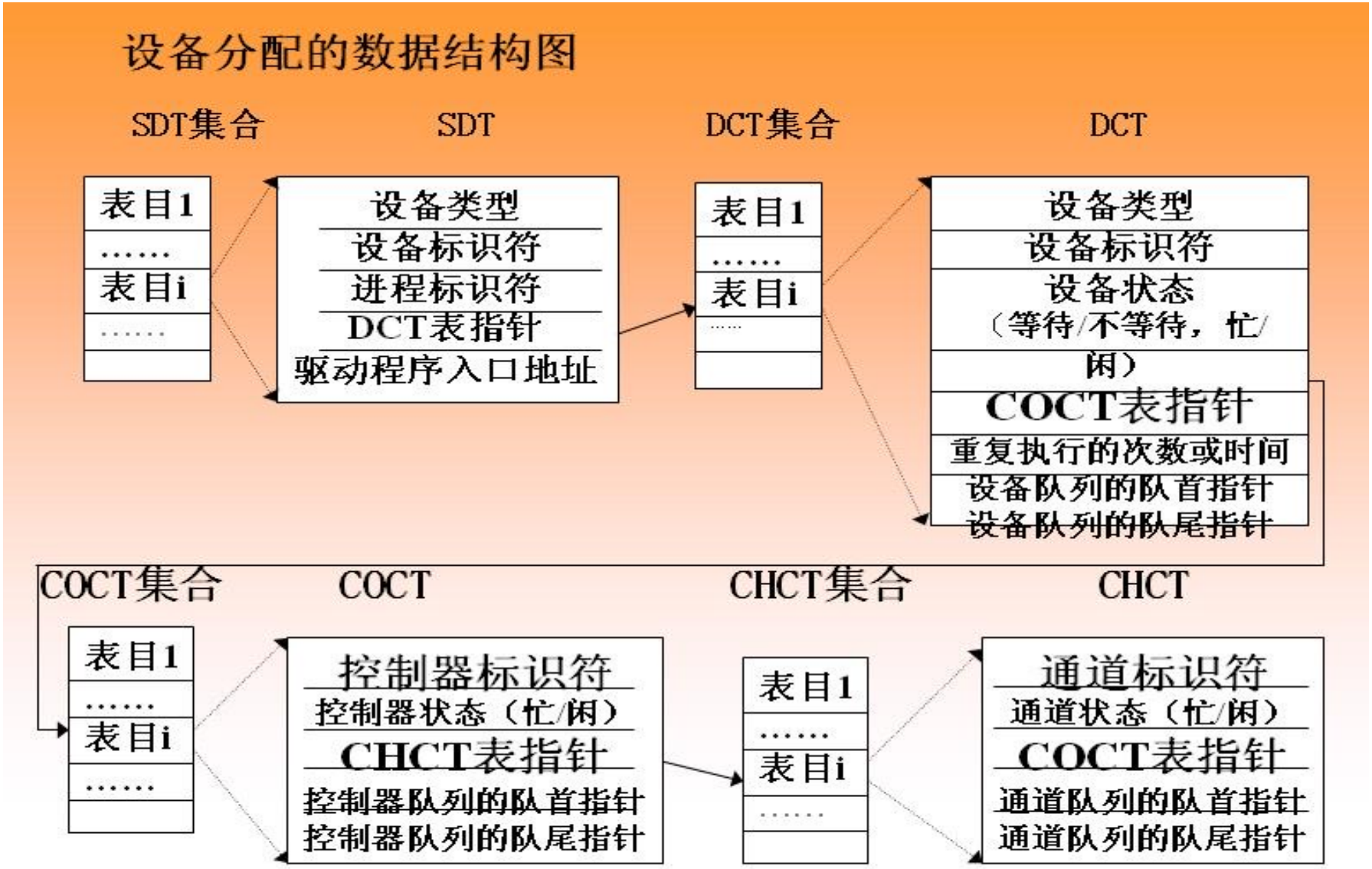
八、设备管理数据结构

系统设备表(System Device Table, SDT)

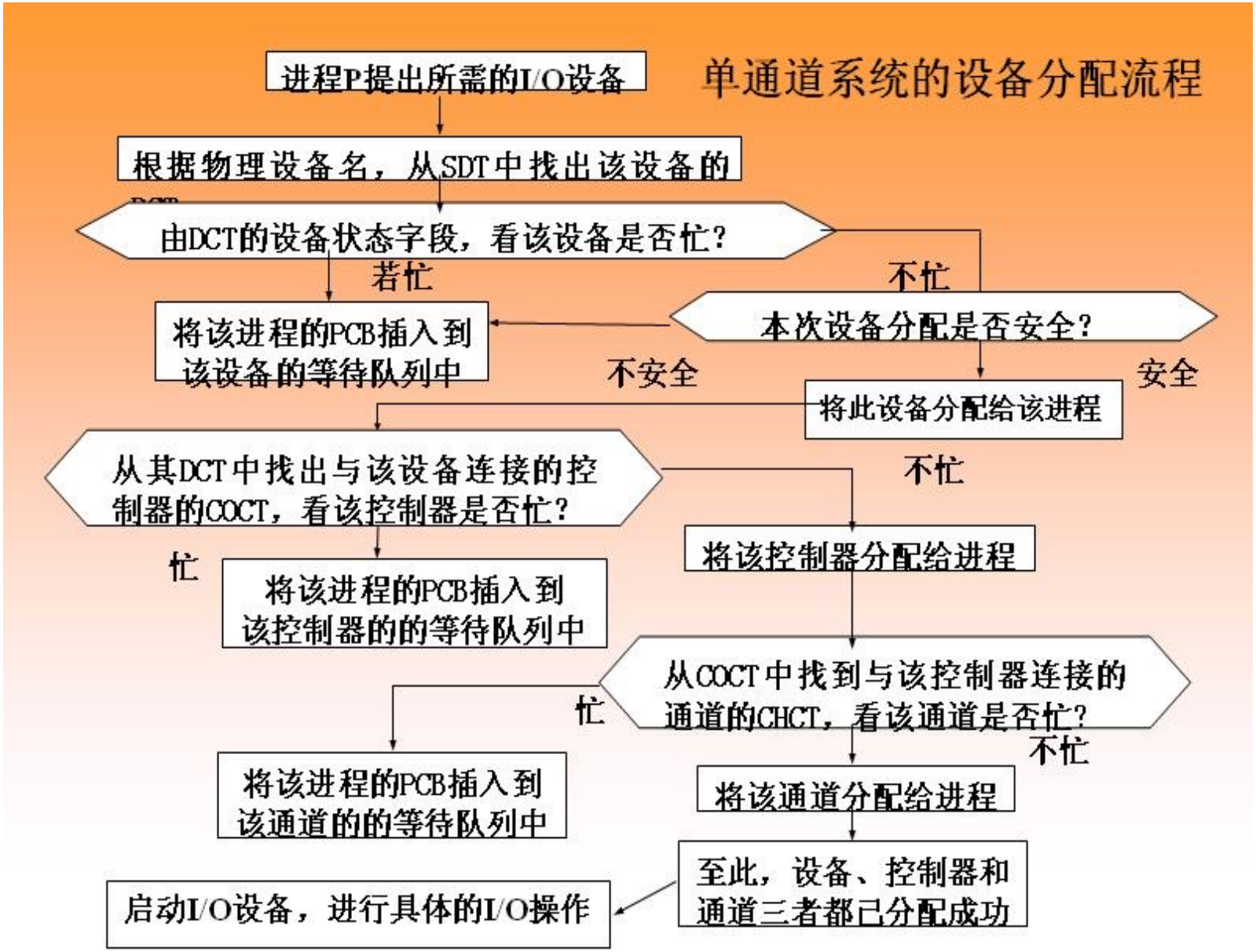
- 系统范围的数据结构，其中记录了系统中全部设备的情况。每个设备占一个表目。



八、设备管理数据结构


设备分配的数据结构图
SDT集合
SDT
DCT集合
DCT
表目1
……
表目i
……
设备类型
设备标识符
进程标识符
DCT表指针
驱动程序入口地址
表目1
……
表目i
……
设备类型
设备标识符
设备状态
（等待/不等待，忙/闲）
COCT表指针
重复执行的次数或时间
设备队列的队首指针
设备队列的队尾指针
COCT集合
COCT
CHCT集合
CHCT
表目1
……
表目i
……
控制器标识符
控制器状态（忙/闲）
CHCT表指针
控制器队列的队首指针
控制器队列的队尾指针
表目1
……
表目i
……
通道标识符
通道状态（忙/闲）
COCT表指针
通道队列的队首指针
通道队列的队尾指针

八、设备管理数据结构


单通道系统的设备分配流程
进程P提出所需的I/O设备
根据物理设备名，从SDT中找出该设备的
由DCT的设备状态字段，看该设备是否忙？
若忙
不忙
将该进程的PCB插入到该设备的等待队列中
本次设备分配是否安全？
不安全
安全
将此设备分配给该进程
从其DCT中找出与该设备连接的控制器的COCT，看该控制器是否忙？
不忙
忙
将该控制器分配给进程
将该进程的PCB插入到该控制器的的等待队列中
从COCT中找到与该控制器连接的通道的CHCT，看该通道是否忙？
忙
不忙
将该进程的PCB插入到该通道的的等待队列中
将该通道分配给进程
至此，设备、控制器和通道三者都已分配成功
启动I/O设备，进行具体的I/O操作