

I/O(设备)管理

1. IO系统给的组成

1.1. IO系统硬件：设备+控制器+通道+总线

1.1.1. IO设备

性能指标有：数据传输速率，数据传输单位，设备共享属性

分类：

1 按使用特性：存储(磁盘)+人机交互(键盘/显示器/打印机)+网络通信(网口)

2 按信息交换单位：

1. 字符设备：用于数据IO，如键盘/打印机/显示器
2. 块设备：用于存储信息，如磁盘
3. 其他：如时钟

3 传输速率：低速(键盘/鼠标)+中速(打印机)+高速设备(磁盘)

4 按共享属性：

1. 独占：不允许多进程访问，如打印机
2. 共享：允许多进程访问，如磁盘
3. 虚拟设备：一个设备逻辑上被共享给多个进程，如虚拟后的打印机

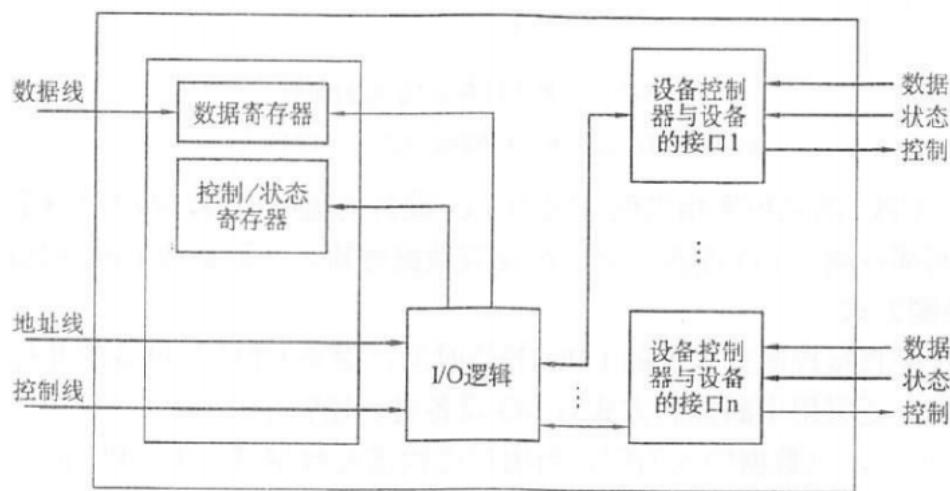
1.1.2. 设备控制器

1 IO设备的组成：电子部分(设备控制器，介于CPU和IO设备之间)+机械部分

2 设备控制器功能：

1. 接收CPU指令，在CPU/设备间交换数据
2. 记录设备状态供CPU查询
3. 识别所控制的设备的地址(设备数=设备地址数)
4. 缓冲CPU和设备间的数据，进行IO差错控制

3 设备控制器结构



1.1.3. IO通道

- 1 概念：独立于CPU的处理器，负责IO控制
- 2 功能：统一管理外设，代替CPU独立执行I/O任务，使I/O与CPU可并行，
- 3 但是指令单一，无自己内存(所以通道程序放主存中)，贵

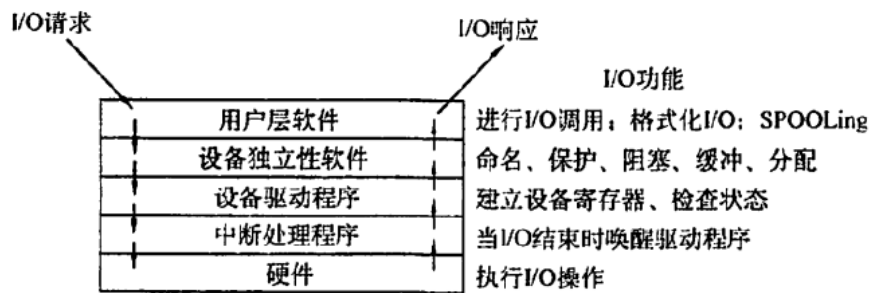


1.2. IO软件

1.2.1. I/O软件功能

- 1 提供设备与用户的接口
- 2 分配/释放设备：使用设备前，需分配设备/通道/控制器，然后把设备给进程
- 3 设备访问/控制：包括并发访问&差错处理
- 4 I/O缓冲调度：让CPU和IO速度匹配
- 5 设备独立性(无关性)：应用程序独立于物理设备，编程时不出现设备名，否则拔掉设备就会出Bug

1.2.2. 结构层次



1 用户层软件：用户通过它发出IO请求，发出请求后用户进程中断等待，随后请求到达下一层 ↓

2 设备独立性软件：提供统一接口，对软件屏蔽了硬件，将请求转发给合适设备驱动程序

3 设备驱动程序：与特定硬件通信，将从上层收到的指令转化为硬件指令，发给指定设备

4 硬件：接收来自设备驱动程序的指令，执行指令

+ 中断处理程序：硬件IO完后，由它通知CPU前来处理中断，确定中断原因+唤醒之前阻塞的驱动程序，完成IO

2. IO控制方式

2.1. 程序IO(轮询)方式

CPU与设备串行工作，CPU循环测试(守着IO全过程)，大量CPU时间被浪费

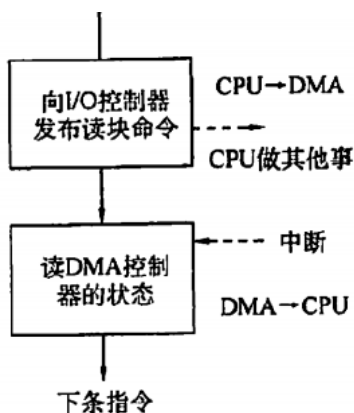
2.2. 中断驱动I/O：现代计算机广泛使用

1 以字(节)为单位进行I/O

2 在I/O设备输入/输出每个数据的过程中，无须CPU干预。

3 仅当传输完一个数据时，才需CPU花极短时间去做些中断处理

2.3. DMA控制方式



1 概述：

1. 外设-内存间开辟通道，直接交换数据(不通过CPU)

2. 以块为单位传输数据

3. 这一过程由DMA控制器完成

3 中断控制&DMA:

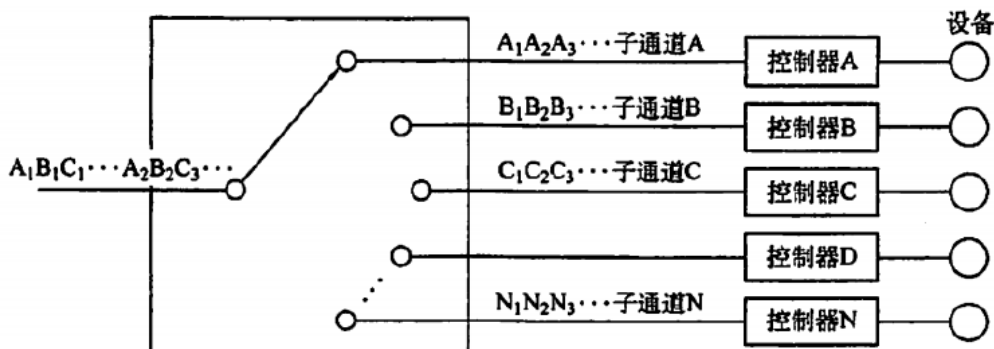
1. DMA在块数据传完后才中断CPU，频率低得多
2. 中断方式处理中断的是CPU，DMA控制方式则是DMA处理中断

6 特点:

1. 设备可以和CPU并行工作
2. 数据只能单向传输，DMA一次也只能执行一条IO指令

2.4. 通道控制方式：外设-内存直接交换数据

1 字节多路通道：多路低速

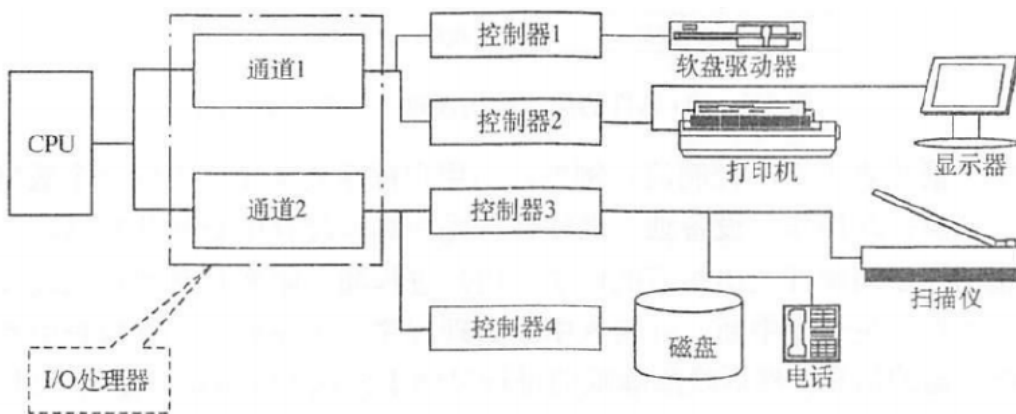


1. 作用：连接多个以字节为单位传输的慢速设备
2. 字节交叉：多个设备轮流在同一通道上发送和接收数据

2 数组选择通道：单路高速

高速设备的通信通道，传输单位为数组，会被一台设备霸占，效率低

3 数组多路通道：多路高速



1. IO通道工作方式：接受CPU委托，独立执行通道程序，实现I/O设备与内存间的信息交换
2. 与DMA控制的差别：DMA要CPU控制块大小和传输内存地址，通道则全权负责

3. IO缓冲

3.1. 缓冲引入

- 1 目的：缓和CPU与I/O设备间速度不匹配的矛盾，减少CPU中断频率，提高CPU和IO设备并行性
- 2 缓冲的设计：进程先输出数据给缓冲，慢速设备从缓冲中读取

3 缓冲的实现：硬件缓冲器(贵)，内存缓冲区(内存中专门划一块来缓存IO数据)

3.2. 三种缓冲

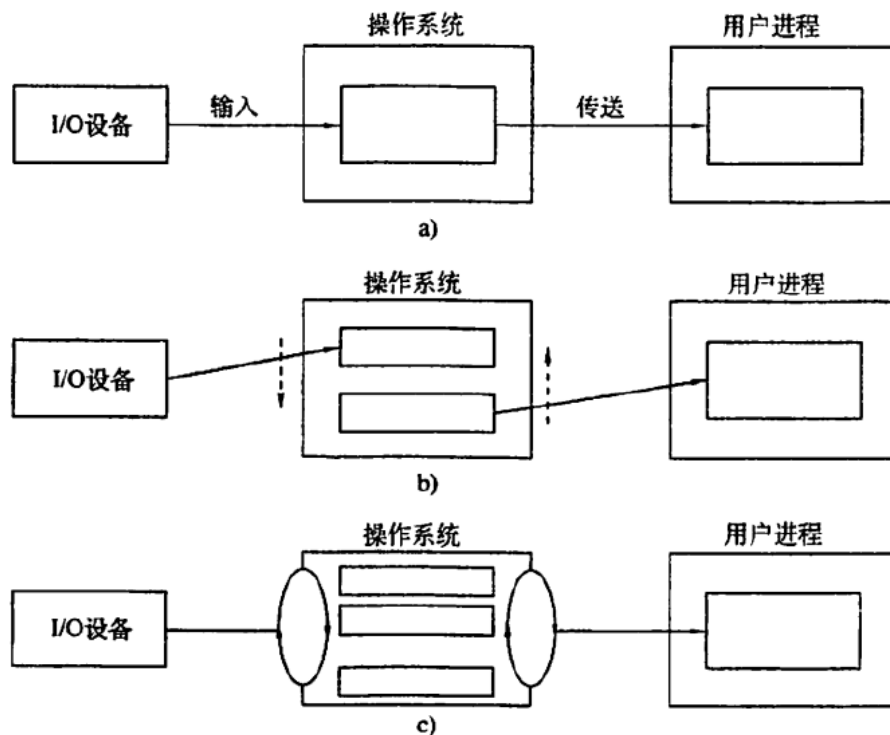


图 5-7 缓冲区工作方式

a) 单缓冲 b) 双缓冲 c) 循环缓冲

1 单缓冲：只有一个IO缓冲区，在内存中，用户发出IO请求时会分配这个缓冲区

2 双缓冲：交替使用两个缓冲区(一个传数据给进程，另一个同时读数据)

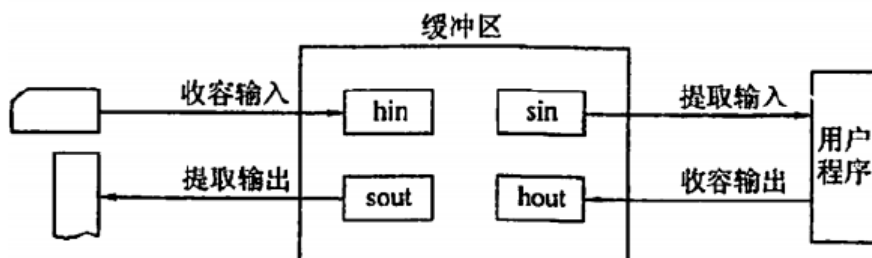
3 循环缓冲：

1. 结构：多个大小相等的缓冲区构成循环链表

2. in/out指针：in指向首个可输入空缓冲区，out指向首个可提取数据满缓冲区

3. IO过程：数据通过in放入缓冲→通过out输出

3.3. 缓冲池：目前广泛使用的



1 缓冲区分类：收容(h)/提取(s)+输入(in)/输出(out)→四种组合

2 缓冲池组成：

1. 空缓冲区→空缓冲队列

2. 输入缓冲区：装满输入数据→输入队列

3. 输出缓冲区输出数据队列

3 输入数据：

1. 收容输入：往空队列中的一个空区输入数据→丢给输入队列→进入缓冲区
2. 提取输入：进程从缓冲区取走数据

4 输出数据

1. 收容输出：往空队列中的一个空区输入数据→丢给输出队列→进入缓冲区
2. 提取输出：设备从缓冲区取走数据

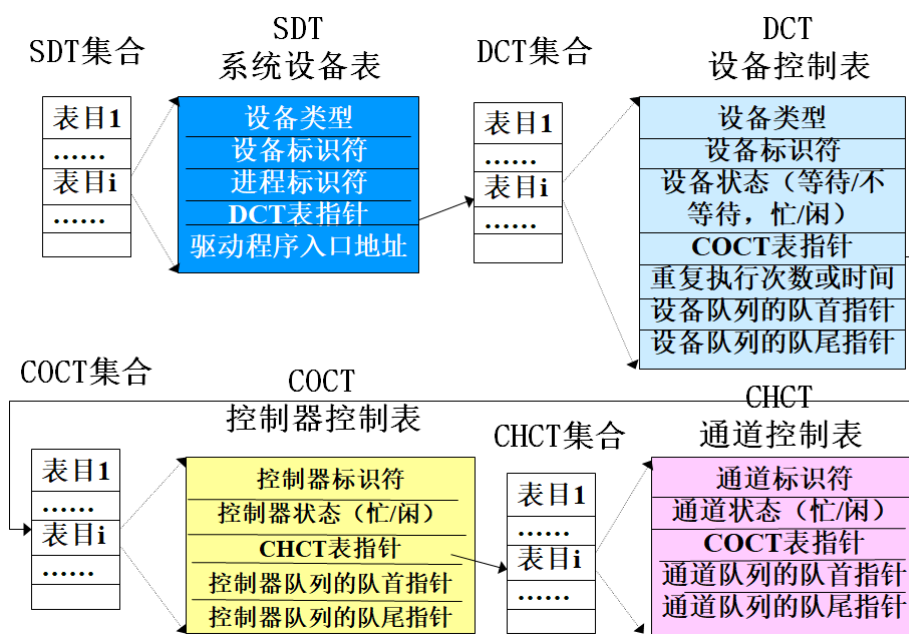
3.4. 高速缓存与缓冲区

1 存放内容不同：Cache缓存低速设备的数据备份，缓冲区存的是中转数据

2 目的不同：Cache是为减少内存访问次数，缓冲区解决速度不匹配

4. 设备的分配：有且只能由OS完成

4.1. 设备管理中的数据结构



1 SDT系统设备表：整个系统只有一张，一个表目待变一个连接的设备

2 DCT设备控制表：与控设备——对应

3 COCT控制器控制表：与控制器——对应

4 CHCT通道控制表：与通道——对应

4.2. 设备分配策略/考虑因素

4.2.1. 设备使用性质

- 1 一进程独占：可以静态/动态分配
- 2 多进程共享：提出I/O请求的不同进程以排队方式分时地占用设备进行I/O
- 3 虚拟化：多进程共享，但每个进程看似独占

4.2.2. 设备分配算法

FCFS，优先级高这优先(优先级相同时采用FCFS)

4.2.3. 安全性：设备分配时不死锁

- 1 静态分配(不死锁)：作业执行前就分配所有设备和控制器，执行时始终占用
- 2 动态分配：进程执行时根据进程的请求分配/释放设备
 - 1. 安全分配(不死锁)：进程每请求一次IO，就被阻塞一次，IO完成后才唤醒，效率低
 - 2. 不安全分配(可能死锁)：进程发出IO请求，进程还可能继续发出IO请求

4.3. 设备独立性

- 1 目的：提高OS的可适应性和可扩展性
- 2 含义：应用程序独立于具体使用的物理设备
- 3 逻辑设备&物理设备
 - 1. OS使用物理设备名，应用程序使用逻辑设备名
 - 2. OS将逻辑设备名→物理设备名
- 4 逻辑设备表(LUT)：OS中每个用户一个表

| 逻辑设备名 | 物理设备名 | 驱动程序入口地址 |
|------------|-------|----------|
| /dev/tty | 3 | 1024 |
| /dev/print | 5 | 2046 |
| ... | ... | ... |
| | | |

| 逻辑设备名 | 系统设备表指针 |
|------------|---------|
| /dev/tty | 3 |
| /dev/print | 5 |
| ... | ... |
| | |

4.4. 设备分配程序

- 1 单通路I/O系统的设备分配：进程请求分配→一次性分配设备+控制器+通道，忙则到等待队列去
- 2 多通路I/O系统的设备分配：一个设备连接多个设备控制器，设备控制器连接多通道
 - 1. 进程提出IO请求
 - 2. 找到第一个空闲设备，验证安全后分配(反之等待)
 - 3. 找到第一个与设备相连的空闲控制器，分配(若无返回第一步)
 - 4. 找到第一个与控制器相连的空闲通道，分配(若无返回第二步)
 - 5. IO启动

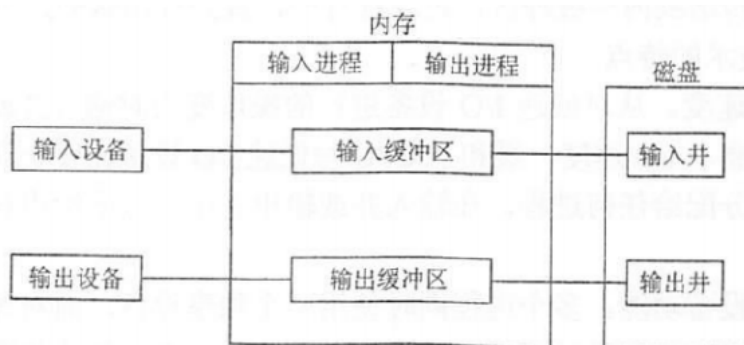
5. 假脱机技术(SPOOLing)

5.1. 概述

- 1 目的：将独占设备改为共享设备技术，使得主机直接控制下实现脱机IO
- 2 原理：

1. 一道程序模拟脱机输入时的外围控制机：数据从低速IO设备→高速磁盘
2. 一道程序模拟脱机输出时的外围控制机：数据从磁盘→低速输出设备

5.2. 组成



- 1 输入/输出井：都为磁盘上存储区
 1. 输入井：模拟脱机输入的磁盘，收容IO输入数据
 2. 输出井：模拟脱机输出的磁盘，收容用户程序输出的数据
- 2 输入/输出缓冲区：在内存中，暂存中转数据
- 3 输入输出进程：模拟脱机输入/输出的外围控制机

5.3. 实现&工作流程

- 1 用户请求打印输出
- 2 输出进程在输出井申请一块空闲盘块区，把打印数据塞进去
- 3 输出进程申请一张空白的用户请求打印表，填入打印要求，然后把表挂上请求打印队列
- 4 输出进程从请求打印队列中取出请求打印表，根据表中要求再从输出井中取出打印内容

6. 设备处理：驱动程序+中断处理


6.1. 设备驱动程序

- 1 目的：封装不同的IO设备及其具体差异，使得OS能统一处理他们
- 2 含义：处理或操作硬件控制器的软件
- 3 处理过程：

抽象要求转为具体的→检查IO合法性→检测设备状态→传输必要参数→设置方式→启动IO

6.2. 中断处理程序

- 1 中断过程：
 1. 设备控制器准备好服务(输入数据已有/输出完成/检测到错误)，会向CPU发中断请求
 2. CPU收到中断请求后，转而去执行中断程序
- 2 中断处理程序的步骤
 1. 唤醒被阻塞的驱动程序进程
 2. 保护被中断进程的CPU环境

- 
3. 分析中断原因，转入相应的设备中断处理程序
 4. 进行中断处理
 5. 恢复被中断进程的现场