
河南大学
操作系统
计算机学院

第 6 章

输入输出系统

大纲

1 6.1 I/O 系统的功能、模型和接口

2 6.2 I/O 设备和设备控制器

3 6.3 中断机构和中断处理程序

4 6.4 设备驱动程序

5 6.5 与设备无关的 I/O 软件

输入输出系统概述

- 输入输出系统是操作系统对计算机系统中除 CPU 和内存之外的外部设备进行管理、对数据传输进行控制的模块，是操作系统资源管理中最复杂、最多样化的部分。
- 外设种类繁多，结构各异。输入输出数据信号类型不同。传输速度差异很大。
- I/O 性能经常成为系统性能的瓶颈。
- CPU 性能不等于系统性能，响应时间也是一个重要因素。CPU 性能越高，与 I/O 差距越大。
- 提高设备利用率的关键是实现设备的并行操作。一方面，设备要与 CPU 并行，另一方面，设备之间也要并行。

1 6.1　I/O 系统的功能、模型和接口

2 6.2　I/O 设备和设备控制器

3 6.3　中断机构和中断处理程序

4 6.4　设备驱动程序

5 6.5　与设备无关的 I/O 软件

I/O 系统的基本功能

- I/O 系统：用于实现数据输入、输出及数据存储的系统。
- I/O 系统的任务
  1. 对 I/O 设备进行控制，完成用户提出的输入/输出要求。
  2. 根据设备请求的情况，按照一定的算法实现对 I/O 设备的合理分配。
  3. 提高设备利用率以及设备与 CPU 的并行操作程度。
- I/O 系统的基本功能
  1. 隐藏物理设备的细节（向上提供抽象的命令）
  2. 实现与设备的无关性（提高可移植性；即插即用）

     在隐藏物理设备的细节的基础上，用户不仅可以使用抽象的命令，也可以使用抽象的逻辑设备名访问设备（不必指明具体的设备）。提高了 OS 的可移植性。

I/O 系统的基本功能

3. 提高处理机和 I/O 设备的利用率（并行）
4. 对 I/O 设备进行控制（驱动程序）

   目前对 I/O 设备进行控制有四种方式：

   ①采用轮询的可编程 I/O 方式

   ②采用中断的可编程 I/O 方式

   ③直接存储器访问方式

   ④I/O 通道方式
5. 确保对设备的正确共享

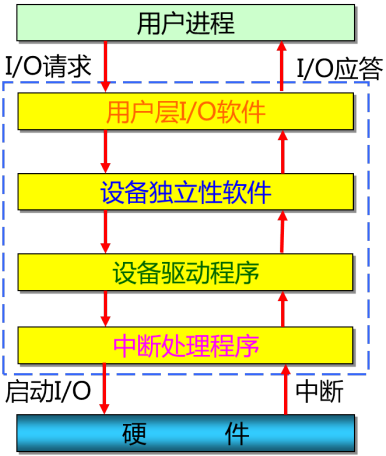
   独占设备：要求互斥访问。共享设备：允许多个进程同时访问。
6. 其他功能（错误处理、缓冲管理等）

I/O 软件的层次结构

I/O 软件的层次结构

1. 用户层 I/O 软件
2. 设备独立性软件
3. 设备驱动程序
4. 中断处理程序

I/O 软件的四个层次



实现与用户的交互接口。进行I/O系统调用，Spooling。

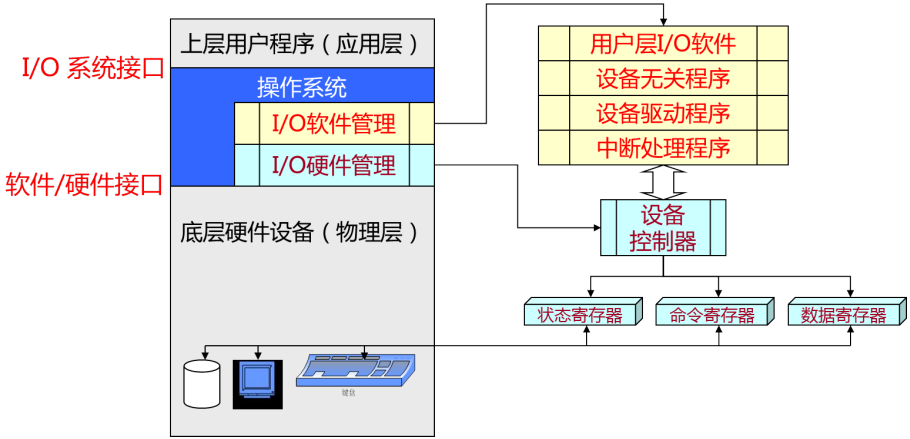
实现与设备的统一接口。设备命名，保护，缓冲，分配。

具体实现对设备发出的操作指令。设置寄存器，检查寄存器状态。

当I/O结束时，唤醒进程。

执行I/O操作

I/O 系统层次结构


I/O 系统接口
软件/硬件接口
上层用户程序（应用层）
操作系统
I/O软件管理
I/O硬件管理
底层硬件设备（物理层）
用户层I/O软件
设备无关程序
设备驱动程序
中断处理程序
设备
控制器
状态寄存器
命令寄存器
数据寄存器

I/O 系统层次结构

- I/O 系统的上、下接口
  1. I/O 系统接口：I/O 系统与上层之间的接口。
     ①块设备接口 ②流设备接口 ③网络通信接口
  2. 软件/硬件（SW/HW）接口：I/O 系统与底层硬件之间的接口。
- I/O 系统的分层（I/O 系统本身的三个层次）：
  1. 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，处理完后恢复现场，并返回到被中断的进程。
  2. 设备驱动程序：用来具体实现系统对设备发出的操作指令，驱动 I/O 设备工作。
  3. 设备独立性软件：使 I/O 软件独立于具体的物理设备。

I/O 系统接口

1 块设备接口

- 块设备：信息交换的基本单位为字符块（比如一个扇区）。属于有结构设备，块大小一般为 512B-4KB。可寻址，即可随机地读/写任意一块。典型的块设备：磁盘等。
- 隐藏了磁盘的二维结构。块设备接口把磁盘扇区依次编号，把磁盘的二维结构变为一维线性结构。
- 把抽象命令映射为底层操作。如把抽象命令中的逻辑块号转换为磁盘的盘面、磁道和扇区号。

I/O 系统接口

2 流设备接口（字符设备接口）

- 字符设备：信息交换的基本单位为字符。属于无结构类型，不可寻址（不能指定数据的输入源地址及输出的目标地址）。典型的字符设备：键盘、打印机和显示器等。
- get 和 put 操作。由于字符设备不可寻址，只能采取顺序访问的方法。通常建立一个字符缓冲区。get 操作从缓冲区取出字符，put 操作把字符放进缓冲区。
- io-control 指令。io-control 指令是设备驱动程序中对 I/O 进行管理的函数。用于预置、读取设备的控制参数。

3 网络通信接口

1 6.1 I/O 系统的功能、模型和接口

2 6.2 I/O 设备和设备控制器

3 6.3 中断机构和中断处理程序

4 6.4 设备驱动程序

5 6.5 与设备无关的 I/O 软件

I/O 设备的类型

I/O 设备的类型

- 按设备的从属关系分类
  - 系统设备：指操作系统生成时即安装操作系统时，就纳入系统管理范围的各种标准设备。如键盘、显示器、打印机等。
  - 用户设备：它是指系统设备之外的非标准设备，在安装操作系统时没有配置，而由用户自己安装配置的设备。如扫描仪等。
- 按使用特性分类
  - 存储设备。
  - I/O 设备。又可分为输入设备和输出设备。

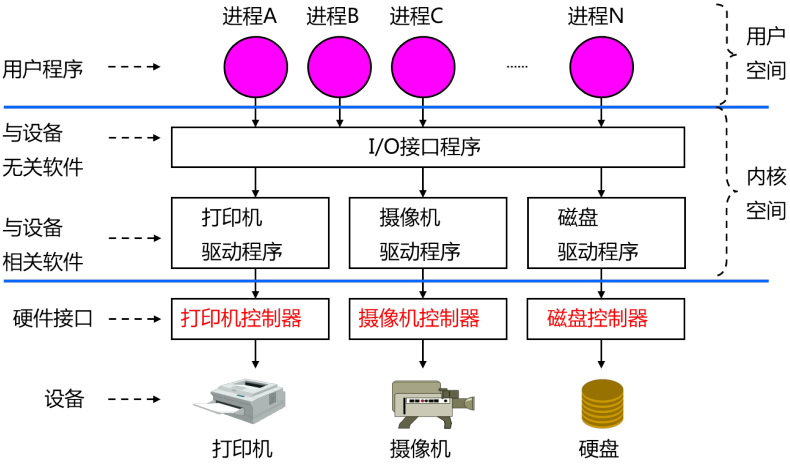
I/O 设备的类型

- 按传输速率分类
  - 低速设备（每秒几字节至数百字节）。如键盘、鼠标、语音输入设备等。
  - 中速设备（每秒数千字节至数万字节）。如打印机等。
  - 高速设备（每秒数兆字节以上）。如磁盘等。
- 按使用方式/共享属性分类
  - 独享/独占设备，如打印机等。包括所有的字符设备
  - 共享设备，如磁盘等。包括大多数的块设备
  - 虚拟设备。通过一定的辅助存储器和控制程序，把一台独占设备虚拟为若干台共享设备。实际上这种“共享”设备并不存在，因此称为“虚拟设备”。

设备控制器

- 传统的设备 = 机械部分 + 电子部分
- 电子部分在系统的控制下驱动机械部分运转，完成 I/O 操作。
- 由于设备中电子部分比机械部分的速度快得多，为了降低硬件成本，将电子部分从设备中分离出来作为一个独立的部件，这就是设备控制器。控制器常采用印刷电路卡的形式插入计算机中（接口卡）。
- 一个控制器可与多个设备相连（因为控制器的速度快于机械部分），交替地或分时地控制与其相连的设备。例如，磁盘控制器可以控制多个磁盘驱动器。

设备控制器


进程A
进程B
进程C
.....
进程N
用户空间
用户程序
与设备无关软件
I/O接口程序
内核空间
与设备相关软件
打印机驱动程序
摄像机驱动程序
磁盘驱动程序
硬件接口
打印机控制器
摄像机控制器
磁盘控制器
设备
打印机
摄像机
硬盘

设备控制器

- 设备控制器负责控制一个或多个 I/O 设备，实现处理机与 I/O 设备之间的数据交换。
- 设备控制器是处于 CPU 与 I/O 设备之间的接口。
- 设备控制器是一个可编址设备。当它仅控制一个设备时，它只有一个唯一的设备地址，连接多个设备时，则含有多个设备地址。
- 功能：接收和识别命令 控制寄存器、实现数据交换 数据寄存器、了解设备状态 状态寄存器、识别设备地址 地址译码器、数据缓冲 缓冲器、差错控制。

设备控制器的组成

1. 设备控制器与处理机的接口

   用于实现 CPU 与设备控制器之间的通信。共有三类信号线: 数据线、地址线和控制线。

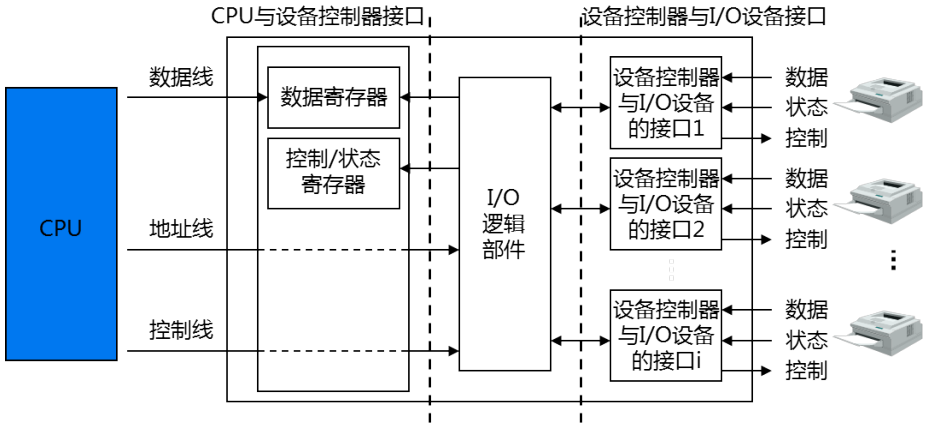
2. 设备控制器与设备接口

   一个接口连接一台设备。在每个接口中都存在数据、控制和状态三种类型的信号。

3. I/O 逻辑：用于实现对设备的控制。通过一组控制线与处理机交互，接收命令并对命令和地址进行译码。

4. 寄存器：控制寄存器（存放命令及参数）、数据寄存器（存放数据）、状态寄存器（记录设备状态）。

设备控制器的组成


CPU与设备控制器接口
设备控制器与I/O设备接口
CPU
数据线
地址线
控制线
数据寄存器
控制/状态
寄存器
I/O
逻辑
部件
设备控制器
与I/O设备
的接口1
设备控制器
与I/O设备
的接口2
设备控制器
与I/O设备
的接口i
数据
状态
控制
数据
状态
控制
数据
状态
控制

内存映像 I/O

- 驱动程序把抽象的 I/O 命令通过驱动程序变成具体的命令和参数，装入设备控制器的寄存器中，由控制器执行。
- 有两种实现方式：
  - 把CPU 寄存器的内容复制到控制器寄存器中：
    Io-store cpu-reg, dev-no, dev-reg
  - 把CPU 寄存器的内容复制到内存地址为 k 的单元中：
    Store cpu-reg, k
- 缺点：访问内存和访问设备需要两个不同的指令。

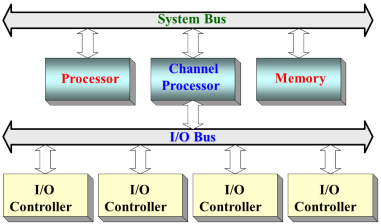
内存映像 I/O

- 内存映像 I/O
  - 不区分内存单元地址和控制器寄存器地址（寄存器占用内存地址的一部分），采用统一的 I/O 指令。

    Store cpu-reg, k
  - k 小于 n 时，被认为是内存地址，大于等于 n 时，被认为是某个控制器寄存器地址。
- 优点：不需要专门的 I/O 指令，统一了对内存和控制器的访问方法。凡是可对内存操作的指令都可对 I/O 控制器操作。
- 缺点：占用内存地址空间。

通道

- 通道是一种特殊的专门执行 I/O 指令的处理机，与 CPU 共享内存，可以有自己的总线。
- 引入通道的目的
  - 解脱 CPU 对 I/O 的组织、管理。
  - CPU 只需发送 I/O 命令给通道，通道通过调用内存中的相应通道程序完成任务，仅当通道完成了规定的 I/O 任务后，才向 CPU 发中断信号。



通道

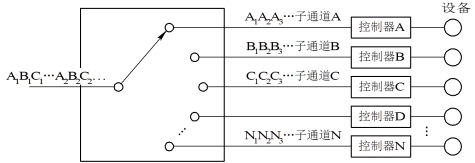
- 通道与一般处理机的差异
  - 执行的命令主要局限于与 I/O 操作有关的指令。
  - 通道没有自己的内存（与 CPU 共享内存）。
- 通道包含通道指令（空操作，读操作，写操作，控制，转移操作），并可执行用这些指令编写的通道程序。

通道类型

1. 字节多路通道
2. 数组选择通道
3. 数组多路通道

字节多路通道

- 工作原理: 数据传送是按字节交叉方式工作。
  - 一个主通道含有多个子通道 A、B、C、· · · 。
  - 每子通道通过一个控制器与一台中/低速的 I/O 设备相连，可同时并行向主通道传数据。
  - 各子通道以时间片轮转方式（完成一个字节）按字节交叉使用主通道。
- 优点：可连多台中/低速设备；能分时并行操作。
- 缺点：传输率较低。



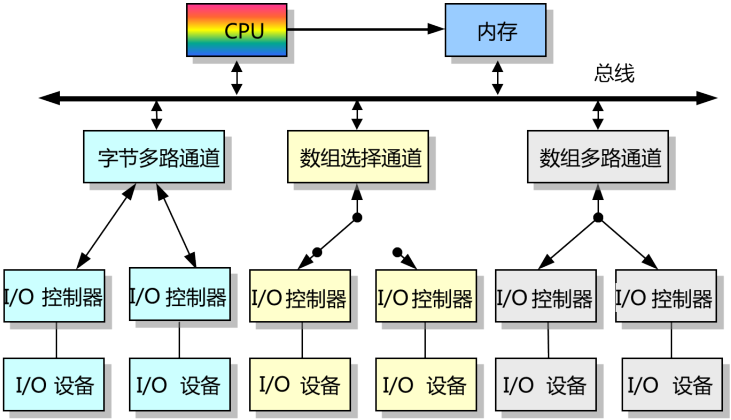
数组选择通道

- 数据传送是按成组方式进行工作，每次传输一批数据。主要用于连接高速 I/O 设备。
  - 一个主通道含有多个子通道 A、B、C、···。
  - 每子通道通过一控制器与一台高速的 I/O 设备相连，在一段时间内只能选择一个子通道程序执行。
- 优点：可连多台高速设备；传输率较高。
- 缺点：一段时间内只能执行一道通道程序。某子通道不传数据，而使主通道闲置，其它子通道也不能传数据。所以通道的利用率很低。

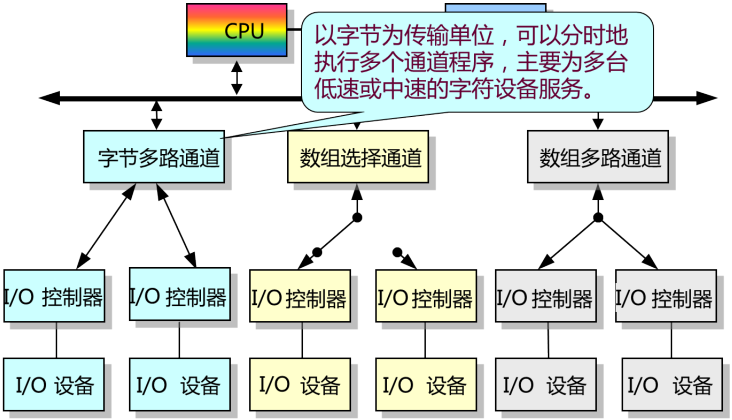
数组多路通道

- 数据传送仍是按数组方式工作。工作原理（结合前两者：并行 + 数组）
  - 一个主通道含有多个子通道 A、B、C、···。
  - 每子通道通过一控制器与一台中/高速的 I/O 设备相连，可同时并行向主通道传数据。
  - 各子通道以时间片轮转方式按数组方式使用主通道。
- 优点：可连多台中/高速设备；能分时并行操作，传输率较高。

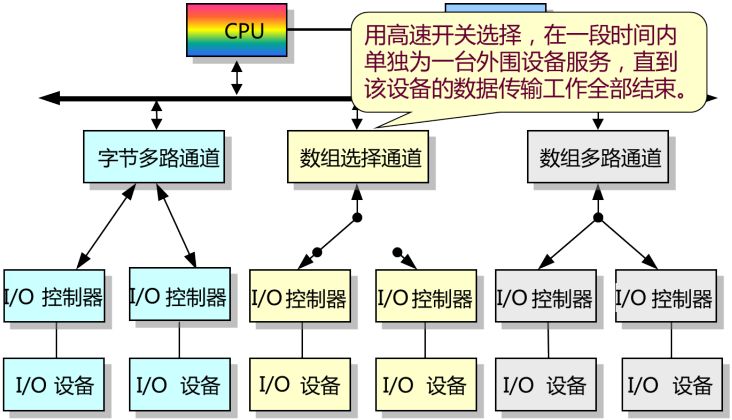
通道类型


CPU
内存
总线
字节多路通道
数组选择通道
数组多路通道
I/O 控制器
I/O 控制器
I/O控制器
I/O控制器
I/O控制器
I/O 控制器
I/O 设备
I/O 设备
I/O 设备
I/O 设备
I/O 设备
I/O 设备

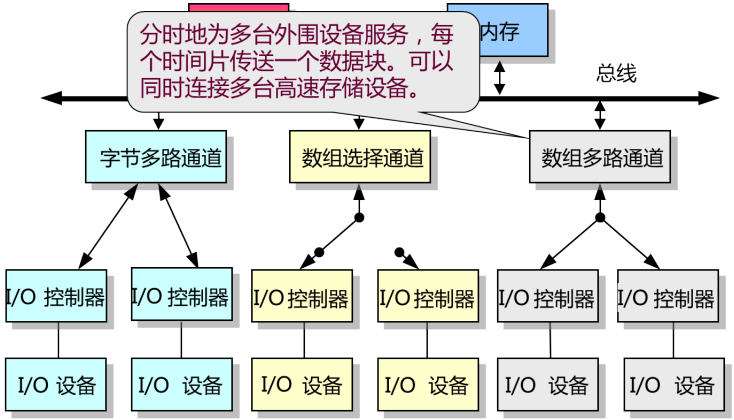
通道类型


CPU
以字节为传输单位，可以分时地执行多个通道程序，主要为多台低速或中速的字符设备服务。
字节多路通道
数组选择通道
数组多路通道
I/O 控制器
I/O 控制器
I/O控制器
I/O控制器
I/O控制器
I/O 控制器
I/O 设备
I/O 设备
I/O 设备
I/O 设备
I/O 设备
I/O 设备

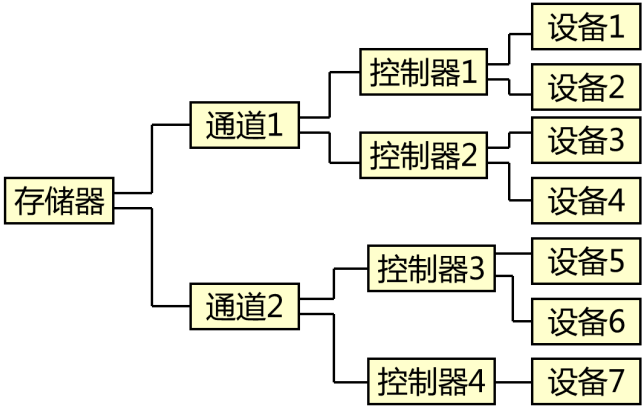
通道类型


CPU
用高速开关选择，在一段时间内
单独为一台外围设备服务，直到
该设备的数据传输工作全部结束。
字节多路通道
数组选择通道
数组多路通道
I/O 控制器
I/O 控制器
I/O控制器
I/O控制器
I/O控制器
I/O控制器
I/O 设备
I/O 设备
I/O 设备
I/O 设备
I/O 设备
I/O 设备

通道类型

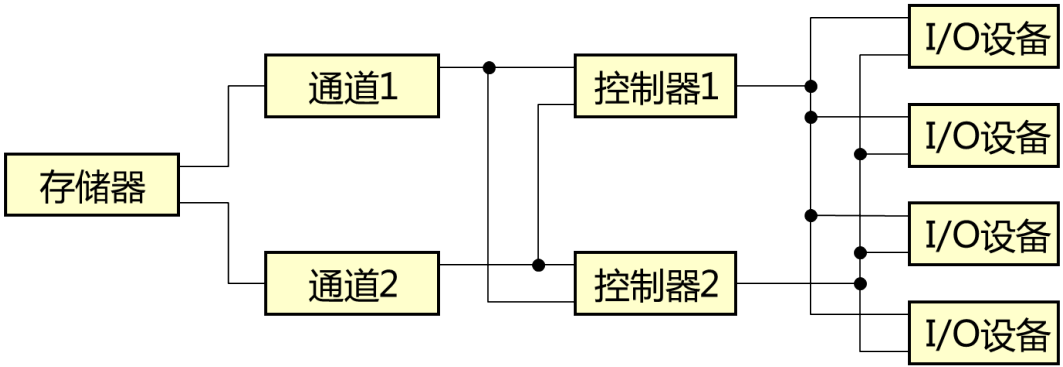

分时地为多台外围设备服务，每个时间片传送一个数据块。可以同时连接多台高速存储设备。
内存
总线
字节多路通道
数组选择通道
数组多路通道
I/O 控制器
I/O 控制器
I/O控制器
I/O控制器
I/O控制器
I/O 控制器
I/O 设备
I/O 设备
I/O 设备
I/O 设备
I/O 设备
I/O 设备

“瓶颈”问题



单通路 I/O 系统

“瓶颈”问题

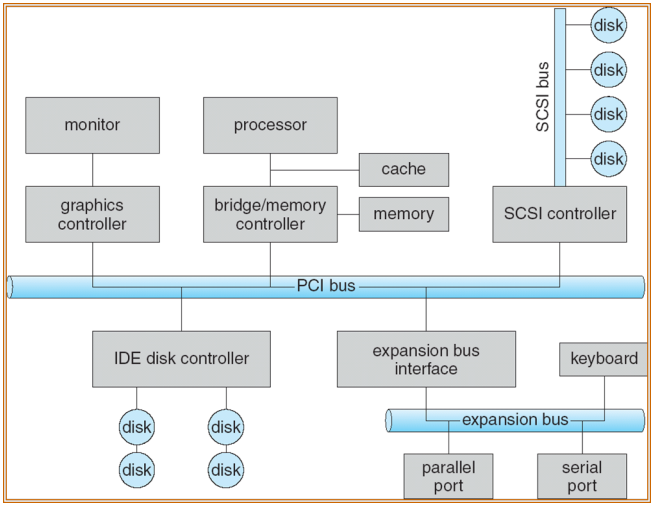


多通路 I/O 系统

总线与局部总线

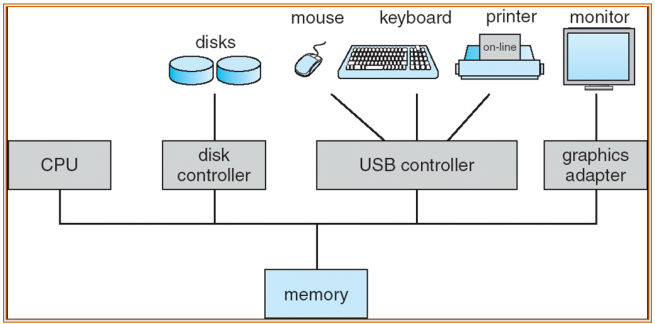
- 总线
  - 总线是构成计算机系统的一种互连机构，是多个系统功能部件之间进行数据传送的公共通路。
  - 在物理上总线是一组信号线的集合（数据线、地址线、控制线）。
- 局部总线
  - 局部总线是指来自处理器的延伸线路，在 CPU 及高速外围设备之间提供了一座桥梁，可缩短设备取得总线控制权所需的时间，提高数据吞吐量。
  - 可以把一些高速外设，如图形卡、硬盘控制器等通过局部总线直接挂接到 CPU 总线上。

局部总线（PCI 总线）


disk
disk
disk
disk
SCSI bus
monitor
processor
cache
graphics controller
bridge/memory controller
memory
SCSI controller
PCI bus
IDE disk controller
expansion bus interface
keyboard
disk
disk
disk
disk
expansion bus
parallel port
serial port

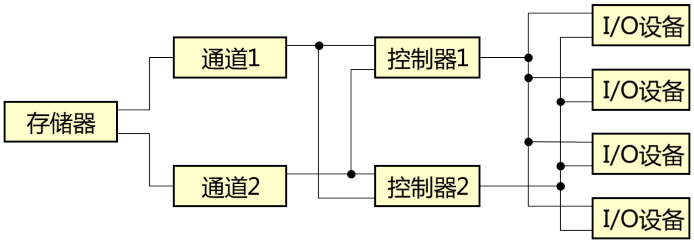
I/O 系统结构

- 三级结构 I/O 系统
  - 存储器 ⇒ 设备控制器 ⇒ 设备
  - 设备控制器与设备是一对多的关系，系统通过它与设备通信。
  - 缺点：总线瓶颈。



I/O 系统结构

- 四级结构 I/O 系统
  - 存储器 ⇒ 通道 ⇒ 设备控制器 ⇒ 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。



- PCI-E（PCI-Express）总线

1 6.1 I/O 系统的功能、模型和接口

2 6.2 I/O 设备和设备控制器

3 6.3 中断机构和中断处理程序

4 6.4 设备驱动程序

5 6.5 与设备无关的 I/O 软件

中断简介

- 中断：CPU 对外部某个事件作出的一种响应。某个事件发生时，系统中止现运行程序的执行，引出处理事件程序对相应事件进行处理，处理完毕后返回断点继续执行。
- 陷入：由 CPU 内部事件引起的中断。如非法指令、地址越界等。（内中断）
- 中断处理程序：对中断事件进行处理的程序。如时钟中断处理、打印机完成中断处理、打印机缺纸中断处理等。

中断简介

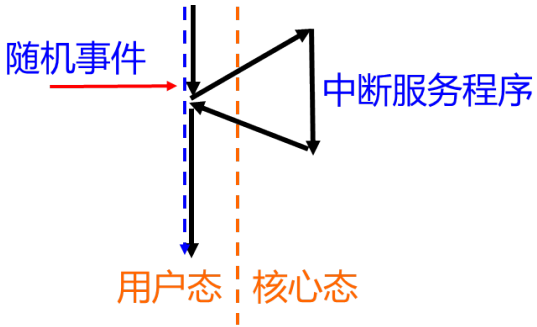
- 中断向量表：表项为中断处理程序的入口地址。当设备发出中断请求信号后，根据中断号查中断向量表，取得该设备中断处理程序的入口地址。
- 中断优先级

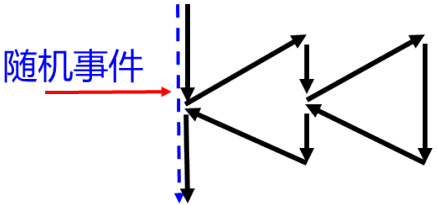
对多中断源的处理方式

1. 屏蔽中断：在处理一个中断时屏蔽掉所有的中断。所有的中断都会按次序顺序处理。优点是简单，但不适合实时性要求较高的系统。
2. 嵌套中断：①如果同时有多个中断请求，CPU 优先响应优先级最高的中断请求。②高优先级中断请求抢占低优先级中断的处理机。

中断简介

- 中断与嵌套中断





中断处理程序

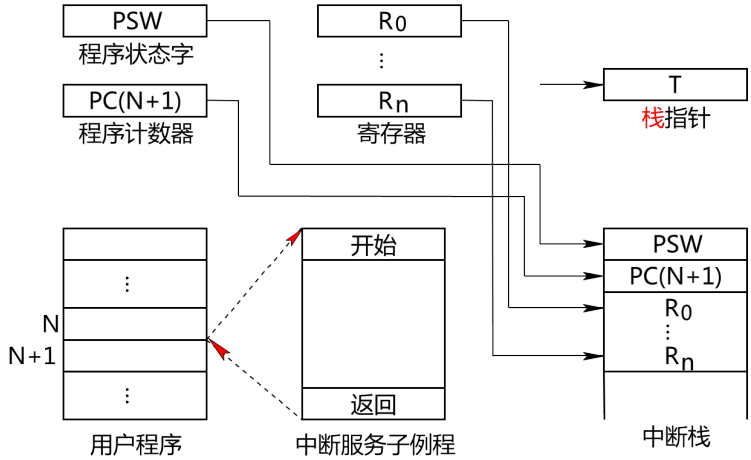
- 当一个进程请求 I/O 时，进程将被阻塞，直到 I/O 完成。
- I/O 完成后，设备控制器向 CPU 发出一个中断请求，CPU 响应中断请求后转向中断处理程序。
- 中断处理程序执行相应的处理后，唤醒被阻塞的进程。

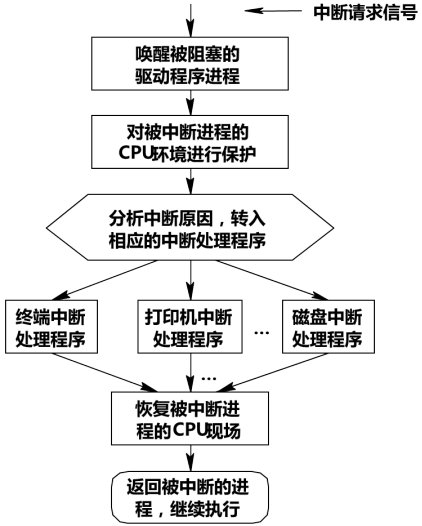
中断处理程序是操作系统中与硬件最接近的一部分

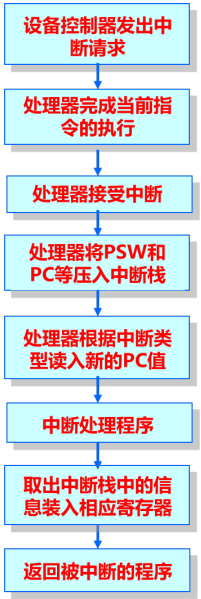
中断处理程序的处理过程

1. 检测是否有未响应的中断请求。
2. 保护被中断进程的 CPU 环境。
3. 分析中断原因（确定引起本次中断的 I/O 设备），将相应的设备中断处理程序的入口地址装入到程序计数器中，转入中断处理程序。
4. 进行中断处理（正常中断、异常中断）。
5. 恢复被中断进程的现场并退出中断。
   - 如果采用屏蔽中断方式，返回被中断进程。
   - 如果采用嵌套中断方式，没有优先级更高的中断请求，返回被中断进程。否则处理优先级最高的中断请求。

中断处理程序的处理过程




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
分析中断原因，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


设备控制器发出中断请求
处理器完成当前指令的执行
处理器接受中断
处理器将PSW和PC等压入中断栈
处理器根据中断类型读入新的PC值
中断处理程序
取出中断栈中的信息装入相应寄存器
返回被中断的程序

1 6.1 I/O 系统的功能、模型和接口

2 6.2 I/O 设备和设备控制器

3 6.3 中断机构和中断处理程序

4 6.4 设备驱动程序

5 6.5 与设备无关的 I/O 软件

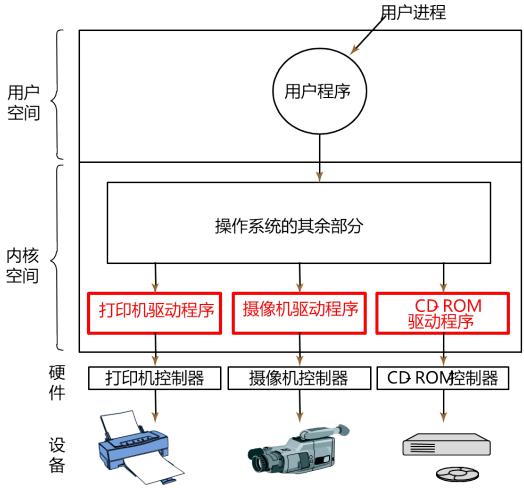
系统处理输入输出请求的步骤

- I/O 软件系统接受用户发出的 I/O 请求，需要通过系统调用取得操作系统的服务。
- 执行到与 I/O 请求相对应的系统调用后，转去执行操作系统的核心程序（设备驱动程序），此时处理机状态由用户态转到核心态。
- 启动设备工作。
- I/O 完成后，由通道（或设备）产生中断信号。CPU 接到中断请求后，如条件符合，则响应中断，转去执行相应的中断处理程序。唤醒因等待 I/O 而阻塞的进程，调度用户进程继续运行。

设备驱动程序

- 设备驱动程序通常又称为设备处理程序，它是 I/O 进程与设备控制器之间的通信程序，它常以进程的形式存在。
- 一个设备驱动程序通常由若干子程序组成，每个子程序实施一种或一组物理操作。如读、写子程序、设备初始化、关闭设备、控制和配置设备子程序等。通过这些子程序向上层提供了与硬件无关的接口。
- 每一类设备都需要配置一种驱动程序。

设备驱动程序在系统中的位置



设备驱动程序的功能

- 将接收到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。
- 检查用户 I/O 请求的合法性、I/O 设备状态、传递有关参数、设置设备的工作方式。
- 发出 I/O 命令。按处理机的 I/O 请求去启动指定的设备进行 I/O 操作。
- 及时响应由控制器或通道发来的中断请求，并进行相应处理。

设备驱动程序的特点

- 设备驱动程序是实现在与设备无关软件与设备控制器之间的一个通信和转换程序。把抽象的命令转为具体的 I/O 操作。
- 与 I/O 设备的特性紧密相关。
- 与 I/O 控制方式紧密相关（中断驱动、DMA）。
- 与硬件紧密相关，因而其中一部分程序必须用汇编语言编写。很多驱动程序的基本部分已经固化在 ROM 中。
- 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。

设备处理方式

设备处理的三种方式

1. 操作系统为每一类设备设置一个进程，专门执行这类设备的 I/O 操作。
2. 在整个系统中设置一个进程，执行所有的 I/O 操作。
3. 不设置专门的设备处理进程，而为各类设备设置相应的设备驱动程序，供用户进程或系统进程调用。这类方式目前使用最多。

设备驱动程序的处理过程

1. 将接收到的抽象要求转换为具体要求。
2. 检查用户 I/O 请求的合法性。如：试图从打印机读数据为非法请求。
3. 读出和检查 I/O 设备状态。驱动程序把设备控制器中状态寄存器的内容读入 CPU 寄存器，检测设备是否就绪。
4. 传送必要参数，如内存始址和字节数。
5. 设置设备的工作方式，如同步、异步、单工、双工等。
6. 按处理机的 I/O 请求去启动指定的设备进行 I/O 操作。驱动程序进程把自己阻塞起来，直到中断到来才唤醒。

信息单向传送为单工，不能同时双向传送称为半双工，同时双向传送则称为全双工

对 I/O 设备的控制方式

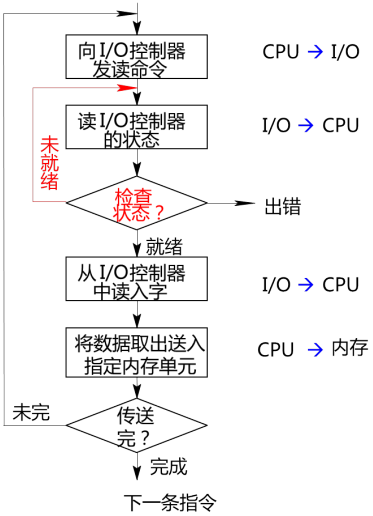
I/O 设备控制方式

1. 采用轮询的可编程 I/O 方式（程序直接控制方式）
2. 采用中断的可编程 I/O 方式
3. 直接存储器访问方式（DMA）
4. I/O 通道方式

采用轮询的可编程 I/O 方式

- 早期计算机系统无中断机构、系统对 I/O 的控制采用程序直接控制方式，或称为程序 I/O 方式、忙 / 等待方式。
- 当 CPU 向设备控制器发送完命令后（状态寄存器中的忙/闲标志 busy 置为 1），CPU 就循环检测设备控制器中的状态寄存器，看 I/O 操作是否完成，直到 I/O 完成（busy=0）。
- CPU 需要花代价不断查询 I/O 状态（轮询）。
- CPU 大部分时间都处于检查和等待状态，整个计算系统的效率十分低下。

采用轮询的可编程 I/O 方式

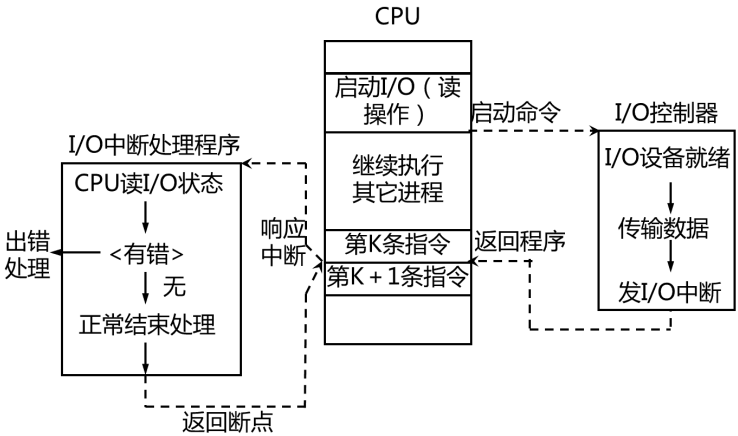


- 处理机向控制器发出一条I/O指令时，要同时把状态寄存器中的busy置为1，然后便不断地循环测试busy。
- 当busy=1时，表示输入机尚未输完一个字(符)，处理机应继续对该标志进行测试，直至busy=0
- 以字(节)为单位进行I/O。

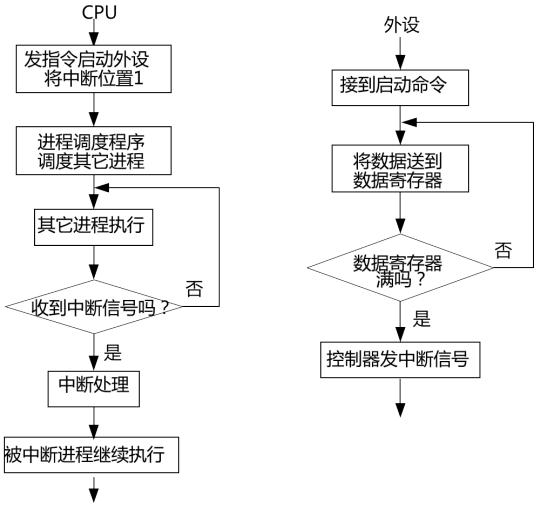
采用中断的可编程 I/O 方式

- 进程向 CPU 发出指令启动 I/O 设备输入数据。
- 该进程放弃处理机（CPU 转去做其它工作），等待输入完成。
- 输入完成后，I/O 控制器向 CPU 发出中断请求，CPU 收到后，转向中断服务程序。中断服务程序将输入寄存器中的数据送指定内存单元，并将原进程唤醒，继续执行。

采用中断的可编程 I/O 方式



采用中断的可编程 I/O 方式



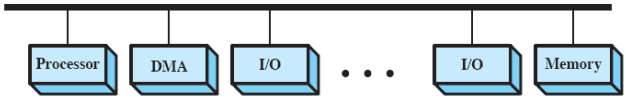
采用中断的可编程 I/O 方式

- 优点：CPU 不需等待数据传输完成，I/O 设备与 CPU 并行工作，CPU 的利用率因此提高。
- 例如，从终端输入一个字符的时间约为 100 ms，而将字符送入终端缓冲区的时间小于 0.1 ms。若采用程序 I/O 方式，CPU 约有 99.9 ms 的时间处于忙等待。采用中断驱动方式，CPU 可利用这 99.9 ms 的时间去做其它事情，而仅用 0.1 ms 的时间来处理中断请求。
- 缺点：CPU 在响应中断后，还需要时间来执行中断服务程序。如果数据量大，需要多次执行中断程序，CPU 的效率仍然不高。

直接存储器访问方式（DMA）

- 引入
  - 程序直接控制方式是以字 (节) 为单位进行 I/O。
  - 中断控制方式仍是以字 (节) 为单位进行 I/O。也就是说在采用中断控制方式的 I/O 中，CPU 每字节的传送就产生一次中断。
- 直接存储器访问 (DMA)I/O 控制方式
  - 数据传输的基本单位是数据块。
  - 数据从设备直接送入内存或相反。
  - 整块数据的传送在 DMA 控制器的控制下完成，CPU 仅在数据传输的起止时参与。

直接存储器访问方式（DMA）



向I/O控制器发布读块命令 → CPU → DMA

CPU做其它事

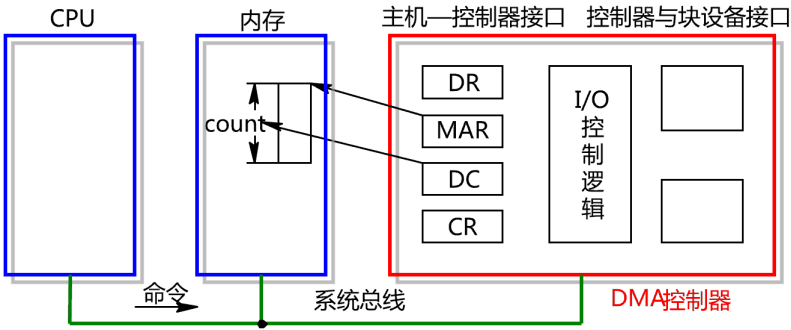
读 DMA 控制器的状态 ← 中断　DMA → CPU

下一条指令

DMA 控制器

- DMA 控制器的组成：
  - 主机与 DMA 控制器的接口
  - DMA 控制器与块设备的接口
  - I/O 逻辑控制
- DMA 控制器中设置如下四类寄存器：
  - 命令/状态寄存器 (CR)
  - 内存地址寄存器 (MAR)：输入时存放设备到内存的内存始址；输出存放由内存到设备的内存源地址。
  - 数据寄存器 (DR)
  - 数据计数器 (DC)：存放本次 CPU 要读或写的字节数。

DMA 控制器



- 命令/状态寄存器(CR)
- 数据寄存器(DR)
- 内存地址寄存器(MAR)
- 数据计数器(DC)

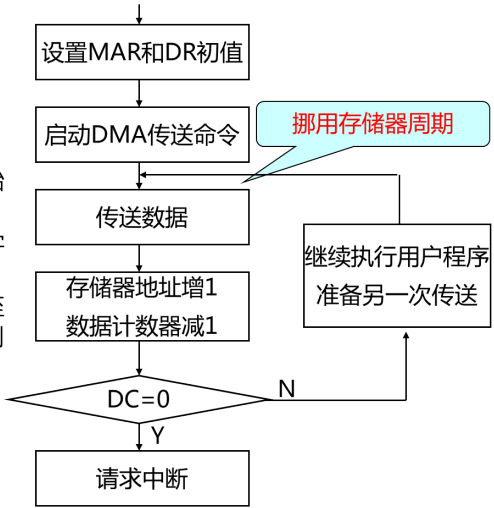
DMA 工作过程

1. 需要 I/O 的进程向 CPU 发出指令，向 DMA 控制器写入数据存放的内存始址、传送的字节数，并置中断位和启动位，启动 I/O 设备输入数据并允许中断。
2. 进程放弃处理机等待输入完成，处理机被其它进程占据。
3. DMA 控制器挪用存储器周期，把一个字节的数据写入内存。
4. DMA 控制器传送完数据后向 CPU 发中断请求，CPU 响应后转向中断服务程序唤醒进程，返回被中断进程。
5. 在以后该进程再被调度，从内存取出数据进行处理。

DMA 工作过程

- 当CPU从磁盘读入数据块时
  - 向DMA控制器发送读命令送到 CR。
  - 发送数据读入的内存始址送入MAR。
  - 发送本次要读数据的字节数送入DC。
  - 将磁盘中的源地址送至DMA控制器的I/O控制逻辑上。

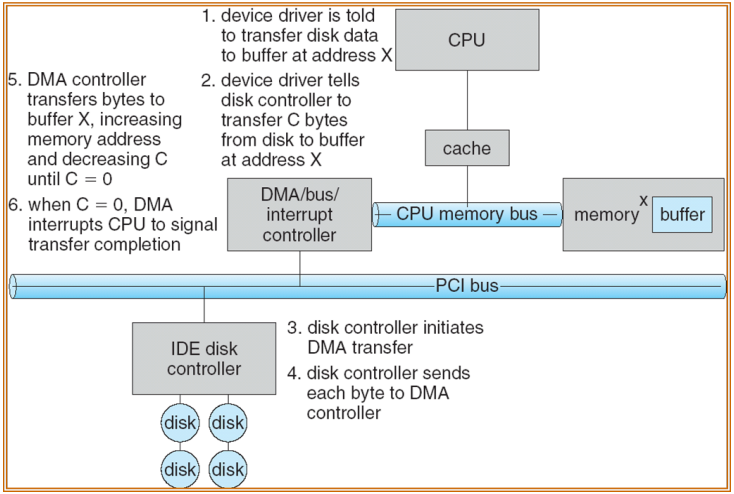
- 命令/状态寄存器(CR)
- 内存地址寄存器(MAR)
- 数据寄存器(DR)
- 数据计数器(DC)



DMA 工作过程

- 周期挪用是利用 CPU 不访问存储器的那些周期来实现 DMA 操作。
- 周期挪用不减慢 CPU 的操作，但数据传送过程是不连续和不规则的。
- DMA 控制器在获得总线控制权后（窃取总线控制权、挪用存储器周期），直接在内存与外设之间实现数据传送。
- CPU 和 DMA 传送不完全并行，会有总线竞争。处理器用总线时可能稍作等待，但不会引起中断，不引起程序上下文的切换。

DMA 工作过程


1. device driver is told to transfer disk data to buffer at address X
2. device driver tells disk controller to transfer C bytes from disk to buffer at address X
3. disk controller initiates DMA transfer
4. disk controller sends each byte to DMA controller
5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0
6. when C = 0, DMA interrupts CPU to signal transfer completion
CPU
cache
DMA/bus/ interrupt controller
CPU memory bus
memory
x
buffer
PCI bus
IDE disk controller
disk
disk
disk
disk

I/O 通道控制方式

- 通道相当于一个独立于 CPU 的专管 I/O 控制的处理机，用于控制设备与内存直接进行数据交换。
- 在程序 I/O 方式和中断方式中，CPU 以字节为单位进行干预；DMA 方式中以数据块为单位进行干预。I/O 通道方式中进一步减少到以一组数据块为单位进行干预。
- 通道有自己的一套简单的指令系统，称为通道指令。每条通道指令规定了设备的一种操作，通道指令序列便是通道程序，通道执行通道程序来完成规定动作。
- 通道靠执行通道程序软件完成数据传输，通道控制器的功能比 DMA 控制器更强大，它能够承担外设的大部分工作。通道有自己的总线，不需要窃取总线控制权。

通道控制工作过程

1. 需要 I/O 的进程向 CPU 发出指令，CPU 发出启动设备指令，指明 I/O 操作、设备号和对应的通道。
2. 该进程放弃 CPU 等待输入完成，CPU 被其它进程占据。
3. 通道接收到 CPU 发来的启动指令后，取出内存中的通道程序执行，控制设备将数据传送到内存指定区域。
4. 传送完数据后，通道向 CPU 发中断请求，CPU 响应后转向中断服务程序，唤醒进程，并返回被中断进程。
5. 在以后该进程再被调度，从内存取出数据进行处理。

通道程序

通道指令的格式

- 操作码：规定了指令所要执行的操作，如读、写等。
- 计数：表示本条指令要读（写）数据的字节数。
- 内存地址：数据要送入的内存地址或从内存何处取出数据。
- 通道程序结束位 P：P=1 表示本条指令是通道程序的最后一条指令。
- 记录结束位 R：R=0 表示本条指令与下一条指令所处理的数据属于一个记录，R=1 表示该指令处理的数据是最后一条记录。

通道程序

- 下面给出了一个由六条通道指令构成的通道程序。
- 该程序的功能是将内存中不同地址的数据写成多个记录。

操作	P	R	计数（字节）	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	1650
Write	1	1	250	2720

中断、DMA、通道

- DMA 方式与中断方式的区别
  - 中断方式是在数据寄存器满之后发中断请求，而 DMA 方式则是在数据块全部传送结束时请求中断。
  - 中断方式的数据传送是在中断处理时由 CPU 控制完成的，而 DMA 方式则是在 DMA 控制器的控制下不经过 CPU 控制完成的。
- DMA 方式与通道方式的区别
  - 在 DMA 方式中，数据的传送方向、存放数据的内存始址以及传送的数据块长度等都由 CPU 控制，而在通道方式中，这些都由通道来进行控制。

中断、DMA、通道

例：假设幼儿园一个老师带 10 个孩子，要给每个孩子分 4 块糖。

1. 方法一：她先给孩子甲一块糖，盯着甲吃完，然后再给第二块，等吃完第二块又给第三块，吃完第三块又给第四块。接着给孩子乙，其过程与孩子甲完全一样。依次类推，直至到第 10 个孩子发完四块糖。孩子们吃糖时她一直在守候，什么事也不能干。（程序直接控制方式）
2. 方法二：每人发一块糖各自去吃，并约定谁吃完后就向她举手报告，再发第二块。在未接到孩子们吃完糖的报告以前，她还可以腾出时间给孩子们改作业。（中断控制方式）

中断、DMA、通道

例：假设幼儿园一个老师带 10 个孩子，要给每个孩子分 4 块糖。

3. 方法三：每人拿 4 块糖各自去吃，吃完 4 块糖后再向她报告。这种方法工作效率大大提高，她可以腾出更多的时间改作业。（直接存储器访问方式 (DMA)）
4. 方法四：把发糖的事交给另一个人分管，只是必要时她才过问一下。（通道控制方式）

中断、DMA、通道

例：假设幼儿园一个老师带 10 个孩子，要给每个孩子分 4 块糖。

3. 方法三：每人拿 4 块糖各自去吃，吃完 4 块糖后再向她报告。这种方法工作效率大大提高，她可以腾出更多的时间改作业。（直接存储器访问方式 (DMA)）
4. 方法四：把发糖的事交给另一个人分管，只是必要时她才过问一下。（通道控制方式）

是否还有其它 I/O 控制方式？

外围处理机 I/O 控制方式

- 外围处理机 (Peripheral Processing Unit, PPU) I/O 控制方式是通道方式的进一步发展。由于 PPU 基本上独立于主机工作，结构上更接近一般处理机。它有自己的指令系统，完成算术/逻辑运算，读/写主存储器，直接与外设交换信息等。有的外围处理机就直接选用已有的通用机。
- 可以在系统中配置多台 PPU，分别承担 I/O 控制、通信、维护诊断等任务。从某种意义上说，这种系统已变成分布式的多机系统。
- 外围处理机 I/O 控制方式一般应用于大型计算机系统。

1 6.1　I/O 系统的功能、模型和接口

2 6.2　I/O 设备和设备控制器

3 6.3　中断机构和中断处理程序

4 6.4　设备驱动程序

5 6.5　与设备无关的 I/O 软件

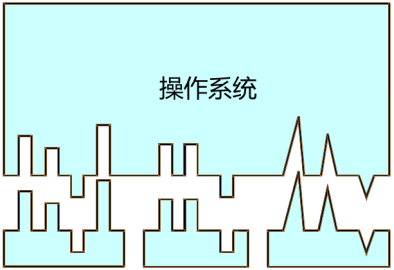
设备独立性的基本概念（设备无关性）

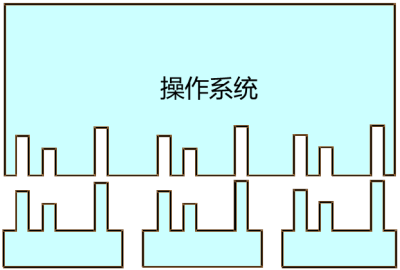
- 早期的 OS 中，使用 I/O 设备都是用物理设备名，这使得应用程序与物理设备直接相关。
- 设备独立性的概念：为提高 OS 的可适应性和可扩展性，将应用程序独立于具体使用的物理设备。
- 为了实现设备独立性而引入了逻辑设备和物理设备两个概念。
  - 逻辑设备（应用程序中）
  - 物理设备（执行中）
- 逻辑设备表（LUT）：逻辑设备到物理设备的映射关系。

与设备无关的软件

- 驱动程序是一个与硬件紧密相关的软件。为了实现设备独立性，必须再在驱动程序之上设置一层软件，称为设备独立性软件（与设备无关的软件）。
- 设备独立性软件执行所有设备的公有操作，具体如下：
  1. 设备驱动程序的统一接口（与设备无关；逻辑设备名映射到具体的物理设备；访问控制）
  2. 缓冲管理
  3. 差错控制
  4. 对独占设备的分配与回收（由系统统一管理，不允许进程自行使用）
  5. 提供独立于设备的逻辑数据块

统一的设备驱动程序接口





(a)无标准的驱动程序接口

(b)标准驱动程序接口

实现设备独立性功能的优点

- 设备分配时的灵活性
  - 逻辑设备和物理设备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：某逻辑名可对应一类设备，提高容错性；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
  - 用于 I/O 操作的设备可以更换（重定向），而不必改变应用程序，只需改变逻辑设备表（LUT）的映射关系。
  - 如：输出设备由显示器改为打印机

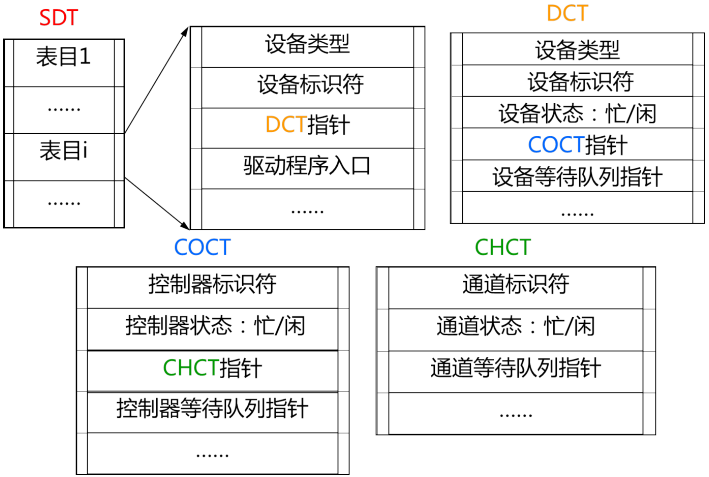
设备分配的数据结构

为了实现对独占设备的分配，系统需要配置如下的数据结构：

设备分配的数据结构

1. 系统设备表 SDT(system device table)
2. 设备控制表 DCT(device control table)
3. 控制器控制表 COCT(controller control table)
4. 通道控制表 CHCT（ channel control table ）

系统设备表SDT、设备控制表DCT
控制器控制表COCT、通道控制表CHCT



设备分配时应考虑的因素

1. 设备的使用性质/固有属性
   - 设备的固有属性：独占性、共享性、可虚拟性
   - 独享分配、共享分配、虚拟分配
   - 可虚拟设备是指一台物理设备在采用虚拟技术后，可变成多台逻辑上的所谓虚拟设备。
     （独占设备 ⇒ 共享设备）
2. 设备分配算法
   - 先请求先服务
   - 优先级高者优先

设备分配时应考虑的因素

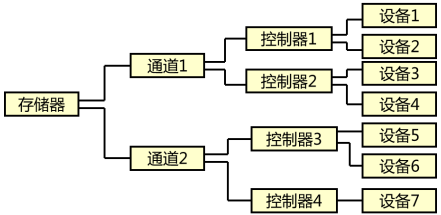
3 设备分配的安全性

- 安全分配方式。进程发出 I/O 请求后，便进入阻塞状态，直到其 I/O 操作完成时才被唤醒。摒弃了死锁的四个必要条件之一的“请求和保持”条件，从而使设备分配是安全的。
- 缺点：进程进展缓慢，CPU 与 I/O 设备串行工作。
- 不安全分配方式。进程在发出 I/O 请求后仍继续运行，需要时又发出第二、第三个 I/O 请求。仅当进程所请求的设备被另一进程占用时，请求进程才进入阻塞状态。
- 优点：可同时操作多个设备，使进程推进迅速。
- 缺点：分配不安全，可能造成死锁。需要计算本次分配的安全性。

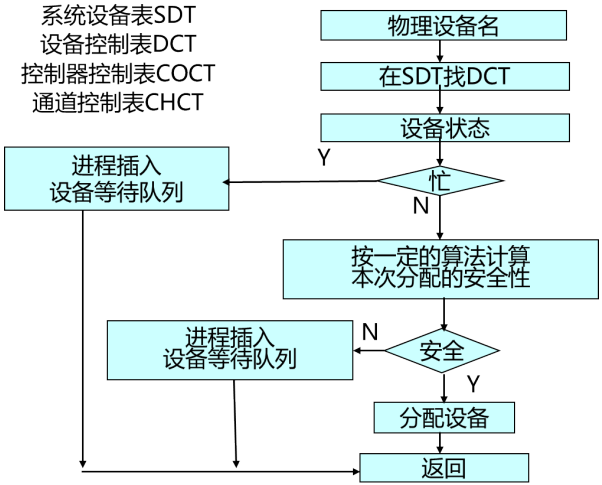
独占设备的分配程序

基本的设备分配程序

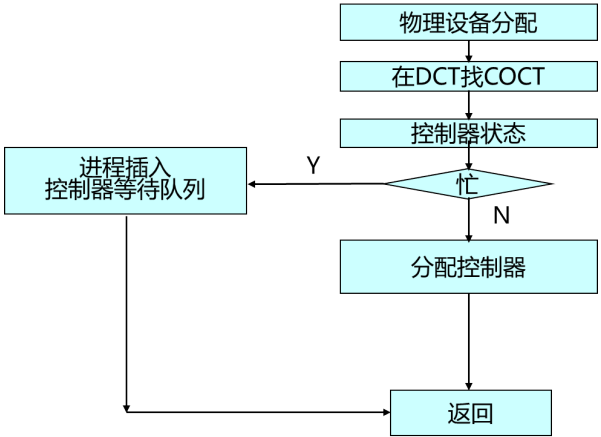
1. 分配设备
2. 分配设备控制器
3. 分配通道



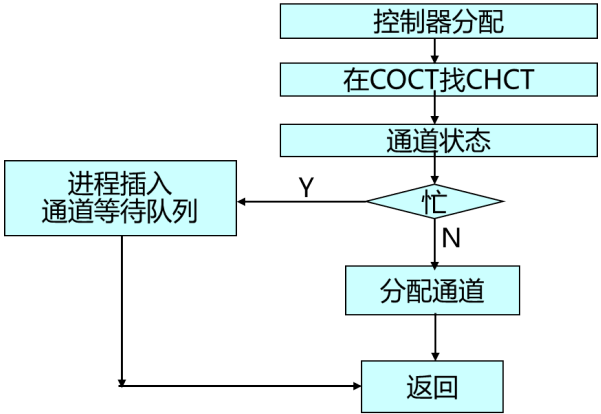
独占设备的分配程序


系统设备表SDT
设备控制表DCT
控制器控制表COCT
通道控制表CHCT
物理设备名
在SDT找DCT
设备状态
忙
Y
N
进程插入
设备等待队列
按一定的算法计算
本次分配的安全性
安全
N
进程插入
设备等待队列
Y
分配设备
返回

独占设备的分配程序



独占设备的分配程序

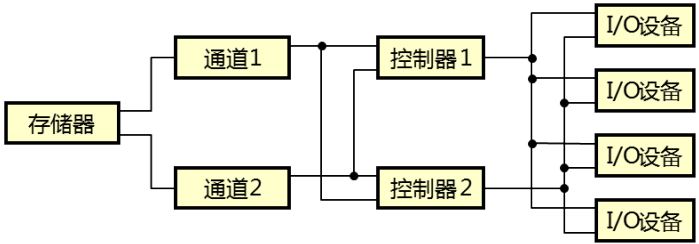


设备分配程序的改进

- 问题一：进程以物理设备名来提出 I/O 请求。
- 问题二：采用的是单通路的 I/O 系统结构（“瓶颈”）。

设备分配程序的改进

- 问题一：进程以物理设备名来提出 I/O 请求。
- 问题二：采用的是单通路的 I/O 系统结构（“瓶颈”）。
- 改进一：增加设备的独立性（进程以逻辑设备名提出 I/O 请求）。
- 改进二：考虑多通路情况。



逻辑设备名到物理设备名映射的实现

- 逻辑设备表 LUT(Logical Unit Table)
- 每个表目中包含三项：逻辑设备名、物理设备名和设备驱动程序的入口地址。
- 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。
- LUT 设置的两种方式
  - 整个系统设置一张 LUT 。要求逻辑名不重复。
  - 每个用户设一张 LUT，该表放入进程的 PCB 中。可限制用户对某些设备的使用。

逻辑设备表 LUT(Logical Unit Table)

整个系统设置一个逻辑设备表LUT

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
…	…	…

每个用户设一个逻辑设备表LUT

逻辑设备名	系统设备表指针
/dev/tty	3
/dev/print	5
…	…

谢 谢

School of Computer & Information Engineering
Henan University
Kaifeng, Henan Province
475001
China