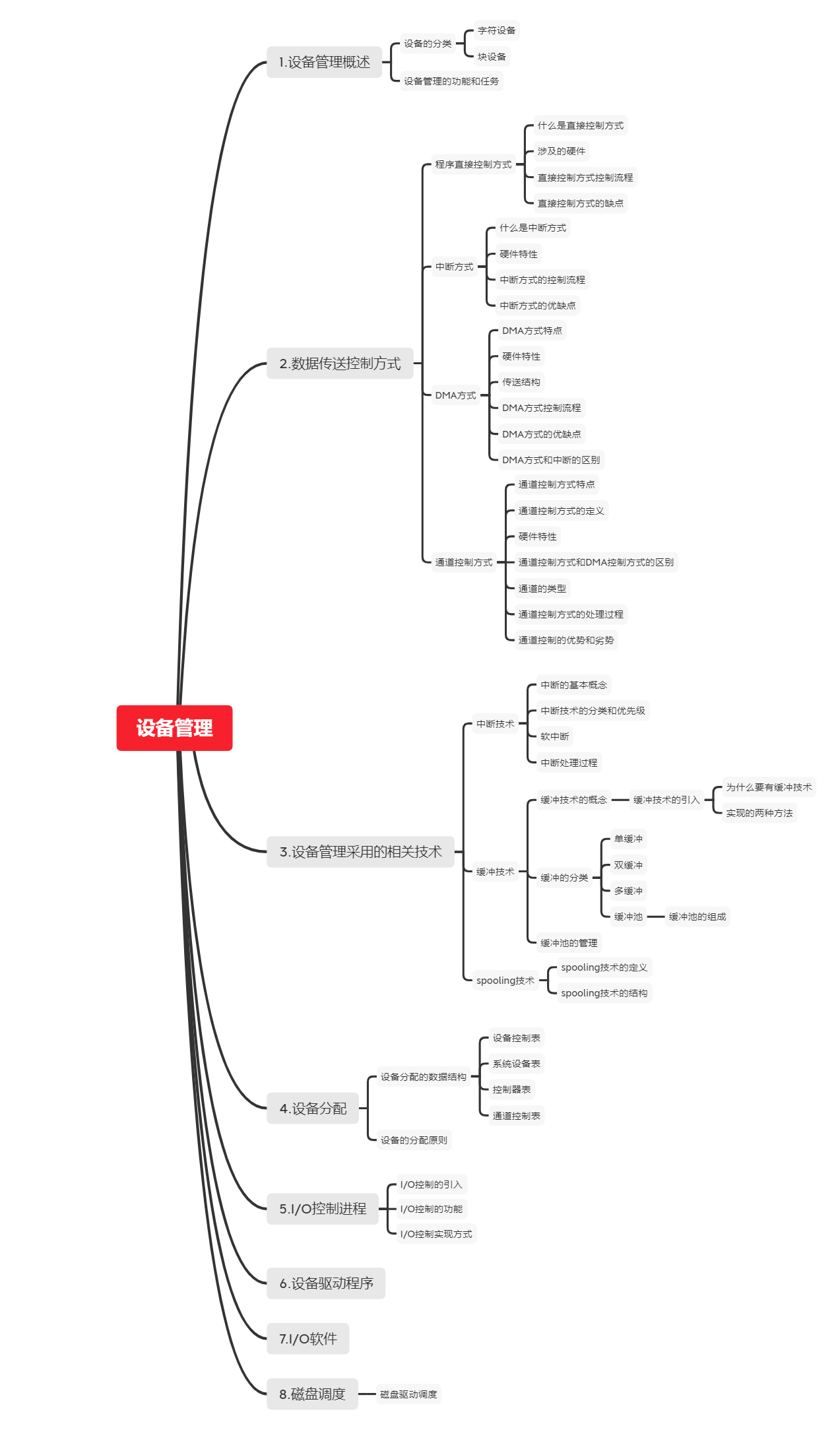
设备管理2

2021年4月5日

13:12

# 



# 9.1引言

## 9.1.1设备的分类

Unix中把设备分为两类，字符设备和块设备。

字符设备：以为字符为单位组织和处理信息的设备，如键盘。

块设备：以字符串位单位组织和处理信息的设备，如磁盘。

## 9.1.2设备管理的功能和任务

1. **主要任务：**
   1. 选择和分配输入输出设备以便进行数据传输操作。
   2. 控制输入输出设备和CPU，内存之间的数据交换。
   3. 把用户和硬件区分开来，用户编写程序时不涉及具体硬件。
2. **主要功能：**
   1. 提供和进程管理系统的接口。
   2. 进行设备分配。
   3. 提高设备间，设备与CPU之间，进程间的并行操作度。
   4. 进行缓冲区管理。

# 9.2数据传送控制方式

1. **数据传送方式主要有四种**：
   1. 程序直接控制方式
   2. 中断方式
   3. DMA方式
   4. 通道控制方式
2. **选择和衡量控制方式的原则：**
   1. 数据传送速度足够高，能满足用户的需要但是又不丢失数据。
   2. 系统开销小所需的处理控制程序少。
   3. 能充分发挥硬件资源的能力，使IO设备尽量的忙，而CPU等待时间少。

## 9.2.1程序直接控制方式

1. **程序直接控制方式的定义**：

用户进程来直接控制内存或CPU和外围设别之间的信息传递。

1. **硬件特性：**
   1. 设备拥有控制状态寄存器

设备标志触发器，记录了设备的完成状态。

1. 数据缓冲寄存器

输入设别每进行一次操作，首先把数据放入数据缓冲寄存器，然后CPU再把其中的数据取走；输出数据时，CPU先把数据输出到数据缓冲寄存器，输出设备在进行将其取走。

1. **控制流程：**
   1. 请求数据操作的进程通过CPU发出启动设备准备数据的启动指令Stat，用户进程进入等待状态
   2. CPU不停的用一条测试指令检查设备的控制状态寄存器。
   3. 设备的控制状态寄存器为完成状态时（即标记触发器为Done），开始向CPU或内存传递数据。

CPU 
Start 
Done? 
ne 

1. **优点：**
   1. 控制简单，不需要很多硬件支持。
2. **缺点：**
   1. CPU 和外围设备只能进行串行工作。
   2. CPU在一段时间内只能和一台设备交换数据。
   3. 无法发现由于设备或其他硬件产生的错误。只适用于CPU较慢，外围设备较少的系统。

## 9.2.2中断方式

1. **什么是中断方式？**

为了减少程序直接控制方式中CPU等待时间以及提高系统的并行工作程度，中断方式被用来控制外围设备和内存与CPU之间的数据传送。这种方式要求CPU与设备之间有相应的中断请求线和设备控制器的控制状态寄存器有相应的中断允许位。

1. **硬件特性：**
   1. 控制状态寄存器，且有相应的中断允许位
   2. 数据缓冲寄存器
2. **控制流程：**
   1. 需要数据的进程通过CPU向设备发送Start命令，打开设备中断允许位。
   2. 当前进程放弃处理机，进入等待状态，系统调度其他程序。
   3. 设备输入完成时，I/O控制器通过中断请求，向CPU发出中断信号，CPU中断当前进程，转向中断处理程序对数据传送工作进行相应处理。
   4. 当调度到需要该数据的进程时，进程从内存中取出

设 备 
兰 〔 PLI 发 莱 的 s 惬 rt 0 
爸 据 荇 总 入 彘 寄 存 器 
萍 寄 存 器 苈 ？ 
控 老 器 发 中 断 信 号 
CPU 
苘 谩 S rt 命 
中 断 0 到 1 
满 程 满 他 程 
其 它 程 行 
收 到 中 断 号 ？ 
中 断 处 
中 断 程 继 续 斑 行 

1. 传送结构图

《 / O 设 备 
《 / O 设 备 
丨 / O 控 制 器 
/ 数 据 总 线 
丨 / O 控 制 器 
启 动 位 
地 址 总 线 
制 总 线 
Start 
中 断 位 
内 存 
CPU 
控 制 状 态 寄 存 器 
数 据 缓 冲 寄 存 器 

1. **优点：**
   1. 设备和CPU做到并行。外部设备进行IO操作的同时，CPU已经被调度程序分配给其他进程
   2. 设备与设备并行。某个输入输出设备在工作时，CPU在另外的进程执行上下文时，也可以启动不同设备的启动指令。
2. **缺点：**
   1. 中断消耗大量CPU时间。一次数据传送过程中，发生的中断次数较多。
   2. 中断次数急剧增加造成CPU无法响应中断和出现数据丢失。
   3. 外围设备速率较高的情况下可能造成缓冲寄存器的数据由于CPU来不及取走二丢失。

## 9.2.3DMA方式

1. **主要特点：**

在外围设备和内存之间开辟直接的数据交换通路，DMA代替CPU控制内存和设备之间进行成批的数据交换。每一个设备都需要一个DMA控制器。

1. **硬件特性：**

控制状态寄存器

数据缓冲寄存器

传送字节寄存器

内存地址寄存器

1. **传送结构**

CPU 
》 旧 设 蚤 
DMAGIJ# 
中 断 0 
崆 引 状 态 寄 存 器 
数 茫 冲 寄 存 器 
传 送 字 节 数 计 数 器 
丙 存 地 址 寄 存 器 

1. **控制流程：**
   1. 进程需要输入数据时，CPU把需要存放数据的内存地址，字节数发送给DMA控制器，设置允许中断。
   2. 请求数据的进程阻塞，系统调度其他程序。
   3. 输入设备不断地将数据写入内存，知道所要求字节全部传送完毕。
   4. DMA发出中断信号，CPU接收到中断信号后转中断处理程序进行善后处理。
   5. 中断处理结束时，CPU返回被中断进程处执行或被调度到新的进程上下文环境中执行。

设 备 
[ 〕 MA 崆 器 壹 
S 地 旨 0 
启 动 蚤 爸 据 
据 如 据 彘 寄 
河 寄 存 器 中 的 的 
入 的 存 
《 送 字 节 数 
计 数 器 与 的 存 地 
寄 存 器 的 
发 送 中 断 信 号 
僖 止 》 / O 
CPU 
苘 [ 〕 MA 崆 器 发 
送 st t 唯 0 
发 送 的 存 地 址 ， 字 
节 数 到 [ 〕 MA ， 荇 
允 许 中 断 
兰 前 进 程 等 待 满 
度 其 它 程 
收 到 中 断 号 
转 中 断 处 邏 
满 中 断 程 

1. **优点：**
   1. 大大减小CPU中断次数
   2. 解决了并行操作设备过多时数据丢失问题
2. **缺点：**
   1. 对外围设备的管理和某些操作仍由CPU控制
   2. 多个DMA控制器同时使用会引起内存地址冲突，使管理过程变复杂
3. **DMA方式和中断的区别**

1.中断方式是程序切换，需要保护和恢复现场；DMA除了开始和结尾，不占用CPU资源

2.中断的发生时间限制的是指令执行完毕之后；而DMA则更小，可以发生在每个机器周期结束时。

3.中断传送过程需要CPU干预；DMA则不需要。

4.DMA的请求优先级高于中断

5.中断可以处理异常；DMA只能局限于完成传送数据块的I/O操作。

## 9.2.4通道控制方式

1. **主要特点：**

与DMA相似，实现设备和内存直接交换数据一个通道可以控制多台设备。

1. **定义：**

通道是一个独立于CPU的专管输入输出控制的处理机，控制设备与内存直接进行数据交换。有自己的通道指令，这些指令由CPU启动，并在操作结束时向CPU发出中断信号。

1. **硬件特性：**

通道可以分时执行多个通道指令程序

通道有三种类型：

1. 字节多路通道：以字节为单位传送数据，用来连接低速设备。
2. 数组多路通道：以块为单位传送数据，用来连接中速设备。
3. 选择通道：前两者可以分时执行多个通道指令程序，选择通道只能同时控制一台设备进行I/O操作。用于连接高速设备。
4. **通道控制方式与DMA控制方式的区别**

1. 在DMA方式中，数据的传送方向，存放数据的内存地址以及传送的数据块长度等都由CPU控制，而在通道方式中，这些都由专管输入输出的硬件---通道来控制。
2. DMA方式时，每台设备至少一个DMA控制器相比，通道控制方式可以做到一个通道控制多台设备与内存之间进行数据交换。
3. 在通道控制方式中，I/O控制器中没有传送字节数计数器和内存地址寄存器，但是多了通道设备控制器和指令执行机构。在通道控制方式下，CPU只需要发出启动指令和指出通道相应的操作和I/O设备，该指令就可以启动通道并使该通道内存取出执行。
4. **通道类型：**

按照信息交换方式不同，一个系统中可设立三种类型的通道，即字节多路通道，数组多路通道，选择通道。

1. 字节多路通道：以字节为单位传送数据，他主要用来连接大量的低速设备，如终端。
2. 数组多路通道：以块为单位传送数据，他具有传送速率高和能分时操作不同设备等优点。数组多路通道主要用来连接中速块设备。
3. 选择通道：选择通道不支持分时操作不同的设备，不过他的传送速率是最高的，常应用于高速的设备。

》 / O 控 制 器 
》 / O 控 制 器 
》 ／ O 扌 空 制 器 
CPU 
字 节 多 路 通 道 
磁 带 机 
磁 带 机 
字 节 多 路 通 道 
》 ／ O 扌 空 制 器 
》 ／ O 扌 空 制 器 
内 存 
字 节 多 路 通 道 
》 ／ O 扌 空 制 器 
》 / O 控 制 器 
磁 盘 机 
磁 盘 机 

1. **控制流程：**
   1. 进程请求输入数据时，CPU发Start指令，指明I/O操作，设备号和对应通道。
   2. 通道接收到指令后，读出存放在内存中的指令，设置对应设备I/O控制器的状态。
   3. 设备把数据送往内存中的指定区域。
   4. 数据传送结束时I/O控制器向CPU发送中断请求。
   5. 中断结束后CPU返回被中断进程继续执行。
2. **通道控制方式的优缺点**

优点：

通道控制方式解决了I/O操作的独立性和各部件工作的并行性。把CPU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，各通道上的外设也能实现并行操作，从而可提高整个系统的效率。

缺点：

由于需要更多硬件（通道处理器），因此其成本较高。通道控制方式通常应用于大型数据交互的场合。

# 9.3中断技术

## 9.3.1中断的基本概念

1. **中断：**

中断是指计算机在执行程序期间，系统内发生任何非寻常的或者非预期的急需处理事件，使得CPU暂时中断当前正在执行的程序而转去执行相应的事件处理程序，待处理完毕后又返回原来被中断处继续执行或调度新的进程执行的过程。

1. **中断源：**

引起中断发生的事件称为中断源。

1. **中断请求：**

中断源向CPU发出请求中断处理信号称为中断请求。

1. **中断响应：**

CPU接受中断请求后转相应的中断处理程序称为中断响应。

1. **禁止中断：**

CPU内部处理机状态字（PSW）的中断允许位已被清除，从而不允许响应中断。

禁止中断也称关中断，相反称为开中断。

1. **中断屏蔽：**

在中断请求产生之后，系统用软件方式有选择地封锁部分中断而允许其余部分的中断仍能得到响应。

有些中断不能屏蔽甚至不能禁止，这些中断具有最高优先级，无论CPU是否禁止中断，这些中断请求必须得到CPU的立即响应。

## 9.3.2中断的分类与优先级

1. 外中断：外中断是指来自处理机和内存外部的中断。

包括I/O设备发出的I/O中断，外部信号中断（Esc）,定时器引起的中断，调试程序中断等。

1. 内中断和陷阱：内中断是指来自处理机和内存内部的中断。内中断一般称为陷阱。

包括程序运算引起的错误，入地址非法，校验错误，页面失效，非法指令等。

1. 中断优先级：

为了禁止中断和屏蔽中断，CPU的处理机状态字（PSW）中也设置响应的优先级，如果中断源的优先级高于PSW的优先级，则CPU响应中断，反之屏蔽该中断请求。

1. 中断和陷阱的区别：
   1. 陷阱通常由处理机正在执行的指令引起，而中断由与现行指令无关的中断引起。
   2. 陷阱处理程序提供的服务为当前进程所用，而中断处理程序提供的服务则不是为了当前进程。
   3. CPU在执行完一条指令后，下一条指令开始响应之前的中断。而一条指令执行中也可以响应陷阱。
   4. 中断处理程序在系统上下文执行，陷阱处理程序在各自进程上下文中执行。

## 9.3.3软中断

1. 软中断：通信进程之间用来模拟硬中断的一种信号通讯方式。
2. 与硬中断的相同点：

中断源发出中断请求或软中断信号后，CPU或接受进程在适当的时机自动进行中断处理或软中断信号所对应的功能。

## 9.3.4中断处理过程

1. CPU 检查响应中断的条件是否满足，不满足则中断无法进行。
2. 如果CPU 响应中断，则CPU关中断。
3. 保存被中断进程现场。
4. 饭分析中断原因，调用中断处理自程序。
5. 执行中断处理自程序。
6. 退出中断，恢复被中断进程的现场或者调度新进程占据处理机。
7. 开中断，CPU继续执行。

中 断 源 
所 0 应 中 断 条 僻 ？ 
不 应 
关 中 断 
保 满 中 断 程 现 肠 
分 析 中 断 原 因 
甲 中 断 茎 过 
塗 行 中 断 处 邏 程 
土 中 断 ， 0 中 断 
程 荭 者 新 程 
开 中 断 

# 9.4缓冲技术

## 9.4.1缓冲的引入

1. 引入缓存的目的：

为了匹配外设与CPU之间的速度，减少中断次数和CPU中断处理时间，解决DMA和通道方式的瓶颈问题。

1. 使用缓冲技术的原因：
   1. 改善CPU与I/O设备间速度不匹配的矛盾
   2. 可以减少对 CPU的中断频率，放宽对中断响应时间的限制
   3. 提高 CPU和 I/O设备之间的并行性
2. 实现方式分类：
   1. 采用专用硬件缓冲器，例如I/O控制器中的数据缓冲寄存器。
   2. 在内存中划出一个具有n个单元的专用缓冲区。内存缓冲又称作软件缓冲。

## 9.4.2缓冲的种类

1. **单缓冲**

在设备和处理机之间设置一个缓冲器。不允许多个进程同时对一个缓冲区操作，设备和设备不能通过单缓冲达到并行。

《 / 0 设 备 
缓 冲 区 
内 存 

1. **双缓冲**

双缓冲是在设备和处理机之间设置两个缓冲区，当一个缓冲区中的数据尚未被处理时可使用另外一个缓冲区存放从设备读入的数据，以此来进一步提高CPU和外设的并行程度。

只是设备和设备，设备和CPU并行的简单模型，不能用于实际并行操作。

A 缓 冲 区 
用 户 洪 程 
作 系 
B 黑 冲 区 

1. **多缓冲**

多个缓冲区连接起来组成两部分，一部分专门用于输入，另一部分专门用于数据输出缓冲结构。

1. **缓冲池**

多个缓冲区连接起来统一管理，既可以用于输入又可以用于输出。

## 9.4.3缓冲池管理

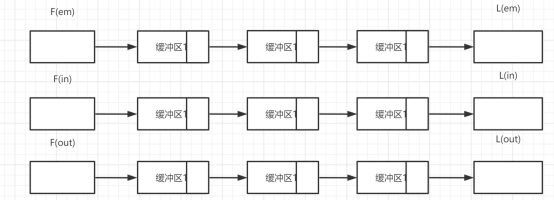
1. **缓冲池的结构：**

缓冲池由多个缓冲区组成，缓冲区由两部分组成：

1. 缓冲首部：表示该缓冲区和用于管理缓冲区。
2. 缓冲体：存放数据

设 备 号 
数 据 块 号 
缓 冲 区 
缓 冲 区 
缓 冲 区 
缓 冲 器 号 
缓 冲 体 
互 斥 标 识 号 
连 接 扌 旨 针 
缓 冲 区 
缓 冲 池 

1. **缓冲区按三种使用情况分为三个队列：**
   1. 空白缓冲队列，队首指针F(em)，队尾L(em)
   2. 装满输入数据的输入缓冲队列。队首指针F(in)，队尾L(in)
   3. 装满输出数据的输出缓冲队列。队首指针F(out)，队尾L(out)



1. **四种工作缓冲区：**
   1. 用于收容设备输入数据的 收容输入缓冲区 hin。
   2. 用于提取设备输入数据的 提取输入缓冲区 sin。
   3. 用于收容CPU输入数据的 收容输出缓冲区 hout。
   4. 用于体悟CPU输入数据的 提取输出缓冲区 sout。

囗 目 
囗 囗 
CPU 

1. **缓冲池管理的操作**
   1. 从三种缓冲区队列中按一定的选取规则取出一个缓冲区的过程take\_buf(type);
   2. 把缓冲区按一定的选取规则插入相应的缓冲区队列过程add\_buf(type,number);
   3. 供进程申请缓冲区用的过程get\_buf(type,number);
   4. 供进程将缓冲区放入相应缓冲区队列的过程put\_buf(type,work\_buf);

注意：

参数type表示缓冲区队列类型

Number为缓冲区号

Work\_buf则表示工作缓冲区类型

1. **缓冲池工作过程**
   1. 输入收容过程：
      1. 首先输入进程调用get\_buf(em,number)过程从空白缓冲区队列中取出一个缓冲号为number的空白缓冲区，将其作为收容输入缓冲区hin。
      2. 当收容输入缓冲区满了之后，系统调用put\_buf(in,hin)将该缓冲区插入输入缓冲区队列in中。
   2. 输出提取过程：
      1. get\_buf(in,number)从输入缓冲区队列中取出一个缓冲号为number的装满输入数据的缓冲区sin。
      2. put\_buf(em,sin)将该缓冲区释放和插入em空白缓冲区队列中

入 进 埕 
匾 0 用 
get buf(em.number) 
尷 令 窒 彐 地 冲 区 认 
列 中 窒 三 冲 区 
将 窒 E 黷 ； 中 区 作 为 收 吝 输 
入 地 冲 区 hin 
通 过 用 put buf(in,hin) 
将 过 入 数 的 地 冲 区 
入 蝓 人 冲 区 从 列 in 
画 过 用 
get but(in,number) 
后 令 从 蝓 入 ； 中 区 鯡 冽 
出 蝓 入 數 的 途 冲 
将 此 冲 区 作 为 取 輸 
入 冲 区 
悍 完 的 过 
b 可 m 匈 n 》 恬 畲 
放 区 压 入 空 
鬚 冲 区 鯡 冽 

## 9.4.3 spooling技术

1. **spooling技术的定义：**

Spooling实际上是用一类物理设备模拟另一类物理设备的技术，使独占使用的设备变成多台虚拟设备的一种技术，也是一种速度匹配技术。

1. **Spooling系统的组成：**
   1. 预输入程序
   2. 缓输出程序
   3. 井管理程序
   4. 输出和输入井

作 业 2 
作 业 1 
作 业 结 果 
作 业 结 果 
作 业 结 果 
输 入 设 备 
输 出 设 备 
预 输 入 程 序 
作 业 调 度 程 序 
当 前 运 行 的 作 业 
# 管 理 程 序 
缓 输 出 进 程 
输 入 共 
作 业 1 信 息 
作 业 n 信 息 
输 出 共 
作 业 1 结 果 
作 业 n 结 果 

1. **Spooling作业表**

# Spooling系统中拥有一张作业表用来记录系统的所有作业的作业名，状态和预输入表位置等信息。

1. **输入井中作业的四个状态：**
   1. 提交状态。正从输入设备预输入。
   2. 后备状态。预输入结束但未被执行。
   3. 执行状态。
   4. 完成状态。

# 9.5设备分配

进程必须向设备管理程序提出资源请求，然后由设备分配程序根据响应得分配算法为进程分配资源。得不到资源得进程会进入等待，直到资源被释放。

## 9.5.1设备分配用数据结构

1. 设备控制表 DCT
   1. 设备控制表反应设备得特性，设备和I/O控制器的连接情况。
   2. 每个设备都有一个，在该设备和系统连接时创建。
   3. 包含内容：
      1. 设备标识符，用来区分设备。
      2. 设备类型，终端设备，块设备或字符设备。
      3. 设备地址或设备号，可以和内存统一编址，也可以单独编址。
      4. 设备状态，忙碌或者空闲。
      5. 等待队列指针，等待使用该设备的进程的队列。
      6. I/O控制器指针
2. 系统设备表 SDT
   1. 记录已被连接到系统中的所有物理设备情况，为每个物理设备设一个表项
   2. 整个系统一张
   3. 表项包含内容：
      1. DCT指针
      2. 正在使用设备的进程标识
      3. 设备类型和设备标识符
3. 控制器表 COCT
   1. 反应I/O控制器的使用状态以及和通道连接的状态
   2. 每个控制器一张
   3. 在DMA方式中，没有该表
4. 通道控制表 CHCT
   1. 只在通道控制方式的系统中存在
   2. 每个通道一张
   3. 包含内容：
      1. 通道标识符
      2. 通道忙/闲标识
      3. 等待队列指针，等待获得该通道的进程构成的队列。
5. 注意：

一个进程只有获得了 通道，控制器，所需设备 之后才具备I/O操作的物理条件。

## 9.5.2设备分配原则

1. 设备分配总原则：

既要充分发挥设备使用率，又要避免由于不合理的分配方法造成进程死锁，还要做到把用户程序和具体物理设备隔离开来，让用户面对逻辑设备而非物理设备。

1. 两种设备分配方式
   1. 静态分配：

在用户作业开始执行之前，系统分配该作业所要求的全部设备，控制器和通道，且一直占用到作业撤销为止。

不会出现死锁，但效率低下，其不符合设备总原则。

1. 动态分配：

在进程执行过程中根据进程需要进行。当进程需要设别时通过系统调用命令向系统提出设备请求，系统按照事先规定的策略分配给进程，用完之后立即释放。

有可能造成死锁。

1. 设备分配策略
   1. 先请求先分配
   2. 优先级高者分配

对于相同优先级，先请求先分配。

# 9.6 I/O控制进程

## 9.6.1 I/O控制引入

从用户进程的输入输出开始，给用户进程分配设备和启动有关设备进行I/O操作，以及I/O操作完成之后响应中断，进行善后处理为止的整个系统称为I/O控制。

## 9.6.2 I/O控制的功能

1. I/O控制模块

外 沿 中 断 0 求 
中 断 原 因 分 桁 
醛 中 断 处 过 
中 断 0 应 
甲 产 河 / 00 求 
] / 00 求 处 邏 
》 / 0 控 老 
河 区 管 
启 动 》 / 0 旨 0 ， 谩 蚤 
驱 动 程 耵 湮 多 程 

## 9.6.3 I/O控制的实现

1. I/O控制过程的三种实现方式：
   1. 作为请求I/O操作的一部分实现。
   2. 作为当前进程的一部分实现。
   3. I/O控制由专门的系统进程——I/O进程完成。
2. I/O进程的三种实现方式：
   1. 每个设备设一个专门的I/O进程，且该设备只能在系统状态下执行。
   2. 整个系统设一个I/O进程，全面负责系统的数据传送工作。
   3. 每个设备设一个专门的I/O进程，但给进程可在用户态和系统态下运行。

# 9.7设备驱动程序

1. 设备驱动程序：

驱动物理设备和DMA控制器或I/O控制器直接进行I/O操作的子程序集合。

设备驱动程序负责设置相应设备有关寄存器的值，启动设备进行I/O操作，指定操作的类型和数据流向。

1. 设备开关表：

设备开关表中给出相应设备的各种操作子程序，也是I/O进程的一个数据结构。

# 9.8 I/O软件

1. 设备管理软件设计水平决定了设备管理效率，，一般分层设计，底层与设备相关，高层的软件向用户提供接口。
2. 设计I/O软件的目的：

设备独立性和统一命名。当设备更新时，没有必要重新编写全部设备驱动程序。

1. I/O设备管理软件一般分为四层：
   1. 中断处理程序
   2. 设备驱动程序
   3. 与设备无关的系统软件
   4. 用户级软件
2. 每层主要功能

《 ／ 0 后 求 
甲 户 过 程 
谩 夤 元 关 软 件 
中 断 处 还 程 
《 / 0 应 笞 
进 行 》 ／ 0 啁 用 ． 式 化 》 / 0 ． Spooling 
扉 ， 俣 护 ． 塞 ， 軋 分 配 
I/O 绻 生 时 0 醖 妪 动 程 
行 ] / 0 在 

# 9.9 磁盘调度

磁盘是可以被多个进程共享的设备，但是系统在每一时刻只允许一个进程启动磁盘进行I/O操作，其余进程等待。

1. 磁盘调度分为两类：
   1. 移臂调度
   2. 旋转调度

注意：

必须先进行移臂调度，才能进行旋转调度。

磁盘调度的目标是使磁盘的平均寻道时间最少。

1. 常用磁盘驱动调度算法：
   1. 先来先服务(FCFS)。公平简单，但是平均寻道时间较长。
   2. 最短寻道时间优先(SSTF)。

选择要访问的磁道与当前磁头所在磁道最近的进程。使得每次寻道时间最短，但不保证平均寻道时间最短。

1. 扫描算法(SCAN)。

不仅考虑了要访问的磁道与当前磁道的距离，更有限考虑磁头当前移动的方向。避免饥饿现象出现。该算法规律与电梯运动规律相似，又叫电梯调度算法。

1. 单向扫描算法(CSCAN)。和扫描算法类似，但磁头只做单向移动。