

操作系统

Xia Tian

Renmin University of China

June 6, 2019



- 5.0 引言
- 5.1 I/O 系统的硬件组成
- 5.2 I/O 控制方式
- 5.3 缓冲管理
- 5.4 设备分配
- 5.5 设备处理
- 5.6 磁盘存储器管理

问题



- 设备管理的主要目标是什么？
- 对应的主要解决途径是什么？



- “设备”泛指计算机系统的外部设备，即除主机以外的其他所有设备。在多道程序设计环境下，计算机系统允许多个用户作业同时在内存，它们的运行势必涉及到 I/O 设备。
- 带来三个问题



- “设备”泛指计算机系统的外部设备，即除主机以外的其他所有设备。在多道程序设计环境下，计算机系统允许多个用户作业同时在内存，它们的运行势必涉及到 I/O 设备。
- 带来三个问题
 - * 对于设备本身，如何有效利用的问题；
 - * 对于设备和 CPU，如何发挥并行工作能力的问题；
 - * 对于设备和用户，有一个如何方便使用的问题。

- 设备无关性
 - * 操作系统必须向设备发送命令，捕捉中断，并处理设备的各种错误，还应提供其他部分使用设备的简单接口，如有可能，这个接口对所有设备都相同，这就是所谓的“与设备无关性”。
- 虚拟设备
 - * 在众多的 I/O 设备中，并不是所有的设备都是可以共享使用的。也就是说，当一个作业使用某种设备时，另一个要使用的作业只能暂时等待，一直到对方使用完毕，它才能去使用，这影响了系统效率的发挥。在设备管理中，仍然可以借助于磁盘，把只能独享的设备变为共享，这就是所谓的“虚拟设备”



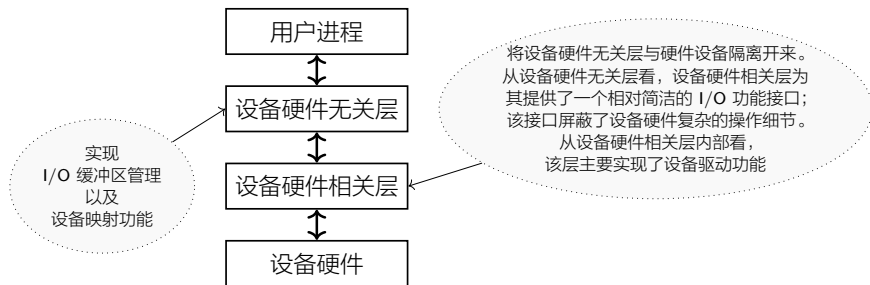
引言：设备管理的功能

- 缓冲区管理
- 设备分配
- 设备处理
- 虚拟设备
- 设备独立性

通用设备管理分层模型



- 现代操作系统都采用分层结构构建设备管理模型，一种常见的设备管理模型如图





5.1 I/O 系统的硬件组成

- 5.1.1 IO 设备
- 5.1.2 设备控制器
- 5.1.3 IO 通道
- 5.1.4 总线系统

5.1.1 IO 设备



1. IO 设备的类型
2. 设备与控制器之间的接口



I/O 设备的类型 I

- (1) 按速度分:
 - * 低速设备: 典型设备有键盘、鼠标器、语音的输入和输出等设备
 - * 中速设备: 典型设备有行式打印机、激光打印机等
 - * 高速设备: 典型设备有磁带机、磁盘机、光盘机等
- (2) 按信息交换的单位分类
 - * 块设备 (Block Device): 用于存储信息, 属于有结构设备。典型的块设备是磁盘。磁盘设备的基本特征是其传输速率较高, 另一特征是寻址, 即对它可随机地读/写任一块; 此外, 磁盘设备的 I/O 常采用 DMA 方式
 - * 字符设备 (Character Device): 用于数据的输入和输出, 属于无结构设备。典型字符设备如交互式终端、打印机等。基本特征是其传输速率较低, 另一特征是寻址; 此外, 常采用中断驱动方式



I/O 设备的类型 II

- (3) 按设备的共享属性分:
 - * 独占: 如临界资源
 - * 共享: 磁盘
 - * 虚拟: 本身为独占, 但将其虚拟为几个逻辑设备。

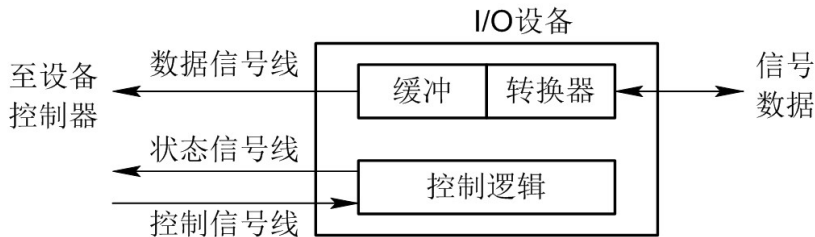


设备与控制器之间的接口

- CPU \leftrightarrow 控制器 \leftrightarrow 设备
- 传统的设备由机械部分和电子部分组成，电子部分在系统的控制下驱动机械部分运转，形成 I/O 操作。
- 电子部分比机械部分速度快，为降低硬件成本，将电子部分从设备中分立出来作为一个独立的部件，即设备控制器。
- 设备不直接与 CPU 通信，而是通过设备控制器通信。



设备与设备控制器间的接口



- 数据信号线
 - * 在设备与设备控制器之间传送数据信号
- 状态信号线
 - * 传送指示设备当前状态的信号
- 控制信号线
 - * 设备控制器向 I/O 设备发送控制信号用



设备与控制器之间的接口传递的信号

1. 数据信号：——双向，有缓存
 2. 控制信号：控制器发给设备；要求其完成相关操作
 3. 状态信号：设备发给控制器，后者“显示”；
- IO 设备控制的变化
 - * 两级 (CPU \leftrightarrow IO 设备)
 - * 三级 (CPU \leftrightarrow 控制器 \leftrightarrow IO 设备)
 - * 四级 (CPU \leftrightarrow IO 通道 \leftrightarrow 控制器 \leftrightarrow IO 设备)

讨论：促使 IO 控制不断发展的推动因素





讨论：促使 IO 控制不断发展的推动因素

- 减少 CPU 对 IO 的干预，充分利用 CPU 的处理能力
- 缓和速度不匹配问题
- 提高 CPU 和 IO 的并行程度，使 CPU 和 IO 尽可能忙碌。



5.1.2 设备控制器

- 分类
 - * 控制块设备的控制器
 - * 控制字符设备的控制器
- 设备控制器的基本功能
- 设备控制器的组成



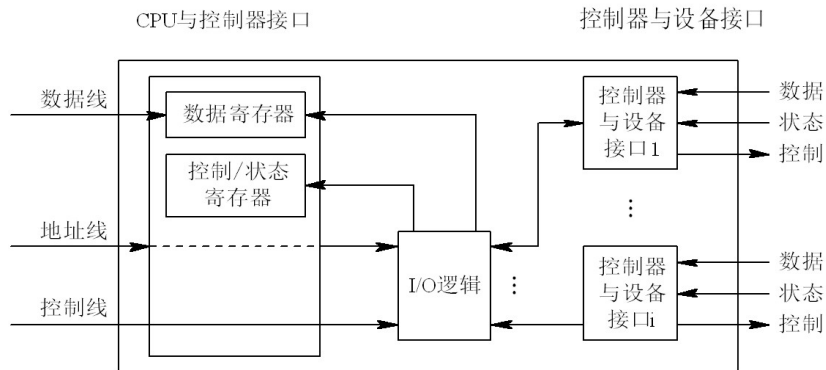
设备控制器的基本功能

- 接收 CPU 命令，控制 I/O 设备工作，解放 CPU.
- 1. 接收和识别命令。
 - * 应有相应的 Register 来存放命令（“命令寄存器”）
- 2. 数据交换: CPU \leftrightarrow 控制器的数据寄存器 \leftrightarrow 设备
- 3. 设备状态的了解和报告
 - * 设备控制器中的“状态寄存器”
- 4. 地址识别
 - * CPU 通过“地址”与设备通信，设备控制器应能识别它所控制的设备地址以及其各寄存器的地址。
- 5. 数据缓冲
 - * 适配高速的 CUP 和内存与低速的外设之间的问题
- 6. 差错控制
 - * 对由 IO 设备传来的数据进行差错检测



- 设备控制器与处理机的接口
 - * 实现 CPU 与控制器通信，共有三类信号线：数据线、地址线和控制线
 - * 两类寄存器：数据寄存器、控制/状态寄存器
- 设备控制器与设备的接口
 - * 一个设备控制器连接一个或多个设备，每个接口中都存有数据、控制、状态三种类型的信号
- I/O 逻辑：在其控制下完成与 CPU、设备的通信。

设备控制器的组成示意图





5.1.3 IO 通道

- 通道
 - * 一种特殊的执行 I/O 指令的处理机，与 CPU 共享内存，可以有自己的总线。
- 引入目的
 - * 解脱 CPU 对 I/O 的组织、管理：建立独立的 I/O 操作，不仅使数据的传送能力独立于 CPU，而且对有关对 I/O 操作的组织、管理及其结束处理也尽量独立，以保证 CPU 有更多的时间去进行数据处理。
- 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。
- I/O 通道是一种特殊的处理机，具有执行 I/O 指令的能力，并通过执行通道 (I/O) 程序来控制 I/O 操作



I/O 通道与一般的处理机的区别

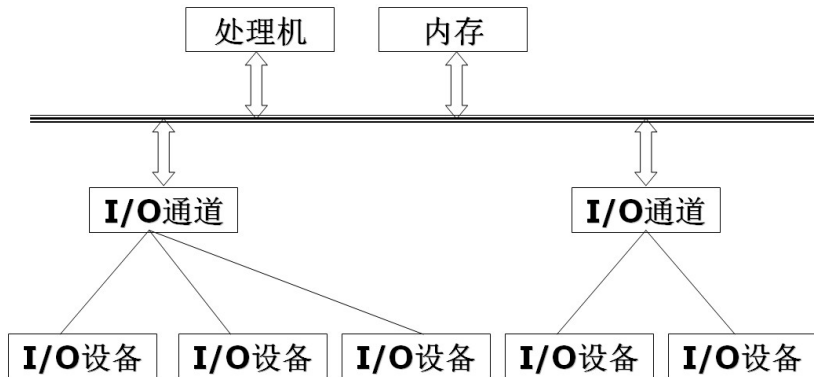
- 指令类型单一
- 通道没有自己的内存，通道与 CPU 共享内存
- 因为简单，所以价格便宜



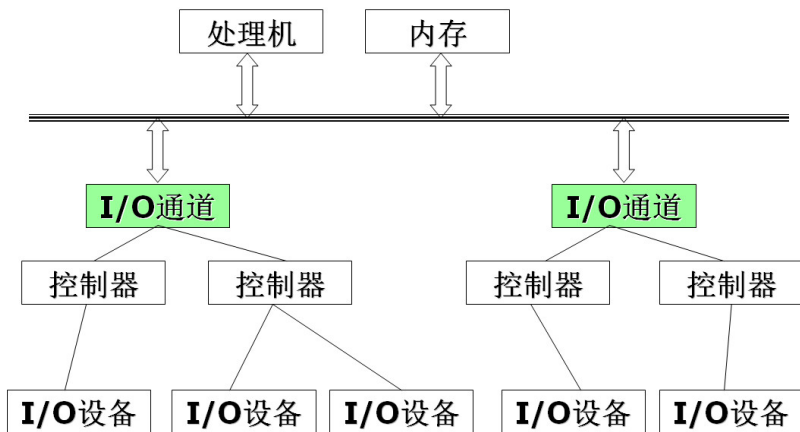
IO 通道的类型

- 1. 字节多路通道：
 - * 各子通道以时间片轮转方式共享通道，适用于低、中速设备。
- 2. 数组选择通道：
 - * 无子通道，仅一主通道，某时间由某设备独占，适于高速设备。
 - * 但通道未共享，利用率低。
- 3. 数组多路通道：
 - * 多子通道不是以时间片方式，而是“按需分配”，综合了前面 2 种通道类型的优点。

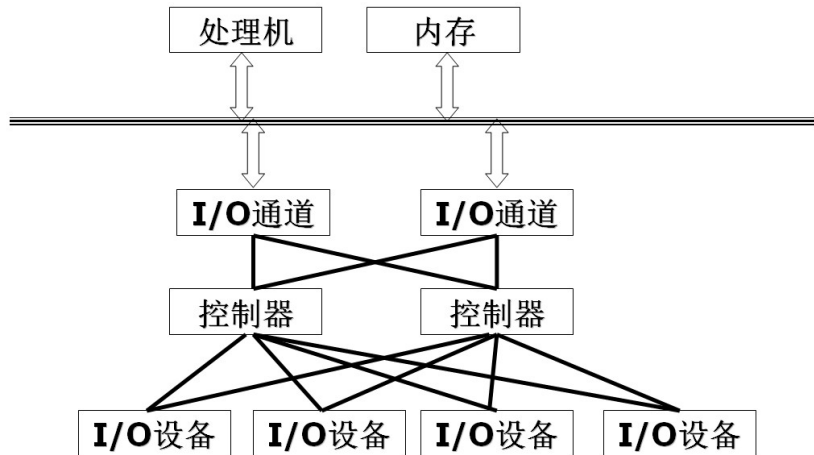
I/O 设备通道连接方式



单通路通道“瓶颈”问题

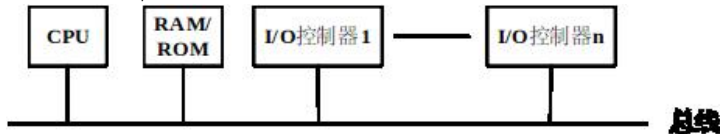


采用复联方式-多通路



5.1.4 总线系统

● 一、微机 I/O 系统



* 处理机与 I/O 设备之间的基本连接都是通过总线实现的。即处理机连接在总线上，与设备无关。设备则根据需要连接在相应的总线上，可多可少，结构和安装均十分灵活

○ 如：磁盘设备，打印设备; 缺点：总线瓶颈，CPU 瓶颈。

* ISA/EISA/Local BUS/VESA/PCI

● 二、主机 I/O 系统（四级结构）

* 计算机—I/O 通道—I/O 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，解决其负担。



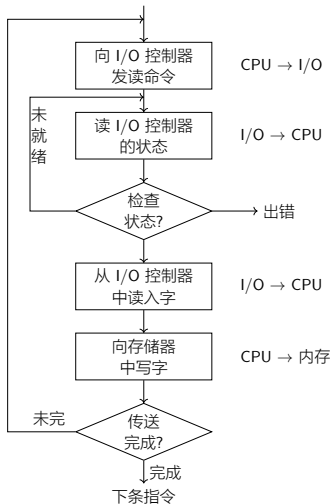
5.2 I/O 控制方式

- 四个发展阶段：
 - * 程序 I/O
 - * 中断 I/O
 - * DMA 控制
 - * 通道控制。
- 趋势：提高并行度。



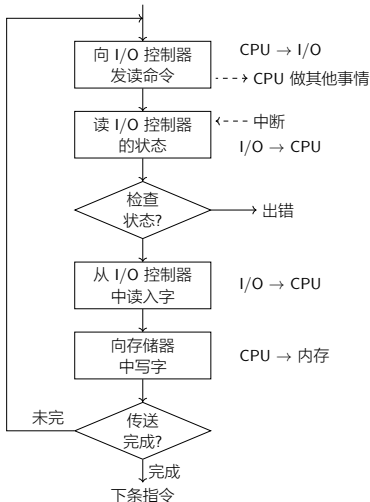
5.2.1 程序 I/O (忙—等待方式)

- 查询方式: CPU 需花代价不断查询 I/O 状态
- CPU 资源浪费极大。
- 例: $99.9\text{ms} + 0.1\text{ms} = 100\text{ms}$
- 99.9 在忙等



5.2.2 中断 I/O

- 向 I/O 发命令 → 返回 → 执行其它任务。
- I/O 中断产生 → CPU 转相应中断处理程序。
- 如：读数据，读完后以中断方式通知 CPU，CPU 完成数据从 I/O → 内存
- 以字（节）为单位进行干预
- CPU、设备并行工作
- 提高了系统的资源利用率和吞吐量



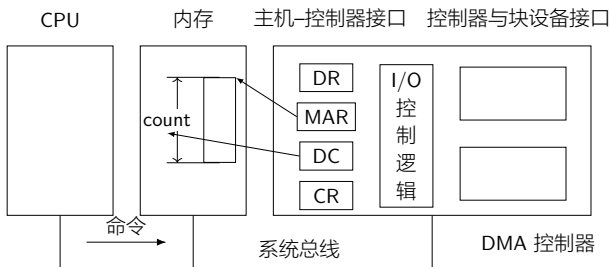


5.2.3 DMA 方式——用于块设备中

- 一、DMA(Direct Memory Access) 控制方式的引入
 - * 中断 I/O，CPU 按字节干预，即每字节传送产生一次中断。
 - * DMA：由 DMA 控制器直接控制总线传递数据块。DMA 控制器完成从 I/O——内存。
- 特点：
 - * 数据传输的基本单位是数据块，即在 CPU 与 I/O 设备之间，每次传送至少一个数据块；
 - * 所传送数据从设备直接送入内存，或者相反；
 - * 仅在传送一个或多个数据块的开始和结束时，才需 CPU 干预，整块数据的传送由控制器控制完成。
- 二、组成
 - * 一组寄存器 + 控制逻辑。
 - * CR (命令/状态) ; DR (数据) ; MAR (内存地址) ; DC (计数)

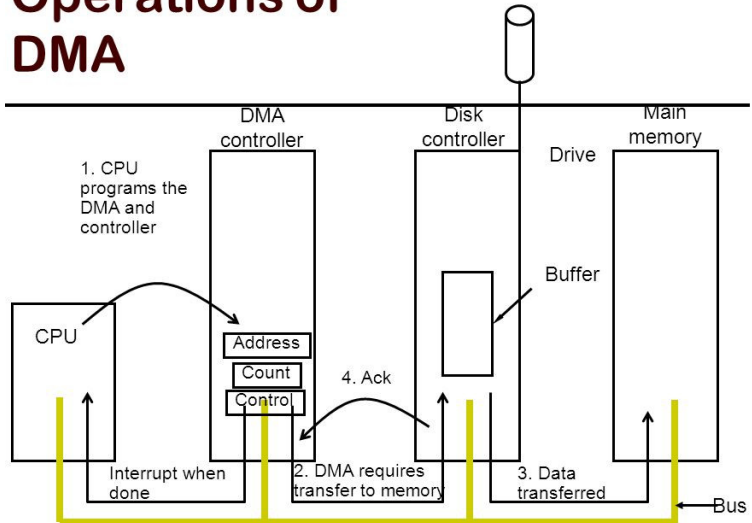


DMA 控制器的组成图

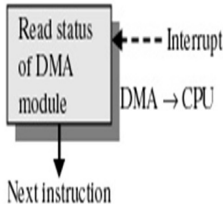
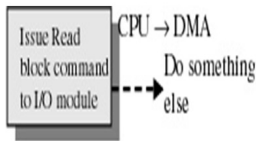
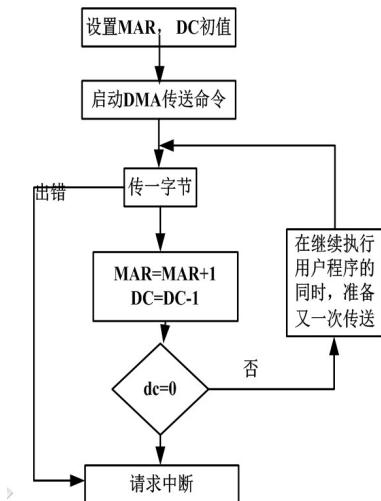


- * 数据寄存器 DR: 用于暂存从设备到内存, 或从内存到设备的数据
- * 内存地址寄存器 MAR: 在输入时, 它存放把数据从设备传送到内存的起始目标地址; 在输出时, 它存放由内存到设备的内存源地址
- * 数据计数器 DC: 存放本次 CPU 要读或写的字 (节) 数
- * 命令/状态寄存器 CR。用于接收从 CPU 发来的 I/O 命令或有关控制信息, 或设备的状态

Operations of DMA



DMA 工作过程



(c) Direct memory access



讨论：中断驱动 IO 和 DMA 的区别

- 中断频率
- 数据的传送方式



5.2.4 I/O 通道控制方式 (1)

- I/O 通道控制方式的引入
 - * DMA 方式：对许多离散块的读取仍需要多次中断。
 - * 对一组数据块的读 (或写) 及有关的管理为单位的干预
 - * 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



5-2-4 I/O 通道控制方式 (2)

- 通道方式：CPU 只需给出
 - * (1) 通道程序首址。
 - * (2) 要访问 I/O 设备后，通道程序就可完成一组块操作



通道程序的指令信息

- (1) 操作码: 指令所执行的操作
- (2) 内存地址: 字符送入内存 (读操作) 或从内存取出 (写操作) 时的内存首址
- (3) 计数: 要读写的字节数
- (4) 通道程序结束位 P: $P=1$ 表示最后一条指令
- (5) 记录结束标志 R, $R=0$, 表示本指令和下一条指令处理的数据同属于一个记录。

通道程序



- (1) 操作码 (2) 内存地址
- (3) 计数 (4) 通道程序结束位 P
- (5) 记录结束标志 R

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

5.3 缓冲管理

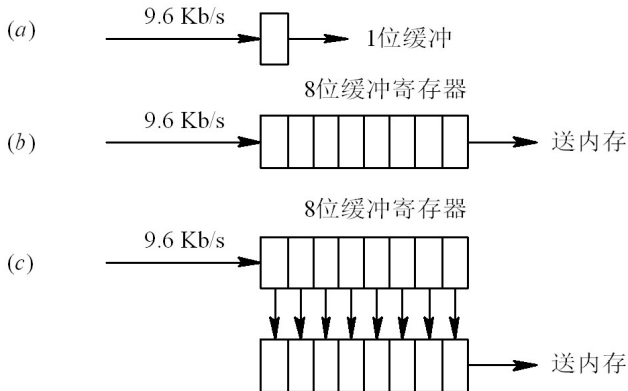




5.3.1 缓冲的引入

- 1. 缓和 CPU 和 I/O 设备间速度不匹配的矛盾。
 - * 如：计算——打印 buffer——打印
- 2. 减少对 CPU 的中断频率
 - * 如：buffer 越大，“buffer 满”信号发生频率越低。
- 3. 提高 CPU 和 I/O 并行性

例子

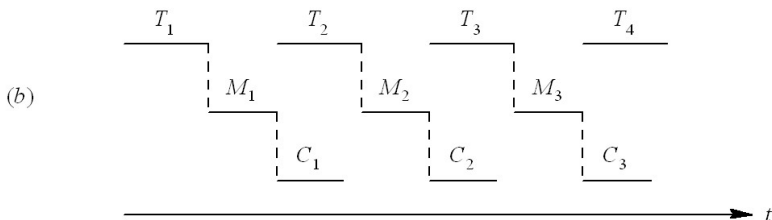
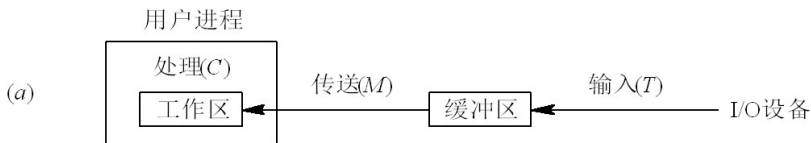


- (a) CPU 中断频率: 9.6Kb/s; CPU 响应时间: 约 100us
- (b) CPU 中断频率: 1.2Kb/s; CPU 响应时间: 约 800us
- (c) CPU 中断频率: 0.6Kb/s; CPU 响应时间: 约 1600us



5.3.2 单缓冲和双缓冲 (1)

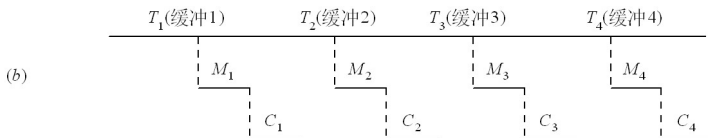
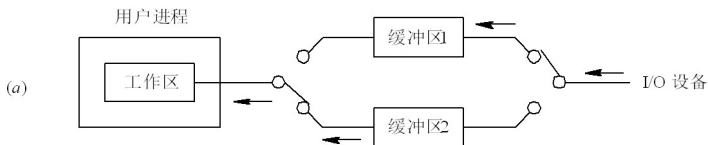
- 单缓冲 (Single Buffer)
- 系统对数据的处理时间: $\text{Max}(C, T) + M$





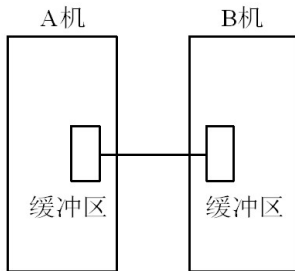
5.3.2 单缓冲和双缓冲 (2)

- 双缓冲 (Double Buffer)
- 系统对数据的处理时间: $Max(C + M, T) \approx Max(C, T)$

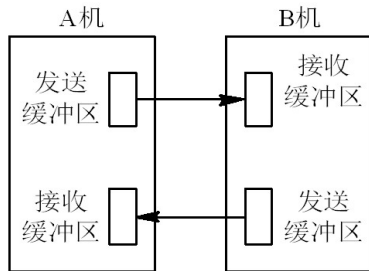


5.3.2 单缓冲和双缓冲 (3)

- 双机通信时缓冲区的设置



(a) 单缓冲



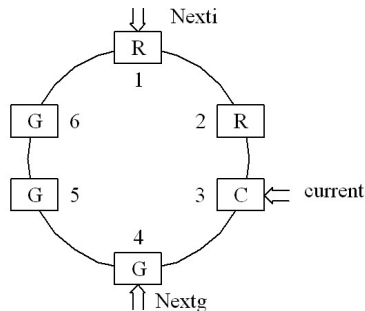
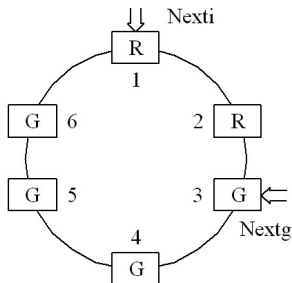
(b) 双缓冲



5.3.3 循环缓冲

- 1. 循环缓冲的组成
 - * (1) 多个缓冲区，三种类型
 - 用于装输入数据的空缓冲区 R
 - 已装满数据的缓冲区 G
 - 计算进程正在使用的现行工作缓冲区 C
 - * (2) 多个指针
 - 指示计算进程下一个可用缓冲区 G 的指针 NextG
 - 指示输入进程下次可用的空缓冲区 R 的指针 Nexti
 - 指示计算进程正在使用的缓冲区 C 的指针 Current

循环缓冲组成示意图



- R: 空缓冲区; G: 已装满数据的缓冲区
- C: 正在使用的现行工作缓冲区
- Nextg: 指示计算进程下次可用的数据缓冲区的指针
- Nexti: 指示输入进程下次可用的空缓冲区的指针
- current: 指示计算进程正在使用的数据缓冲区的指针



循环多缓冲的使用

- nextg: 指示下一个计算进程应取数据的 buf
- nexti: 指示下一个空 buf 供输入进程使用.
- Getbuf:
 - * 计算进程: 取 nextg 对应缓冲区提供使用, 将 Nextg 置为空,
 $\text{Nextg} = (\text{Nextg} + 1) \text{ Mod } N$
 - * 输入进程: 将 Nexti 对应缓冲区提供使用, 将 Nexti 置为满,
 $\text{Nexti} = (\text{Nexti} + 1) \text{ Mod } N$
- Releasebuf:
 - * 输入进程: 当把缓冲区 C 装满后, 则改为 G 缓冲区;
 - * 计算进程: 当把缓冲区 C 中的数据提取完毕后, 则把 C 改为 R;

循环多缓冲的同步问题



- Nexti 追上 Nextg:
 - * 表示输入速度 $>$ 输出速度，全部 buf 满，这时输入进程阻塞，系统受计算能力限制。
- Nextg 追上 Nexti:
 - * 输入速度 $<$ 输出速度，全部 buf 空，这时输出进程阻塞。系统受 I/O 能力限制。



5.3.4 缓冲池

- 缓冲池：系统提供的公用缓冲
 - * 前面的缓冲区仅适用于特定的 IO 进程和计算进程，属于专用缓冲。
 - * 提供可供进程共享的公用缓冲，提高缓冲区利用率

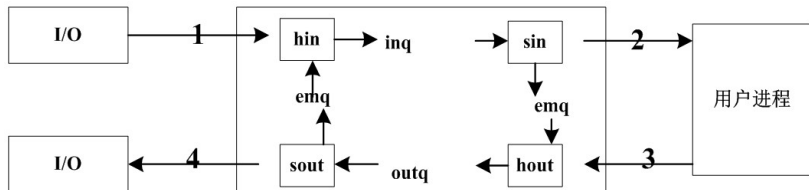


1. 缓冲池的组成

- 一、组成：
 - * 3 个队列：
 - 空缓冲队列 emq
 - 输入队列 inq
 - 输出队列 outq
 - * 四个工作缓冲区：
 - hin: 收容输入数据
 - sin: 提取输入数据
 - hout: 收容输出数据
 - sout: 提取输出数据

2. 缓冲池的 4 种工作方式

- 1. 收容输入;
- 2. 提取输入
- 3. 收容输出;
- 4. 提取输出



工作方式



- 1. `hin=getbuf(emq);`
`putbuf(inq,hin)`
- 2. `sin=getbuf(inq);`
计算;
`putbuf(emq,sin)`
- 3. `hout=getbuf(emq);`
`putbuf(outq, hout);`
- 4. `sout=getbuf(outq);`
输出;
`putbuf(emq,sout)`



3. Getbuf 和 Putbuf 过程

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

- MS(**type**) 为 **type** 类型队列的互斥信号量，保证队列的互斥访问
- RS(**type**) 为 **type** 类型队列的资源信号量，记录资源数量





5.4 设备分配

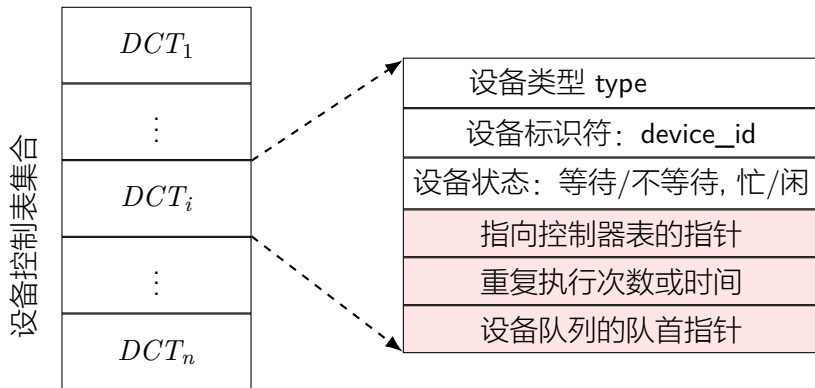
- 多道程序环境下，设备供所有进程共享，为防止进程对系统资源的无序竞争，规定系统设备只能由系统统一分配。设备分配程序按照一定的策略，把设备分配给请求用户。
- 包括：对设备、设备控制器、通道的分配



5.4.1 设备分配的数据结构

- 一、设备控制表 DCT:
 - * 每个设备一张，记录本设备的情况
- 二、控制器控制表 (COCT), 通道表 (CHCT), 系统设备表 (SDT)
 - * SDT: 记录了系统中全部设备及其驱动程序地址等信息。

设备控制表 DCT



- 一个设备一张设备控制表，记录本设备的情况

控制器控制表



- 一个控制器一个 COCT 表，记录本控制器情况

控制器标识符: controller_id
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

控制器表 COCT

通道控制表



- 一个通道一张通道控制表

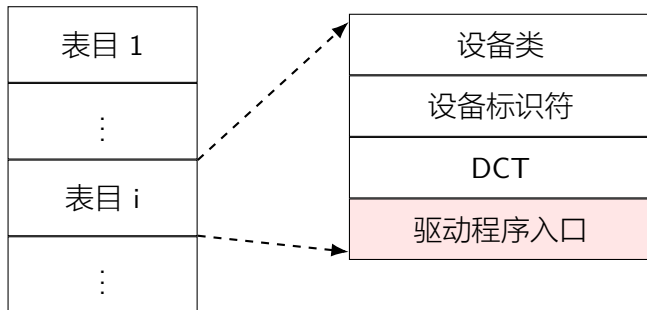
通道标识符: channel_id
通道状态: 忙/闲
与通道连接的控制器表指针
通道队列的队首指针
通道队列的队尾指针

通道表 CHCT

系统设备表



- 记录系统中全部设备的情况



系统设备表 SDT



5.4.2 设备分配应考虑的因素 I

- 一、设备的固有属性：
 - * (1) 独享设备：采用独享分配策略
 - * (2) 共享设备：注意调度
 - * (3) 虚拟设备
- 二、分配算法：
 - * (1) FIFO;
 - * (2) 优先权。
- 三、安全性：
 - * 方式 1：安全分配 (同步)：每进程获得一 I/O 后，即 block，直到其 I/O 完成。
 - 即打破了死锁条件。
 - 缺点：CPU、I/O 对该进程是串行，进程进展缓慢。



5.4.2 设备分配应考虑的因素 II

- * 方式 2: 不安全分配 (异步): 进程在发出 IO 请求后仍继续运行, 可继续请求第二个 IO 需求;
 - 需进行安全性检查, 进程执行效率高。
- 四、设备独立性
 - * 5.4.3 内容



5.4.3 设备独立性

- 1、设备独立性概念
 - * 即设备无关性，指应用程序独立于具体使用的物理设备。
 - 例：打印时的打印机选择
 - * 为实现设备独立性引入逻辑设备和物理设备概念
 - 在应用程序中，使用逻辑设备名称来请求使用某类设备；而系统在实际执行时，还必须使用物理设备名称。因此，系统须具有将逻辑设备名称转换为某物理设备名称的功能，类似于存储器管理中的逻辑地址与物理地址关系。
 - * 逻辑设备表（LUT：Logical Unit Table）：

逻辑设备	物理设备	Driver 入口
------	------	-----------



1、设备独立性的概念-续

- 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。
- 优点：
 - * 设备分配更灵活；
 - 逻辑设备和物理设备间可以是多—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：
 - 某逻辑名可对应这一类设备，提高均衡性与容错性。
 - 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。
 - 不变程序，只需改变 LUT 表的映射关系。
 - 如调式时输出到屏幕，测试成功后输出到打印机



2、设备独立性软件

- 执行所有设备的公有操作
 - * 分配回收：对独立设备的分配与回收
 - * 名字映射：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
 - * 保护：对设备进行保护，禁止用户直接访问设备
 - * 缓冲管理：即对字符设备和块设备的缓冲区进行有效的管理，以提高 I/O 的效率；
 - * 差错控制：处理设备驱动程序无法处理的错误
- 向用户层软件提供统一接口
 - * 无论何种设备，它们向用户所提供的接口应该是相同的，如统一的 Read 和 Write 操作

```
struct general_op{  
    int (*read)(...)  
    int (*write)(...)  
};
```

```
driver1: struct  general_op dev_op={  
    dev1_read,  
    dev1_write  
};
```

```
driver2: struct  general_op dev_op={  
    dev2_read,  
    dev2_write  
};
```

```
Gen_read(fd, ...){  
    dev_op=map(fd);  
    dev_op->read(...);  
}
```



3、逻辑设备名到物理设备名映射的实现

- 1) 逻辑设备表：用于将应用程序中所使用的逻辑设备名映射为物理设备名
- 2) LUT 的设置问题：
 - * (a) 系统中只设置一张 LUT：所有用户之间不允许有相关的逻辑设备名
 - * (b) 一个用户一张 LUT：多用户系统中通常都配置系统设备表，故只需给出系统设备表的指针即可

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)



5.4.4 独占设备的分配程序 I

- 1. 基本的设备分配程序
 - * 1) 分配设备
 - * 2) 分配控制器
 - * 3) 分配通道
 - * 只有在设备、控制器和通道三者都分配成功时，此次设备分配才算成功



5.4.4 独占设备的分配程序 II

- 2. 设备分配程序的改进
 - * 1) 增加设备的独立性：使用逻辑设备名请求 I/O
 - 系统首先从 SDT 中找出第一个该类设备的 DCT，如忙，则找下一个，仅当所有该类设备都忙时，才把进程挂在该类设备的等待队列上。
 - * 2) 考虑多通路情况
 - 设备（控制器）所连接的第一个控制器（通道）忙时，则查看所连接的第二个控制器（通道），只有都忙时，才把进程挂在控制器（通道）的等待队列上



分配程序示例 I

- 进程 n 请求设备:

```
search (sdt, phdevice);
```

```
if not busy (phdevice) then
```

```
begin
```

```
    compute(safe); //对独占设备
```

```
    if safe then
```

```
        alloc (n, phdevice);
```

```
    else begin
```

```
        insert (DL(phdevice), n); //将n插入设备等待队列DL上
```

```
        return;
```

```
    end;
```

```
end;
```

```
else begin //设备忙
```

```
    insert (DL (phdevice), n);
```




分配程序示例 II

```
return;
```

```
end;
```

```
//device分配成功
```

```
controllerid=controllerid (COCT ptr(dct));
```

```
if not busy (COCT (controllerid)) then
```

```
    alloc (n, controllerid);
```

```
else begin
```

```
    insert (col, n);
```

```
    return;
```

```
end;
```

```
//控制器分配成功
```

```
channeled=channeled(chatptr (controllerid));
```



分配程序示例 III

```
if not busy (chct (channelid)) then  
    allocation (n, channelid);  
else begin  
    insert (chl, n)  
    return;  
end;
```

- 优化:
 - * 1) 增加设备的独立性
 - * 2) 考虑多通路情况



5.4.5 SPOOLing 技术

虚拟性：单一物理 CPU 被虚拟成多台逻辑 CPU，同样可以想法把单一物理设备虚拟为多台逻辑设备

- 1. SPOOLing 的基本概念
- 2. SPOOLing 组成
- 3. 共享打印机
- 4. SPOOLing 的特点



1. SPOOLing 的基本概念

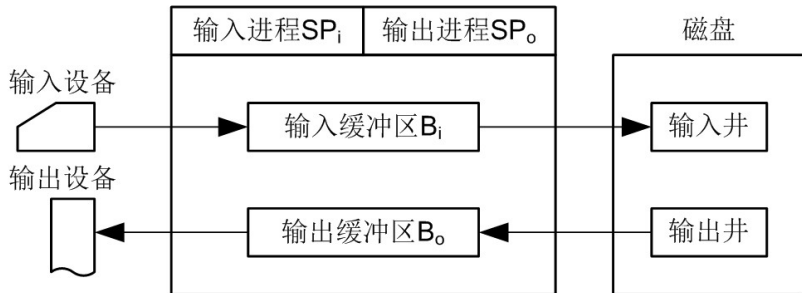
- 把在联机情况下实现的同时外围操作称为 SPOOLing (Simultaneous Peripheral Operating On-Line)，或称为假脱机操作。
- 作用：
 - * 通过缓冲方式，将独占设备改造为共享设备



2. SPOOLing 组成

- 1. 输入井和输出井:
 - * 在磁盘上开辟的 2 个大存储空间, 模拟输入和输出设备。
- 2. 输入 buf 和输出 buf (内存中)
 - * (1) 输入设备 → 输入 buf → 输入井 → 用户区
 - * (2) 用户区 → 输出井 → 输出 buf → 设备
- 3. 输入 SP_i 和输出 SP_o 进程。
 - * 分别控制 (1), (2) 的动作。
 - * SP_i 相当于脱机输入控制器。
 - * SP_o 相当于脱机输出控制器。

SPOOLing 系统的组成示意图





3. 共享打印机

- 共享打印机技术已被广泛地用于多用户系统和局域网络中。当用户进程请求打印输出时，SPOOLing 系统同意为它打印输出，但并不真正立即把打印机分配给该用户进程，而只为它做两件事：
 - * (1) 由输出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) 输出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



4. SPOOLing 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/出井
- 3. 实现了虚拟设备功能
 - * 对于独占设备，用户都感到是自己独占了该设备。

讨论：SPOOLing 和缓冲的区别





讨论：SPOOLing 和缓冲的区别

- 目的不同：
 - * SPOOLing 解决的是独占 I/O 设备如何共享使用的问题
 - * 缓冲解决的是 I/O 设备和 CPU 的速度不匹配的问题。
- 数据存放的位置
 - * SPOOLing 在磁盘上开辟输入井和输出井
 - * 缓冲在内存中设置缓冲区
- 管理
 - * SPOOLing 由井管理程序负责
 - * 缓冲由缓冲区管理软件负责



5.5 设备处理

- 设备处理程序即是设备驱动程序
 - * 设备处理程序用于实现 I/O 进程与设备控制器之间的通信，由于常以进程形式存在，故可简称为设备驱动进程
 - * 负责接收上层软件发来的抽象要求，再转换为具体要求后，发送给设备控制器，启动设备执行；或者将设备控制器发来的信号传送给上层软件。



5.5.1 设备驱动程序的功能和特点

- 1. 设备驱动程序的功能：
 - * (1) 接收由 I/O 进程发来的命令和参数，并将命令中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，了解 I/O 设备状态，传递有关参数，设置设备的工作方式
 - * (3) 发出 I/O 命令，如果设备空闲，便立即启动 I/O 设备去完成指定的 I/O 操作；如果设备处于忙碌状态，则将请求者的请求块挂在设备队列上等待。
 - * (4) 及时响应由控制器或通道发来的中断请求，并根据其中断类型调用相应的中断处理程序进行处理。
 - * (5) 对于设置有通道的计算机系统，驱动程序还应能够根据用户的 I/O 请求，自动地构成通道程序。



5.5.1 设备驱动程序的功能和特点-II

- 2. 设备处理的三类方式
 - * (1) 为每一类设备设置一个进程，专门用于执行这类设备的 I/O 操作。
 - * (2) 在整个系统中设置一个 I/O 进程，专门用于执行系统中所有各类设备的 I/O 操作。或者设置一个输入进程和一个输出进程，处理系统中所有设备的输入和输出操作。
 - * (3) 不设置专门的设备处理进程，而只为各类设备设置相应的设备处理程序 (模块)，供用户进程或系统进程调用。



5.5.1 设备驱动程序的功能和特点-III

- 3. 设备驱动程序的特点：属于低级的系统例程
 - * (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。如中断驱动或 DMA 方式
 - * (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。许多驱动固化在 ROM 中



5.5.2 设备驱动程序处理过程

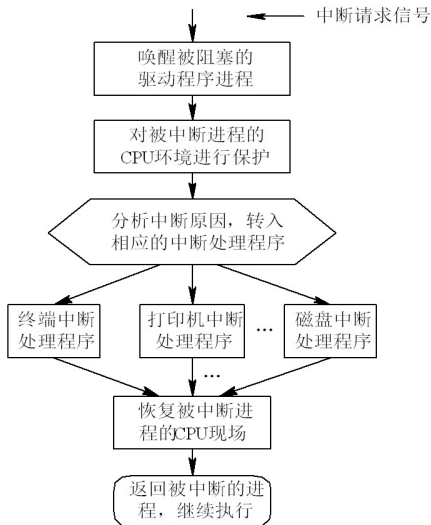
- 包括
 - * 启动过程
 - * 中断处理过程
- 启动过程
 - * (1) 将抽象要求转化为具体要求，如抽象的盘块号转换为盘面、磁道及扇区号。
 - * (2) 检查 I/O 请求合法性，如试图从打印机输入数据
 - * (3) 读出和检查设备状态，如只能在就绪时写入数据
 - * (4) 传送必要的参数，如块设备需要的字节数参数
 - * (5) 设置工作方式
 - * (6) 启动 I/O 设备，向控制器中的命令寄存器发控制命令



5.5.3 中断处理程序

- 流程
 - * 设备启动 → I/O 完成 → 发送中断 → CPU 调用中断处理过程
- 中断处理过程
 - * (1) 唤醒被阻塞的驱动程序进程
 - * (2) 保护被中断进程环境
 - * (3) 转入相应的设备处理程序
 - * (4) 中断处理
 - * (5) 恢复被中断进程的现场

中断处理流程







5.6 磁盘存储器管理

- 磁盘存储容量巨大，速度快，支持随机存取，因此现代计算机都配置了磁盘存储器。改善磁盘性能，是现代操作系统的重要任务之一。
- 日常问题
 - * 磁盘擦写次数
 - *



5.6 磁盘存储器管理

- 5.6.0 硬盘的发展历史
- 5.6.1 磁盘性能简述
- 5.6.2 磁盘调度
- 5.6.3 磁盘高速缓存
- 5.6.4 提高磁盘 I/O 速度的其他方法
- 5.6.5 独立磁盘冗余阵列

硬盘的发展历史 I





硬盘的发展历史 II

1956 年, IBM 发明了第一块硬盘 RAMAC 350 (Random Access Method of Accounting and Control), 支持随机读取, 硬盘比冰箱大, 容量是 5MB



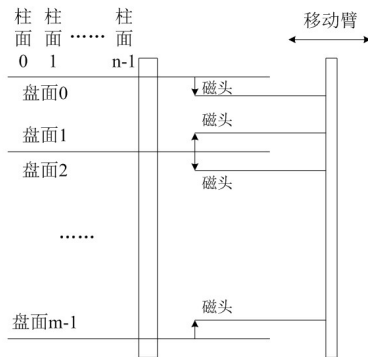
硬盘的发展历史 III

数据是新时代的石油。今天被记录下来的数据容量每年增长 30% 到 40%，但现代硬盘的容量增长速度不到这个增速的一半。幸运的是，绝大部分信息并不需要即时存取。对于此类信息磁带是完美的解决方案。是的，世界大部分的数据仍然保存在磁带上，包括基础科学如粒子物理学和射电天文学，文化遗产和国家档案，电影，金融、保险和石油勘探等。最近微软的 Azure Archive Storage 还使用了 IBM 的磁带存储设备。磁带的一大优势是廉价，而且其容量仍然在不断增长，平均每年增长 33%，意味着每两年或三年其容量将会翻一番。磁带可能属于最后一批仍然遵循摩尔定律的信息技术。

<https://www.solidot.org/story?sid=57763>



5.6.1 磁盘性能简述

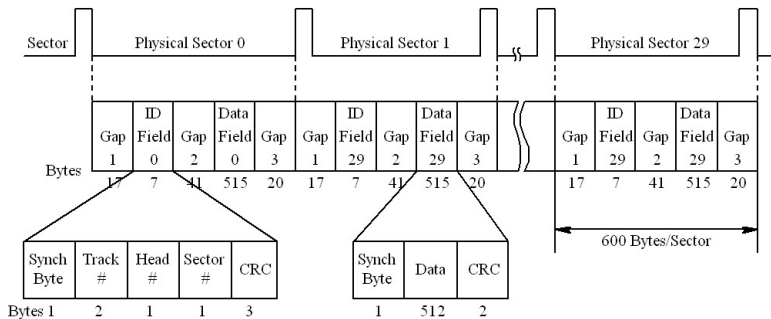


1. 物理结构

- * 包括一或多个盘片，每个盘片分两面
- * 每个盘面分若干磁道，各盘面上序号相同的磁道构成一个柱面



2. 磁盘格式化



格式化的处理



- 格式化就是把一张空白的盘划分成一个个小的区域，并编号，供计算机储存，读取数据。没有这个工作的话，计算机就不知道在哪写，从哪读。



3. 磁盘的类型

- 1) 固定头磁盘
 - * 每条磁道上都有一个读写磁头，实现并行读/写，有效地提高了磁盘的 I/O 速度。主要用于大容量磁盘上
- 2) 移动头磁盘
 - * 每个盘面配有一个磁头，为访问磁盘上的所有磁道，磁头必须能移动以寻道。
 - * 串行方式读/写，致使其 I/O 速度较慢
 - * 结构简单，广泛应用于中小型磁盘设备中



4. 磁盘访问时间 I

- 寻道时间 + 旋转延迟时间 + 传输时间

- 1) 寻道时间 T_s (seek time)

- * 这是指把磁臂 (磁头) 移动到指定磁道上所经历的时间。该时间是启动磁臂的时间 s 与磁头移动 n 条磁道所花费的时间之和, 即

$$T_s = m \times n + s$$

- * 其中, m 是一常数, 与磁盘驱动器的速度有关, 对一般磁盘, $m=0.2$; 对高速磁盘, $m \leq 0.1m$, 磁臂的启动时间约为 2 ms 。这样, 对一般的温盘, 其寻道时间将随寻道距离的增加而增大, 大体上是 $5\text{--}30 \text{ ms}$ 。



4. 磁盘访问时间 II

- 2) 旋转延迟时间 T_r (rotational delay)
 - * 这是指定扇区移动到磁头下面所经历的时间
 - * 平均旋转延迟时间:

$$E(T_r) = \frac{1}{2r}$$

- * r : 磁盘转速
- * 典型的旋转速度为 5400r/min, 即每转需要 11.1ms, 平均旋转延迟为 5.55ms



4. 磁盘访问时间 III

- 3) 传输时间 T_t (transfer time)

- * 这是指把数据从磁盘读出或向磁盘写入数据所经历的时间。 T_t 的大小与每次所读/写的字节数 b 和旋转速度有关:

$$T_t = \frac{b}{rN}$$

- * 其中, r 为磁盘每秒钟的转数; N 为一条磁道上的字节数, 当一次读/写的字节数相当于半条磁道上的字节数时, T_t 与 T_r 相同

- 由于特定磁盘，只有集中放数据，集中读写（读写字节数 b 大）才能更好提高传输效率，例：
 - * 寻道时间: 20ms
 - * 磁盘通道传输速率: 1MB/s
 - * 转速 $r=3600\text{rpm}$
 - * 每扇区 512 字节
 - * 每磁道 32 扇区
 - * 目标：读 128k 数据



Question

- 每转所需要的时间是多少？
- 计算平均旋转延迟时间？
- 每一条磁道上的字节数是多少？
- 读取每个扇区数据的传输时间为多少？
- 128K 数据需要存放多少个扇区？
- 128K 数据需要存放多少个磁道？



- 每转所需要的时间为: $60 \times 1000 / 3600 = 16.7 \text{ ms}$
- 平均旋转延迟时间为: $60 \times 1000 / 3600 / 2 = 8.3 \text{ ms}$
- 一条磁道上的字节数为: $512 \times 32 = 16384$ 字节, 即 16K
- 读取每个扇区数据的传输时间: $0.5 \text{ K} / 16 \text{ K} \times (\text{每转的时间}) = (1/32) \times (60 \times 1000 / 3600) \text{ ms} = 0.521 \text{ ms}$
- 128K 数据需要存放多少个扇区: $128 / (512 / 1024) = 256$
- 128K 数据需要存放多少个磁道: $256 / 32 = 8$



时间比较

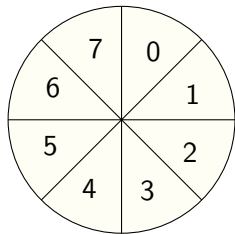
- 顺序组织

- * $20 + (8.3 + 16.7) \times 8 = 220(\text{ms})$

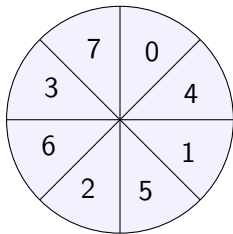
- 随机组织

- * $(20 + 8.3 + 0.5) \times 256 = 7373(\text{ms})$

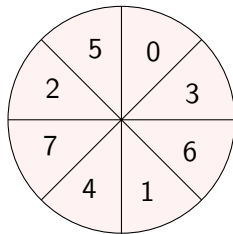
磁盘交叉编址



无交叉编址



单交叉编址



双交叉编址



- 假定磁盘转速为 $20\text{ms}/\text{圈}$ ，磁盘格式化时每个磁道被划分成 10 个扇区，今有 10 个逻辑记录（每个记录的大小刚好和扇区的大小相等）存放在同一磁道上，处理程序每次从磁盘读出一个记录后要花 4ms 进行处理，现要求顺序处理这 10 个记录，若磁头现在正处在首个逻辑记录的始点位置，请问：
 - * 1. 按逆时针方向安排 10 个逻辑记录（磁盘顺时针方向转），处理程序处理完这 10 个记录所花费的时间是多少？
 - * 2. 按最优化分布重新安排这 10 个逻辑记录，写出记录的安排，并计算出所需要处理的时间。



5.6.2 磁盘调度

目标: 减少寻道时间

在多道程序环境下，OS 为每个 I/O 设备维护一条请求队列，对一个磁盘，队列中可能有来自多个进程的许多 I/O 请求，如果随机地从队列中选择一个进程，则磁道访问完全随机，性能很差，因此需要更好的调度策略。



5.6.2 磁盘调度

- 一、FCFS (First Come, First Served)
 - * 也可以称为 FIFO
 - * 特点：简单，寻道时间长，相当于随机访问模式。
 - * 另一种较好的调度策略—LIFO
- 二、SSTF (最短寻道优先)

练习

假设当前时刻先后需要访问的磁道编号分别为：55, 58, 39, 18, 90, 160, 150, 38, 184，假设磁头位于磁道 100 位置
分别计算 FCFS 和 SSTF 的平均寻道长度

FSFS vs. SSTF



Table: FCFS 调度算法

下一个访问磁道	移动距离
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

假设均从 100 磁道开始

Table: SSTF 调度算法

下一个访问磁道	移动距离
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



5.6.2 磁盘调度 - cont.

- 三、扫描算法。
 - * 1. 进程“饥饿现象”
 - SSTF 存在。
 - * 2. SCAN 算法 (电梯调度算法):
 - 在移动方向固定的情况下采用了 SSTF，以避免饥饿现象



5.6.2 磁盘调度 - cont.

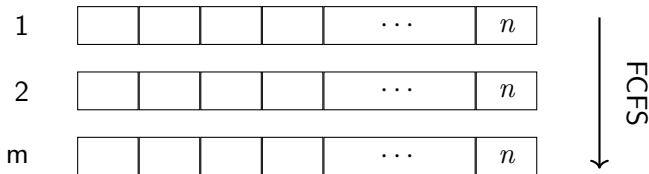
- 四、循环扫描 CSCAN (图 9-5)

- * 一个方向读完，不是象 SCAN 那样回头，而是循环。
- * 最迟访问时间: $2T \times T + S_{\max}$

- 五、N—Step—SCAN 和 FSCAN 算法。

- * 1. N—Step—SCAN

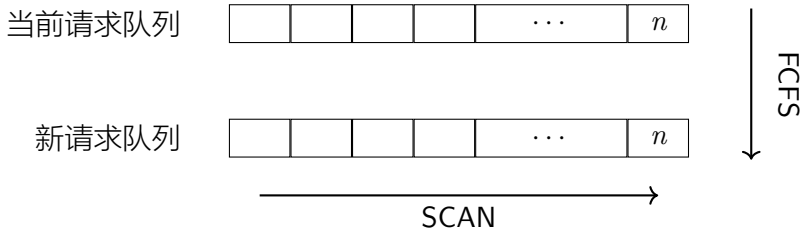
- 粘臂：由于连续对某磁道访问引起的垄断访问，将磁盘请求队列分为长为 N 的子队列 m 个，如下图处理。当 $N=1$ 时，为 FCFS。当 N 取值很大时，为 SCAN。





5.6.2 磁盘调度 - cont.

* 2. FSCAN



FSFS vs. SSTF



- 假设当前时刻先后需要访问的磁道编号分别为：55,58,39,18,90,160,150,38,184，假设磁头位于磁道 100 位置
 - * 分别计算 SCAN 和 CSCAN 的平均寻道长度 (磁头向增加方向移动)

SCAN vs. CSCAN



Table: SCAN 调度算法

下一个访问磁道	移动距离
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

Table: CSCAN 调度算法

下一个访问磁道	移动距离
150	50
160	10
184	24
18	166
38	20
39	1
55	16
58	3
90	32

平均寻道长度: 35.8



5.6.3 磁盘高速缓存 I

- 形式
 - * 逻辑上是磁盘、物理上是驻留在内存中的盘块
 - * 固定大小和可变大小
- 数据交付方式
 - * 数据交付指将磁盘高速缓存中的数据传送给请求者进程
 - * 步骤：先查缓存、后查磁盘并更新缓存
 - * 方式：
 - 数据交付
 - 指针交付：只指向高速缓存中某区域的指针



5.6.3 磁盘高速缓存 II

- 置换算法
 - * 最近最久
 - * 访问频率
 - * 可预见性
 - * 数据一致性：将需要一致性的块放在替换队列的头部，优先回写。
- 周期性回写磁盘
 - * 例：
 - Unix 周期性调用 SYNC
 - ms - dos 采用写穿透方式



5.6.4 提高磁盘 I/O 速度的其它方法

- 提前读
- 延迟写
 - * 访问频率高的磁盘块放在替换队列的尾部，减少回写次数
- 优化物理块的分布
 - * 目的是减小磁头移动距离
 - * 簇分配方式：一个簇为多个连续的块
- 虚拟盘（RAM 盘）
 - * 和磁盘高速缓存区别：虚拟盘由用户控制；磁盘高速缓存由系统控制。



5.6.5 独立磁盘冗余阵列

- 辅存性能的提高速度远远低于处理器和内存的提高速度，使得磁盘存储系统称为提高计算机整体性能的主要问题
- 如单个组件性能提升有限，可采用多个组件并行的方式
- 加州大学伯克利分校的研究小组于 1988 年首次提出 RAID 术语
 - * Redundant Array of Independent Disks: 独立磁盘冗余阵列
- 思路
 - * RAID 是一组物理磁盘驱动器，OS 把他们看做是一个单独的逻辑驱动器
 - * 数据分布在物理驱动器阵列中
 - * 使用冗余的磁盘容量保存校验信息，从而保证当一个磁盘失效时，数据具有可恢复性。
- RAID 共有 7 个级别，从 0 到 6，这些级别并不隐含一种层次关系



RAID 的特点

- 并行交叉存取（条化存取）
- 冗余存取
- 校验存取
- 优点
 - * 可靠性高
 - * 磁盘 I/O 速度高
 - * 性价比高



- 容错性：有 冗余类型：奇偶校验
- 热备盘选项：有 读性能：高
- 随机写性能：低 连续写性能：低
- 需要的磁盘数：三个或更多
- 可用容量： $(n-1)/n$ 的总磁盘容量（ n 为磁盘数）
- 典型应用：随机数据传输要求安全性高，如金融、税务等。



本章实验

- 调查：硬盘的发展历史
- 实现 SSTF 算法和 SCAN 算法
 - * 要求
 - 给出任意的输入流、计算平均寻道长度。
 - 输入流长度、磁头移动方向可定制。
 - 测试：设有 100 各磁道，访问序列如下：
 - 23, 5, 98, 14, 66, 25, 78, 34, 66, 74, 56, 87, 12, 39, 71, 49, 58
 - 当前磁头在 50 道，上次访问的磁道是 18 道，SCAN 算法由内到外移动

END

展示作业 I



1. 协程与进程
2. CPU 与 GPU
3. 中国芯片发展状况
4. 开源软件的发展历史和重要开源项目介绍
5. 计算机存储系统的现状与发展趋势
6. Linux 常用命令
7. 区块链
8. Java 编程语言的发展趋势
9. 著名 IT 公司介绍
10. 著名 IT 人物介绍
11. 函数式编程
12. 信息的度量
13. 操作系统扩展学习资料汇编

致谢



本讲义的内容来自于参考教材及互联网上的部分讲义和公开资料，包括但不限于：

计算机操作系统、深入理解计算机系统、操作系统精髓、操作系统之哲学原理、带你逛西雅图活电脑博物馆...

遗漏之处，请留言联系以便补充。