考研 408—操作系统讲义【I/O 系统】

一. I/O 系统的功能和层次模型

IO 系统管理的主要对象是 IO 设备和相应的设备控制器。其主要作用是完成用户的 IO 请求，提高 IO 速率以及提高设备的利用率，并为高层的进程提供方便的接口。

1. I/O 系统的基本功能

I. 方便用户使用 I/O 设备

i) 隐藏物理设备细节

I/O 设备通常有很多种类型，每一种又有传输方向、数据粒度、传输速度等不同特性，将这些复杂的细节隐藏，仅向上层暴露简单的接口。

ii) 与设备的无关性

用户不但要用抽象的 I/O 命令，还要能够使用现成的逻辑设备名称来控制选择 I/O 设备，并且添加新的设备驱动程序。

II. 提高 CPU 和 I/O 设备利用率

i) 提高设备利用率

尽可能让 CPU 和 I/O 设备并行执行

ii) 对 I/O 设备进行控制

- 轮询 I/O 设备的可编程 I/O 方式
- 中断 I/O
- DMA 直接存储器访问
- I/O 通道

III. 共享设备、系统有序执行

i) 确保对设备的正确共享

- 独占设备：进程间互斥访问该类资源
- 共享设备：如磁盘，多个进程可交叉读写磁盘

ii) 2.错误处理

- 临时性错误：重试操作
- 持久性操作：向上层汇报

2. I/O 系统的层次结构

I. 用户层 I/O 软件

实现与用户的交互，用户可以直接调用此层提供的接口、函数等；

II. 设备独立性软件

用于实现用户程序和设备驱动器的统一接口、设备命名、设备保护以及设备分配和释放等，同时为数据的传输提供必要的空间

III. 设备驱动程序

与硬件直接相关，用于具体实现系统施加给硬件设备的指令

IV. 中断处理程序

保护被中断的 CPU 环境，转入中断处理程序，处理，返回恢复现场

3. I/O 系统的接口

I. 块设备接口

该接口反映了大部分磁盘存储器的本质特征，用于控制该类设备的输入输出。

i) 块设备：数据存取以块为单位

ii) 隐藏了磁盘的二维结构：将二位的物理磁盘结构转换成一维的线性结构

iii) 将抽象命令映射为低层操作：将上层的简单接口命令转换成底层能够识别的操作

II. 流设备接口,又称字符设备接口。

i) 字符设备：数据存取以字符为单位

ii) get 和 put 操作：字符设备不可寻址，只能采用顺序存取方式，通常带有一个缓冲区

iii) in-control 指令：字符设备类型众多，为了以统一的方式处理，必需提供一个通用的指令，该指令包括的参数对应不同设备

III. 网络通信接口

与网络协议和网络层次有关，操作系统要提供面向网络的各种接口。

i) 网络协议：UDP、TCP 等

ii) 网络体系结构：OSI 七层模型，但实际上用的 TCP/IP 协议，只有五层

二. I/O 硬件

4. I/O 设备分类

I.

按信息交换单位
- 块设备
  - 以数据块为单位组织和传递数据信息
  - 基本特性
    - i) 传输速率高，数兆位/s
    - ii) 可寻址，可随机读写任意一块
    - iii) 磁盘I/O采用DMA方式
  - 如磁盘，光盘
- 字符设备（流设备）
  - 以单个字符为单位传送数据
  - 一般用于数据输入/输出（交互式终端：打印机）
  - 基本特征
    - i) 传输速率低
    - ii) 不可寻址，不可指定，输入值地址、输出目标地址
    - iii) 字符设备I/O，采用中断驱动方式

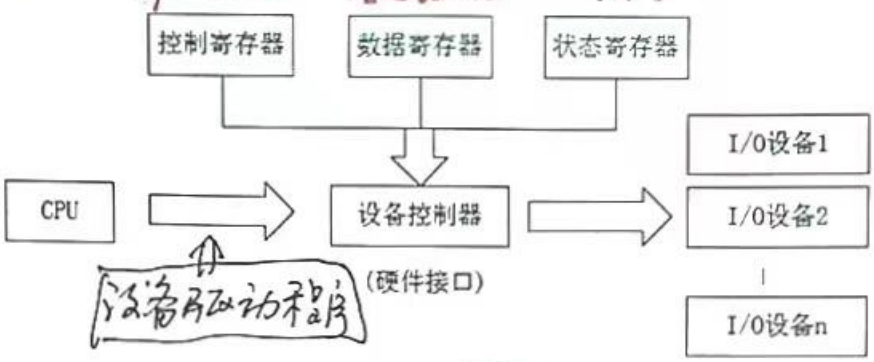
II.

按使用特征
- 存储设备
- ~~I/O设备~~

III.

按传输速率
- 低速设备（键盘、mouse）：每秒数字节～数百字节
- 中速设备（打印机）：每秒数千字节～数万字节
- 高速设备（磁带、磁盘、光盘）：每秒数十万～数千兆字节

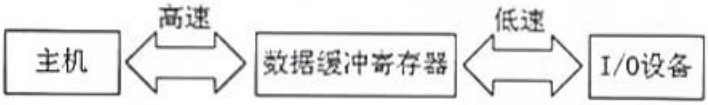
5. 设备控制器：



I. **控制寄存器**：CPU 通过向控制器写命令字执行 I/O 操作，设备控制器按命令要求独立控制器 I/O 设备完成输入/输出。CPU 可转去执行其他操作（实现 CPU 和 I/O 并行操作）。

II. **状态寄存器**：I/O 设备完成情况与设备状态由控制器存入状态寄存器=>控制器和设备之间进行数据交换=>产生中断=>CPU 通过状态寄存器了解操作结果和设备状态。

III. **数据寄存器**：解决 CPU 内存和 I/O 设备之间速率不匹配的问题，并且保证可以并行工作。



6. I/O 寄存器访问方式：

I. **方式 1**：给每个控制寄存器分配一个端口号，通过专门的 I/O 指令完成读写操作

II. **方式 2**：内存映射 I/O：
I/O 寄存器是内存的一部分，系统为每个控制器分配一个唯一的地址，系统可以通过访问的形式访问 I/O 寄存器。

7. I/O 通道(大、中型操作系统)

IV. 目的：

iii) 建立独立的 I/O 操作(组织、管理和结束)，使由 CPU 处理的 I/O 工作转由通道完成（解放 CPU，实现并行）

iv) 减少 CPU 对 I/O 干预

v) 数据传送安全由通道控制

V. 什么是 I/O 通道？

i) 是一种特殊的处理机（独立于 CPU），具有通过执行通道程序完成 I/O 操作的指令

特点：指令单一(局限于与 I/O 操作相关的指令)，与 CPU 共享内存

ii) 通道有自己的指令系统（通道指令）

VI. 基本过程：

CPU 向通道发出 I/O 指令（要执行的 I/O 指令和要访问的 I/O 设备）->通道接收指令（CPU 可转去执行其他工作）->从内存取出通道程序处理 I/O（通道程序存于内存，可以控制设备与内存完成数据交换）->完成操作向 CPU 发出中断信号

VII. 通道类型

i) 字节多路通道

- 低中速连接子通道时间片轮转方式共享主通道
- 字节多路通道不适于连接高速设备，这推动了按数组方式进行数据传送的数组选择通道的形成。

ii) 数组选择通道

- 这种通道可以连接多台高速设备，但只含有一个分配型子通道，在一段时间内只能执行一道通道程序， 控制一台设备进行数据传送，直至该设备传送完毕释放该通道。这种通道的利用率很低。

iii) 数组多路通道

- 含有多个非分配型子通道，前两种通道的组合，通道利用率较好

VIII. 瓶颈问题

- **原因**：通道不足
- **解决办法**：增加设备到主机间的通路，而不增加通道（结果类似 RS 触发器）

8. 中断

中断：在操作系统中地位主要，多道程序得以实现的基础

I. 分类：

i) 中断（外部触发）：CPU 以外的外部事件引起，对外部 I/O 设备发出的中断信号响应（I/O 中断）

ii) 陷入（内部原因，除 0）：异常，有 CPU 内部事件引起的中断（非法指令，地址越界）

II. 中断向量表：

i) 中断信号：对应相应的中断处理程序

ii) 中断程序的入口地址表：中断响应时，系统通过中断号查找中断向量表，获得相应的中断处理程序的入口地址

III. 中断优先级：对紧急程度不同的中断处理方式

IV. 对多中断源的处理方式：

i) 屏蔽中断：当处理机正在处理一个中断时，将屏蔽掉所有的中断，直到处理机已处理完本次中断，再去检查是否有中断产生。所有中断按顺序处理，优点是简单，但不能用于实时性要求较高的中断请求。

ii) 嵌套中断：在设置了中断优先级的系统中，当同时有多个不同优先级的中断请求，CPU 优先响应优先级最高的中断请求，高优先级的中断请求可以抢占正在运行的低优先级中断的处理机。

三. 中断处理程序和设备控制程序

1. 中断处理程序：

I. 功能：

i) 测试是否有未响应的中断信号

ii) 保护被中断进程的 CPU 环境
iii) 转入相应的设备处理程序
iv) 中断处理
v) 恢复 CPU 的现场并退出中断

II. 工作过程：
i) 检查本次 I/O 操作的完成情况
ii) 进行 I/O 结束或错误处理
iii) 唤醒被 I/O 操作阻塞的进程
iv) 启动下一个请求
v) 中断返回

2. 设备驱动程序：是 I/O 进程与设备控制器之间的通信程序，又由于它常以进程的形式存在，故以后就简称为设备驱动进程
I. 主要任务：接受上层软件的抽象命令=>转换成具体的指令=>发送给设备寄存器=>启动设备进行数据传输
II. 具体功能：
i) 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。例如，将磁盘块号转换为磁盘的盘面、 磁道号及扇区号。
ii) 检查用户 I/O 请求的合法性，了解 I/O 设备的状态，传递有关参数，设置设备的工作方式。
iii) 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。
iv) 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。
v) 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。
vi) 启动 I/O 设备，并检查启动是否成功，如成功则将控制返回给 I/O 控制系统，在 I/O 设备忙于传送数据时，该用户进程把自己阻塞，直至中断到来才将它唤醒，而 CPU 可干别的事。

III. 设备驱动程序的处理过程：
- 第一步：将用户和上层软件对设备控制的抽象要求转换成对设备的具体要求，如对抽象要求的盘块号转换为磁盘的盘面、磁道及扇区。
- 第二步：检查 I/O 请求的合理性。
- 第三步：读出和检查设备的状态，确保设备处于就绪态。
- 第四步：传送必要的参数，如传送的字节数，数据在主存的首址等。

3. 对 I/O 设备的控制方式
I. I/O 控制的宗旨
i) 减少 CPU 对 I/O 控制的干预
ii) 充分利用 CPU 完成数据处理工作
II. I/O 控制方式
i) 轮询的可编程 I/O 方式

ii) 中断驱动 I/O 方式

iii) DMA 控制方式

a. 主机与 DMA 控制器的接口

b. DMA 控制器与块设备的接口

c. I/O 控制逻辑

iv) I/O 通道控制方式

I/O 通道方式是 DMA 方式的发展，把对一个数据块的读或写为单位的干预，减少为对一组数据块的读或写以及有关的控制和管理为单位的干预。

可实现 CPU、通道、I/O 设备三者的并行操作

四. 与设备无关的 I/O 软件

1. 基本概念
   - I. 应用程序独立于具体使用的物理设备。
   - II. 驱动程序是一个与硬件(或设备)紧密相关的软件。为实现设备独立性，须在驱动程序上设置一层软件，称为设备独立性软件。
2. 设备独立性(Device Independence)的优点
   - I. 以物理设备名使用设备
   - II. 引入了逻辑设备名
   - III. 逻辑设备名称到物理设备名称的转换（易于实现I/O重定向）
3. 与设备无关的软件
   - I. 设备驱动程序的统一接口
   - II. 缓存管理
   - III. 差错控制
   - IV. 对独立设备的分配与回收
   - V. 独立于设备的逻辑数据块
4. **设备分配中的数据结构**

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）由：**设备控制表（DCT）、控制器控制表（COCT）、通道控制表（CHCT）和系统设备表（SDT）**

设备控制表 DCT

设备类型
设备标识符
设备状态：忙/闲　等待/不等待
与设备连接的 COCT 指针
重复执行次数或时间
设备队列的队首指针
设备队列的队尾指针

控制器控制表 COCT

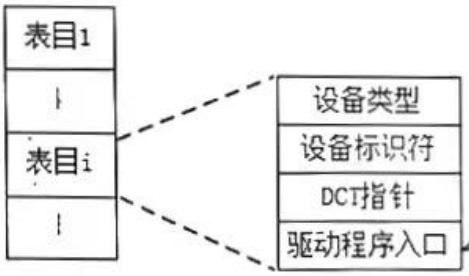
控制器标识符
控制器状态：忙/闲
与控制器连接的 CHCT 指针
控制器队列的队首指针
控制器队列的队尾指针

通道控制表 CHCT

通道标识符
通道状态：忙/闲
与通道连接的 CHCT 指针
通道队列的队首指针
通道队列的队尾指针

显然，在有通道的系统中，一个进程只有获得了通道，控制器和所需设备三者之后，才具备了进行 I/O 操作的物理条件

5. 系统设备表 SDT



逻辑设备表 LUT

I. 分配的流程，从资源多的到资源紧张的:LUT->SDT->DCT->COCT->CHCT

II. 在申请设备的过程中，根据用户请求的 I/O 设备的逻辑名，查找逻辑设备和物理设备的映射表；以物理设备为索引，查找 SDT，找到该设备所连接的 DCT；继续查找与该设备连接的 COCT 和 CHCT，就找到了一条通路。

五. 用户层的 I/O 软件

1. 系统调用与库函数
   I. OS 向用户提供的所有功能，用户进程都必须通过系统调用来获取
   II. 在 C 语言以及 UNIX 系统中，系统调用（如 read）与各系统调用所使用的库函数（如 read）之间几乎是一一对应的。而微软的叫 Win32API
2. 假脱机系统（spooling）：同时联机外围操作技术，spooling 技术是对脱机输入/输出系统的模拟
   I. 主要组成

i) 输入/输出井：
   a. 磁盘中两大存储区
   b. 输入井：模拟脱机输入时的磁盘，用于暂存 I/O 设备输　　入的数据
   c. 输出井：模拟脱机输出时的磁盘，用于暂存用户程序否的输出数据

ii) 输入/输出缓冲区：
   a. 内存中开辟的两个缓冲区
   b. 输入缓冲区：用于暂存由输入设备送来的数据，在传送到输出井
   c. 输出缓存区：用来暂存从输出井送来的数据，在传送给输出设备

iii) 输入/输出进程
   a. 输入进程：又称预输入进程。用来模拟脱机输入时的外围控制机，将用户要求的数据从输入设备传到输入缓冲区，在存放到输入井
   b. 输出进程：也称缓输出进程。用来模拟脱机输出时的外围控制机，将用户要求输出的数据，从内存送到输出井，再将输出井中的数据经输出缓冲区送到输出设备

iv) 井管理程序：
   a. 用于控制作业与磁盘井之间信息的交换

II. 特点（体现操作系统的虚拟性）
提高了 I/O 的速度
对数据所进行的 I/O 操作，由对低速设备演变为对输入井或输出井中的数据存取。

III. 将独占设备改造为共享设备
实际分给用户进程的不是打印设备，而是共享输出井中的存储区域

IV. 实现了虚拟设备功能
将独占设备变成多台独占的虚拟设备。

3. 假脱机打印系统

利用 SPOOLing 技术可将独占的打印机改造成一台供多个用户共享的设备，当用户进程请求打印输出时，假脱机打印系统并不真正把打印机分配给用户进程，而是由假脱机管理进程为它做两件事情

I. 在输出井中为之申请一个空闲的磁盘块区，并将要打印的数据送入其中。
II. 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，然后将该表挂到假脱机文件队列上。

4. 缓冲区管理

I. 缓冲的引入（原因）
   i) 缓和 CPU 与 I/O 设备间速度不匹配的矛盾
   ii) 减少对 CPU 的中断频率，放宽对 CPU 中断响应时间的限制
   iii) 提高 CPU 和 I/O 设备之间的并行性
   iv) 解决数据粒度不匹配的问题

II. 缓冲区分类
   i) 单缓冲区
   即在 CPU 计算的时候，将数据数据输入到缓冲区(大小取决与 T 和 C 的大小)
   ii) 双缓冲区

即允许 CPU 连续工作（T 不断）

iii) 环形缓冲区（专为生产者和消费者打造）

a. 组成
   - 多个缓冲区
   - 多个指针

b. 使用
   - Getbuf 过程
   - Releasebuf 过程

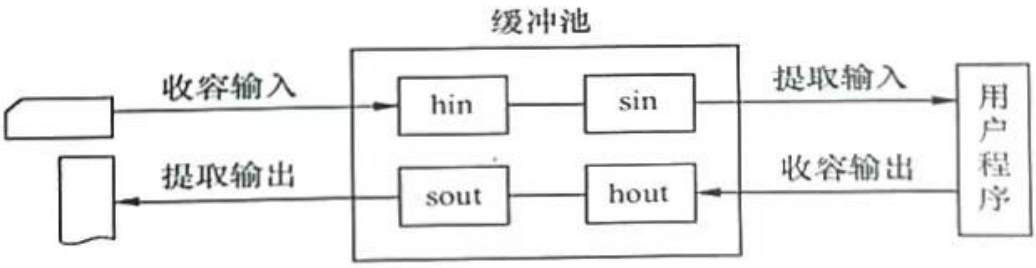
c. 同步问题

iv) 缓冲池（理解为更大的缓冲区）：缓冲池由系统中共用的缓冲区组成。这些缓冲区按使用状况可以分为：空缓冲队列、装满输入数据的缓冲队列（输入队列）、装满输出数据的缓冲队列（输出队列）。

a. 组成
   - 空白缓冲队列（emq）：由空缓冲区链接而成 F(emq)，L(emq)分别指向该队列首尾缓冲区
   - 输入队列（inq）：由装满输入数据的缓冲区链接而成 F(inq)，L(inq)分别指向该队列首尾缓冲区
   - 输出队列（outq）：由装满输出数据的缓冲区链接而成 F(outq)，L(outq)分别指向该队列首尾缓冲

b. Getbuf 和 Putbuf 过程
   - 收容：缓冲池接收外界数据
   - 提取：外界从缓冲池获得数据

c. 缓冲区工作方式（从缓冲区的角度来看）



- 收容输入：在输入进程需要输入数据时，调用 Getbuf（emq）过程，从空缓冲区队列 emq 的队首摘下一个空缓冲区，把它作为收容输入工作缓冲区 hin。然后，把数据输入其中，装满后再调用 Putbuf（inq，hin）过程，将该缓冲区挂在输入队列 inq 的队尾。
- 提取输入：当计算进程需要输入数据时，调用 Getbuf（inq）过程，从输入队列取一个缓冲区作为提取输入工作缓冲区 sin，计算进程从中提取数据。计算进程用完该数据后，再调用 Putbuf（emq，sin）过程，将该缓冲区挂到空缓冲队列 emq 上。

- 收容输出：当计算进程需要输出时调用 Getbuf（emq）过程，从空缓冲队列 emq 的队首取得一个空缓冲，作为收容输出工作缓冲区 hout。当其中装满输出数据后，又调用 Putbuf（outq，hout）过程，将该缓冲区挂在输出队列 outq 末尾。
- 提取输出：要输出时，由输出进程调用 Getbuf（outq）过程，从输出队列的队首取一个装满输出数据的缓冲区，作为提取输出工作缓冲区 sout。在数据提取完后，再调用 Putbuf（emq，sout）过程，将它挂在空缓冲队列 emq 的末尾。

专题 A ：

1. 试说明 I/O 控制发展的主要因素

答：促使 I/O 控制不断发展的几个主要因素如下:

(1) 力图减少 CPU 对 I/O 设备的干预，把 CPU 从繁杂的 I/O 控制中解脱出来，以充分发挥 CPU 数据处理的能力。

(2) 缓和 CPU 的高速性和 I/O 设备的低速性之间速度不匹配的矛盾，以提高 CPU 的利用率和系统的吞吐量。

(3) 提高 CPU 和 I/O 设备操作的并行程度，使 CPU 和 I/O 设备都处于忙碌状态，从而提高整个系统的资源利用率和系统吞吐量。

事实上，I/O 的控制系统由两级(CPU-I/O 设备)发展到三级(CPU-控制器-I/O 设备)，进而发展到四级(CPU-I/O 通道-控制器-I/O 设备)，都是上述三种因素促进的结果。

2. 什么是中断？CPU 响应中断的而过程

答:中断是指计算机在执行期间，系统内部或外部设备发生了某一急需处理的事件，使得 CPU 暂时停止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，继续执行被中断的程序的过程。

CPU 响应中断并进行中断处理的一般过程如下:

(1) 保存现场。CPU 收到中断信号后，通常由硬件自动将处理机状态字 PSW 和程序计数器 PC 中的内容，保存到中断保留区(栈)中。

(2) 转中断处理程序进行中断处理。由硬件分析中断的原因，并从相应的中断向量中获得中断处理程序的入口地址，装入 CPU 的程序计数器中，从而使处理机转向相应的中断处理程序。

(3) 中断返回。中断处理完成后通过中断返回指令，将保存在中断栈中的被中断进程的现场信息取出，并装入到相应的寄存器中，从而使处理机返回到被中断程序的断点执行。

专题 B

1. ~~DMA 控制方式和 I/O 通道控制方式的区别~~？ 中断控制方式和 DMA 方式的区别

答：区别是:

(1) I/O 中断频率。在中断方式中，每当输入数据缓冲寄存器中装满输入数据或将输出数据缓冲寄存器中的数据输出之后，设备控制器便发生一次中断，由于设备控制器中配置的数据缓冲寄存器通常较小，如 1 个字节或 1 个字，因此中断比较频繁。而在 DMA 方式中，在 DMA 控制器的控制下，一次能完成一批连续数据的传输，并在整批数据

传送完后才发生一次中断，因此可大大减少 CPU 处理 I/O 中断的时间。

(2) 数据的传送方式。在中断方式中，由 CPU 直接将输出数据写入控制器的数据缓冲寄存器供设备输出，或在中断发生后直接从数据缓冲寄存器中取出输入数据供进程处理，即数据传送必须经过 CPU;而 DMA 方式中，数据的传输在 DMA 控制器的控制下直接在内存和 I/O 设备间进行，CPU 只需将数据传输的磁盘地址、内存地址和字节数传给 DMA 控制器即可。

2. 什么是虚拟机设备？实现虚拟设备的关键技术是什么？

答:虚拟设备是指通过某种虚拟技术，将一台物理设备变换成若干台逻辑设备，从而实现多个用户对该物理设备的同时共享。由于多台逻辑设备实际上并不存在，而只是给用户的一种感觉，因此被称作虚拟设备。

虚拟设备技术常通过在可共享的、高速的磁盘上开辟两个大的存储空间(即输入井和输出井)以及预输入、缓输出技术来实现。如对一个独占的输入设备，可预先将数据输入磁盘输入井的一个缓冲区中，而在进程要求输入时，可将磁盘输入井中的对应缓冲区分配给它，供它从中读取数据;在用户进程要求输出时，系统可将磁盘输出井中的一个缓冲区分配给它，当将输出数据写入其中之后，用户进程仿佛觉得输出已完成并继续执行下面程序，而在输出设备空闲时，再由输出设备将井中的数据慢慢输出。由于磁盘是一个共设备,因此便将独占的物理设备改造成为多个共享的虚拟设备(相当于输入井或输出井中的一个缓冲区)。预输入和缓输出可通过脱机和假脱机技术实现，而假脱机(即 SPOOLing 技术)是目前使用最广泛的虚拟设备技术。

3. SPOOLing 系统由哪几部分组成？已打印机为例说明如何利用 SPOOLing 技术实现多个进程对打印机的共享？

答:SPOOLing 系统由磁盘上的输入井和输出井，内存中的输入缓冲区和输出缓冲区，输入进程和输出进程以及井管理程序构成。

在用 SPOOLing 技术共享打印机时，对所有提出输出请求的用户进程，系统接受它们的请求时，并不真正把打印机分配给它们，而是由假脱机管理进程为每个进程做两件事情(1)在输出井中为它申请一空闲缓冲区，并将要打印的数据送入其中;(2)为用户进程申请一张空白的用户打印请求表，并将用户的打印请求填入表中，再将该表挂到假脱机文件队列上。至此，用户进程觉得它的打印过程已经完成，而不必等待真正的慢速的打印过程的完成。当打印机空闲时，假脱机打印进程将从假脱机文件队列队首取出一张打印请求表，根据表中的要求将要打印的数据从输出井传送到内存输出缓冲区，再由打印机进行输出印。打印完后，再处理假脱机文件队列中的下一个打印请求表，直至队列为空。这样，虽然系统中只有一台打印机，但系统并未将它分配给任何进程，而只是为每个提出打印请求的进程在输出井中分配一个存储区(相当于一个逻辑设备)，使每个用户进程都觉得自己在独占一台打印机，从而实现了对打印机的共享。

六. 磁盘调度

一个进程需要两种类型的时间，CPU 时间和 IO 时间。 对于 I/O，它请求操作系统访问磁盘。但是，操作系统必须足够满足每个请求，同时操作系统必须保持流程执行的效率和速度。操作系统用来确定接下来要满足的请求的技术称为**磁盘调度**。

1. 磁盘调度有关术语。

I. 寻道时间：寻道时间是将磁盘臂定位到满足读/写请求的指定磁道所用的时间。

II. 倒换延迟：期望的扇区将自己倒换到可以访问 R / W 磁头的位置。

III. 转换时间：这是传输数据所需的时间。

IV. 磁盘访问时间：磁盘访问时间为：

磁盘访问时间=旋转延迟+搜索时间+传输时间

V. 磁盘响应时间：这是每个请求等待 IO 操作所花费时间的平均值。

VI. 磁盘调度的目的：磁盘调度算法的主要目的是从 IO 请求队列中选择一个磁盘请求，并决定处理该请求的时间表。

VII. 磁盘调度算法的目标

- 公平
- 始终最高
- 最小的遍历时间

VIII. 磁盘调度算法

- FCFS 调度算法
- SSTF(最短寻找时间优先)算法
- SCAN 调度
- C-SCAN 调度
- LOOK 调度
- C-LOOK 调度

2. 磁盘调度算法

一次磁盘读写操作的时间由寻找(寻道)时间、延迟时间和传输时间决定：

- 寻找时间 T_s：活动头磁盘在读写信息前，将磁头移动到指定磁道所需要的时间。这个时间除跨越 n 条磁道的时间外，还包括启动磁臂的时间 s，即：$T_s = m * n + s$。式中，m 是与磁盘驱动器速度有关的常数，约为 0.2ms，磁臂的启动时间约为 2ms
- 延迟时间 T_r：磁头定位到某一磁道的扇区(块号)所需要的时间，设磁盘的旋转速度为 r，则：$T_r = 1 / (2 * r)$。对于硬盘，典型的旋转速度为 5400r/m，相当于一周 11.1ms，则 T_r 为 5.55ms;对于软盘，其旋转速度在 300~600r/m 之间，则 T_r 为 50~100ms。
- 传输时间 T_t：从磁盘读出或向磁盘写入数据所经历的时间，这个时间取决于每次所读/写的字节数 b 和磁盘的旋转速度：$T_t = b / (r * N)$。式中，r 为磁盘每秒钟的转数;N 为一个磁道上的字节数。

在磁盘存取时间的计算中，寻道时间与磁盘调度算法相关，下面将会介绍分析几种算法，而延迟时间和传输时间都与磁盘旋转速度相关，且为线性相关，所以在硬件上，转速是磁盘性能的一个非常重要的参数。

总平均存取时间 T_a 可以表示为：$T_a = T_s + T_r + T_t$

注意：在实际的磁盘 I/O 操作中，存取时间与磁盘调度算法密切相关。调度算法直接决定寻道时间，从而决定总的存取时间。

I. **先来先服务（First Come First Served，FCFS）**

FCFS 算法根据进程请求访问磁盘的先后顺序进行调度，这是一种最简单的调度算法。该算法的优点是具有公平性。如果只有少量进程需要访问，且大部分请求都是访问簇聚的文件扇区，则有望达到较好的性能;但如果有大量进程竞争使用磁盘，那么这

种算法在性能上往往接近于随机调度。所以，实际磁盘调度中考虑一些更为复杂的调度算法。

i) 算法思想:按访问请求到达的先后次序服务。

ii) 举例说明：

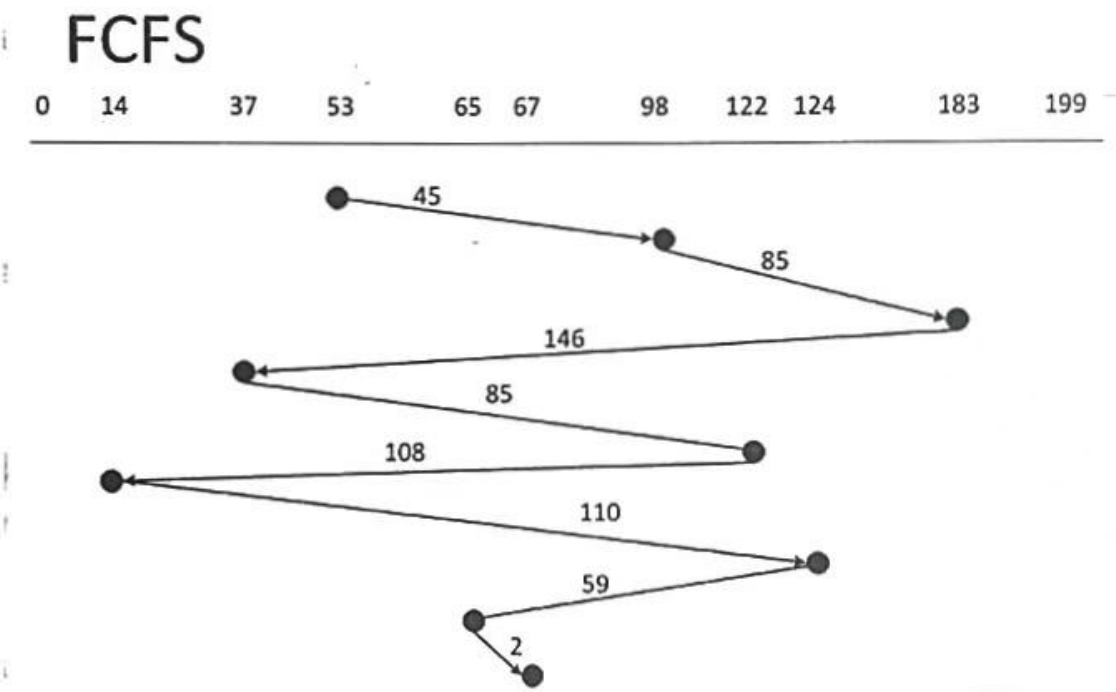
假设磁盘访问序列:98，183，37，122，14，124，65，67。读写头起始位置:53。求:磁头服务序列和磁头移动总距离(道数)。

由题意和先来先服务算法的思想，得到下图所示的磁头移动轨迹。由此:

磁头服务序列为:98，183，37，122，14，124，65，67

磁头移动总距离

=(98-53)+(183-98)+|37-183|+(122-37)+|14-122|+(124-14)+|65-124|+(67-65)=640(磁道)



II. 最短寻道时间优先（Shortest Seek Time First，SSTF）

SSTF 算法选择调度处理的磁道是与当前磁头所在磁道距离最近的磁道，以使每次的寻找时间最短。当然，总是选择最小寻找时间并不能保证平均寻找时间最小，但是能提供比 FCFS 算法更好的性能。这种算法会产生“饥饿”现象。

i) 算法思想：优先选择距当前磁头最近的访问请求进行服务，主要考虑寻道优先。

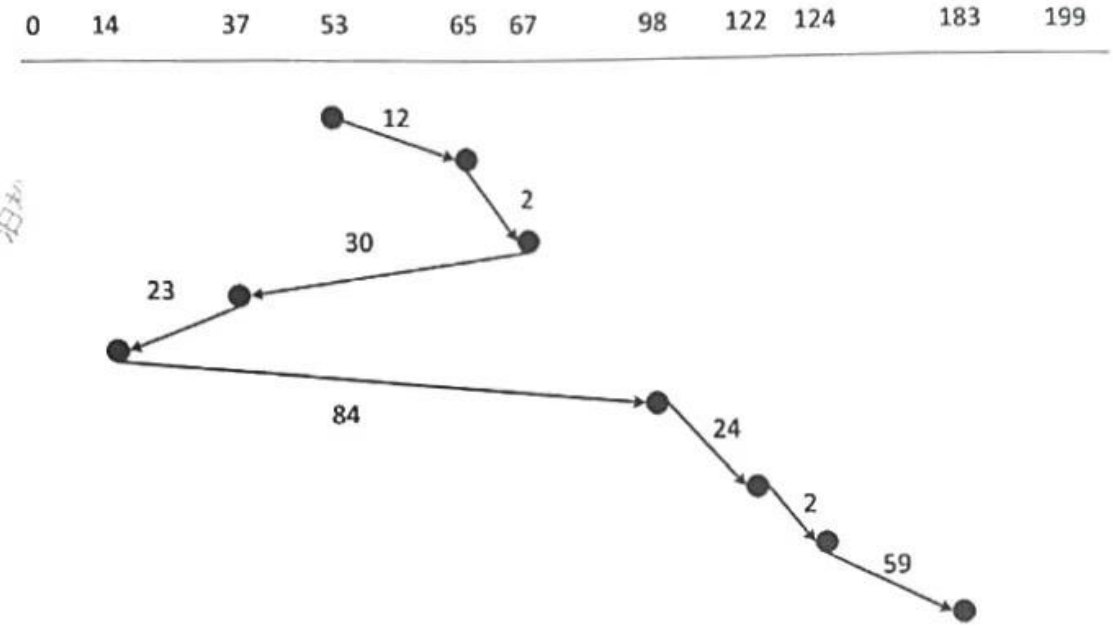
ii) 举例说明：

下面栗子中，磁头当前位于 53 位置，则：

磁头服务序列为:65，67，37，14，98，122，124，183

磁头移动总距离=12 + 2 + 30 +23 + 84 + 24 + 2 + 59 = 236 (磁道)

SSTF



III. **扫描算法（SCAN）或电梯调度算法**

SCAN 算法在磁头当前移动方向上选择与当前磁头所在磁道距离最近的请求作为下一次服务的对象。由于磁头移动规律与电梯运行相似，故又称为电梯调度算法。SCAN 算法对最近扫描过的区域不公平，因此，它在访问局部性方面不如 FCFS 算法和 SSTF 算法好。

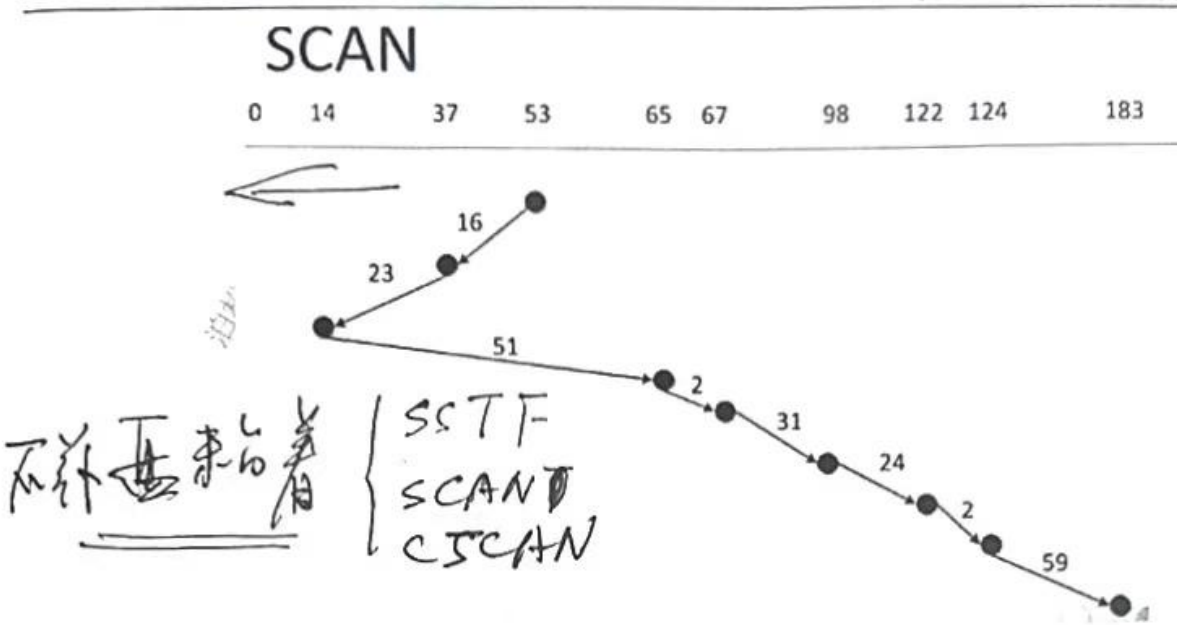
i) 算法思想：当设备无访问请求时，磁头不动；当有访问请求时，磁头按一个方向移动，在移动过程中对遇到的访问请求进行服务，然后判断该方向上是否还有访问请求，如果有则继续扫描；否则改变移动方向，并为经过的访问请求服务，如此反复。

ii) 举例说明：

假设磁头当前在 53 的位置，现在朝 0 方向移动则：

磁头服务序列为:37，14，65，67，98，122，124，183

磁头移动总距离=16 + 23 + 51 + 2 + 31 + 24 + 2 + 59 = 208 (磁道)



不科要求着 { SSTF SCAN CSCAN

Ⅳ. 循环扫描算法（Circular SCAN，CSCAN）

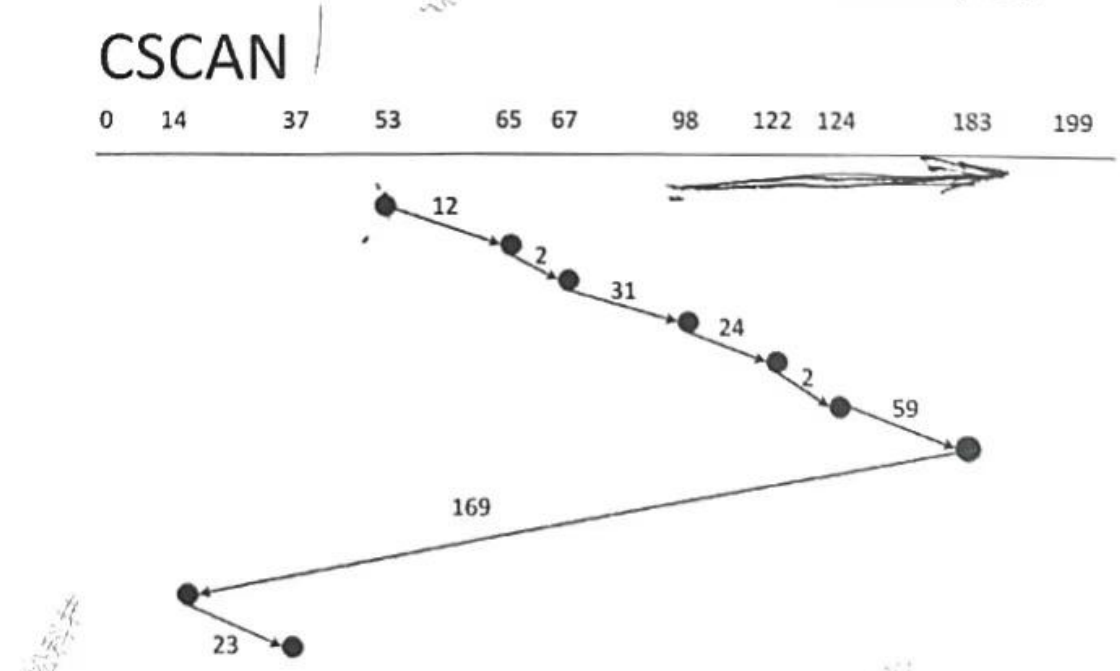
CSCAN 调度算法是在扫描算法的基础上改进的，为了减少延迟，规定磁头单向移动，例如，只是自里向外移动，从当前位置开始沿磁头的移动方向去选择离当前磁头最近的那个柱面访问，如果沿磁头的方向无请求时，磁头立即返回到最里面的欲访问的柱面，再亦即将最小柱面号紧接着最大柱面号构成循环。

i) 举例说明：

假设磁头移动的方向是从柱面 0 到柱面 199 方向。

磁头服务序列为:65，67，98，122，124，183，14，37

磁头移动总距离=12 + 2 + 31 + 24 + 2 + 59 + 169 + 23= 322 (磁道)



V. 磁盘调度算法比较

	优点	缺点
FCFS (先来先服务)	公平、简单	平均寻道距离大，仅应用在磁盘 I/O 较少的场合
SSTF (最短寻道时间优先)	性能比“先来先服务”好	不能保证平均寻道时间最短，可能出现“饥饿”现象
SCAN (扫描/电梯算法)	寻道性能较好，可避免“饥饿”现象	不利于远离磁头一端的访问请求
CSCAN (循环扫描算法)	消除了对两端磁道请求的不公平	兼顾较好的寻道性能和防止饥饿现象，同时减少一个请求可能等待的最长时间。

每张的输入/出。

I/O 设备 —— 对外接

数据的传输

I/O 设备

① 设备地址。

② 设备功能。

I/O 端口

① 设备内部 I/O 寄存器

② 设备内部存储器

③ 设备内部 I/O 控制逻辑

CPU 与 I/O 设备交换信息的方式

① 程序查询方式

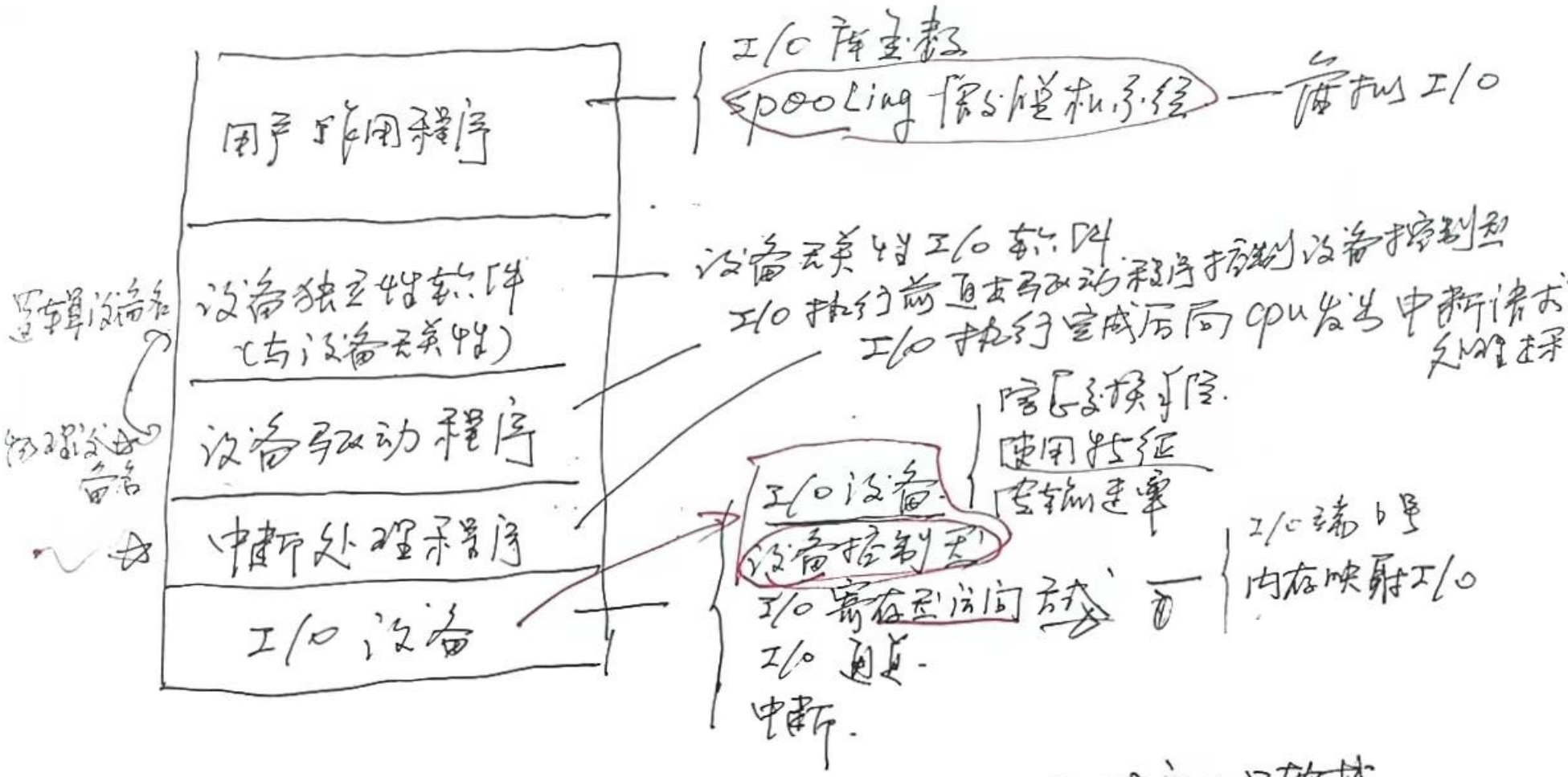
② 程序中断方式

③ 直接存储器访问方式（DMA）

I> 阻塞 I/O 和非阻塞 I/O 的区别
II> 与设备无关性
(实现)
III> 缓冲区
IV> 设备 I/O 的实现方法
V> 小问题:总结
VI> 软中断

A> 因为要等待设备完成 I/O 的动作
B> 用多进程或多线程实现并发的效果
⇒ 缓冲区的作用:为了...
A> 快的 CPU 对慢的 I/O 干预
B> 缓冲区的作用:为了...
A> 因为设备多种多样
B> 【设备无关性】
A> 用户进程、中断处理、设备驱动程序、硬件

I/O 系统层次结构



中断请求 ⇒ 中断向量表 ⇒ 中断号 — 中断程序的 PC 地址

⇒ 调用相应的中断程序 ⇒ 执行中断

中断处理程序
设备控制器
CPU
内存
设备1
设备2
设备n
中断向量表
中断处理程序入口地址
中断服务程序
设备控制器
I/O设备
设备控制器
I/O设备
设备控制器
I/O设备
CPU
内存

设备驱动程序处理步骤.

1> 将用户抽象的I/O请求转换为具体的I/O指令.

2> 检查I/O请求合法性.

3> 读写和检查设备状态.

4> 传送参数.

5> 启动I/O设备

I/O设备控制方式

1> 轮询可编程I/O方式

2> 中断驱动方式 —— 中断可编程I/O方式 (现在)

用户I/O请求 ⇒ cpu向设备控制器发送一条I/O指令

⇒ cpu：阻塞用户进程，将cpu调度给其它就绪进程

I/O设备完成相应的数据传输

⇒ I/O完成后，设备控制器向cpu发送I/O中断请求

⇒ cpu响应中断 ⇒ 进入中断处理

⇒ 唤醒被阻塞进程

优点：cpu、I/O可并行

缺点：① 需在意普遍小，需频繁发生中断，

cpu运行的事相对麻烦

② 数据传送方式：cpu直接干预数据

传送到设备控制器中的数据缓冲

需在到

DMA只需传送数据在内存的起始

地址和字节数

3> DMA(直接存储器访问方式)

A> DMA控制器 —硬件实现
—应用于磁盘访问

B> 主程

用户I/O请求 ⇒ cpu将I/O指令发送到DMA控制器(命令寄存器)

⇒ cpu：同中断方式

I/O：输入：将要存放输入数据的内存始址及字节数存入DMA的内存始址寄存器和PC

存储：将准备存储的磁盘地址、内存始址、字节数存入DMA控制器

输出：同输入

⇒ 阻塞用户进程，将cpu调度给其它就绪进程

⇒ 完成I/O

⇒ 由DMA向cpu发出中断请求

⇒ cpu响应中断

⇒ 唤醒被阻塞进程

4> I/O通道控制方式

① 通道：硬件实现。——独立于CPU的处理机

通道程序：存放于内存。

通道指令：操作系统发给通道可执行的I/O指令。

② 过程。

用于I/O请求 ⇒ CPU只需向I/O通道发送一条I/O指令

⇒ (1) 通过通道程序找到通道程序所在的地址

(2) 要访问的I/O设备

⇒ CPU：调度其它就绪进程。

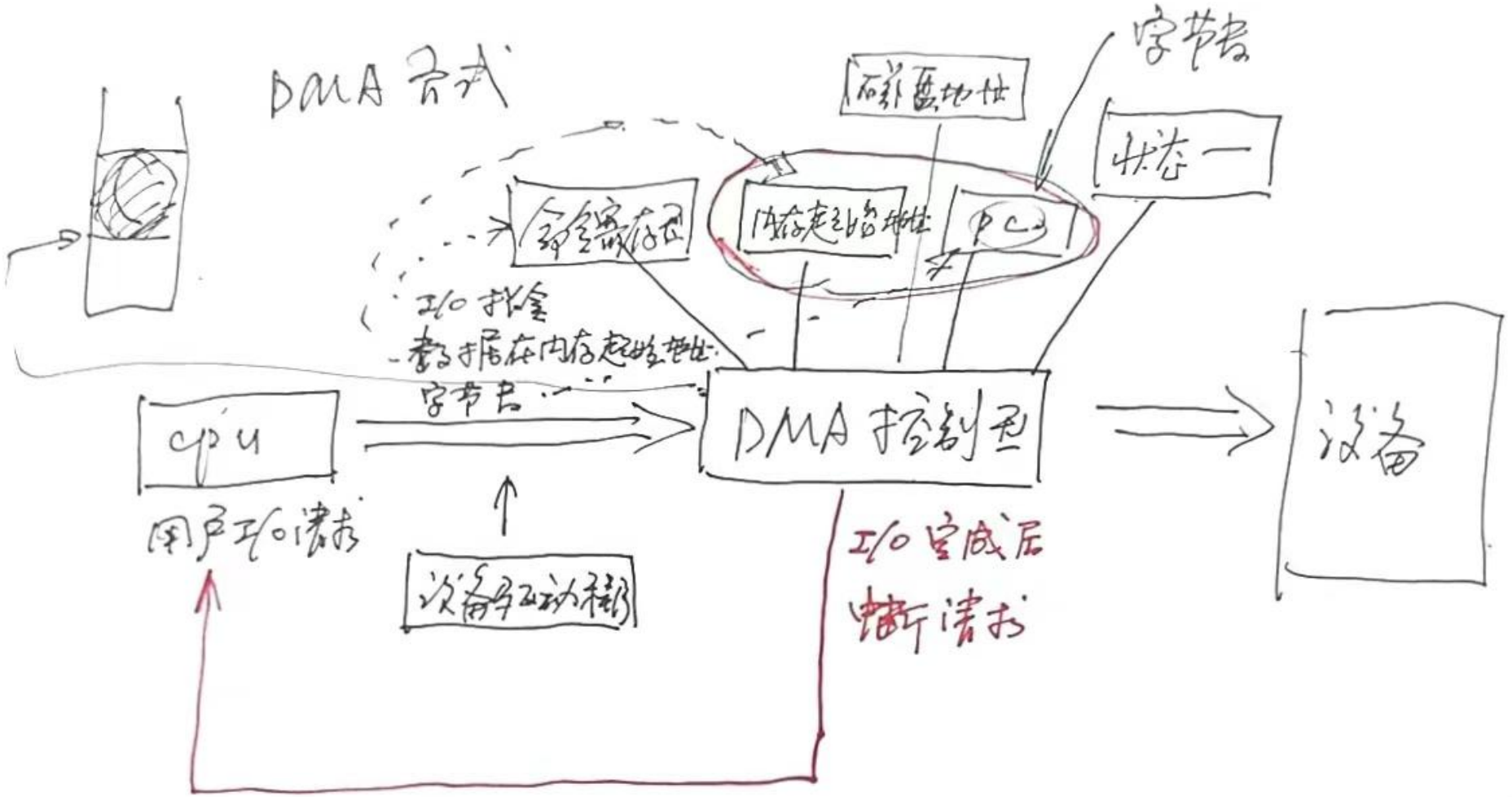
通道：通过通道程序控制完成I/O任务

⇒ 完成后，向CPU发出中断请求。

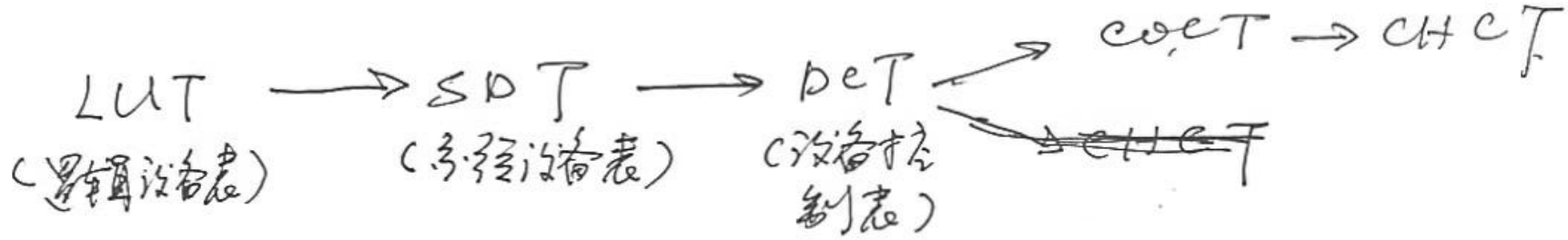
⇒ 同DMA。

区别：对象：CPU、通道、I/O三者之间的并行工作。

☆ CPU只向通道发送一条I/O指令即可



注意：由 DMA 的内存始址和 PC 传送向内存写数据
不需要 CPU 干预 ⇒ 进一步提高了 CPU 利用率。



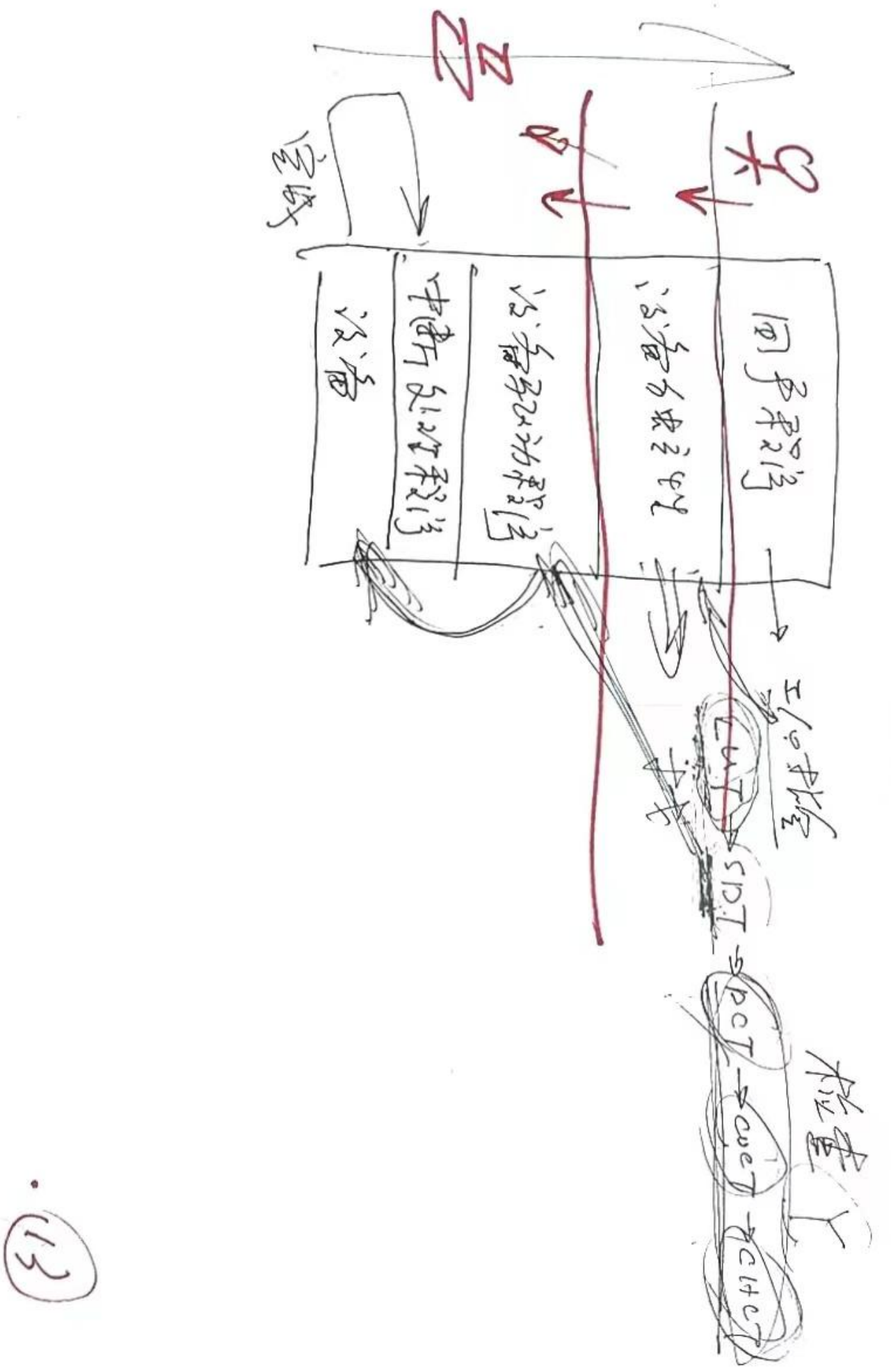
LUT：通过逻辑设备查找对应的物理设备
——知道实用哪个物理设备。

SDT：查找实际用的物理设备的驱动程序。

DCT：当前实际用的物理设备的状态 ↔ 不可用 → 差错管理
可用：通过COCT找到

COCT：若设备可用：查看控制器是否空闲

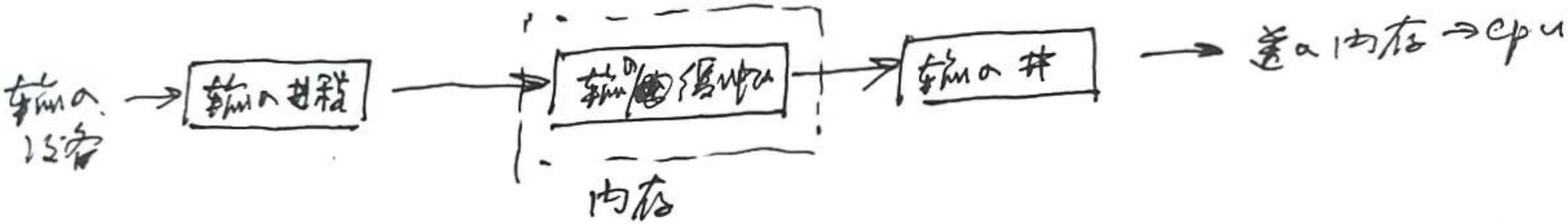
CHCT：查看通道状态 ↔ 忙：对
闲：接收启动设备。



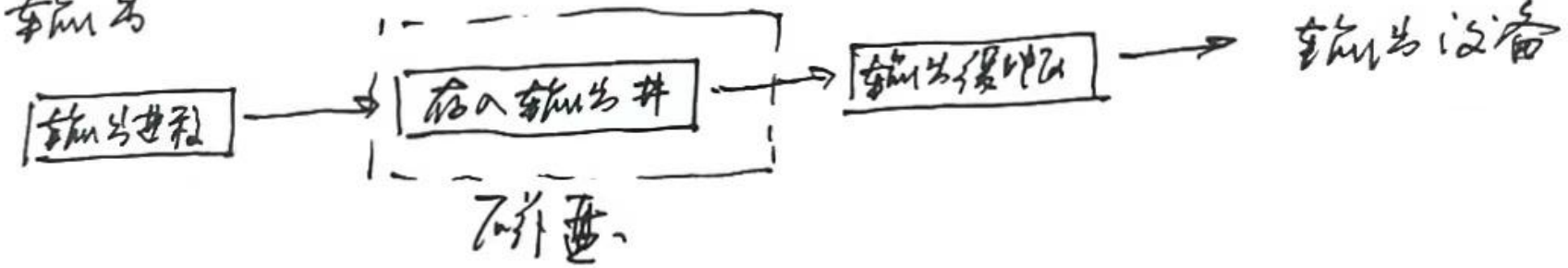
(13)

spooling 系统 组成
- 输入井 → 磁盘
- 输入缓冲区 — 内存
- 输入进程 — 外围控制机
- 井管理程序

输入：（干预输入进程）



输出为

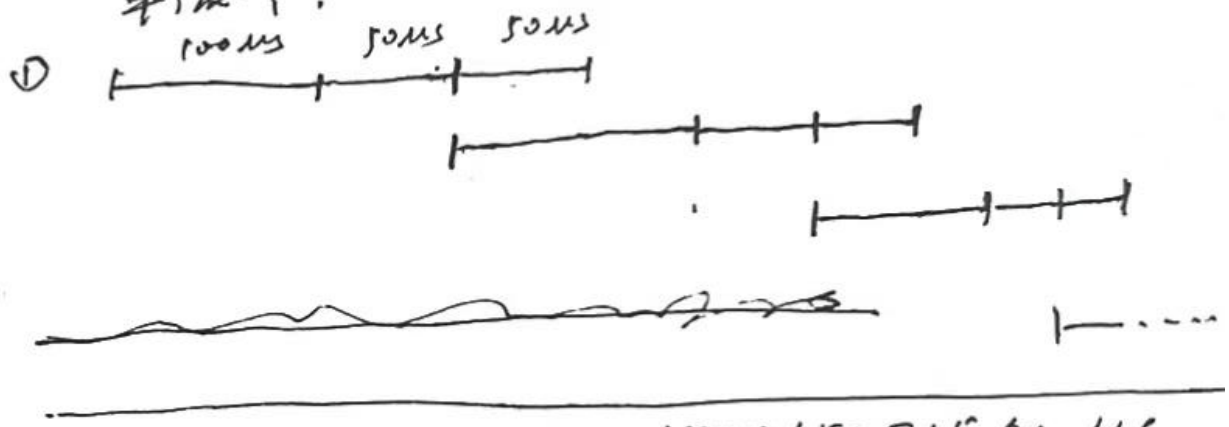


CPU
内存
磁盘
磁头
磁道
扇区
#扇区
内存的缓冲区
spool
内存
磁盘
CPU
内存

2011—1—31

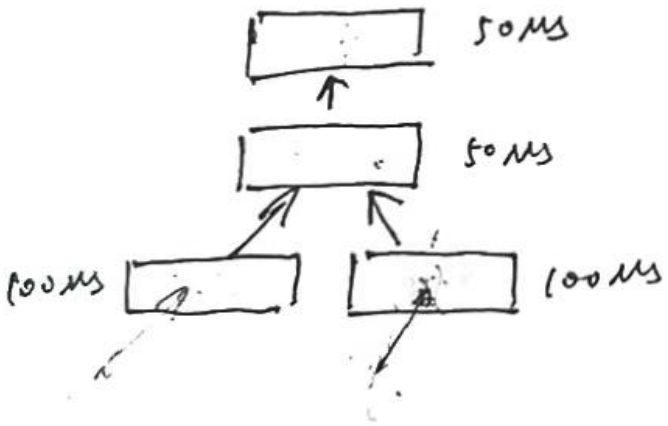
2块 —— 10个磁盘块 —— 每个磁盘块 = 缓冲区

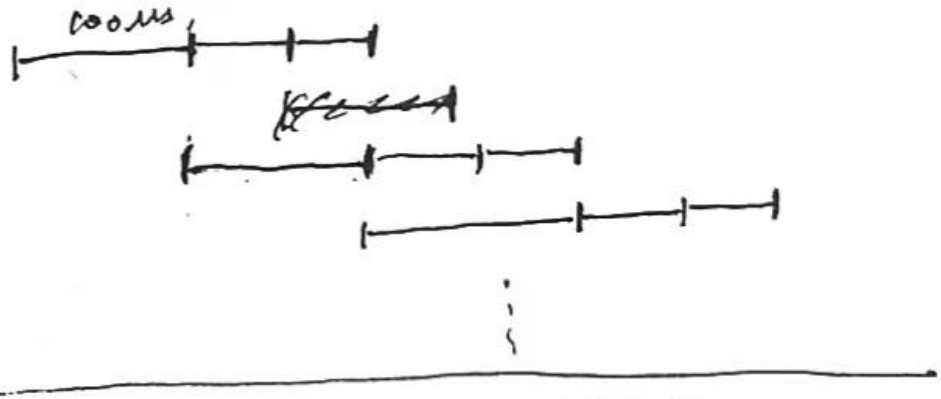
单缓冲：



$10 \times 150 + 50 = 1500 + 50 = 1550\ ms$

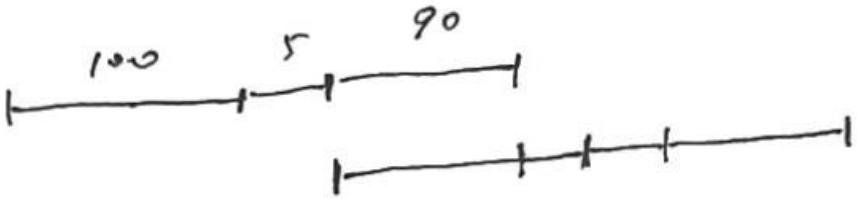
双缓冲





$9 \times 100 + 200 = 1100$

2013-1-27



~~195~~

195×2-90=390-90=300

磁盘
磁头

磁盘面
磁头

SCAN
C SCAN
72+52+19+150+27
122
141
298
323

SCAN
12 35 41 68 105 110 170 180 195
53 98 183 37 122 14 124 65 67
53 65 67 98 122 124 183 199
37 14