第6章　设备管理

- 设备管理涉及计算机系统与外界数据交换和通信联系，计算机系统通过各种各样的I/O设备完成数据的采集、信息的输出等。

目标和功能

- I/O管理是操作系统的主要功能之一，负责管理所有I/O设备。
- 计算机系统中存在着大量的I/O设备，其性能和应用特点可能完全不同，所以要建立一个通用的、一致的设备访问接口，使用户能够方便地使用I/O设备。

6.1 I/O硬件组成

6.2 I/O软件的组成

6.3 磁盘管理

6.1 I/O硬件组成

6.1.1 I/O设备分类

① 字符设备：人机交互设备。是以字符为单位发送和接收数据的，通信速度比较慢。键盘和显示器、鼠标、扫描仪、打印机、绘图仪等。

② 块设备：外部存储器。以块为单位传输数据。常见块尺寸：512B～32KB。如磁盘、磁带、光盘等。

③ 网络通信设备：主要用于与远程设备的通信。传输速度比字符设备快，比块设备慢。如网卡、调制解调器等。

④ 时钟：按预先规定好的时间间隔产生中断。

6.1.2 设备控制器

I／O设备一般由机械和电子两部分组成。电子部分叫做设备控制器或适配器。机械部分是设备本身。在小型和微型机中，设备控制器常采用电路板插入计算机中。控制器卡上通常有一个插座，通过电缆与设备相连。

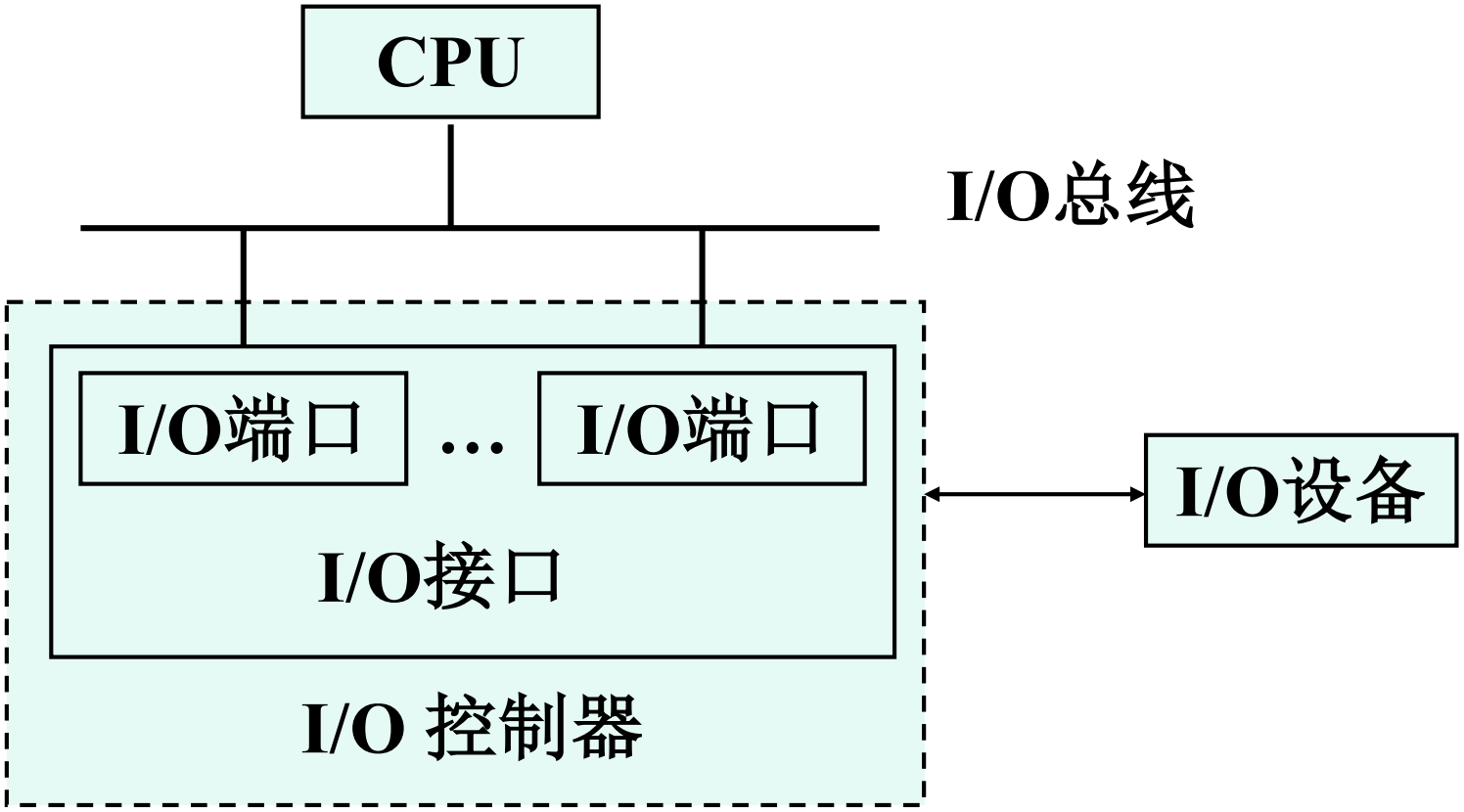
很多控制器可以连接两个、四个，甚至八个相同的设备。常常把这两部分分开处理，以便提供通用的、模块化的设计。

1. I/O体系结构

① I/O端口。是连接到I/O总线上的设备的I/O 地址集。每个设备寄存器有一个端口号。设备寄存器包括：控制寄存器、状态寄存器、数据缓冲寄存器等。

② I/O接口。是处于一组I/O端口和对应的设备控制器之间的一种硬件电路。

③ I/O总线。是CPU与I/O设备之间的通路。



PC的I/O 体系结构

2. 设备控制器

- I/O设备一般由机械和电子两部分组成。机械部分是设备本身。电子部分叫做设备控制器。
- 设备控制器处于CPU和I/O设备之间，接收从CPU发来的命令，控制I/O设备工作。
- 每个控制器有几个寄存器，用来与CPU通信。
  - 控制寄存器：接收CPU发送的读写命令。
  - 状态寄存器：包含设备的状态信息。
  - 数据缓冲寄存器：通常为1B至4B。

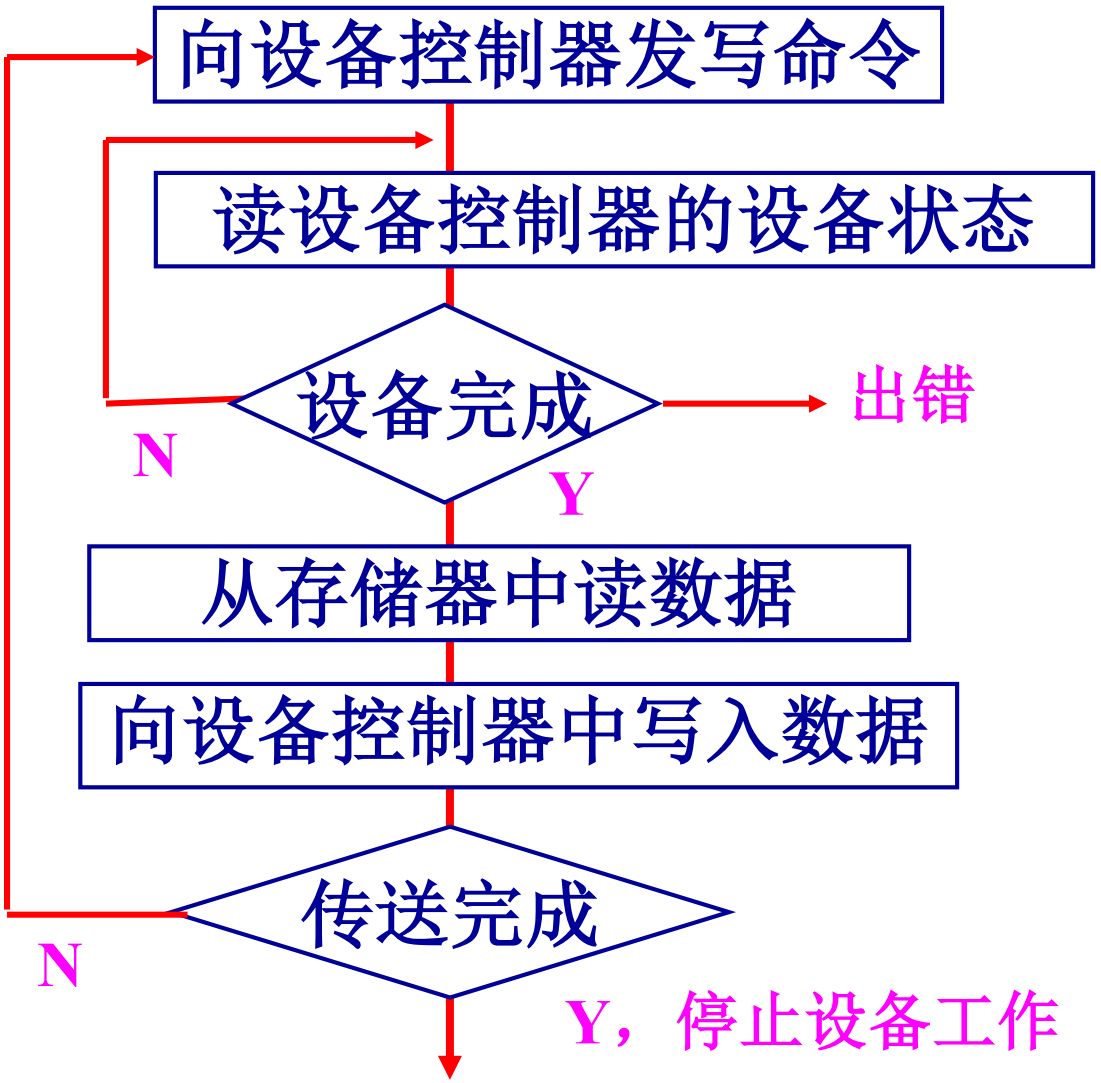
- 除了几个寄存器外，许多设备控制器还有一个操作系统可以读写的数据缓冲区。如在屏幕上显示像素的常规方法是使用一个视频RAM，该RAM基本上只是一个数据缓冲区。
- 磁盘控制器：从磁盘驱动器出来的是一连串的位流，控制器把串行的位流组装为字节，存入控制器内部的缓冲区中，形成以字节为单位的块。对块验证后，再一次一个字节或字地存入内存。

6.1.3 I/O数据传输的控制方式

1. 程序查询方式（polling）
2. 中断方式
3. 直接存储器访问(DMA)方式
4. 通道控制方式

- 程序查询方式下，CPU在启动设备进行一个字符的传输过程中，不断查询设备控制器的状态，检查设备是否已经完成。若完成，再启动设备进行下一个字符传输，否则，一直循环查询，直到设备完成。CPU与设备完全串行工作。
- 该方式的工作过程非常简单，但CPU的利用率低。因为CPU执行指令的速度高出I/O设备几个数量级，所以在循环测试中浪费了大量的CPU处理时间。

1. 程序查询方式



CPU忙等
串行工作

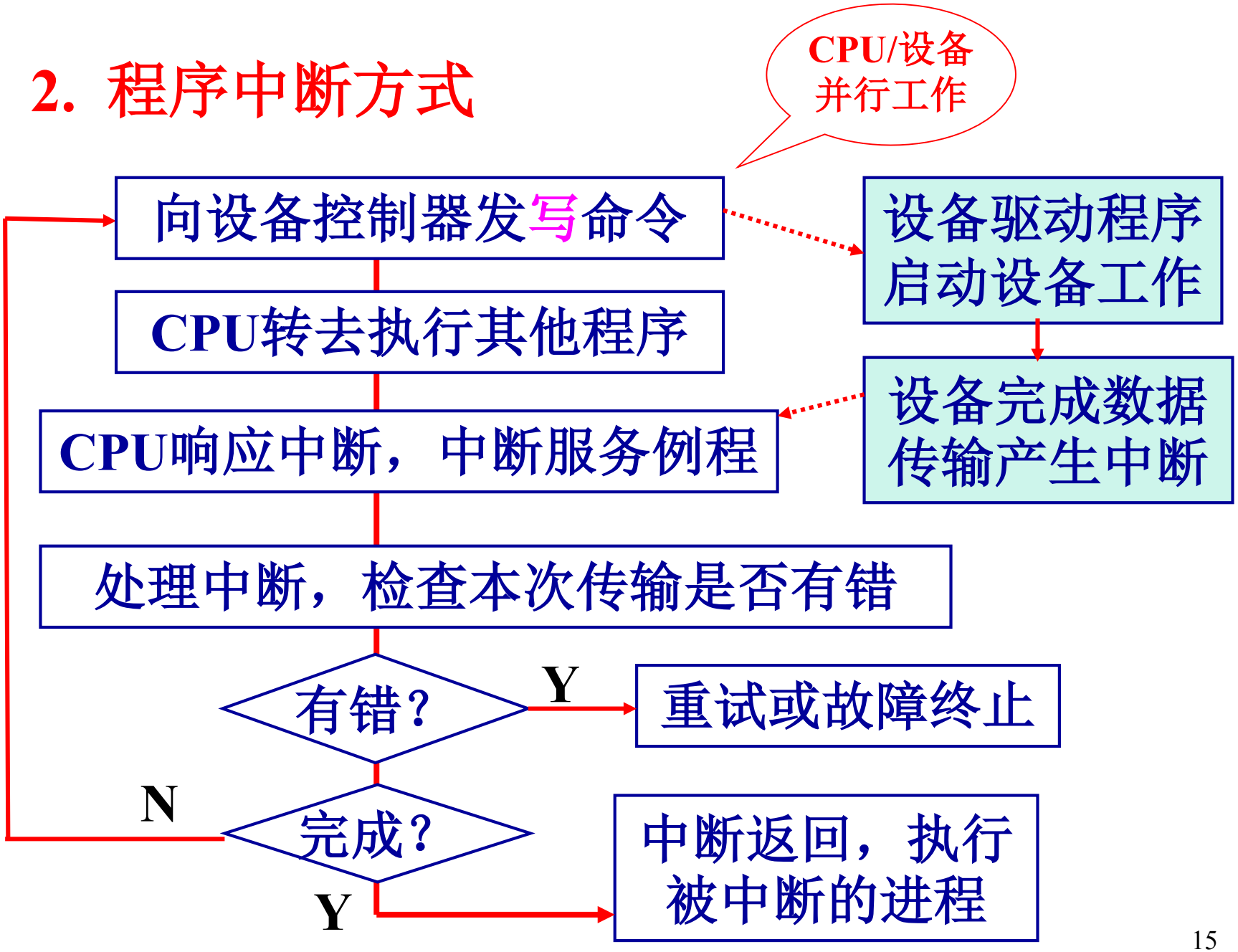
重复执行，
直到一批数
据传输完成。

为了减少程序查询方式中CPU等待时间以及提高系统的并行工作速度，从而提出了中断控制方式。这种方式要求CPU与设备之间有相应的中断请求线，且要求在状态R中有中断允许位。

程序中断方式下，CPU一旦启动设备成功，CPU转去执行另一个程序。当设备完成时，向CPU提出中断请求，CPU执行完当前一条指令，就响应中断，转去执行中断处理程序。从而使CPU可与设备并行操作。

但这种方式仍然存在许多问题，如每台设备输入/输出一个数据都要求中断CPU，这样，在一次数据传送过程中，中断发生次数较多，从而耗去大量CPU处理时间。

2. 程序中断方式


CPU/设备
并行工作
向设备控制器发写命令
CPU转去执行其他程序
CPU响应中断，中断服务例程
处理中断，检查本次传输是否有错
有错？
Y
重试或故障终止
N
完成？
Y
中断返回，执行
被中断的进程
设备驱动程序
启动设备工作
设备完成数据
传输产生中断

3. 直接存储器访问（DMA）

- 有时DMA控制器集成到磁盘控制器中；有时主板上只有一个DMA控制器，可共享。
- 通常，CPU控制地址总线，进行与主存储器的数据交换。
- 允许DMA控制器接管地址总线的控制权，直接控制控制器内部缓冲区与主存之间的数据交换。

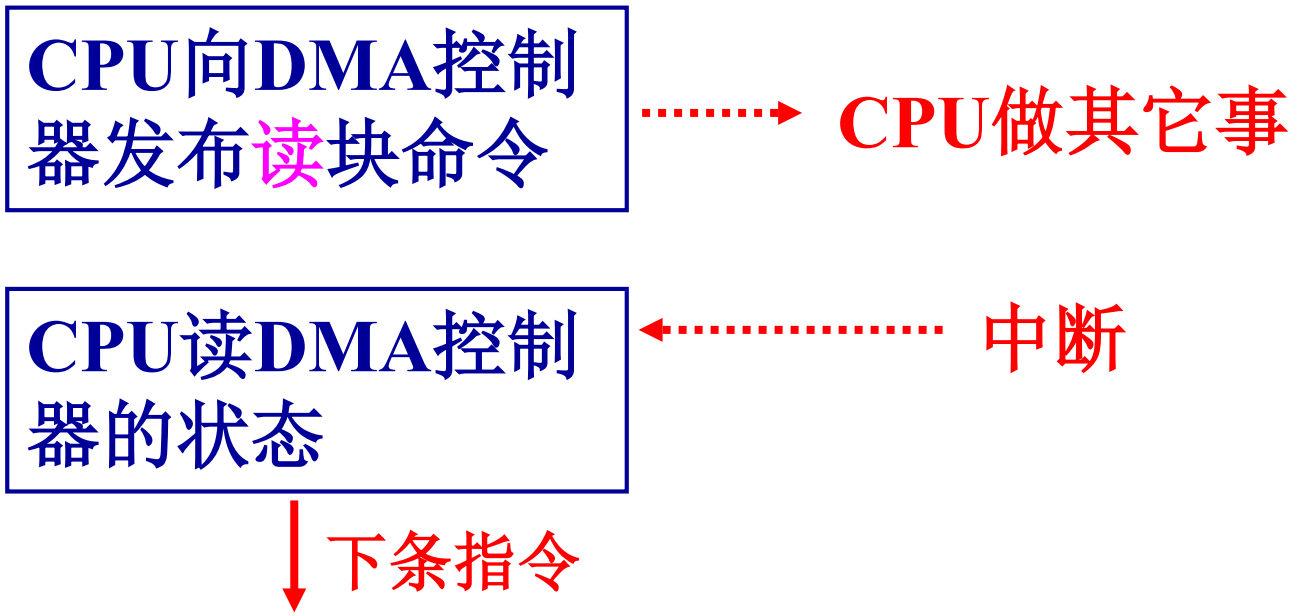
- 许多控制器，特别是块设备的控制器支持直接存贮器存取，即**DMA**。让我们首先看一下不用**DMA**时，磁盘如何读。操作系统将命令写入控制器寄存器中,以实现读某磁盘块。首先，控制器从磁盘驱动器串行地一位一位地读一个块，直到将整块信息放入控制器的内部缓冲区中。

- 其次，它计算检查和，以核实没有读错误发生。然后控制器产生一个中断。CPU响应中断，控制转给操作系统的磁盘中断处理程序。当操作系统开始运行时，它重复地从控制器缓冲区中一次一个字节或一个字地读这个磁盘块的信息，并将其存入存贮器中。
- 显然，这种采用软件的方法由CPU重复地一个字节或一个字地从控制器缓冲区读信息浪费了大量CPU时间。 DMA的采用使CPU摆脱了这种低级工作。

- 在DMA方式中，I/O控制器具有更强的功能。它除了具有上述中断功能外，还有一个DMA控制机构。在DMA控制器控制下，设备和内存之间可成批地进行数据交换，而不用CPU干预。

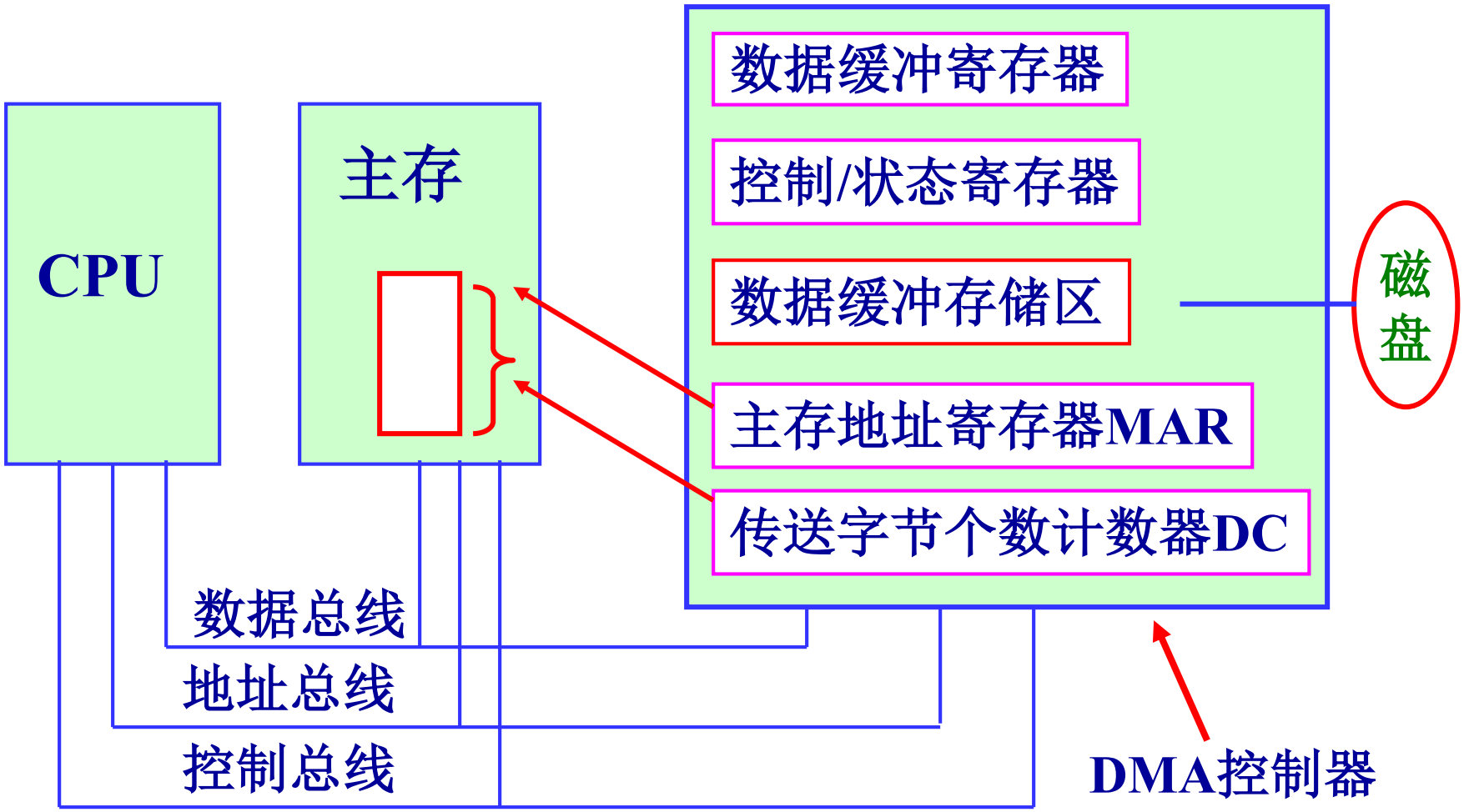
- 这样，既大大减轻了CPU的负担，也使I/O数据传送速度大大提高。这种方式应用于块设备的数据传输。

- 整块数据的传输是在DMA控制下完成的。仅在开始和结束时才需CPU干预。



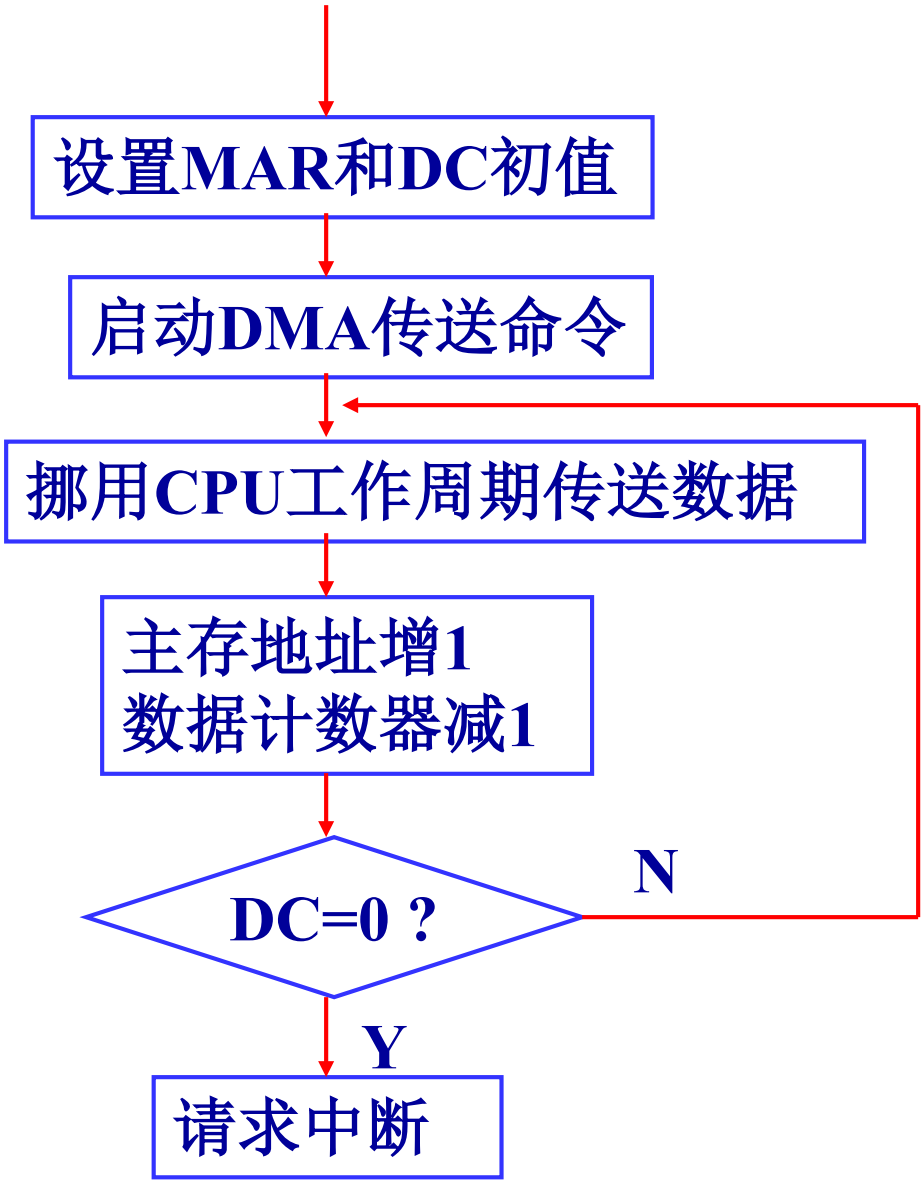
当使用DMA时，磁盘控制器增加了内存地址寄存器和传送字节个数寄存器，从而使磁盘控制器具有智能。CPU除向控制器提供要读块的磁盘地址外，还要向控制器提供两个信息：要读块送往主存的起始地址和要传送的字节数。

DMA控制器独立地进行数据传送



DMA工作过程：

每当磁盘把一块数据读入控制器的数据缓冲区时，检验校验和。DMA控制器取代CPU，接管地址总线的控制权，直接控制与主存的数据交换。使CPU访问总线时速度会变慢。



- 采用DMA方式时，不仅允许CPU控制地址总线，存取主存的程序和数据，而且允许DMA控制器接管地址总线的控制权，控制DMA缓冲区与主存的数据交换。从而使磁盘设备与存贮器之间的数据传送不需要CPU介入，因而减轻了CPU负担。
- 当DMA控制磁盘与存贮器之间进行信息交换时，每当磁盘把一个数据读入控制器的数据缓冲区时，按照DMA控制器中的存贮器地址寄存器内容把数据送入相应的存贮器单元中。然后，DMA硬件自动地把传送的字节计数器减1，把存贮器地址寄存器加1，并恢复CPU对主存贮器的控制权。

- DMA控制器对每一个传送的数据重复上述过程，直到传送字节计数器为“0”时，向CPU产生一个中断信号。
- 当CPU响应中断，由操作系统接管CPU控制权。检查本次传输是否正常完成，若是，因为数据已经传输结束，无需再做块的复制工作，只需检查是否还有I/O请求，若有，再次启动磁盘，否则，停止磁盘传输，并结束中断处理。

DMA方式与中断方式的主要区别

- （1）中断方式是在数据缓冲R满之后，发中断要求CPU进行处理，而DMA方式则是在所要求转送的数据块全部结束时，要求CPU处理。这就大大减少了CPU进行中断处理的次数。
- （2）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控制器的控制下完成的。

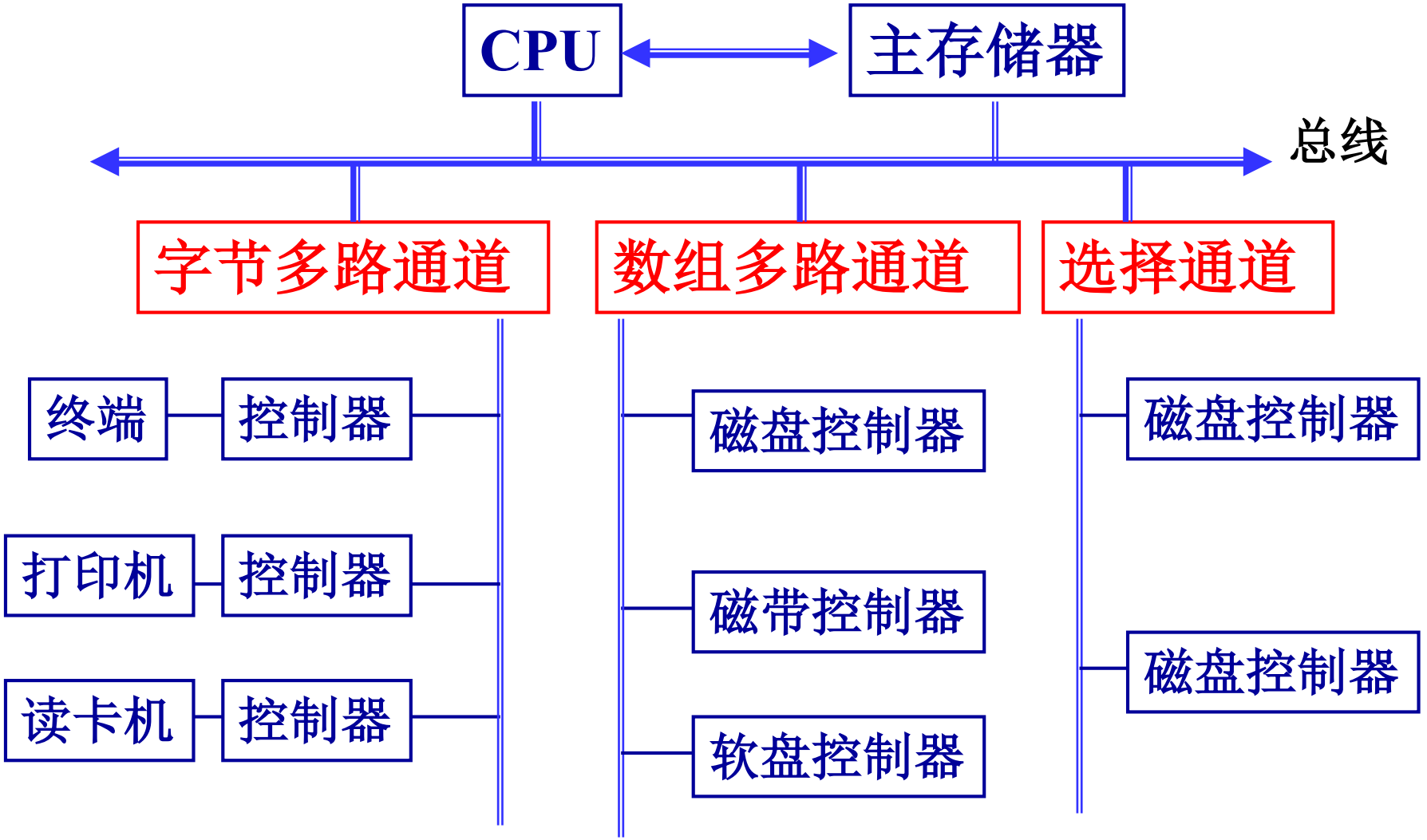
➢不过，DMA方式仍存在一定局限性。如数据传送方向、存放数据的内存起始地址及传送数据的长度等都由CPU控制，并且每台设备需一个DMA控制器，当设备增加时，多个DMA控制器的使用也不经济。

4. 通道控制方式

- 与DMA方式相比，通道所需的CPU干预更少，且可以做到一个通道控制多台设备，进一步减轻了CPU的负担。
- 通道是一种专用的I/O处理机。
- 通道有自己的指令系统，若干条通道命令连接成通道程序。

- 它接收CPU的委托，独立地执行自己的通道程序，管理和控制输入输出设备，实现外围设备与主存贮器之间的成批数据传送。当CPU委托的I/O任务完成后，通道发出中断信号，请求CPU处理。
- 这一方面使CPU摆脱了繁琐的输入输出控制工作，大大提高了CPU与外围设备工作的并行程度，另一方面多总线控制多通道之间也实现了并行操作，提高了整个系统的处理效率。

- 通道是一个专管输入输出控制的处理机，它控制设备与内存直接进行数据交换。它有自己的通道指令，这些通道指令受CPU启动，并在操作结束时向CPU发中断信号。
- 在通道控制方式下，CPU只需发出启动指令，指出通道相应的操作和I/O设备，该指令就可启动通道，并使该通道从内存中调出相应的通道指令执行。



CPU、通道和I/O设备并行工作

- 一个通道可以以分时方式同时执行几个通道指令程序。按照信息的交换方式和控制设备的种类，通道可以分为三种类型：

字节多路通道、选择通道和数组多路通道。

通道的三种类型

- 字节多路通道：以字节为单位传输信息，可以分时地执行多个通道程序，一个通道程序对应一台设备。主要用来连接大量慢速设备。
- 当一个通道程序控制某台设备传送一个字节后，通道硬件就转去执行另一个通道程序，控制另一台设备传送一个字节的信息。
- 如纸带输入 / 出机，卡片输入 / 出机、打印机、终端等。

- 选择通道：每次传送一批数据，传送速度快。在一段时间内只能执行一个通道程序，只允许一台设备传输数据。可用于固定头磁盘等。
- 数组多路通道：结合了选择通道传送速度快和字节多路通道能够分时的优点。先为一台设备执行一条通道指令，再为另一台设备执行一条通道指令。可连接多台活动头磁盘机。
- 它可以启动它们同时执行移臂定位操作，然后，按序交叉地传输一批批数据。数据多路通道实际上是对通道程序采用多道程序设计的硬件实现。

工作过程：

① CPU向通道发出一条I/O指令，给出所要执行的通道程序的首地址和要访问的I/O设备。

② 通道执行通道程序便可完成CPU指定的I/O任务。

③ 完成任务后，通道与设备一起发出中断请求，请求CPU处理。

- **CPU执行用户程序，在遇到I/O请求时，可根据该请求命令生成通道程序并放入内存，之后把该通道程序的首地址放入CAW中；之后执行“启动I/O”指令，启动通道工作。**

通道启动成功后， CPU可继续执行其他程序，通道按照CAW从内存中依次取出通道命令并放入CCW中，通道开始执行通道程序。按I/O地址启动设备控制器，同时将读写命令也传送给它。设备开始在通道控制下进行数据传输。

通道控制设备传输完成后，向CPU提出中断请求，CPU响应中断后，转入中断处理。首先检查是否正确完成。若是，再检查是否有等待通道和设备传输的任务，若有，再次启动通道和设备进行传输。

6.2 I/O软件的组成

I/O软件的基本思想：按分层构建，较低层的软件为较高层的软件服务，使较高层软件独立于硬件，为用户提供统一接口。

6.2.1 I/O软件的目标

1. 设备独立性。用户程序中给出的设备名只是一个逻辑设备名，由OS实现逻辑设备与物理设备的映射。这样，无论系统设备如何改变，用户程序不受影响。
2. 设备的统一命名。与设备独立性密切相关。一个设备的逻辑名只应是一个简单的字符串或一个整数，如 PRN，不依赖于具体的设备。

3. 出错处理。数据传输中的错误应尽可能地在接近硬件层上处理，可重试多次。仅当低层软件无能为力时，才将错误上交高层处理。
4. 缓冲技术。其目的就是设法使数据的到达率和离去率相匹配，以提高系统的吞吐量。
5. 设备的分配。涉及到共享设备（磁盘）和独占设备（打印机）的分配问题。

- 设备分配
- 根据设备各自的属性，系统制定分配策略。
- 设备分配可以采用静态分配和动态分配。静态分配简单，但设备利用率低。动态分配设备利用率高，但容易引起死锁。

- 设备分配是设备管理的功能之一，当进程向系统提出I/O请求之后，设备分配程序将按一定的分配算法为其分配所需的设备。

- 同时还要分配相应的控制器和通道，以保证CPU与设备之间的通信。

设备分配用的数据结构

- 为了实现对I/O设备的管理和控制，需要对每台设备、通道控制器的情况进行登记。设备分配依据的主要数据结构有设备控制表（DCT）、控制器表（COCT）、通道控制表（CHCT）和系统设备表（SDT）。

- 系统为每一个设备配置一张控制表，用于记录设备的特性及与I/O控制器连接的情况。设备控制表中包括设备标识符、设备类型、设备状态、设备等待队列指针、I/O控制器指针等。

➢其中：

设备状态用来指示设备是忙还是闲，设备等待队列指针指向等待使用该设备的进程组成的等待队列，I/O控制器指针指向与该设备相连接的I/O控制器。

➢控制器表也是每个控制器一张，它反映I/O控制器的使用状态以及和通道的连接情况等。

➢每个通道都配有一张通道控制表。它包括通道标识符、通道状态、等待获得该通道的进程等待队列指针等。

- 系统设备表是整个系统一张，它记录已被连接到系统中的所有物理设备的情况，每个物理设备占有一个表目。

- 系统设备表的每个表目包括设备类型、设备标识符、设备控制表的指针等。其中，设备控制表指针指向设备对应的设备控制表。

设备分配的原则

- 在一个系统中，请求设备为其服务的进程数往往多于设备数，这样就出现了多个进程对某类设备的竞争问题。

- 为了保证系统有条不紊地工作，系统在进行设备分配时，应考虑以下几个因素：

- **设备分配的原则**是根据设备特性、用户要求和系统配置情况决定的。

- **设备分配的总原则**是既要充分发挥设备的使用效率，尽可能的让设备忙，但又要避免由于不合理的分配方法造成进程死锁；另外，还要做到把用户程序和具体物理设备隔离开来。

设备分配策略

- I/O设备的分配，除了与I/O设备的固有属性相关外，还与系统所采用的分配算法有关。设备分配主要采用先请求先分配和优先级高者先分配两种算法。

1.先请求先分配

- 当有多个进程对同一设备提出I/O请求时，该算法根据这些进程发出请求的先后次序，将这些进程排成一个设备请求队列，设备分配程序总是把设备首先分配给队首进程。

2.优先级高者先分配

- 按照进程优先级的高低进行设备分配。当多个进程对同一设备提出I/O请求时，哪一个进程的优先级高，就先满足哪个进程的请求，将设备分配给这个进程。

- 对优先级相同的I/O请求，则按先请求先服务的算法排列。

静态分配

- 静态分配是在作业级进行的，用户作业在开始执行前，由系统一次分配该作业所要求的全部设备、控制器和通道。一旦分配之后，这些设备、控制器和通道就一直为该作业所占用，直到该作业被撤消为止。静态分配方式不会出现死锁，但设备的利用率低。

动态分配

- 动态分配是在进程执行过程中根据执行需要进行的设备分配。当进程需要设备时，通过系统调用命令向系统提出设备请求，由系统按照事先规定的策略给进程分配所需要的设备、控制器和通道，一旦用完之后，便立即释放。

- 该动态方式有利于提高设备的利用率，但如果分配算法使用不当，则有可能造成死锁。

设备分配的步骤

- 当某个进程提出I/O请求后，系统的设备分配程序可按如下步骤进行设备分配：

（1）分配设备

- 根据进程提出的物理设备名查找设备表，从中找到该设备的设备控制表。查找设备控制表中的设备状态字段。

- 若该设备处于忙状态，则将进程插入设备等待队列；

- 若该设备处于空闲状态，便按照一定的算法来计算本次设备分配的安全性（即是否会导致死锁）。

- 若分配不会引起死锁则进行分配；否则仍将该进程插入设备等待队列。

（2）分配控制器

- 在系统把设备分配给请求I/O的进程后，再到设备控制表中找到与该设备相连的控制器的控制表，从该表的状态字段中可知该控制器是否忙碌。

- 若控制器忙，则将进程插入等待该控制器的队列；否则将该控制器分配给进程。

（3）分配通道

- 从控制器表中找到与该控制器连接的通道控制表，从该表的状态字段中可知该通道是否忙碌。

- 若通道处于忙状态，则将进程插入等待该进程的队列；否则将该通道分配给进程。

- **此时，进程本次I/O请求所需要的设备、控制器、通道均已分配，可由设备处理程序去实现真正的I/O操作。**

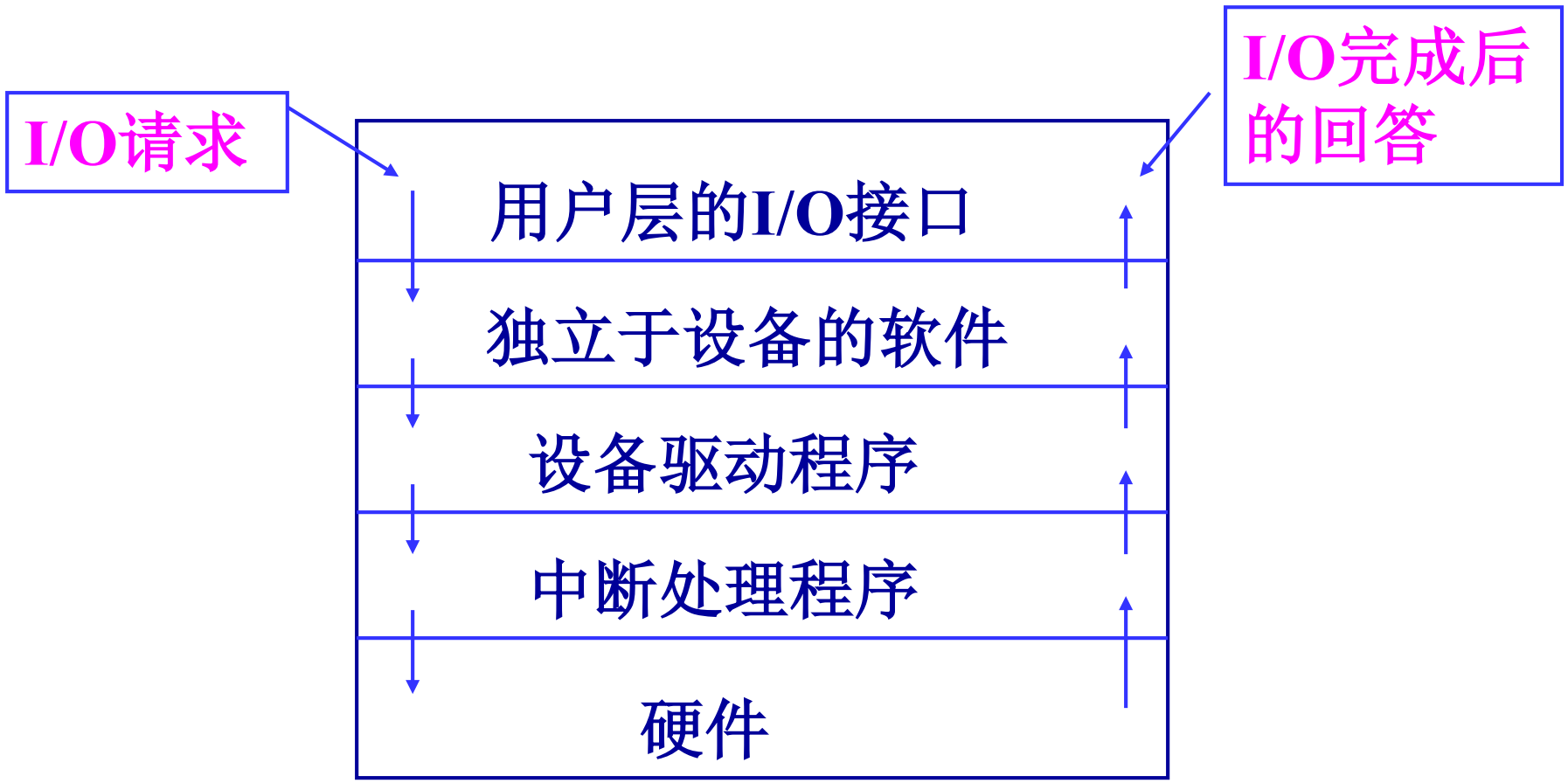
6.2.2 I/O软件的功能

I/O软件的分层：

1. 中断处理程序
2. 设备驱动程序
3. 独立于设备的软件
4. 用户层的I/O接口

在I/O软件中，大部分软件是与设备无关的。

I/O系统的层次结构



- 图中的箭头给出了I/O部分的控制流。现总结如下：
- (1) 用户进程层执行输入输出系统调用，对I/O数据进行格式化，为假脱机输入 / 出作准备；
- (2) 独立于设备的软件实现设备的命名、设备的保护、成块处理、缓冲技术和设备分配；

- (3)设备驱动程序设置设备寄存器、检查设备的执行状态；
- (4)中断处理程序负责I/O完成时进行中断处理，唤醒设备驱动程序进程；
- (5) 硬件层实现物理I/O的操作。

1. 中断处理程序

① 进程在启动一个I/O操作后阻塞起来，I/O操作完成，控制器产生一个中断。

② CPU响应中断，执行中断处理程序。

③ 检查设备状态。

a) 若正常完成，就唤醒等待的进程。然后检查是否还有待处理的I/O请求，若有就启动。

b) 若传输出错，再发启动命令重新传输；或向上层报告“设备错误”的信息。

④ 中断返回被中断的进程，或转进程调度。

2. 设备驱动程序

- 每个设备驱动程序处理一种类型设备。由一些与设备密切相关的代码组成。提供一些与文件类似的API：open, close, read, write, control等。
- 是OS中唯一知道设备控制器的配置情况，如设置有多少个寄存器以及这些寄存器作用。
- 通常包含三部分功能：①设备初始化。②启动设备传输数据的例程。③中断处理例程。

- 设备驱动程序是驱动物理设备和DMA控制器或I/O控制器等直接进行I/O操作的子程序的集合。

- 它负责设置相应设备有关R的值，启动设备进行I/O操作。

系统开关表DST

- 系统中设置DST，是为了更好地对驱动程序进行管理。
- DST中给出了相应设备的各种操作子程度的入口地址，如打开、关闭、启动设备子程序的入口地址。
- 一般来说，它是二维结构，其中的行和列分别为设备类型和驱动程序类型。
- 它也是I/O进程的一个数据结构。I/O控制过程为进程分配设备和缓冲区之后，可以使用DST调用所需的驱动程序进行I/O操作。

设备驱动程序工作过程

1. 接收来自上层软件的抽象请求，如读磁盘的第几块，并执行请求。若忙，则排到I/O请求队列中。
2. 将请求转换成应向控制器发送的命令和参数。
3. 通常，驱动程序进程阻塞等待命令完成，直到中断处理时将其唤醒。有时不必等待，如滚屏操作，把几个字节写到控制器中即可。
4. 检查数据传输是否有错；向上层传送数据。
5. 继续未完成的I/O请求。

3. 独立于设备的软件

（1）基本任务：实现所有设备都需要的功能，且向用户提供一个统一的接口。

（2）设备命名。把设备的符号名映射到正确的设备驱动程序。

UNIX，/dev/tty01→i节点→主设备号（用来确定终端设备驱动程序），次设备号（作为参数用来确定要读/写的是哪一台终端）。

（3）设备保护。防止无权存取设备的用户使用设备。 UNIX的I/O设备作为文件用“rw”位进行保护。禁止用户对I/O设备直接访问，必须通过OS提供的系统调用命令进行I/O操作。

（4）提供与设备无关的块尺寸。应向上层软件提供大小统一的块尺寸。上层软件只与抽象设备打交道，使用等长的逻辑块。

（5）缓冲技术

- 缓和CPU与I/O设备间速度不匹配的矛盾，减少对CPU的中断次数。
- 块设备一次读写一块，用户按任意单位处理数据。
- 以空间换取时间

1. 单缓冲：OS为I/O请求分配一个缓冲区。
2. 双缓冲：建立两个缓冲区，可以平滑I/O设备和进程之间的数据流，改善系统效率。
3. 多缓冲和缓冲池：多进程共享缓冲池。

高速缓存

- 缓冲只保留数据仅有的一个现存拷贝。
- 有时一块内存区域可以同时用于两个目的。

  例如，为了有效调度磁盘I/O，在内存开辟了缓冲区来保留磁盘数据。这些缓冲区也可以用作高速缓存，可被多个进程共享。当内核收到I/O请求时，会首先检查高速缓存里是否有。

（6）负责设备分配和调度

- 静态分配：进程运行前，将需要的设备全部分配给它。简单，不死锁，但利用率低。
- 动态分配：在进程运行过程中，分配设备。设备利用率高，但易引起死锁。

- 独占设备：临界资源，如打印机。
- 共享设备：多个进程可交叉访问。如磁盘。
- 虚拟设备：是指设备本身是独占设备，而经过虚拟技术处理，可以把它改造成共享设备，供多个进程同时使用。

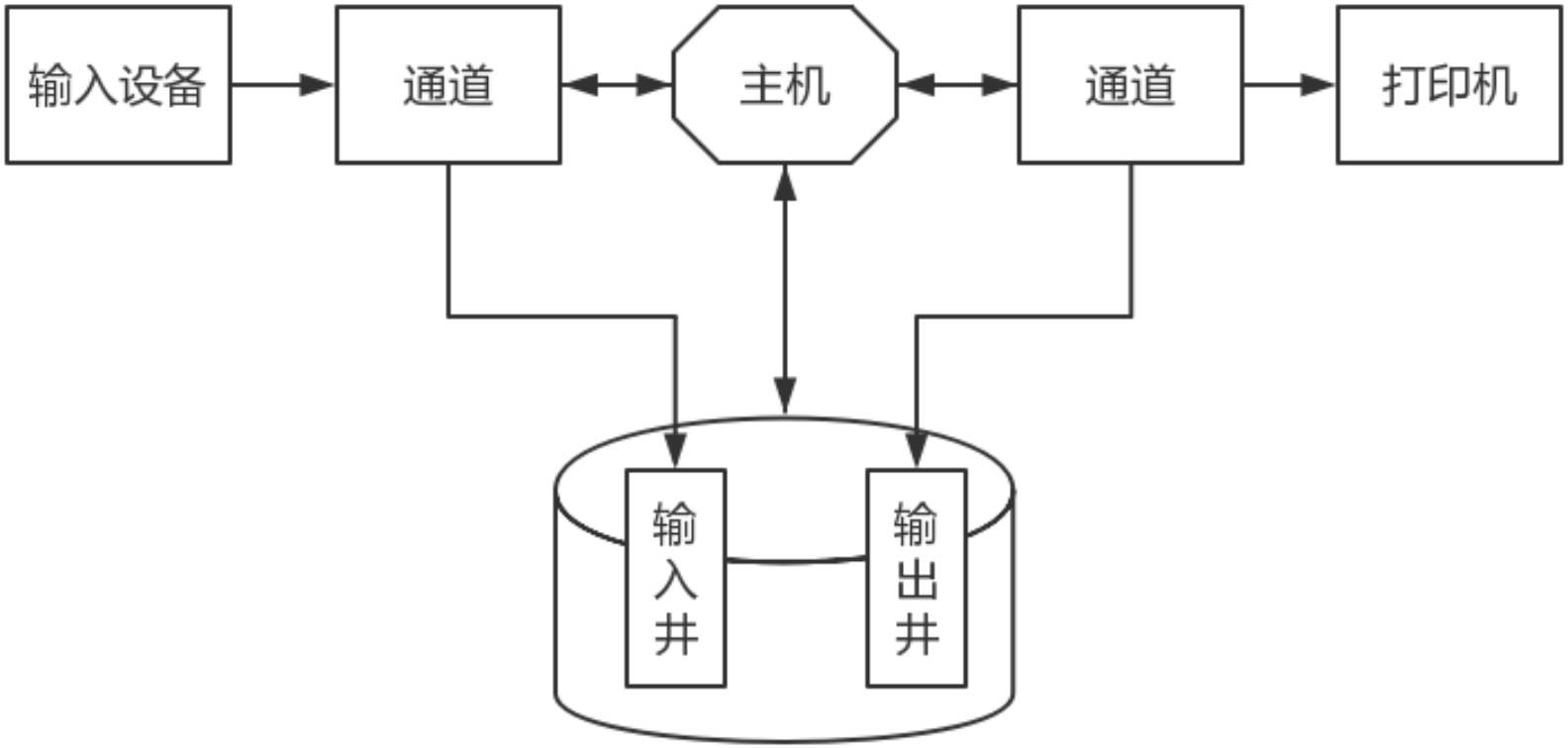
  常用可共享的高速设备来模拟独占的慢速设备。能有效提高独占型设备的利用率。

  Spooling技术是实现虚拟设备的具体技术。它利用可共享磁盘的一部分空间，模拟独占的输入/输出设备。以空间换时间

假脱机输出：以打印机为例

Spooling实际是一种缓冲技术。进程要打印时，系统并不为它分配打印机，而是把待打印的数据缓冲到一个独立的磁盘文件上，形成待打印文件队列。之后，Spooling系统一次一个地将打印队列上的文件送打印机打印。这种技术又叫缓输出技术。

Spooling系统



（7）出错处理

- 绝大多数错误是与设备密切相关的，一般由设备驱动程序来处理。
- 处理设备驱动程序处理不了的错误（重试几次操作后，仍有错误）。将错误信息报告调用者。

4. 用户空间的I/O软件

- 大部分I/O软件都包含在操作系统中，有一小部分是由与用户程序连接在一起的库函数构成的。

[例] 用户程序中的库函数： count=read(fd, buffer, nbytes); 程序运行期间，库函数read将与该程序连接在一起形成一个可执行文件装入主存。

这些函数通常只是将系统调用时所需要的参数放在合适的位置，由系统调用实现真正的操作。如“Printf”将调用“write”系统调用。

读文件的I/O操作步骤

1) 用户进程发出一个读文件的系统调用。
2) 设备独立I/O软件检查参数的正确性。若正确，再检查高速缓存中有无要读的信息块。若有，则从缓冲区直接读到用户区。若无，转3)
3) 执行物理I/O。独立于设备的I/O软件将设备的逻辑名转换成物理名，检查设备操作权限。将I/O请求排队，阻塞用户进程且等待I/O完成。
4) 核心执行设备驱动程序，分配缓冲区，准备接收数据，且向设备控制寄存器发启动读命令。

5) 设备控制器控制设备，执行数据传输。

6) 当采用DMA控制器控制传输时，一个块传输完成，硬件产生一个中断。

7) CPU响应中断，转磁盘的中断处理程序。检查中断原因和设备的执行状态，若出错，则向设备驱动程序发信号，若可重试，则再启动设备重传一次；否则，向上报告错误。若传输正确，将数据传输给指定的用户进程空间，将等待进程唤醒并且放入就绪队列，等待调度。

8) 当用户进程被调度执行时，从I/O系统调用的断点恢复执行。

6.2.3 同步I/O和异步I/O

- 同步I/O：进程发出I/O请求后阻塞等待，直到数据传输完成后被唤醒，之后才能访问被传输的数据。
- 异步I/O：允许进程发出I/O请求后继续运行。将来I/O完成后的通知方式：设置进程地址空间内的某个变量；通过触发信号或软件中断；进程执行流之外的某个回调函数。（Windows的APC）
- 对于不必进行缓冲读写的快速I/O，使用同步I/O更有效；对于需要很长时间的I/O操作，可使用异步I/O。

6.3 磁盘管理

1. 提高磁盘I/O速度的主要途径

- 选择性能好的磁盘。如 IDE、SCSI
- 采用好的调度算法
- 设置磁盘高速缓冲区

2. 磁盘的类型

硬盘和软盘；固定头磁盘和活动头磁盘。

- 固定头磁盘：每条道上都有一个读/写磁头，用于大容量磁盘，并行读/写。
- 移动头磁盘：每个盘面仅配有一个磁头。

3. 访问磁盘块的时间：寻道时间、旋转延迟时间、读/写传输时间。

4. 磁盘分配方法

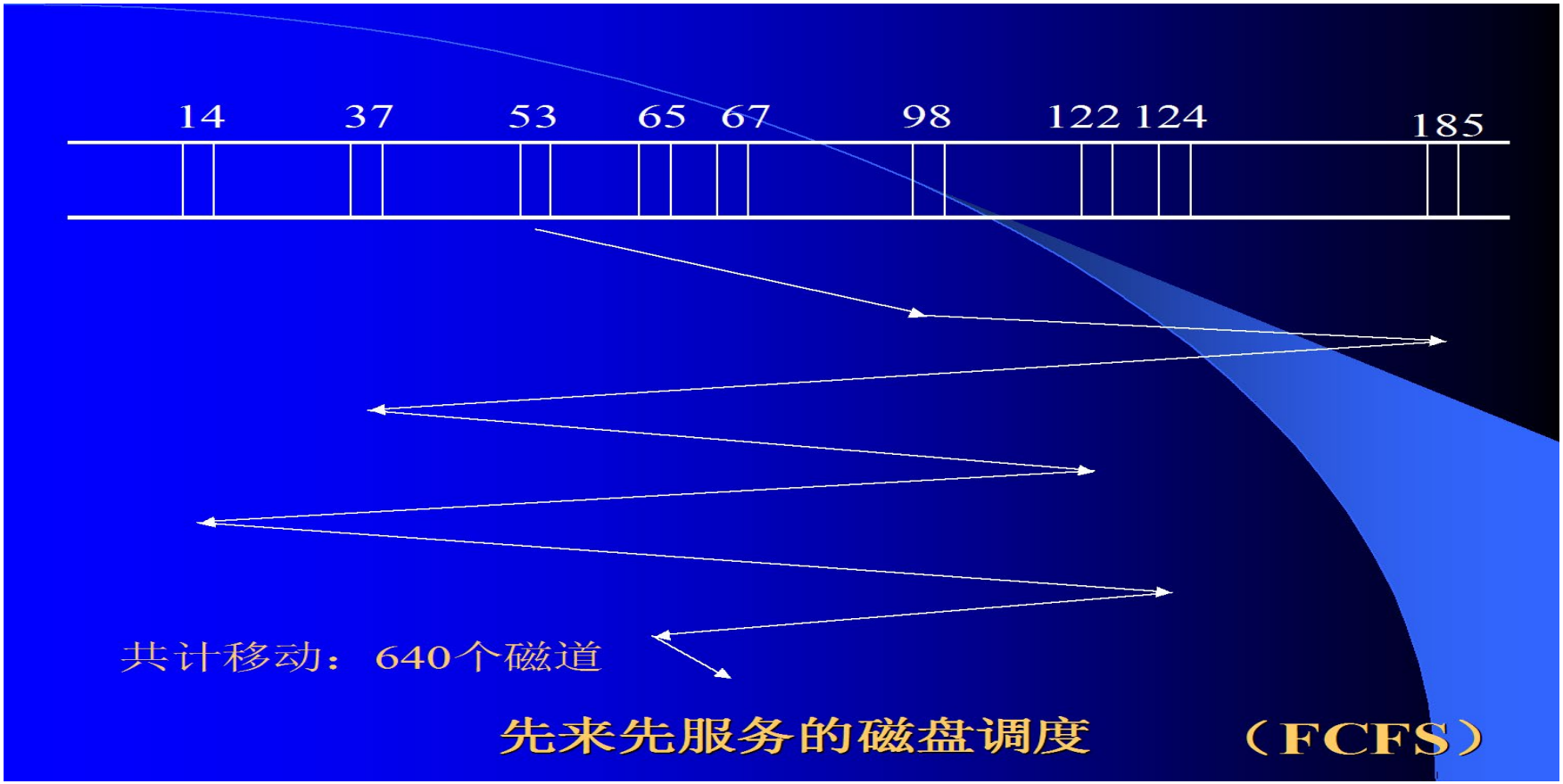
连续分配；链接分配；索引分配

5. 调度算法

- 先来先服务：最简单，易实现，又公平合理。
- 最短寻道时间优先：在将磁头移向下一请求磁道时，总是选择移动距离最小的磁道。
- 扫描法：读/写磁头在由磁盘的一端向另一端移动时，随时处理所到达磁道上的服务请求。
  - ① SCAN：扫描（电梯法）
  - ② C-SCAN：循环扫描
  - ③ LOOK：查询
  - ④ C-LOOK：循环查询

- 例如，有如下的一个磁盘请求序列，其磁道号为
- 98，185，37，122，14，124，65，67
- 假定一开始读/写磁头位于53号磁道。为了满足这一系列请求，磁头要先从53移到98，然后再移到185，37，122…，最后到达67。这样，磁头总共要移动644个磁道。
- 这种调度方式的缺点是完全不考虑队列中各个请求情况。致使磁头频繁地移动，导致最大移动距离。

如果在磁头移到37时，能与14号磁道的请求一起服务，那么就能够省下很多移动时间，从而缩短了每个一请求的处理时间，改善了磁盘的吞吐量。



- 最短寻道时间优先算法SSTF (Shortest-Seek-Time-First)
- 最短寻道时间优先是指在将磁头移向下一请求磁道时，总是选择移动距离最小磁道的一种方法。
- 在上述的等待队列中，采用SSTF算法时，最接近磁头所在位置(53)的请求是第65号磁道，一旦磁头移到65，下一个最接近的是67号磁道，之后至37；

- 接下去服务的顺序是14，98，122，124，最后是185。这种算法使得磁头移动距离的总和只有238个磁道，是FCFS算法的三分之一。

SSTF算法虽然比FCFS算法优越，但它不是最优的。例如，如果将磁头由53号先移至37号(即使它不是最近的)，再移到14号，然后再移到65，67，98，122，124及185号，这样可将总的移动距离减少到210个磁道。一方面使磁头改变方向最少，另一方面大大加快了服务请求。提高了系统处理效率。

扫描法(SCAN)

- SCAN算法是指读／写头在开始由磁盘的一端向另一端移动时，随时处理所到达的任何磁道上的服务请求，直到移到磁盘的另一端为止。
- 在磁盘另一端上，磁头的方向反转，继续完成各磁道上的服务请求。这样，磁头总是连续不断地从磁盘的一端移到另一端。

- 在使用SCAN之前，不仅要知道磁头移动的最后位置，而且还需要知道磁头移动的方向。
- 仍以上面的请求序列为例，如果此时磁头向0号磁道移动，那么当磁头向0号移动时，则先服务37号和14号磁道上的请求，再移至0道。在0道上，磁头反向，并在将磁头移向磁盘另一端时，服务65，67，98，122，124及185号磁道上的请求。

- 如果正好有一个请求在磁头前进方向上到达，那么，这个请求将会立即得到处理。然而，如果一个请求在磁头刚刚移动过后到达，那么它只能等到磁头反方向移到时才能得到处理。
- 扫描法有时叫电梯算法(elevator)。因为这种方式与电梯在各楼层间的往返移动非常类似。

- 假定对磁道的请求是均匀分布的，考虑对磁道的请求密度，当磁头达到一端并反向时，落在磁头之后的请求相对较少，这是由于这些磁道刚刚被处理，而磁盘另一端的请求密度相当地高，且等待的时间较长。
- 为了解决这种情况，引入了循环扫描法C-SCAN(circular-SCAN)算法，以提供比较均衡的等待时间。循环扫描法与SCAN算法类似，也是在将磁头从一端移向另一端时，随时处理到达的请求。但是，当它已到达另一端时，磁头立即返回到开始处。也即回程时，不处理任何请求。

- 循环扫描法将磁盘视为一个圆，最后一个磁道与第一个磁道紧密相接，以此达到对磁道上请求的均衡服务。请求队列中的磁道为98，185，37，122，14，123，65，67，磁头当前位置为53。显然它的服务顺序为65，67，98，122，123，185，14，37。
- 扫描法和循环扫描法都是将磁头由磁盘的一端移向另一端，但实际上没有一个算法是这样实现的。通常，磁头在向任何方向移动时都是只移到最远的一个请求的磁道上。一旦在前进的方向上没有请求到达，磁头就反向移动。

这种方式的扫描及循环扫描称之为查询(LOOK)及循环查询(C-LOOK)，即在将磁头向前移动之前，先查询有无请求，若有，才移动，否则，立即反向。

磁盘调度算法的比较

- 磁盘调度算法很多，如何选择一个有效算法呢?SSTF算法是公认的、最具吸引力的算法， SCAN和C-SCAN对于磁盘负载较重的系统更为合适。然而，任何调度算法性能优劣都是与进程对磁盘的请求数量和方式紧密相关的。当磁盘等待队列中的请求数量很少超过一个时，所有的算法都是等效的。在这种情况下，最好采用FCFS算法。

值得注意的是，文件的分配方法将大大地影响对磁盘的服务请求。当一个程序读磁盘上的一个大的连续文件时，尽管请求读盘的要求很多，但由于各信息块连在一起，磁头的移动距离却很小。若程序读的是一个链接文件或索引文件，尽管这种文件的磁盘空间利用充分，但可能使信息块分布在整个盘上，导致磁头的可观的移动代价。

- 目录及索引块的位置也是重要的。因为每个文件使用前必须打开，在打开一个文件时，必须检查目录结构，此时要频繁地存取目录文件。为此将目录文件放在磁盘的中间，而不是两端，就可以有效地减少磁头的移动。
- 由于上述的考虑，像其它算法一样，应将磁盘调度算法写成一个独立的模块，以便必要时用不同的算法来替换或干脆去掉不用。为了简单，开始时可选用**FCFS**算法或**SSTF**算法。

45. （7分）假设计算机系统采用C-LOOK（循环查询）磁盘调度策略，使用2KB的内存空间记录16384个磁盘块的空闲状态。

（1）请说明在上述条件如何进行磁盘块空闲状态的管理。

（2）设某单面磁盘的旋转速度为每分钟6 000转，每个磁道有100个扇区，相邻磁道间的平均移动时间为1ms。若在某时刻，磁头位于100号磁道处，并沿着磁道号增大的方向移动，磁道号的请求队列为50、90、30、120，对请求队列中的每个磁道需读取1个随机分布的扇区，则读完这4个扇区总共需要多少时间？需要给出计算过程。

（3）如果将磁盘替换为随机访问的Flash半导体存储器（如U盘、SSD盘等），是否有比CSCAN更高效的磁盘调度策略？若有，给出磁盘调度策略的名称并说明理由；若无，说明理由。

参考答案

（1）2KB=2×1024×8bit=16384 bit。因此可使用位示图来管理磁盘块，位为0表示磁盘块空闲，为1表示被占用。

（3）采用FCFS（先来先服务）调度策略更高效。因为FLASH半导体存储器的物理结构不考虑寻道时间和旋转延迟，可直接按I/O请求的先后顺序服务。

（2）循环查询C-LOOK算法。被请求的磁道号顺序为100、120、30、50、90，因此，寻道需要移动的磁道数为：20+90+20+40=170。寻道用去的总时间为：

(20 + 90 + 20 + 40)×1ms = 170ms

磁盘每分钟6000转，转一圈的时间为0.01s，通过一个扇区的时间为0.0001s。

总共要随机读取四个扇区，用去的时间为：

(0.01×0.5 + 0.0001)×4 = 0.0204s = 20.4ms

所以 170ms + 20.4ms = 190.4ms。