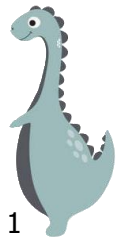
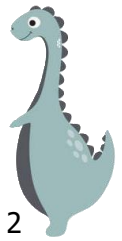
第9章 设备管理

- 本章内容
- 1. 设备管理的基本概念
- 2. 大容量存储器
- 3. I/O系统的层次架构
- 4. 设备驱动程序
- 5. 中断处理程序
- 6. 设备独立内核软件



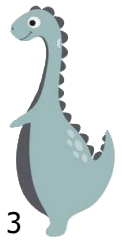
要解决的问题

- 为何设备的差异性那么大？
- 为何要管理设备？
- 如何统一对设备的访问接口？
- 为何要对设备建立抽象？
- 如何感知设备的状态并管理设备？
- 如何提高 CPU 与设备的访问性能？
- 如何保证 I/O 操作的可靠性？

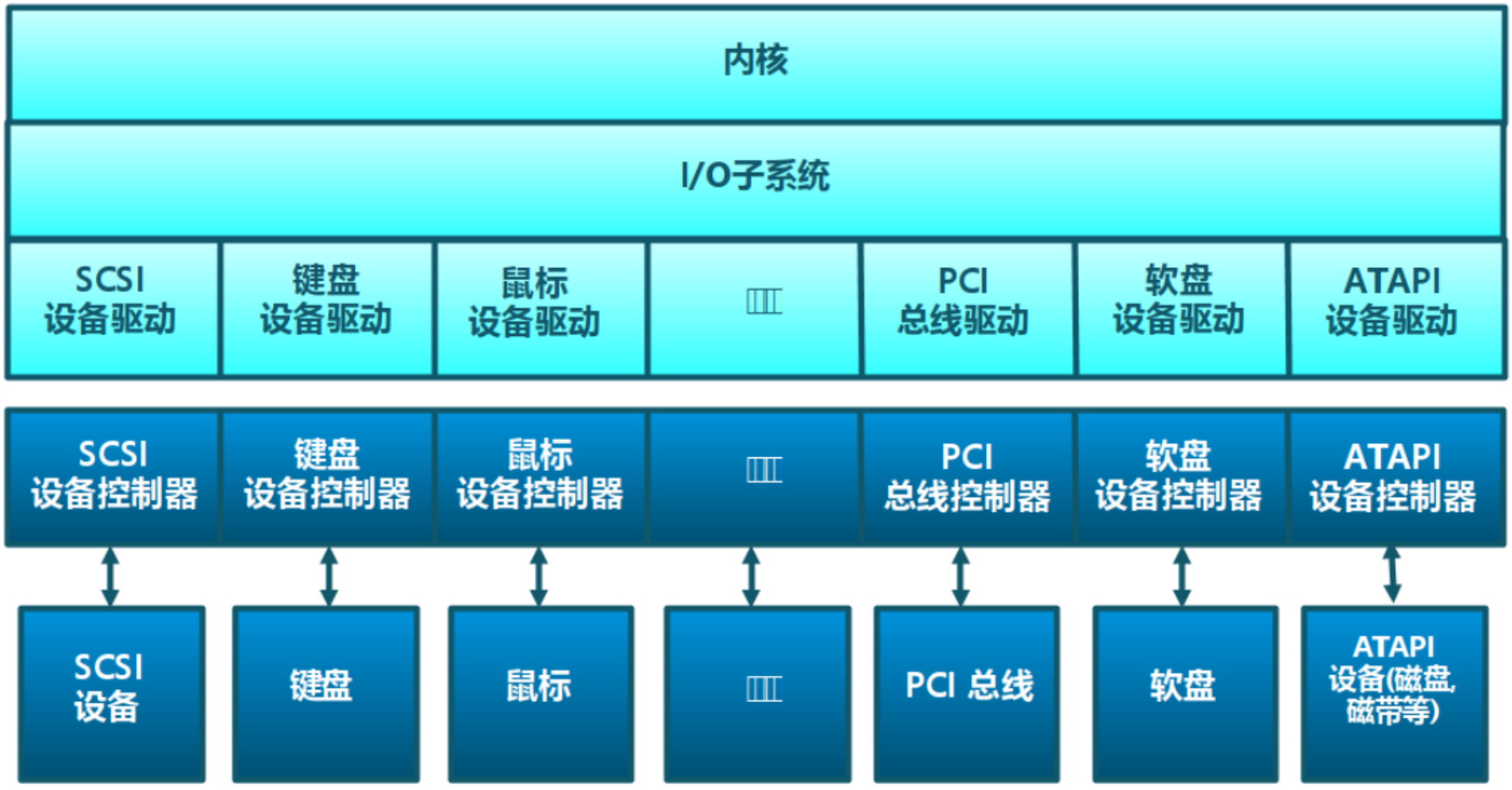


设备管理的任务和功能

- 设备管理的主要任务是：
  - 完成用户提出的I/O请求
  - 分配I/O设备
  - 提高I/O设备的利用率
  - 方便用户使用I/O设备
- 设备管理的功能
  - 设备分配：根据用户的I/O请求，为之分配设备。包含控制器和通道。
  - 设备处理：负责启动设备及I/O操作完成时的中断处理。
  - 缓冲管理：为缓和CPU与I/O速度不匹配的矛盾常设置缓冲区。缓冲管理负责缓冲区的分配和释放及有关管理工作。
  - 设备独立性：又称设备无关性，是指用户编制程序时所使用的设备与物理设备无关。

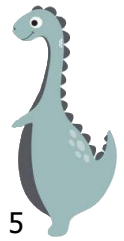


内核I/O结构


内核
I/O子系统
SCSI 设备驱动
键盘 设备驱动
鼠标 设备驱动
PCI 总线驱动
软盘 设备驱动
ATAPI 设备驱动
SCSI 设备控制器
键盘 设备控制器
鼠标 设备控制器
PCI 总线控制器
软盘 设备控制器
ATAPI 设备控制器
SCSI 设备
键盘
鼠标
PCI 总线
软盘
ATAPI 设备(磁盘, 磁带等)

常见设备类型

- 设备的发展历史
  - 简单设备：CPU 可通过 I/O 接口直接控制 I/O 设备
  - 多设备：CPU 与 I/O 设备之间增加了一层 I/O 控制器和总线 BUS
  - 支持中断的设备：提高 CPU 利用率
  - 高吞吐量设备：支持 DMA
  - 各种其他设备：GPU、声卡、智能网卡、RDMA
  - 连接方式：直连、（设备/中断）控制器、总线、分布式



常见设备

- 字符设备：处理信息的基本单位是字符。如键盘、打印机和显示器是字符设备。
- I/O 命令：get( )、put( ) 等
- 通常使用文件访问接口和语义

UART（Universal Asynchronous Receiver/Transmitter） 串口通信

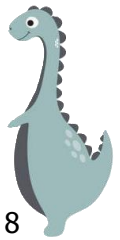
常见设备

- 块设备：处理信息的基本单位是字符块。一般块的大小为512B～4KB，如磁盘、磁带等是块设备。
- I/O 命令：原始 I/O 或文件系统接口、内存映射文件访问。
- 通常使用文件访问接口和语义

常见设备

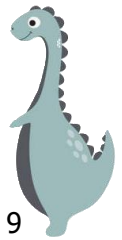
网络设备：

- 格式化报文交换
- I/O 命令：send/receive 网络报文，通过网络接口支持多种网络协议
- 通常使用 socket 访问接口和语义



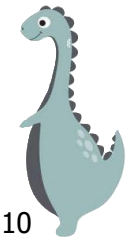
时钟和定时器

- 许多机器都有硬件时钟和定时器以提供如下三个基本函数
  - 获取当前时间
  - 获取已经逝去的时间
  - 设置定时器以在T时触发操作X



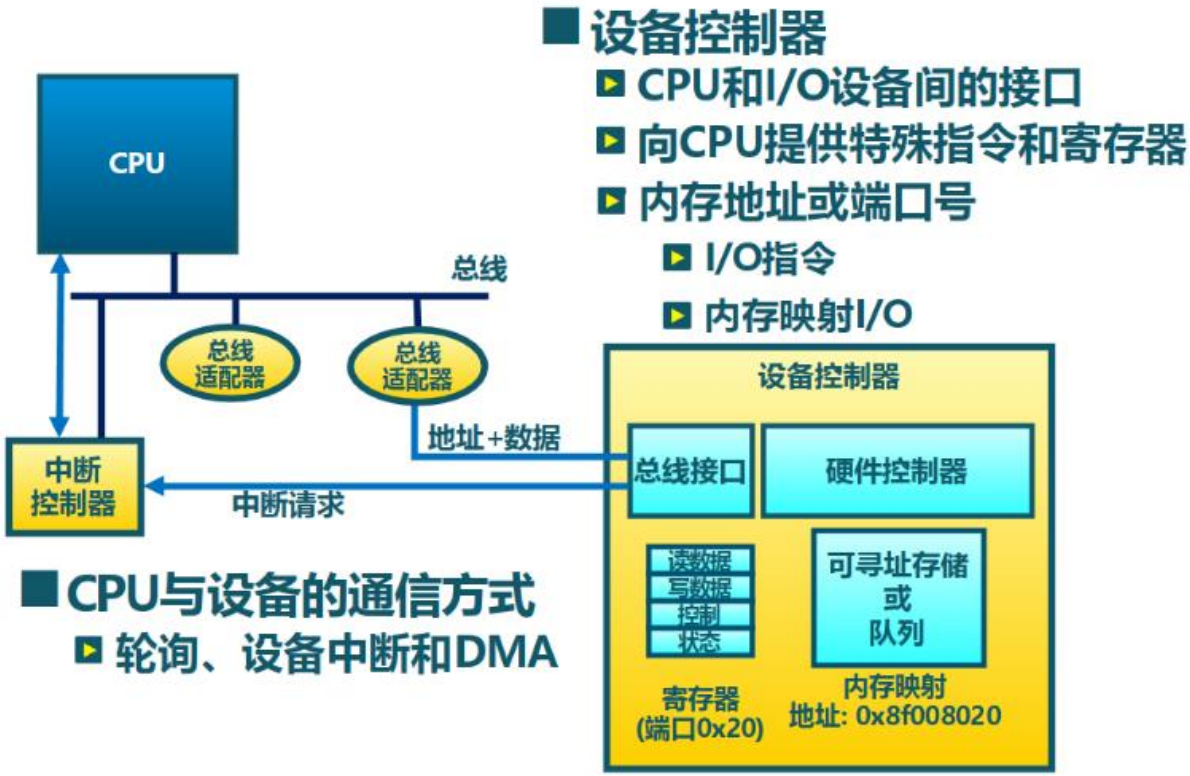
时钟和定时器

- 测量逝去时间和触发器操作的硬件称为可编程间隔定时器
  - 可被设置为等待一定的时间，然后触发中断
  - 也可设置成做一次或重复多次以产生定时中断



设备传输方式

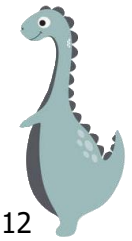
- 程序控制I/O(PIO, Programmed I/O)
- Interrupt based I/O
- 直接内存访问(DMA)



设备传输方式

程序控制 I/O(PIO, Programmed I/O)

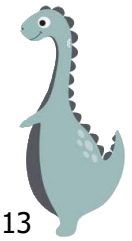
- Port-mapped 的 PIO(PMIO)：通过 CPU 的 in/out 指令
- Memory-mapped 的 PIO(MMIO)：通过 load/store 传输所有数据
- 硬件简单，编程容易
- 消耗的 CPU 时间和数据量成正比
- 适用于简单的、小型的设备 I/O
- I/O 设备通知 CPU：PIO 方式的轮询



设备传输方式

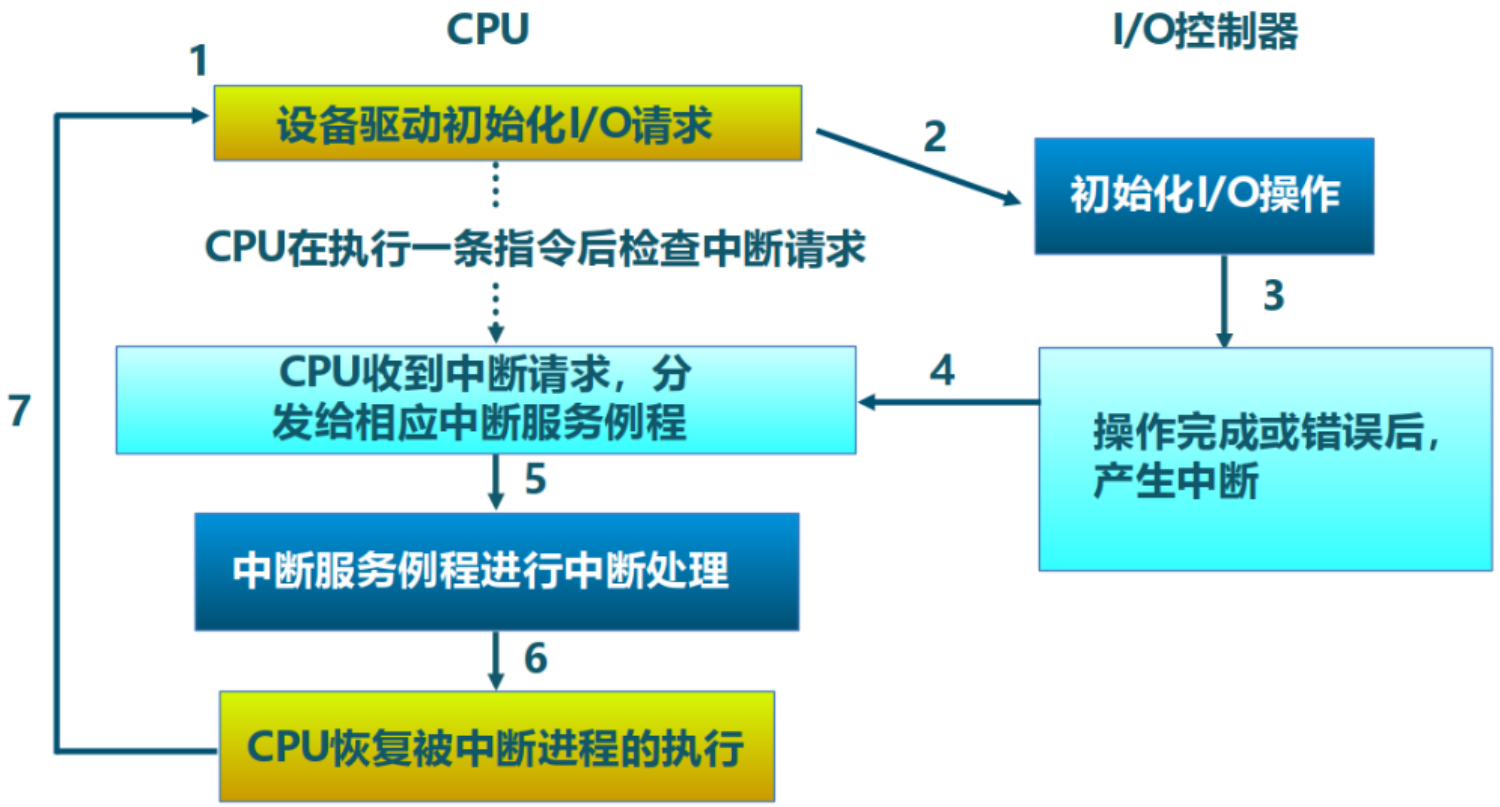
中断传输方式

- I/O 设备想通知 CPU，便会发出中断请求信号
- 可中断的设备和中断类型逐步增加
- 除了需要设置 CPU，还需设置中断控制器
- 编程比较麻烦
- CPU 利用率高
- 适用于比较复杂的 I/O 设备
- I/O 设备通知 CPU：中断方式的提醒



设备传输方式

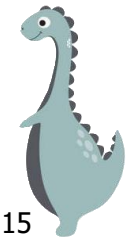
中断传输方式



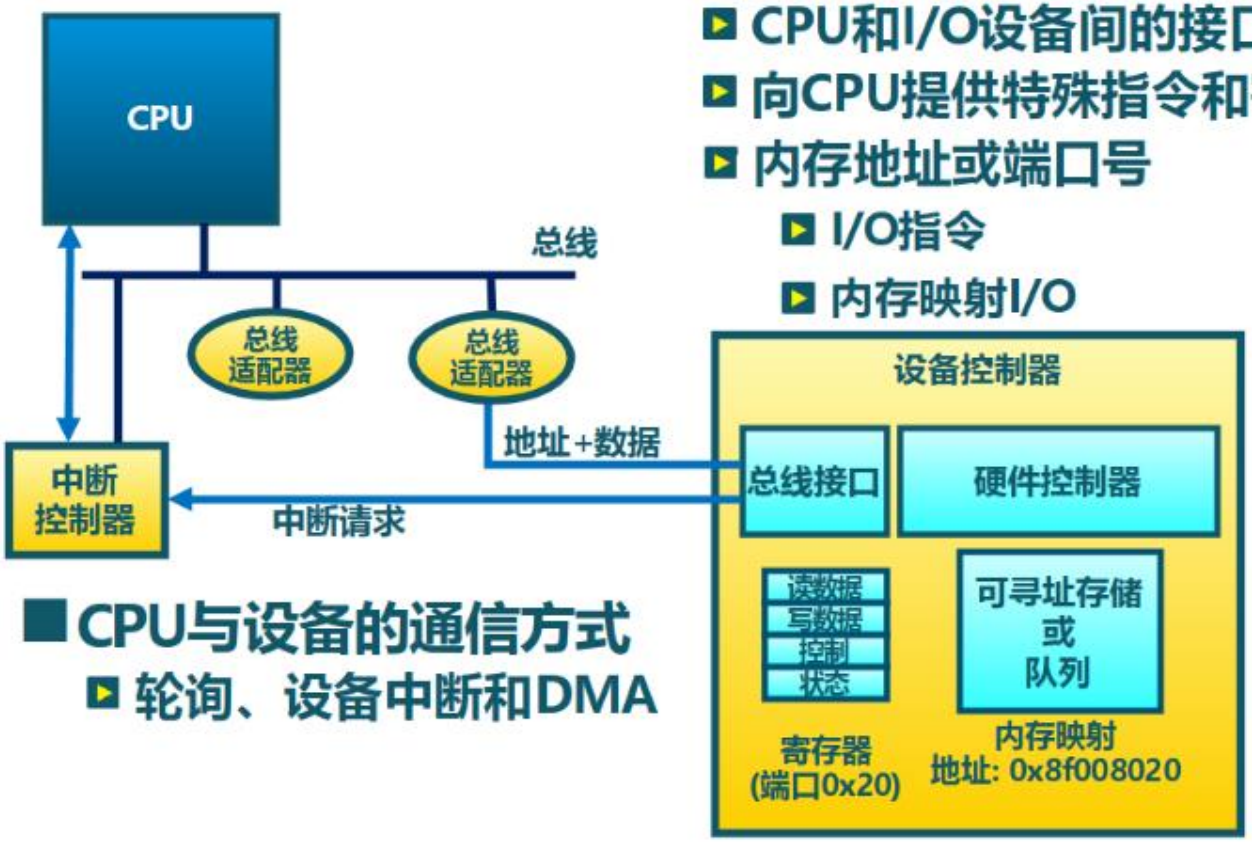
设备传输方式

DMA 传输方式

- 设备控制器可直接访问系统总线
- 控制器直接与内存互相传输数据
- 除了需要设置 CPU，还需设置中断控制器
- 编程比较麻烦，需要 CPU 参与设置
- 设备传输数据不影响 CPU
- 适用于高吞吐量 I/O 设备



CPU与设备的连接

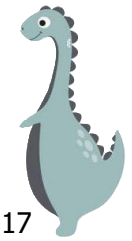


- 设备控制器
  - CPU和I/O设备间的接口
  - 向CPU提供特殊指令和寄存器
  - 内存地址或端口号
    - I/O指令
    - 内存映射I/O

- CPU与设备的通信方式
  - 轮询、设备中断和DMA

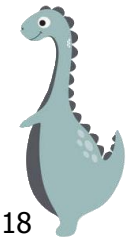
I/O硬件

- 端口：设备与计算机通过端口通信。
- 总线：是一组线和一组严格定义的可以描述在线上传输信息的协议。
- 控制器：用于操作端口、总线或设备的一组电子器件。
- 设备地址，可用于
  - 直接I/O指令 Direct I/O instructions
  - 存储器映射I/O Memory-mapped I/O：设备控制寄存器映射到处理器的地址空间，因此处理器能够象访问普通内存一样访问设备控制寄存器。



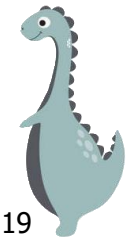
I/O硬件

- I/O端口通常有四种寄存器
  - 状态寄存器 Status registers：包含一些主机可读取的位信息，指示设备的各种状态
  - 控制寄存器 Control registers：可以被主机用来向设备发送命令或改变设备状态。
  - 数据输入寄存器 Data-in registers：可以被主机读取数据
  - 数据输出寄存器 Data-out registers：被主机写入数据以发送数据



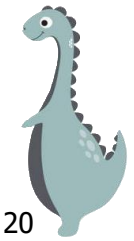
设备控制器

- 设备一般由机械和电子两部分组成，设备的电子部分通常称为**设备控制器**。在微型计算机中，又称为接口卡。
- 控制器处于CPU与I/O设备之间，它接收从CPU发来的命令，并去控制I/O设备工作
- 设备控制器是一个可编址设备，当它控制一个设备时有一个设备地址；当它控制多个设备则应有多个设备地址。



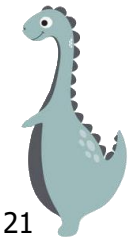
设备控制器的功能

- 接收和识别来自CPU的命令：用控制寄存器。
- 实现CPU与控制器、控制器与设备之间的数据交换：用数据寄存器。
- 记录设备的状态供CPU查询：用状态寄存器。
- 识别控制的每个设备地址：用地址译码器。

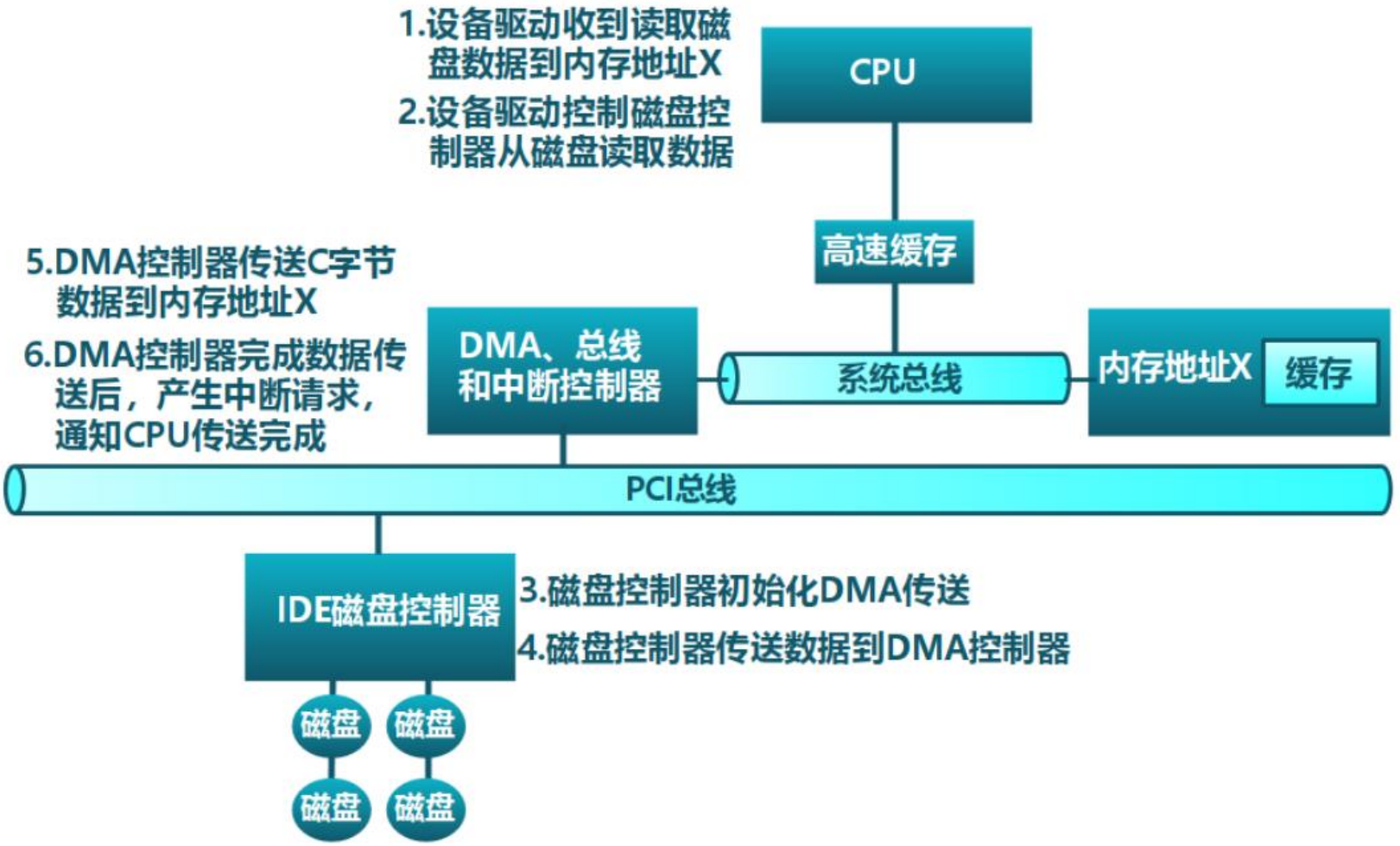


设备控制器的组成

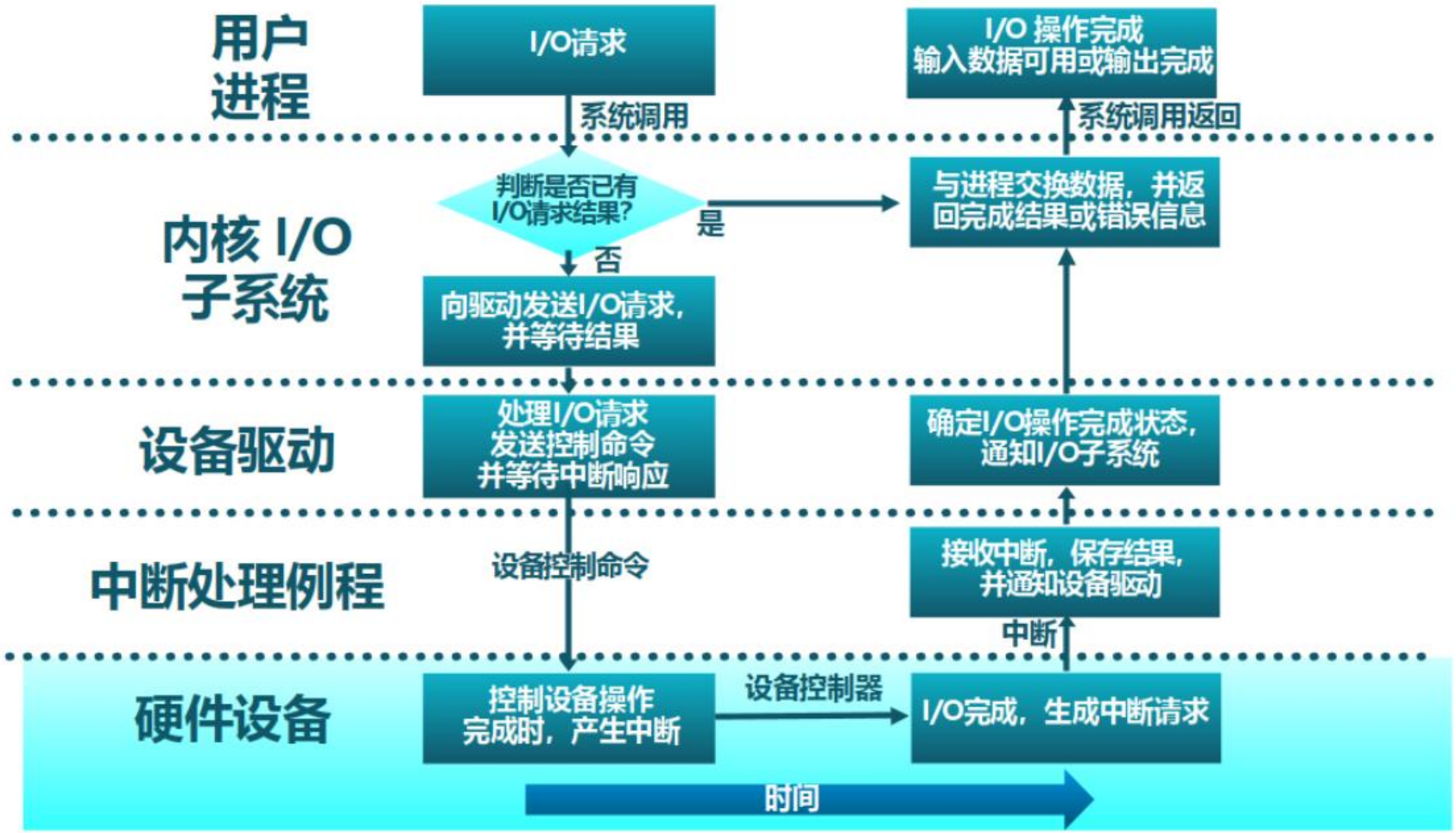
- 设备控制器由三部分组成：
  - 设备控制器与处理机的接口：实现CPU与设备控制器之间的通信。
  - 设备控制器与设备的接口：实现设备与设备控制器之间的通信。
  - I/O逻辑：实现对设备的控制，它负责接收命令、对命令进行译码、再根据译出的命令控制设备。



读取磁盘数据的例子


1.设备驱动收到读取磁盘数据到内存地址X
2.设备驱动控制磁盘控制器从磁盘读取数据
CPU
高速缓存
5.DMA控制器传送C字节数据到内存地址X
6.DMA控制器完成数据传送后，产生中断请求，通知CPU传送完成
DMA、总线和中断控制器
系统总线
内存地址X
缓存
PCI总线
IDE磁盘控制器
3.磁盘控制器初始化DMA传送
4.磁盘控制器传送数据到DMA控制器
磁盘
磁盘
磁盘
磁盘

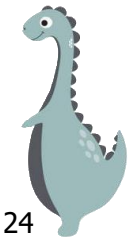
I/O请求周期


用户进程
I/O请求
系统调用
I/O 操作完成
输入数据可用或输出完成
系统调用返回
内核 I/O 子系统
判断是否已有I/O请求结果?
是
否
与进程交换数据，并返回完成结果或错误信息
向驱动发送I/O请求，并等待结果
设备驱动
处理I/O请求
发送控制命令
并等待中断响应
确定I/O操作完成状态，通知I/O子系统
中断处理例程
设备控制命令
接收中断，保存结果，并通知设备驱动
中断
硬件设备
控制设备操作
完成时，产生中断
设备控制器
I/O完成，生成中断请求
时间

I/O执行模型-设备抽象

基于文件的 I/O 设备抽象

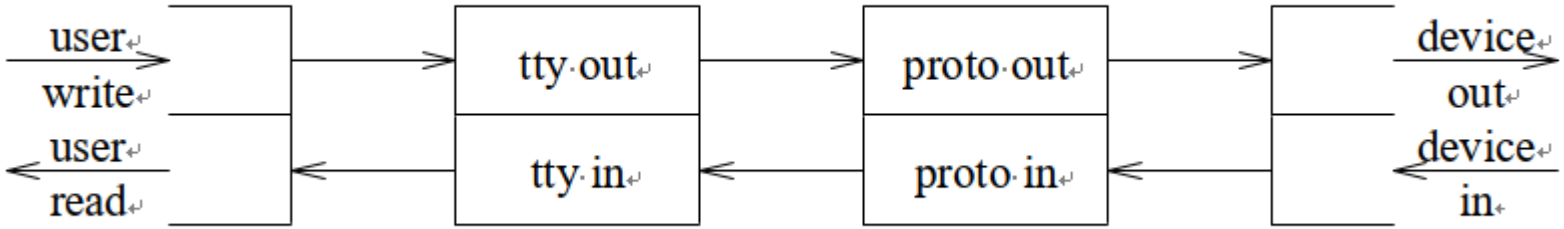
- 访问接口：open/close/read/write
- 特别的系统调用：ioctl ： input/output control
- ioctl 系统调用很灵活，但太灵活了，请求码的定义无规律可循
- ioctl 提供了一种强大但复杂的机制，用于执行特定于设备的操作，允许用户空间程序执行那些标准读写操作无法完成的任务。
- 文件的接口太面向用户应用，不足覆盖到OS对设备进行管理的过程



I/O执行模型-设备抽象

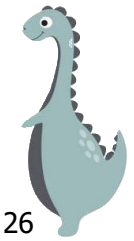
基于流的 I/O 设备抽象

- 流是用户进程和设备或伪设备之间的全双工连接
- 特别的系统调用：ioctl ： input/output control
- ioctl 系统调用很灵活，但太灵活了，请求码的定义无规律可循。



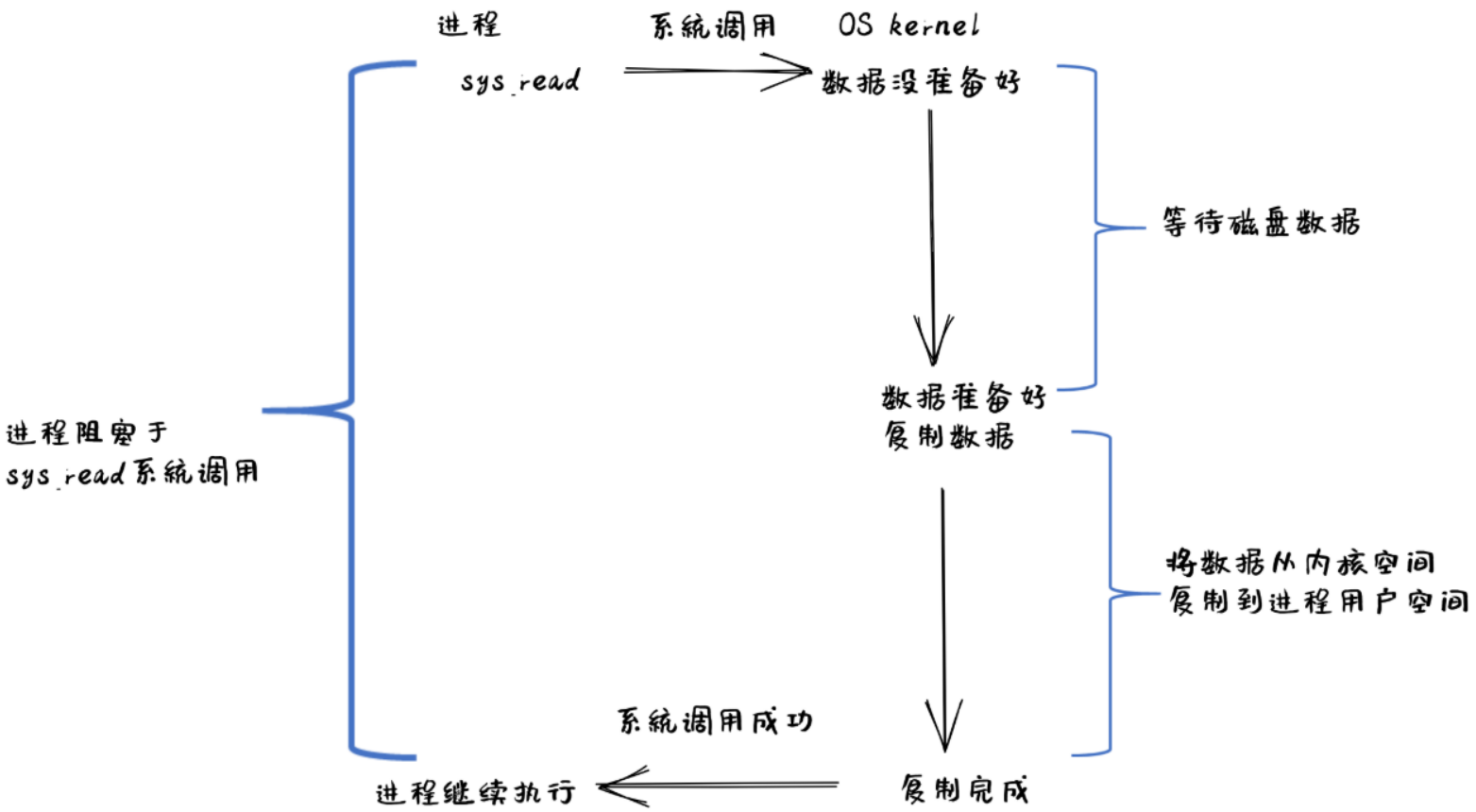
I/O执行模型

- 当一个用户进程发出一个 read I/O 系统调用时，主要经历两个阶段：
- 等待数据准备好；
- 把数据从内核拷贝到用户进程中
- 进程执行状态：阻塞/非阻塞：进程执行系统调用后会被阻塞/非阻塞
- 消息通信机制：
  - 同步：用户进程与操作系统之间的操作是经过双方协调的，步调一致的
  - 异步：用户进程与操作系统之间并不需要协调，都可以随意进行各自的操作



I/O执行模型-阻塞I/O

- 阻塞：进程挂起直到I/O完成



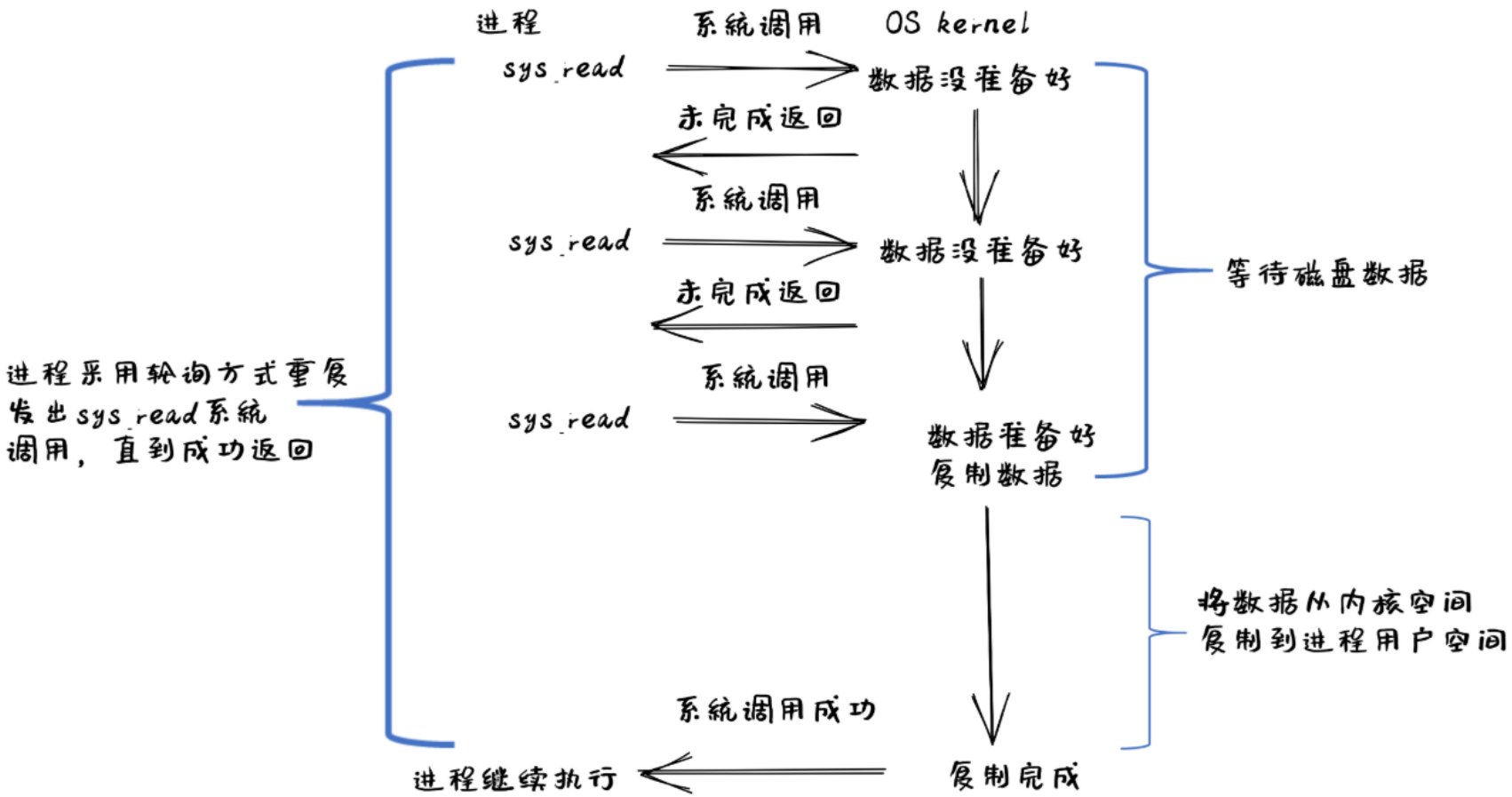
I/O执行模型-阻塞I/O

基于阻塞 I/O（blocking I/O）模型的文件读系统调用 - read 的执行过程是：

1. 用户进程发出 read 系统调用；
2. 内核发现所需数据没在 I/O 缓冲区中，需要向磁盘驱动程序发出 I/O 操作，并让用户进程处于阻塞状态；
3. 磁盘驱动程序把数据从磁盘传到 I/O 缓冲区后，通知内核（一般通过中断机制），内核会把数据从 I/O 缓冲区拷贝到用户进程的 buffer 中，并唤醒用户进程（即用户进程处于就绪态）；
4. 内核从内核态返回到用户态进程，此时 read 系统调用完成。

I/O执行模型-非阻塞I/O

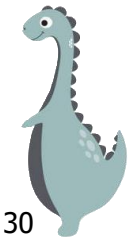
- 非阻塞：I/O调用立即返回



I/O执行模型-非阻塞I/O

基于非阻塞 IO（non-blocking I/O）模型的文件读系统调用 - read 的执行过程：

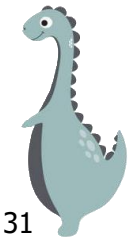
1. 用户进程发出 read 系统调用；
2. 内核发现所需数据没在 I/O 缓冲区中，需要向磁盘驱动程序发出 I/O 操作，并不会让用户进程处于阻塞状态，而是立刻返回一个 error；
3. 用户进程判断结果是一个 error 时，它就知道数据还没有准备好，于是它可以再次发送 read 操作（这一步操作可以重复多次）；



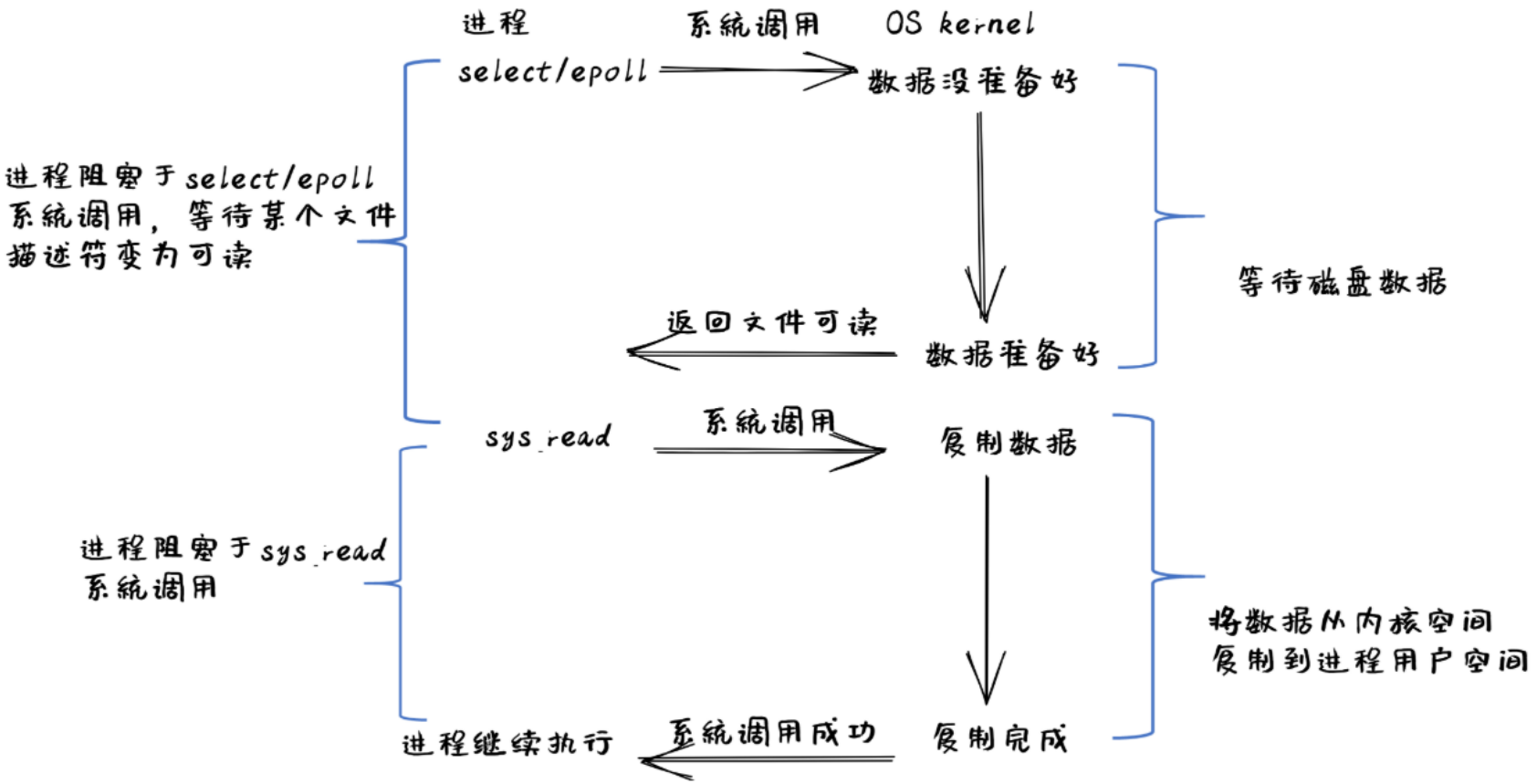
I/O执行模型-非阻塞I/O

4. 磁盘驱动程序把数据从磁盘传到 I/O 缓冲区后，通知内核（一般通过中断机制），内核在收到通知且再次收到了用户进程的 system call 后，会马上把数据从 I/O 缓冲区拷贝到用户进程的 buffer 中；
5. 内核从内核态返回到用户态的用户态进程，此时 read 系统调用完成。

所以，在非阻塞式 I/O 的特点是用户进程不会被内核阻塞，而是需要不断的主动询问内核所需数据准备好了没有。



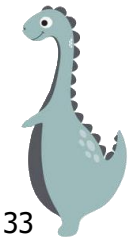
I/O执行模型-多路复用I/O



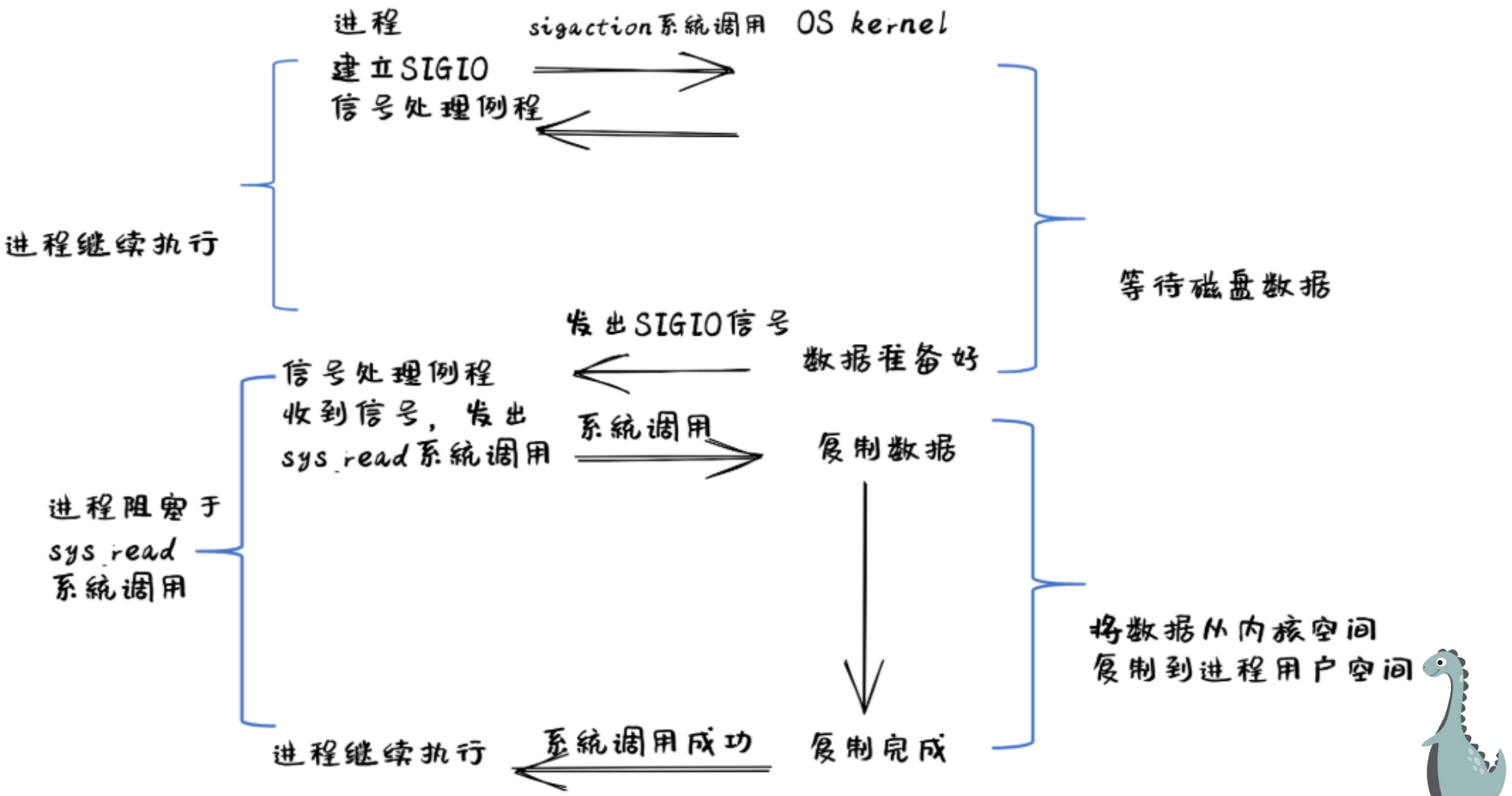
I/O执行模型-多路复用I/O

多路复用 I/O（I/O multiplexing）的文件读系统调用 - read 的执行过程：

1. 对应的 I/O 系统调用是 select 和 epoll 等，可有效处理大量并发的I/O操作；
2. 通过 select 或 epoll 系统调用，用户进程会被阻塞；当某个文件句柄或 socket 有数据到达了，select 或 epoll 系统调用就会返回到用户进程，用户进程再调用 read 系统调用，让内核将数据从内核的I/O 缓冲区拷贝到用户进程的 buffer 中。



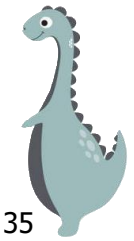
I/O执行模型-信号驱动I/O



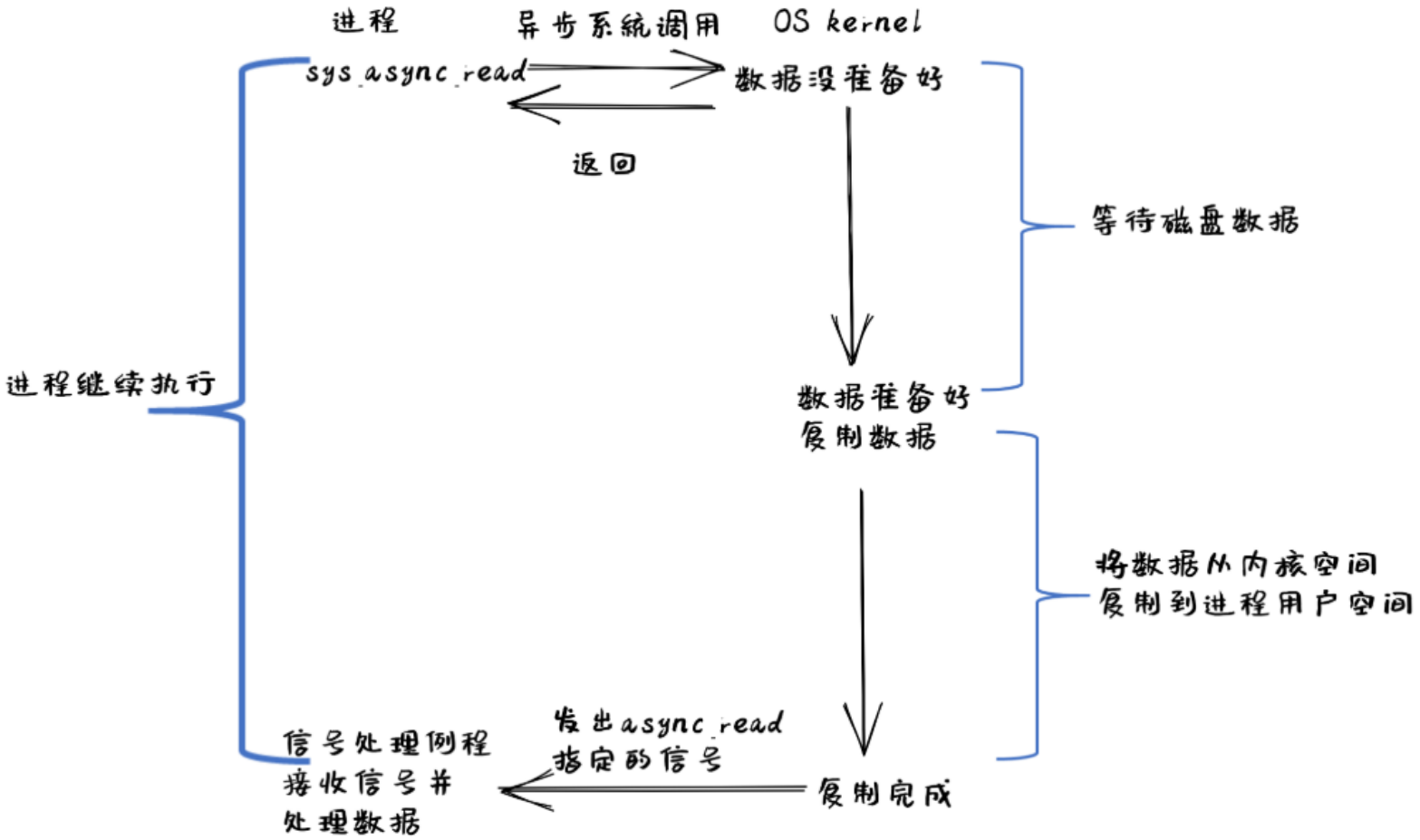
I/O执行模型-信号驱动I/O

1. 当进程发出一个 read 系统调用时，会向内核注册一个信号处理函数，然后系统调用返回，进程不会被阻塞，而是继续执行。
2. 当内核中的 IO 数据就绪时，会发送一个信号给进程，进程便在信号处理函数中调用 IO 读取数据。

- 此模型的特点是，采用了回调机制，这样开发和调试应用的难度加大。

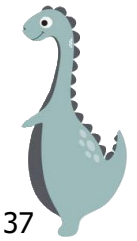


I/O执行模型-异步I/O

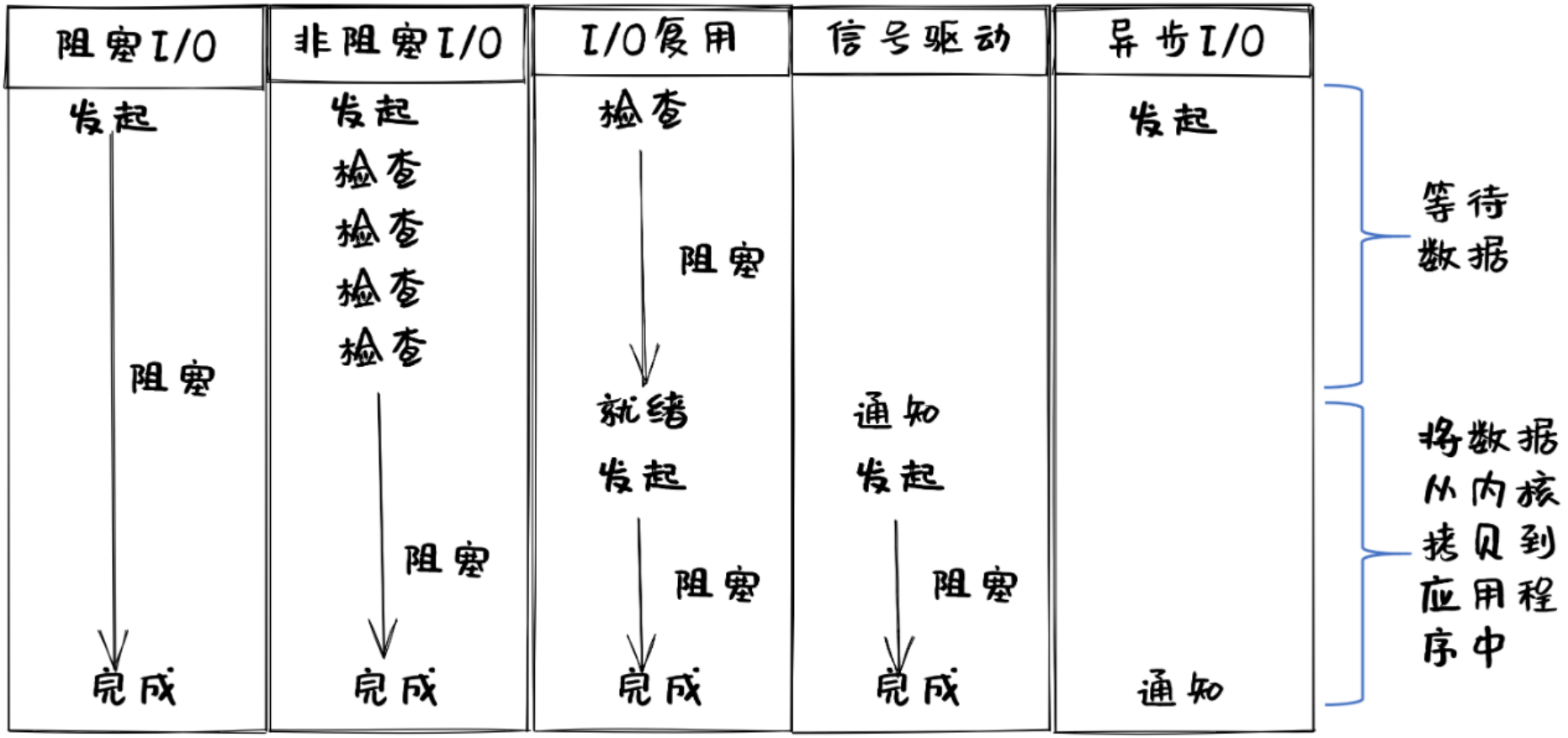

进程
异步系统调用
OS kernel
sys_async_read
数据没准备好
返回
等待磁盘数据
进程继续执行
数据准备好
复制数据
将数据从内核空间
复制到进程用户空间
发出async_read
指定的信号
信号处理例程
接收信号并
处理数据
复制完成

I/O执行模型-异步I/O

1. 用户进程发起 read 异步系统调用之后，立刻就可以开始去做其它的事。
2. 从内核的角度看，当它收到一个 read 异步系统调用之后，首先它会立刻返回，所以不会对用户进程产生任何阻塞情况。
3. kernel 会等待数据准备完成，然后将数据拷贝到用户内存。
4. 当这一切都完成之后，kernel 会通知用户进程，告诉它 read 操作完成了。

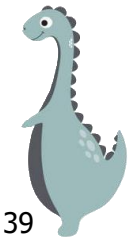


I/O执行模型-比较


阻塞I/O
发起
阻塞
完成
非阻塞I/O
发起
检查
检查
检查
检查
阻塞
完成
I/O复用
检查
阻塞
就绪
发起
阻塞
完成
信号驱动
通知
发起
阻塞
完成
异步I/O
发起
通知
等待数据
将数据从内核拷贝到应用程序中

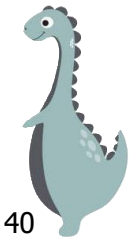
内核I/O子系统

- 许多服务，如调度、缓冲、高速缓存、假脱机、设备预留及错误处理由内核I/O子系统提供



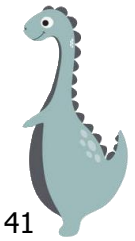
I/O调度

- 调度一组I/O请求就是确定一个好的顺序来执行这些请求。
- 为每个设备维护一个请求等待队列来实现调度。
- 某些操作系统尝试着公平



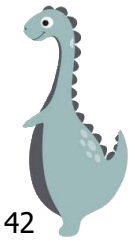
缓冲 Buffering

- 缓冲：当设备间传输数据的时候，暂时存放在内存中。
  - 解决设备速度不匹
  - 解决设备传输块的大小不匹配
  - 为了维持“拷贝语义”



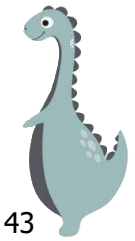
缓冲的引入

- 提高处理机与外设并行的另一项技术是缓冲技术。
- 引入缓冲的主要原因有：
  - 缓和CPU与I/O设备间速度不匹配的矛盾，
  - 提高CPU与I/O设备并行操作的程度，
  - 减少设备对CPU的中断频率，放宽CPU对中断响应时间的限制。



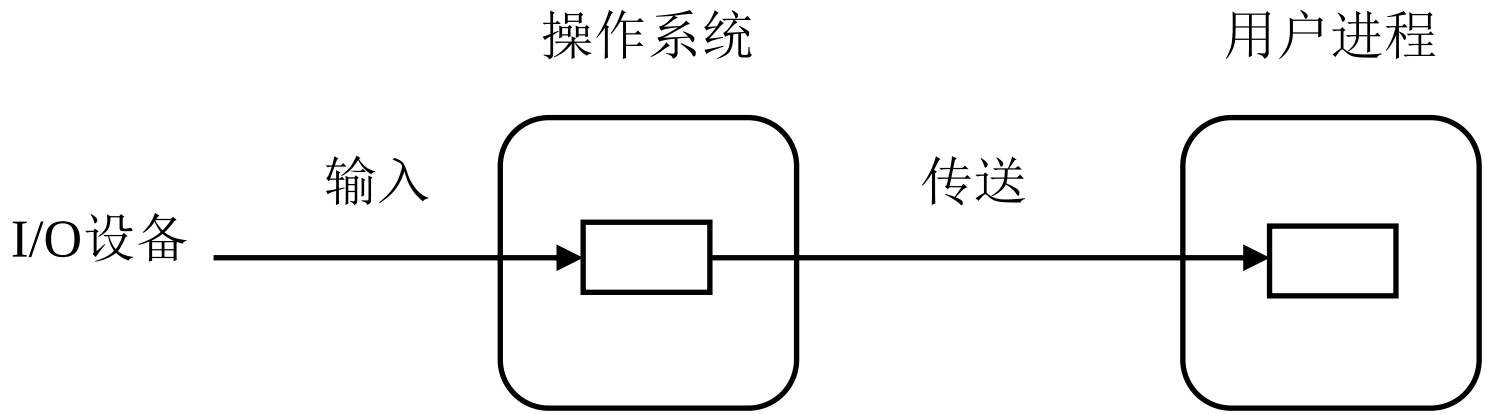
缓冲的实现方法

- 硬件缓冲器：如I/O控制器中的数据缓冲寄存器，但成本太高。
- 软件缓冲：一片内存区域，用来临时存放输入输出数据。
- 缓冲技术分为：
  - 单缓冲
  - 双缓冲
  - 循环缓冲
  - 缓冲池



单缓冲

- 单缓冲是在设备和处理机之间设置一个缓冲区。

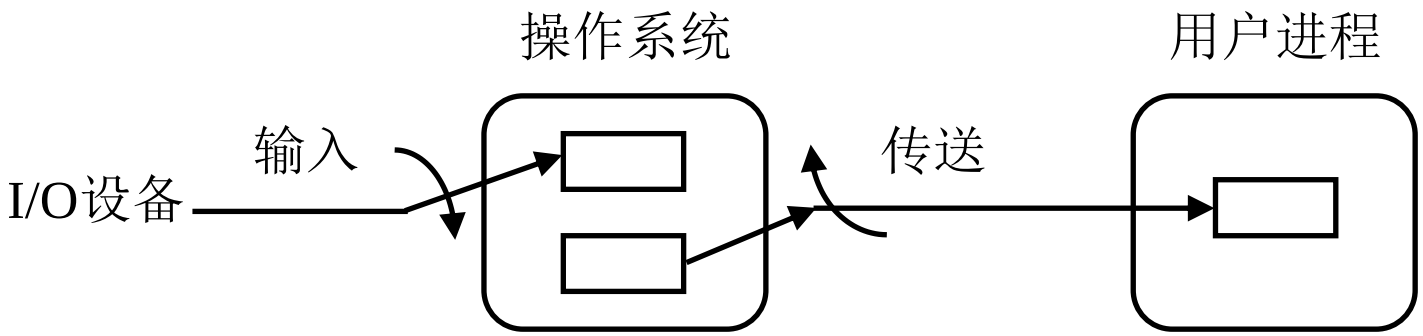


单缓冲（续）

- 在块设备输入时，先从磁盘把一块数据输入至缓冲区，然后OS将缓冲区中的数据送到用户区。在块设备输出时，先将要输出的数据从用户区复制到缓冲区，然后再将缓冲区中的数据写到设备。
- 在字符设备输入时，缓冲区用于暂存用户输入的一行数据。在输入期间，用户进程阻塞以等待一行数据输入完毕；在输出时，用户进程将一行数据送入缓冲区后继续执行计算。当用户进程已有第二行数据要输出时，若第一行数据尚未输出完毕，则用户进程阻塞。

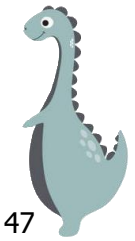
双缓冲

- 引入双缓冲，可以进一步提高处理机与设备的并行操作程度。



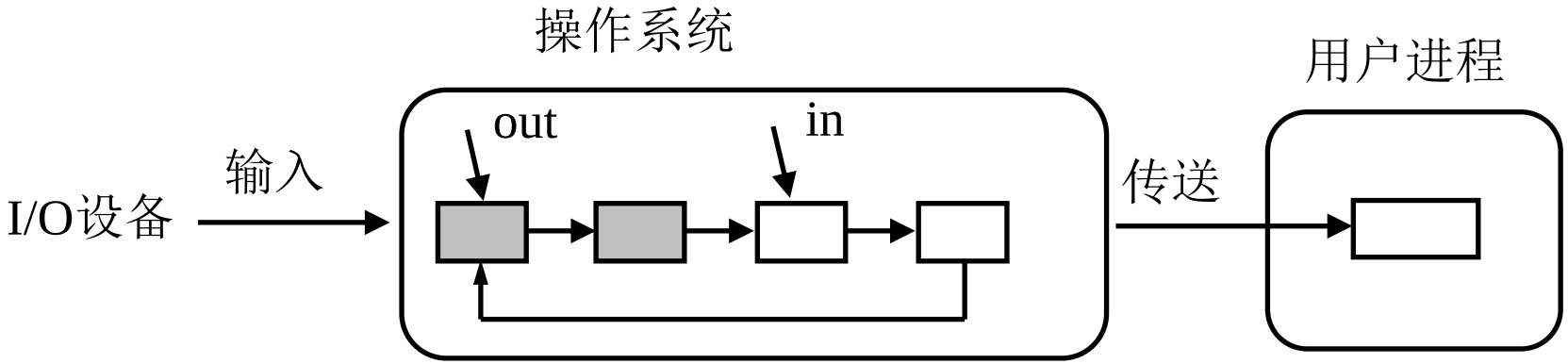
双缓冲（续）

- 在块设备输入时，可先将第一个缓冲区装满，之后便装填第二个缓冲区，与此同时OS可将第一个缓冲区中的数据传到用户区；当第一个缓冲区中的数据处理完后，若第二个缓冲区已装满，则处理机又可处理第二个缓冲区中的数据，而设备又可装填第一个缓冲区。输出与此类似。
- 在字符设备输入时，若采用行输入方式和双缓冲，则用户在输入完第一行后，CPU执行第一行中的命令，而用户可以继续向第二个缓冲区中输入一行数据。



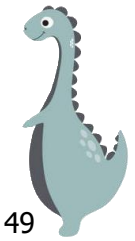
循环缓冲

- 若输入输出速度与数据处理速度相当，则双缓冲能获得较好的效果；若速度相差较大，则可以通过增加缓冲区的数量来改善性能。



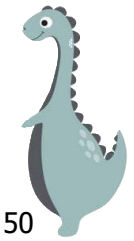
循环缓冲的组成

- 循环缓冲中包含多个大小相等的缓冲区，每个缓冲区中有一个指针指向下一个缓冲区，最后一个缓冲区的指针指向第一个缓冲区，由此构成一个环形。
- 循环缓冲用于输入/输出时，还需要两个指针in和out。
- 对于输入而言，in指向下一个可用的空缓冲区，out指向下一个可以提取数据的满缓冲区。显然，对输出而言正好相反。



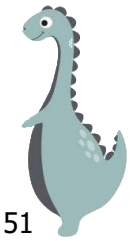
缓冲池

- 循环缓冲适用于合作进程，当系统较大且共享缓冲区的进程较多时，这要消耗大量内存。目前广泛使用的是公用缓冲池。
- 缓冲池由多个缓冲区组成，其中的缓冲区可供多个进程共享，既能用于输入又能用于输出。



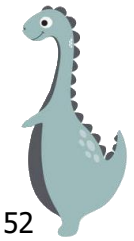
缓冲池的构成

- 缓冲池中有三种类型的缓冲区队列：
  - 空缓冲队列：由空闲缓冲区构成的队列
  - 输入队列：装满输入数据的缓冲区队列
  - 输出队列：装满输出数据的缓冲区队列
- 除上述三个队列之外，还有四种工作缓冲区：
  - 用于收容输入数据的工作缓冲区
  - 用于提取输入数据的工作缓冲区
  - 用于收容输出数据的工作缓冲区
  - 用于提取输出数据的工作缓冲区



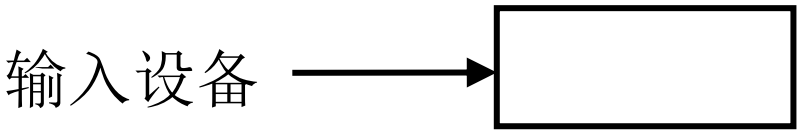
缓冲池的工作方式

- 缓冲池有四种工作方式：
  - 收容输入
  - 提取输入
  - 收容输出
  - 提取输出



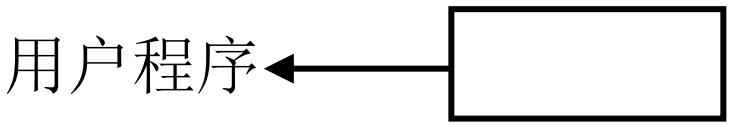
收容输入

- 当输入进程需要输入数据时，便从空缓冲队列的队首摘下一个空缓冲区，把它作为收容输入工作缓冲区，然后把数据输入其中，装满后再将它挂到输入队列队尾。



提取输入

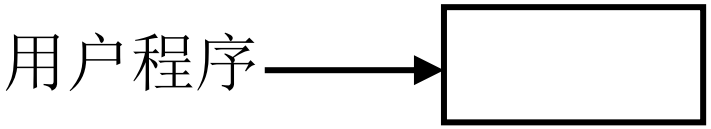
- 当计算进程需要输入数据时，便从输入队列取得一个缓冲区作为提取输入工作缓冲区，计算进程从中提取数据，数据用完后再将它挂到空缓冲队列尾。



提取输入工作缓冲区

收容输出

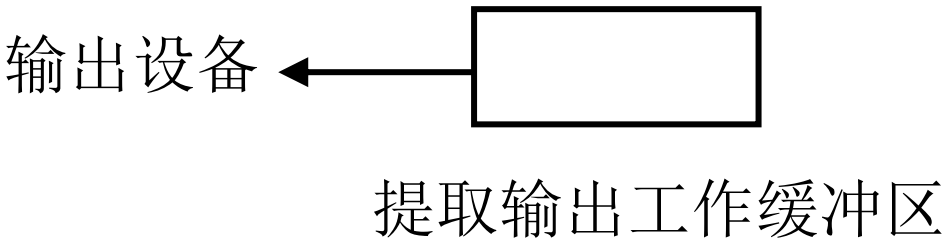
- 当计算进程需要输出数据时，便从空缓冲队列的队首取得一个空缓冲，作为收容输出工作缓冲区，当其中装满输出数据后，再将它挂到输出队列尾。



收容输出工作缓冲区

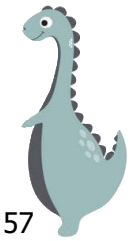
提取输出

- 当要输出时，由输出进程从输出队列中取得一装满输出数据的缓冲区，作为提取输出工作缓冲区，当数据提取完后，再将它挂到空缓冲队列的末尾。



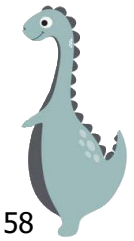
高速缓存cache

- 高速缓存：存放数据副本的高速存储器
- 缓冲与高速缓存的差别
  - 缓冲可能是数据项的唯一副本，而根据定义高速缓存只是提供了一个驻留在其他地方的数据在高速缓存上的一个副本。
  - 高速缓存用于提高访问速度
  - 缓冲用于平衡速度差异



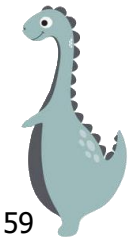
Spooling技术

- Spooling是Simultaneous Peripheral Operating On-Line的缩写，意思是外部设备同时联机操作，又称假脱机操作。
- 在Spooling系统中，用一道程序模拟脱机输入时的外围控制机功能，把低速输入设备上的数据传送到高速磁盘上；再用另一道程序来模拟脱机输出时的外围控制机功能，把数据从磁盘传送到低速输出设备上。



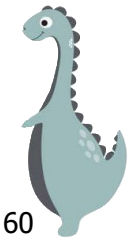
Spooling系统的组成

- Spooling系统由三部分组成：
  - 输入井和输出井
  - 输入缓冲区和输出缓冲区
  - 输入进程和输出进程



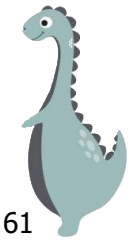
输入井和输出井

- 输入井和输出井是磁盘上的两个存储区域。
- 输入井收容I/O设备输入的数据。
- 输出井收容用户程序的输出数据。



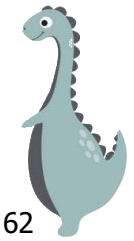
输入缓冲区和输出缓冲区

- 输入缓冲区和输出缓冲区是内存中的两个缓冲区。
- 输入缓冲区用于暂存由输入设备送来的数据，以后再传送到输入井；
- 输出缓冲区用于暂存从输出井送来的数据，以后再传送到输出设备。

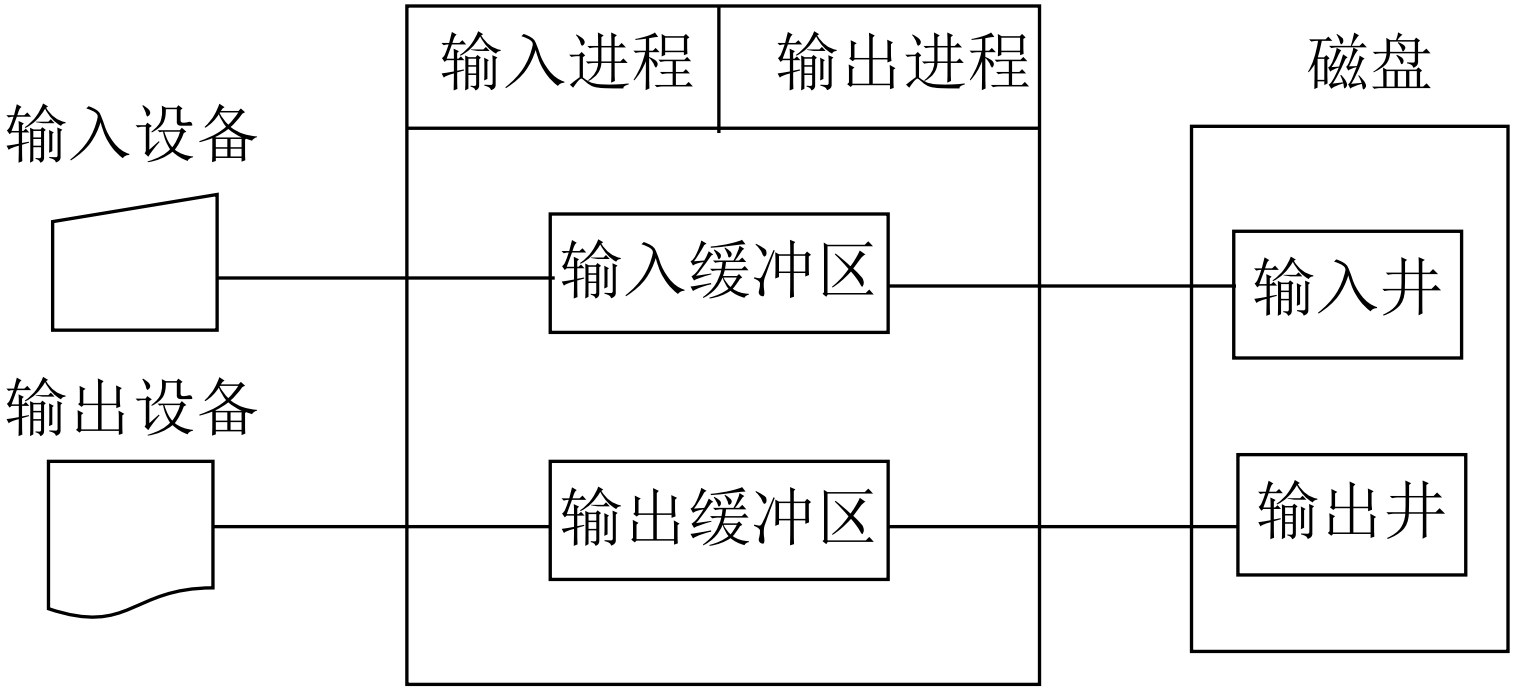


输入进程和输出进程

- 输入进程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通过输入缓冲区再送到输入井，当CPU需要输入数据时直接从输入井读入内存；
- 输出进程模拟脱机输出时的外围控制机，把用户要求输出的数据先从内存送到输出井，待输出设备空闲时，再将输出井中的数据经过输出缓冲区送到输出设备上。

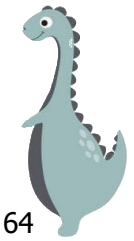


Spooling系统组成图



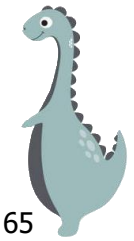
共享打印机1

- 打印机是常用的独享设备，但利用Spooling技术可以将它改造成供多个用户共享的设备。
- 当用户进程请求打印输出时，Spooling系统同意为它打印，但不将打印机真正分配给它，而只为它做两件事：
  - 由输出进程在输出井中为之申请一空闲磁盘区，并将要打印的数据送入其中；
  - 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。



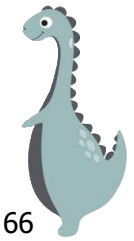
共享打印机2

- 如果打印机空闲，输出进程将从打印队列队首取出一张请求打印表，根据表中的要求将要打印的数据从输出井传送到内存缓冲区，再由打印机进行打印。
- 打印完后，再取下一张表，直至请求队列为空。此时，输出进程阻塞，当再有打印请求时，才将输出进程唤醒。



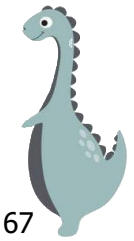
设备预留

- 设备预留：提供对设备的互斥访问
- 预留：亦即为某进程保留该设备的使用权，在该进程获得运行之前，其他申请该设备的进程将得不到使用权。



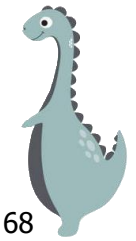
错误处理

- OS可以对短暂出错进行弥补。例如：磁盘read出错可以导致read重试，
- 当I/O失败时，大多会返回一个错误码
- 系统日志记录了出错报告



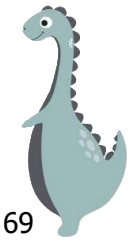
I/O保护 I/O Protection

- 通过发出非法I/O指令，用户程序可以有意或无意中断系统的正常操作
- 为防止用户执行非法I/O，定义所有I/O指令为特权指令。



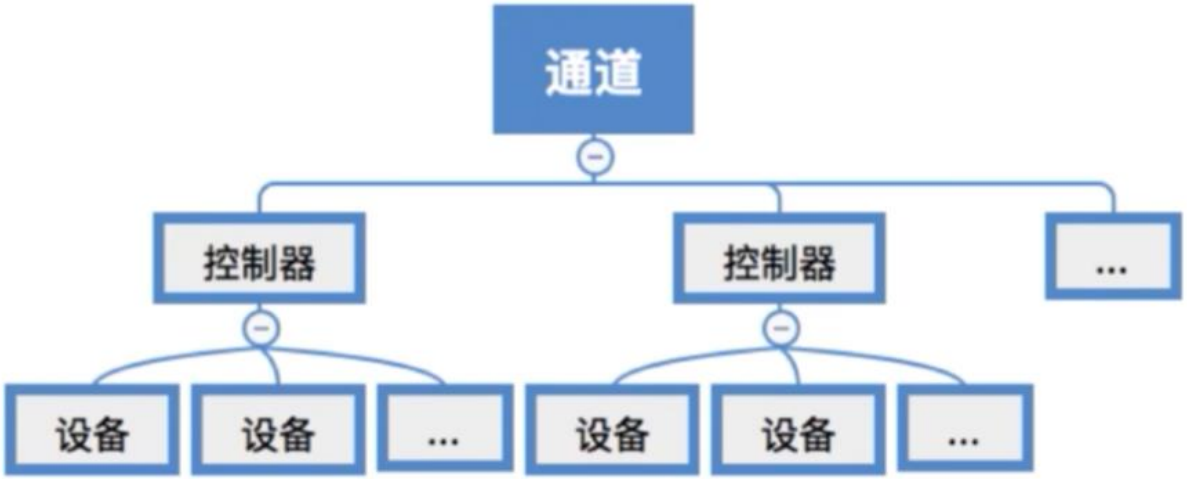
设备分配

- 当进程提出I/O请求时，设备分配程序便按照一定的策略为其分配设备，同时还应分配相应的控制器和通道，以保证CPU与设备之间的通信。



设备管理的数据结构

- 设备管理的主要数据结构有：
  - 设备控制表
  - 控制器控制表
  - 通道控制表
  - 系统设备表



设备控制表（DCT）

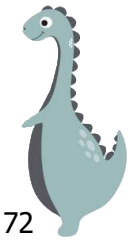
- 系统为每个设备配置一张设备控制表，用于记录设备的特性及与I/O控制器连接的情况。

设备控制表

设备类型
设备标识符
设备状态：忙/闲
指向控制器表的指针
设备等待队列指针
…

设备控制表（DCT）

```
struct Device {
    int device_id;          // 设备唯一标识符
    char device_type[20];   // 设备类型
    int status;             // 设备状态
    char device_address[50];// 设备地址
    struct RequestQueue* queue; // 指向请求队列的指针
};
```



控制器控制表（COCT）

- 控制器控制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。

控制器控制表

控制器标识符
控制器状态：忙/闲
指向通道表的指针
控制器等待队列指针
…

通道控制表（CHCT）

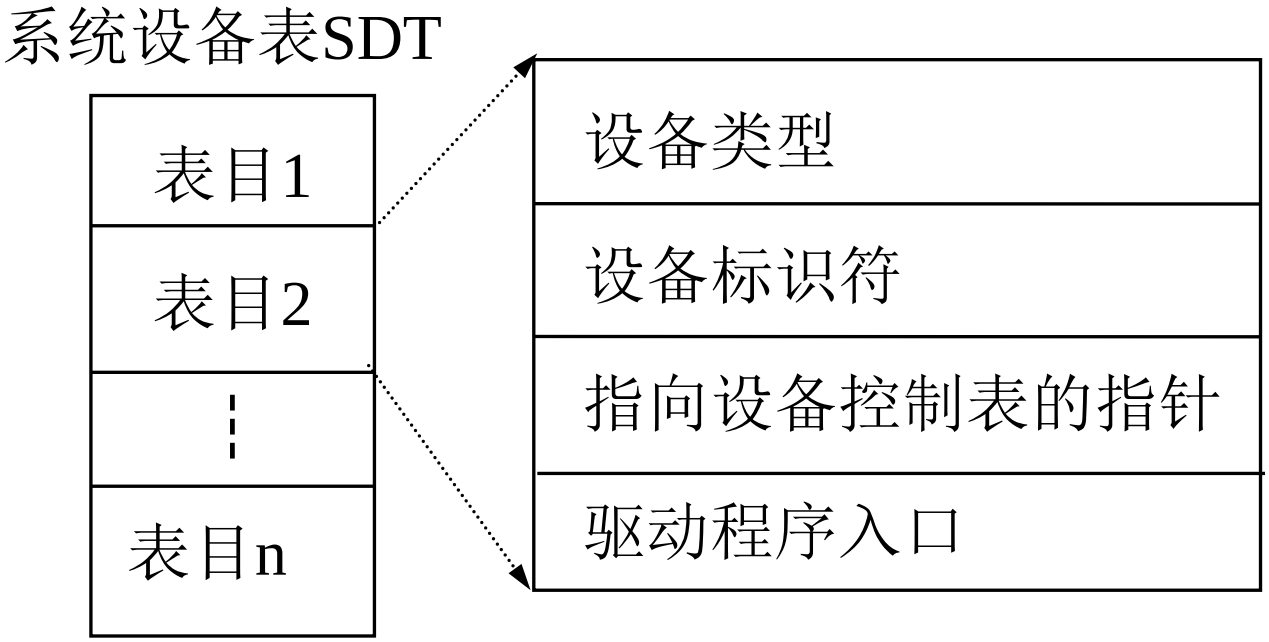
- 每个通道都配有一张通道控制表，它反映通道的使用状态。

通道控制表

通道标识符
通道状态：忙/闲
通道等待队列指针
…

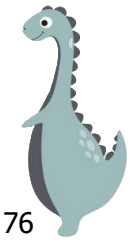
系统设备表（SDT）

- 系统设备表整个系统一张，它记录了系统中所有物理设备的情况，每个物理设备占一个表目。



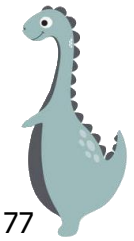
设备分配策略

- 系统在进行设备分配时，应考虑以下因素：
  - 设备的使用性质
  - 设备分配算法
  - 设备分配的安全性
  - 设备独立性



设备的使用性质

- 按共享属性设备有三种类型：
  - 独占：这种设备在一段时间内只允许一个进程独占。
  - 共享：这种设备允许多个进程同时共享。
  - 虚拟：设备本身虽是独占设备，但经过某种技术处理后可改造成虚拟设备。
- 针对上述三种设备可采用三种不同的分配方式。



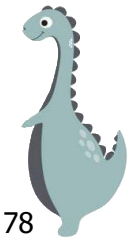
独享分配与共享分配

独享分配

- 在将一个设备分配给某进程后便一直由它独占，直至该进程完成或释放该设备后，系统才能再将该设备分配给其他进程使用。

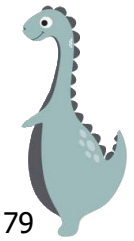
共享分配

将设备同时分配给多个进程使用。但这些进程对设备的访问需要进行合理调度。



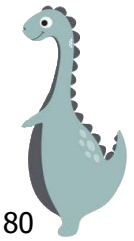
虚拟分配

- 虚拟分配针对虚拟设备，其实现过程是：
  - 当进程申请独占设备时，系统给它分配共享设备上的一部分存储空间；
  - 当进程要与设备交换信息时，系统就把要交换的信息存放在这部分存储空间中；
  - 在适当的时候，将设备上的信息传输到存储空间中或将存储空间中的信息传送到设备。



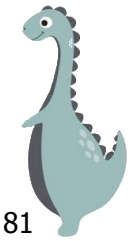
设备分配算法

- 设备分配通常只采用以下两种算法：
  - 先来先服务：根据进程对某设备发出请求的先后次序，将它们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。：
  - 优先级高者优先：按对某设备提出I/O请求的进程优先级由高到低排队，对优先级相同的I/O请求，按先来先服务的算法排队，设备分配程序总是把设备首先分配给队首进程。



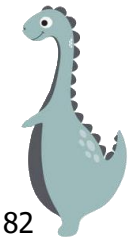
设备分配的安全性

- 设备分配的安全性是指在设备分配中应保证不发生进程死锁。
- 设备有两种分配方式：
  - 静态分配
  - 动态分配



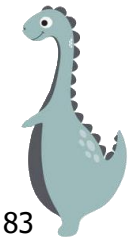
静态分配

- 静态分配：用户作业开始执行前，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，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进程的死锁，但设备的使用效率低。



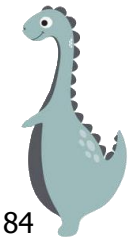
动态分配

- 动态分配：在进程执行过程中根据需要进行设备分配，一旦使用完之后便立即释放。
- 动态分配方式有利于提高设备利用率，但分配算法使用不当有可能造成进程死锁。
- 动态分配又分为：
  - 安全分配
  - 不安全分配



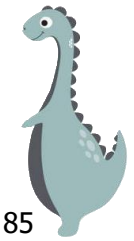
安全分配

- 安全分配：每当进程发出I/O请求后便进入阻塞状态，直到I/O操作完成时才被唤醒并释放资源。
- 特点：设备分配安全，但进程进展缓慢。



不安全分配

- 不安全分配：进程发出I/O请求后继续运行，需要时又可发出第二个I/O请求、第三个I/O请求等。仅当进程所请求的设备被其他进程占用时才进入阻塞状态。
- 特点：进程推进迅速，但可能死锁。



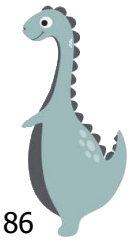
设备分配过程

（1）进程根据物理设备名，查找系统设备表；

（2）根据系统设备表，查找到设备控制表；此时如果设备忙碌，则进程加入该设备的等待队列；

（3）根据设备控制表找到控制器表；此时如果控制器忙碌，则进程加入控制器等待队列；

（4）根据控制器表到通道控制表；此时如果通道忙碌，则进程加入通道等待队列只有设备、控制器、通道三者同时空闲并分配，才算做进程分配设备成功。



设备独立性

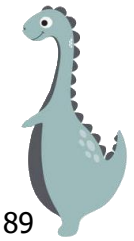
- 设备独立性：又称设备无关性，是指用户编制程序时使用的设备与实际使用的物理设备无关。
- 逻辑设备：用户程序中使用的设备。
- 物理设备：计算机系统中存在的设备。

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/printer	3	1024
/dev/tty	5	2046
…	…	…

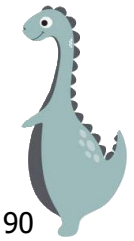
大容量存储器结构简介

- 大容量存储器结构通常由硬盘驱动器（HDD）和固态硬盘（SSD）组成。
- 硬盘驱动器（HDD）是一种机械式存储器，由旋转的磁盘和移动的读写头组成。数据存储在磁盘的表面上，读写头在磁盘上移动以读取或写入数据。HDD的优点是存储容量大、成本相对较低，但速度相对较慢。
- 固态硬盘（SSD）是一种基于闪存存储技术的存储器。它使用闪存芯片来存储数据，没有机械部件，因此读写速度更快，响应时间更短。SSD的优点是速度快、耐用性高，但相对于HDD来说，存储容量较小且价格较高。

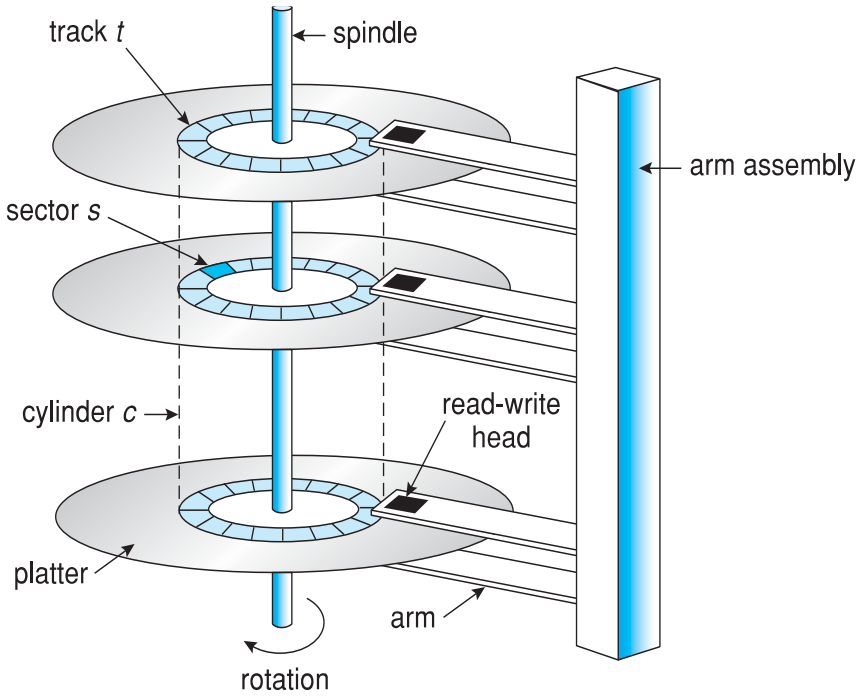


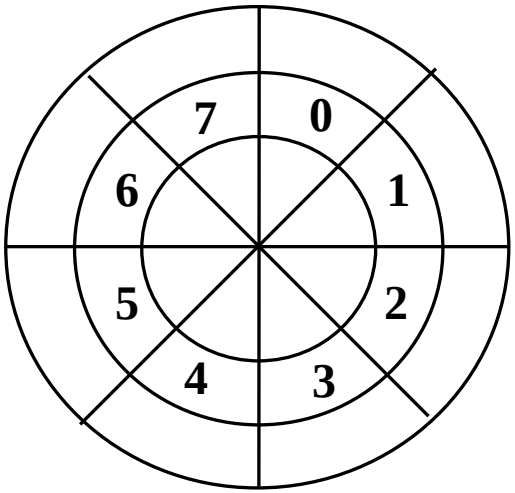
硬盘 Hard Disk

- 磁盘是典型的直接存取设备。
- **磁盘**一般由若干磁盘片组成，可沿一个固定方向高速旋转。每个盘面对应一个磁头，磁臂可沿半径方向移动。
- 磁盘上的一系列同心圆称为**磁道**（track），磁道沿径向又分成大小相等的多个**扇区**（sector），与盘片中心有一定距离的所有磁道组成一个**柱面**（cylinder）。
- 磁盘上的每个物理块可用柱面号，磁头号和扇区号表示。

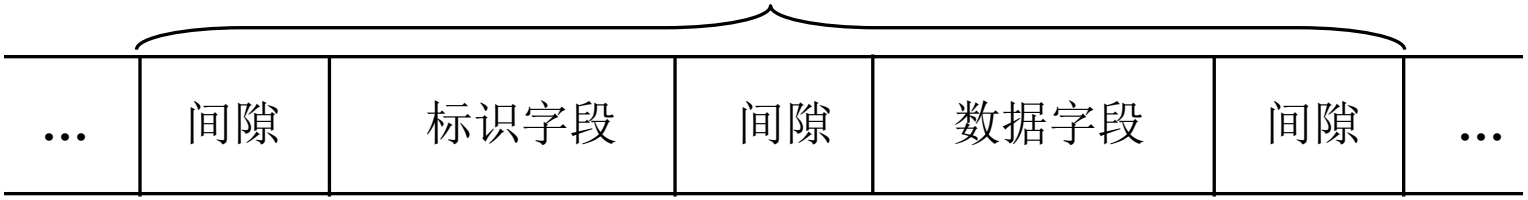


磁盘数据组织和格式示意图



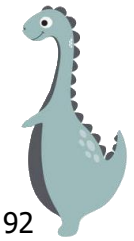


磁道 第i扇区



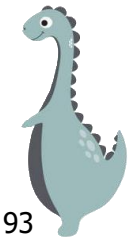
逻辑扇区

- 将一个磁盘上的扇区从0柱面开始，按柱面顺序依次编号，所得到的编号称为逻辑扇区号。
- 逻辑扇区号=柱面号*每柱面扇区数+
  磁头号*每磁头扇区数+
  扇区号
- 其中，柱面号、磁头号、扇区号都从0开始



磁盘结构

- 磁盘设备编址为逻辑块的一维大数组。
- 一维逻辑块的数组按顺序映射到磁盘上的扇区。
- 0扇区是最外边柱面的第一个磁道的第一个扇区。
- 数据首先都映射到一个磁道，其余的数据映射到同一柱面的其他磁道，然后按照从外向里的顺序映射到其余的柱面。

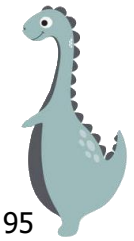


磁盘访问时间

- 磁盘访问时间由三部分组成：
  - **寻道时间**（seek time）：指将磁头从当前位置移动到指定磁道所经历的时间。由启动磁臂时间和磁头移动多条磁道的时间构成。
  - **旋转延迟时间**（rotational latency）：指扇区移动到磁头下面所经历的时间。平均旋转延迟时间是每转所需时间的一半。
  - **传输时间**（tranfer time）：指从磁盘上读出数据或向磁盘写入数据所经历的时间。
- 由于这三部分操作均涉及机械运动，故磁盘块的访问时间约为0.01～0.1s之间，其中寻道时间所占的比例最大。

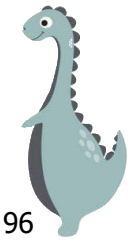
磁盘调度

- 磁盘是可以被多个进程共享的设备
- 当有多个进程都请求访问磁盘时，操作系统需要发挥作用来减少I/O成本
- 当出现一组I/O请求时，磁盘调度程序检查并决定下一个要调度的请求
- 以使各进程对磁盘的平均访问时间（主要是寻道时间）最短为调度目标



先来先服务调度

- **先来先服务算法**按进程请求访问磁盘的先后次序进行调度。
- 特点：简单合理，但未对寻道进行优化。

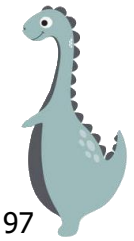


先来先服务调度例

下一磁道号	移动距离
55	45
58	3
39	19
18	21
90	72
160	70
150	10
38	112
184	146

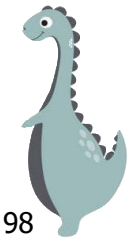
从100号磁道开始，磁盘访问请求为：55、58、39、18、90、160、150、38、184

平均寻道长度为：55.3



最短寻道时间优先调度

- **最短寻道时间优先**算法选择从当前磁头位置所需寻道时间最短的请求作为下一次服务的对象。
- 特点：寻道性能比FCFS好，但不能保证平均寻道时间最短，还可能会使某些请求总也得不到服务。



最短寻道时间优先调度例

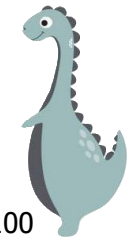
下一磁道号	移动距离
90	10
58	32
55	3
39	16
38	1
18	20
150	132
160	10
184	24

从100号磁道开始，磁盘访问请求为：55、58、39、18、90、160、150、38、184

平均寻道长度为：27.6

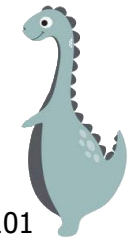
扫描调度

- SSTF有可能引起某些请求的饥饿。
- **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。



扫描调度（LOOK）

- 因这种算法中磁臂移动规律颇似大楼中电梯的运行，故又称为电梯调度算法。
- 特点：具有较好的寻道性能，能避免进程饥饿，但不利于两端磁道的请求。

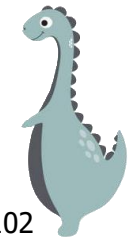


扫描算法例

下一磁道号	移动距离
150	50
160	10
184	24
90	94
58	32
55	3
39	16
38	1
18	20

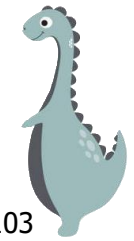
从100号磁道开始，向磁道号增加方向移动。磁盘访问请求为：55、58、39、18、90、160、150、38、184

平均寻道长度为：27.8



循环扫描算法(CSCAN,C-LOOK)

- **CSCAN算法**是SCAN算法的变种，提供了一个更为均匀地等待时间。
- 磁头从磁盘的一端向另一端移动，沿途响应请求。当它到了另一端，就立即回到磁盘的开始处，在返回的途中不响应任何请求。
- 特点：该算法消除了对两端磁道请求的不公平。

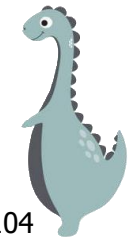


循环扫描算法例

下一磁道号	移动距离
150	50
160	10
184	24
18	166
38	20
39	1
55	16
58	3
90	32

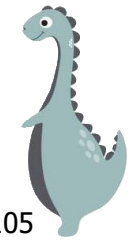
从100号磁道开始，向磁道号增加方向移动。磁盘访问请求为：55、58、39、18、90、160、150、38、184

平均寻道长度为：35.8



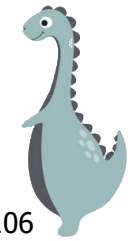
N-Step-SCAN

- 若多个进程反复请求对某一磁道的访问，则磁臂可能停留在某处不动，这一现象称为磁臂粘着。
- N-Step-SCAN算法：将磁盘请求队列分成若干个长度为N的子队列，磁盘调度按FCFS算法依次处理这些子队列，而处理每个队列时按SCAN算法进行，一个队列处理完后，再处理其他队列。



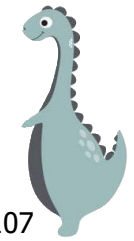
FSCAN算法

- FSCAN算法是N-Step-SCAN算法的简化，它只将磁盘请求队列分成两个子队列。一个是当前所有请求磁盘I/O的进程形成的队列，由磁盘调度按SCAN算法进行处理，另一个队列则是在扫描期间新出现的磁盘请求。



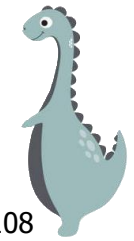
磁盘调度算法的选择

- SSTF比较通用且很有吸引力。
- SCAN和C-SCAN在重磁盘负载的系统中执行得较好。
- 性能依赖于请求的数量和类型。
- 磁盘服务请求受到文件分配方式的影响。
- 磁盘调度算法应该写成操作系统中的一个独立模块，在必要的时候允许用不同的算法来替换。
- SSTF和LOOK都是缺省算法的合理选择。



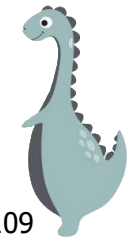
磁盘初始化

- 分区：确定分区的数量、大小和类型，使用分区工具（如磁盘管理工具）创建分区，并将其分配给相应的磁盘空间。
- 创建卷：将分区转换为逻辑存储空间，以便进行文件系统的创建和管理。
- 格式化：选择适当的文件系统类型，使用格式化工具（如格式化命令或磁盘管理工具）对每个卷进行格式化和初始化。



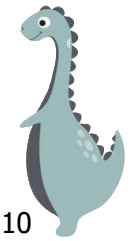
磁盘格式化

- 低级格式化(物理格式化)：在磁盘表面上创建物理磁道和扇区的过程。由磁盘制造商完成的，用户无需干预或执行任何操作。它确保磁盘的物理结构正确，并准备好用于存储数据。
- 高级格式化：在磁盘上创建文件系统结构以及相关的元数据信息的过程。高级格式化会删除磁盘上的所有数据，并为磁盘准备好文件系统以便进行文件和目录的存储和管理。



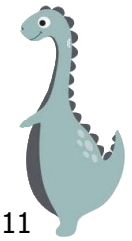
磁盘引导块 Boot Block

- 磁盘引导块（Disk Boot Block），也被称为引导扇区（Boot Sector）或主引导记录（Master Boot Record，MBR），是存储在磁盘的第一个扇区的特殊区域。它是操作系统启动过程中的关键组成部分。
- 引导过程
  - CPU自检
  - 运行ROM中的自举程序 (BIOS for PC)
  - 从引导分区装入第一块
- 绝大多数系统只在启动ROM中保留一个很小的自举装入程序，其作用是进一步从磁盘上调入更为完整的自举程序。它能从磁盘上装入整个操作系统。



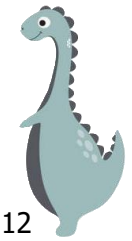
坏块 Bad Block

- 磁盘上的一个或多个扇区可能坏掉。
  - 对于简单磁盘， Format等程序可以标记坏扇区以通知分配程序不使用。
  - 更为复杂的磁盘，对坏块的处理更为智能化。如采用扇区备用或转寄方案，即低级格式化时留一些块作为备用，发现坏块时用备用块逻辑替代坏块；



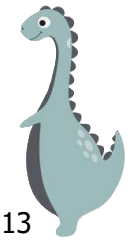
交换空间管理

- 交换空间：虚拟内存使用磁盘空间作为主存的扩展
- 交换空间的使用
  - 保存整个进程映像
  - 存储换出内存的页
- 交换空间的位置
  - 交换空间创建在普通文件系统上。通常是文件系统内的一个简单大文件。这种方式实现简单但效率较低。
  - 交换空间创建在独立的磁盘分区上（如Unix/Linux）
  - 有些OS较为灵活，可以由系统管理员来选择使用以上哪种方式。

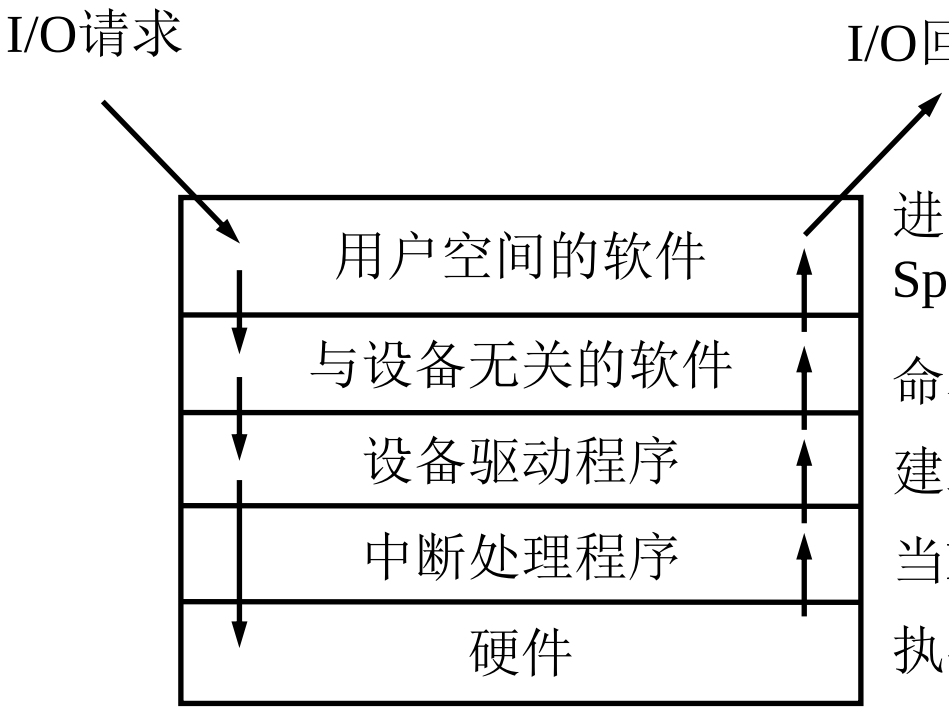


错误处理

- OS可以对短暂出错进行弥补。例如：磁盘read出错可以导致read重试，
- 当I/O失败时，大多会返回一个错误码
- 系统日志记录了出错报告



I/O系统的层次结构



I/O功能

进行I/O调用；格式化I/O；Spooling

命名，保护，阻塞，缓冲，分配

建立设备寄存器；检查状态

当I/O结束时唤醒驱动程序

执行I/O操作

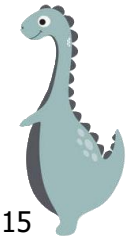
练习题

- 11.13 Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4,999. The drive is currently serving a request at cylinder 2,150, and the previous request was at cylinder 1,805. The queue of pending requests, in FIFO order, is:

  2,069; 1,212; 2,296; 2,800; 544; 1,618; 356; 1,523; 4,965; 3,681

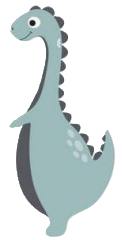
Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

- a. FCFS
- b. SCAN
- c. C-SCAN



练习题

- 12.5How does DMA increase system concurrency? How does it complicate hardware design?
- 12.6 Why is it important to scale up system-bus and device speeds as CPU speed increases?



选择题1

- 缓冲技术中的缓冲池在 _____ 中。

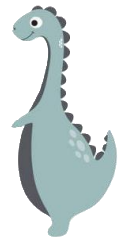
  A. 主存　　B. 外存

  C. ROM　　D. 寄存器

- CPU输出数据的速度远远高于打印机的打印速度，为了解决这一矛盾，可采用_____ 。

  A. 并行技术　　B. 通道技术

  C. 缓冲技术　　D. 虚存技术



选择题2

- 通过硬件和软件的功能扩充，把原来独占的设备改造成能为若干用户共享的设备，这种设备称为 _____ 。

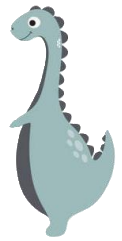
  A. 存储设备　　B. 系统设备

  C. 用户设备　　D. 虚拟设备

- 为了使多个进程能有效地同时处理输入/输出，最好使用 _____ 结构的缓冲技术。

  A. 循环缓冲　　B. 缓冲池

  C. 单缓冲　　D. 双缓冲



选择题3

- 如果I/O设备与存储设备进行数据交换不经过CPU来完成，这种数据交换方式是_____。

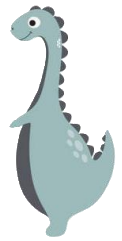
  A. 程序查询　　B. 中断方式

  C. DMA方式　　D. 无条件存取方式

- 在采用Spooling 技术的系统中．用户的打印结果首先被送到_____。

  A. 磁盘固定区域　　B. 内存固定区域

  C. 终端　　D. 打印机



选择题4

- 设备管理程序对设备的管理是借助一些数据结构来进行的，下面的 _____ 不属于设备管理数据结构。

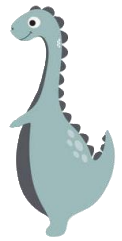
  A. DCT　　B. COCT

  C. CHCT　　D. JCB

- 操作系统中的Spooling技术，实质是将 _____ 转化为共享设备的技术。

  A. 虚拟设备　　B. 独占设备

  C. 脱机设备　　D. 块设备



选择题5

- 按 _____ 分类可将设备分为块设备和字符设备。

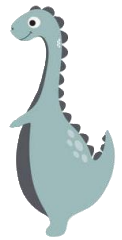
  A. 从属关系　　B. 操作特性

  C. 共享属性　　D. 信息交换单位

- _____ 算法是设备分配常用的一种算法。

  A. 短作业优先　　B. 最佳适应

  C. 先来先服务　　D. 首次适应



选择题6

- 在下面关于设备属性的论述中，正确的论述是_____。

  A．字符设备的一个基本特征是可寻址的。

  B．共享设备必须是可寻址的和可随机访问的设备。

  C．共享设备是指在同一时刻，允许多个进程同时访问的设备。

  D．在分配共享设备和独占设备时，都可能引起进程死锁。

- 通道是一种特殊的___，具有执行I/O指令集的能力。

  A．I/O设备　　B．设备控制器

  C．处理机　　D．I/O控制器

选择题7

- 共享设备磁盘的物理地址为（柱面号，磁头号，扇区号），磁头从当前位置移动到需访问柱面所用的时间称为 ① ，磁头从访问的柱面移动到指定扇区所用时间称为 ② 。
- A. 寻道时间　　B. 传输时间
- C. 旋转等待时间　　D. 周转时间
- 若进程P1访问199号柱面，磁头是从0号柱面移到199柱面的，且在访问期间依次出现了P2申请读299号柱面，P3申请写209号柱面，P4申请读199号柱面，访问完199号柱面以后，如果采用：先来先服务算法，将依次访问 ①；最短寻道时间优先算法，将依次访问 ②；扫描算法，将依次访问 ③ 。
- A. 299，199，209　　B. 299，209，199
- C. 199，209，299　　D. 209，199，299

选择题8

- 存放在磁盘上的文件 _____ 。

  A. 只能随机访问　　B. 只能顺序访问

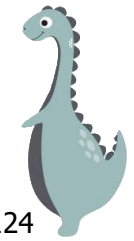
  C. 既可随机访问，又可顺序访问

  D. 不能随机访问

- 用磁带作文件存储介质时，文件只能组织成 _____ 。

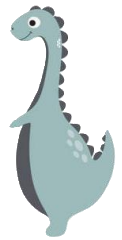
  A. 目录文件　　B. 链接文件

  C. 索引文件　　D. 顺序文件



填空题1

- 进行设备分配时所需的数据表格主要有①、②、③和④。
- 引起中断发生的事件称为_____。
- 常用的I/O控制方式有程序直接控制方式、中断控制方式、①和②。
- 通道是一个独立于①的专管②，它控制③与内存之间的信息交换。

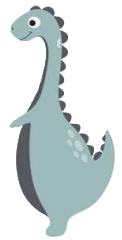


填空题2

- SPOOLing系统是由磁盘中的 ① 和 ② ，内存中的 ③ 和 ④ 以及 ⑤ 和 ⑥ 所构成。
- 设备分配程序分配外部设备时，先分配 ① ，再分配 ② ，最后分配 ③ 。
- 中断方式适合于 ① ，DMA方式适合于 ② 。
- 缓冲区的组织方式可分为 ① 、 ② 、 ③ 和缓冲池。

填空题3

- 缓冲池中有三种类型的缓冲队列：__①__ 、__②__ 、__③__ 。
- 大多数设备控制器由三部分构成：__①__ 、__②__ 、__③__ 。
- 活动头磁盘的访问时间包括 __①__ 、__②__ 和__③__ 。
- _____ 算法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。



考研题1

- 程序员利用系统调用打开I/O设备时，通常使用的设备标识是（）。09

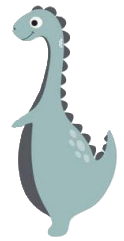
  A、逻辑设备名　　B、物理设备名

  C、主设备号　　D、从设备号

- 下列选项中，能引起外部中断的事件是（）。

  A、键盘输入　　B、除数为0

  C、浮点运算下溢　　D、访存缺页



考研题2

- 本地用户通过键盘登陆系统时，首先获得键盘输入信息的程序是____。10

A.命令解释程序　　B.中断处理程序

C.系统调用程序　　D.用户登陆程序

考研题3

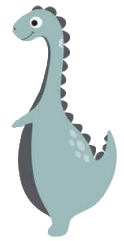
- 用户程序发出磁盘I/O请求后，系统的正确处理流程是_____。

A、用户程序→系统调用处理程序→中断处理程序→设备驱动程序

B、用户程序→系统调用处理程序→设备驱动程序→中断处理程序

C、用户程序→设备驱动程序→系统调用处理程序→中断处理程序

D、用户程序→设备驱动程序→中断处理程序→系统调用处理程序



考研题4

- 某文件占10个磁盘块，现要把该文件磁盘块逐个读入主存缓冲区，并送用户区进行分析。假设一个缓冲区与一个磁盘块大小相同，把一个磁盘块读入缓存的时间为100μs，将缓冲区的数据传送到用户区的时间是50μs，CPU对一块数据进行分析的时间是50μs。在单缓冲区及双缓冲区结构下，读入并分析完该文件的时间分别是____。11
- A、1500μs，1000μs　B、1550μs，1100μs
- C、1550μs，1550μs　D、2000μs，2000μs

考研题5

- 中断处理和子程序调用都需要压栈保护现场，中断处理一定会保存而子程序调用不需要保存其内容的是（）。12

A．程序计数器　　B.程序状态寄存器

C．通用数据寄存器　　D.通用地址寄存器

- 操作系统的I/O子系统通常由四个层次组成，每一层明确定义了与邻近层次的接口。其合理的层次组织排列顺序是()。12

A、用户级I/O软件、设备无关软件、设备驱动程序、中断处理程序

B、用户级I/O软件、设备无关软件、中断处理程序、设备驱动程序

C、用户级I/O软件、设备驱动程序、设备无关软件、中断处理程序

D、用户级I/O软件、中断处理程序、设备无关软件、设备驱动程序

考研题6

- 假设磁头当前位于第105道，正在向磁道序号增加的方向移动。现有一个磁道访问序列请求为35、45、12、68、110、180、170、195，采用SCAN算法得到的磁道访问序列为（）。09

  A、110、170、180、195、68、45、35、12

  B、110、68、45、35、12、170、180、195

  C、110、170、180、195、12、35、45、68

  D、12、35、45、68、110、170、180、195

- 下列选项中，不能改善磁盘I/O性能的是（）12

  A. 重排I/O请求次序

  B. 在一个磁盘上设置多个分区

  C. 预读和滞后写

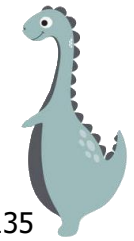
  D. 优化文件物理块的分布

考研题7

- 假设计算机系统采用CSCAN（循环扫描）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）请说明在上述条件下如何进行磁盘块空闲状态管理。
- （2）设某单面磁盘旋转速度为每分钟6000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（如图所示），磁道号请求队列为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，则读完这个扇区点共需要多少时间？要求给出计算过程。

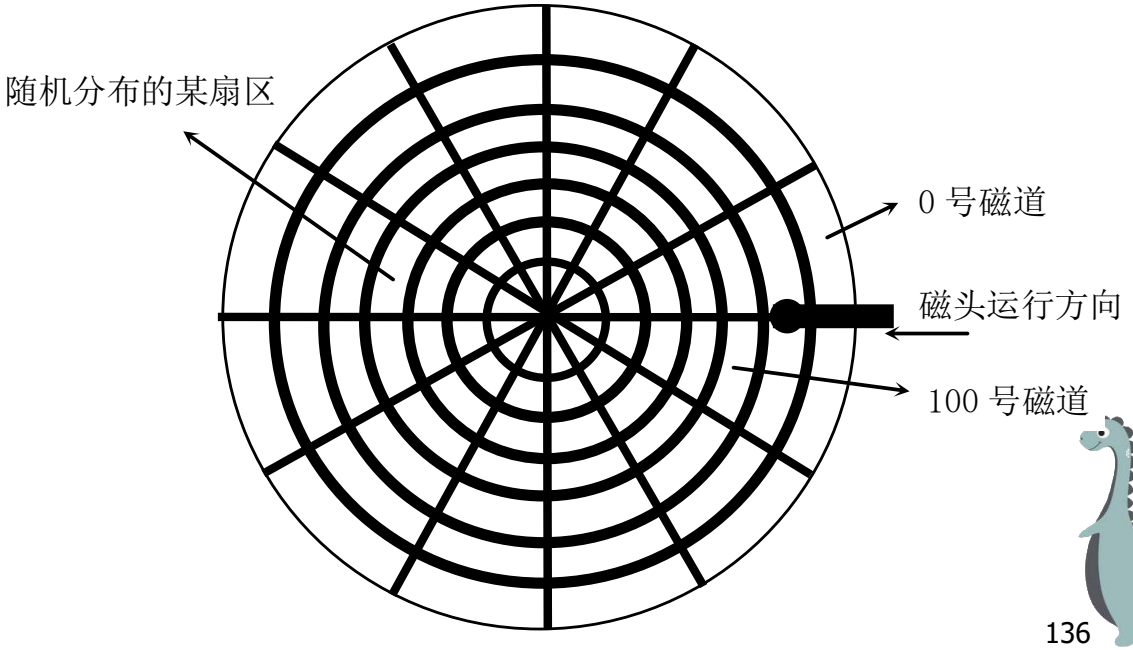
考研题7-2

- （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。



考研题7-3

- （1）使用位示图法表示磁盘的空闲状态（1分），每一位表示一个磁道块是否为空闲，共需要16384/32=512个字=512×4个字节=2KB，正好可放在系统提供的内存中（1分）。



考研题7-4

- （2）采用CSCAN调度算法，访问磁道的顺序为120、30、50、90，则移动磁道长度为20+90+20+40=170，总的移动磁道时间为170×1ms=170ms（1分）。
- 每分钟6000转，则平均旋转廷迟为60/(6000×2)=5ms，总的旋转廷迟时间=5ms×4=20ms（1分）。
- 每分钟6000转，则读取一个磁道上一个扇区的平均读取时间为10ms/100=0.1ms,总的读取扇区的时间=0.1ms×4=0.4ms。
- 读取上述磁道上所有扇区所花的总时间=170ms+20ms+0.4ms=190.4ms（1分）。

考研题7-5

- （3）采用FCFS（先来先服务）调度策略更高效（1分）。因为Flash半导体存储器的物理结构不需要考虑寻道时间和旋转廷迟，可直接按I/O请求的先后顺序服务（1分）。
- 提示：Flash存储器（闪存）属可改写ROM，是一种长寿命的非易失性（在断电情况下仍能保持所存储的数据信息）的存储器，数据删除不是以单个的字节为单位而是以固定的区块为单位，区块大小一般为256KB到20MB。