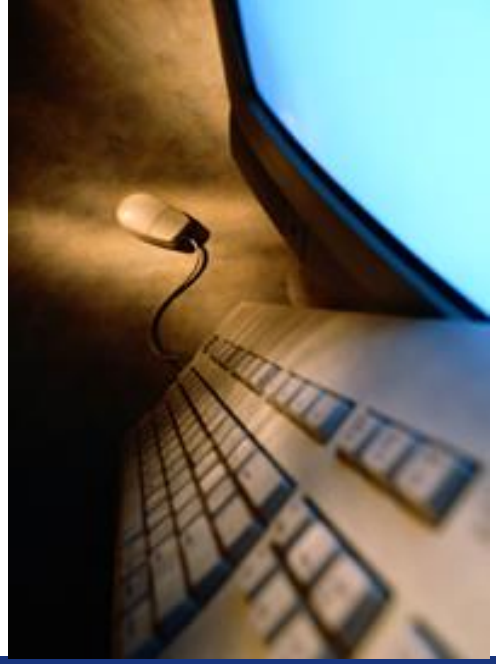
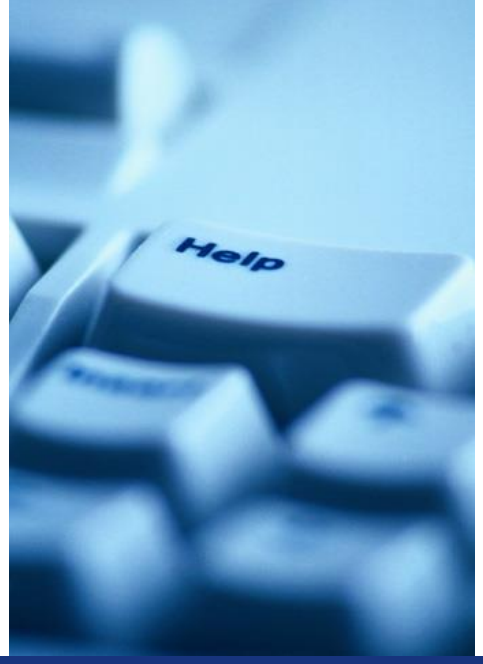
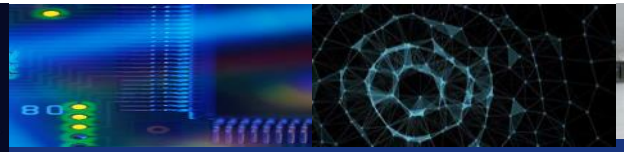



操作系统原理

Operating Systems Principles

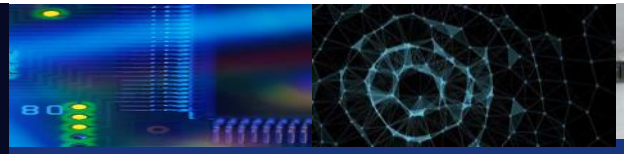
张青
计算机学院

第十二讲 — I/O管理



目标

- 剖析操作系统的I/O子系统；
- 讨论I/O硬件的原理和复杂性；
- 解释I/O硬件和软件的性能问题；

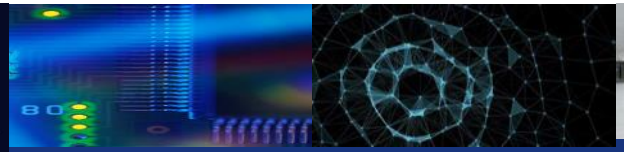


概述

- **计算机设备的控制是操作系统设计人员的主要关注之一**
  - I/O设备的功能和速度差异很大，需要不同方法来控制设备
  - 对于I/O设备的控制方法构成了内核的I/O子系统
- **计算机设备包括：存储设备（磁盘、磁带）、传输设备（网络连接、蓝牙）和人机交互设备（屏幕、键盘、鼠标、音视频输入和输出），以及其它专用设备（比如军事设备）**
- **I/O设备技术呈现两个冲突趋势：**
  - 软件和硬件的接口标准化日益增长
  - I/O设备的种类也日益增多，导致有些新设备与以前设备因差异太大而难以集成到计算机和操作系统
- **I/O设备的要素：Ports(端口), busses（总线）, device controllers（设备控制器）connect to various devices**
- **Device drivers （设备驱动程序） 封装设备的细节与特点**
  - 为I/O子系统提供了统一的设备访问接口

I/O硬件

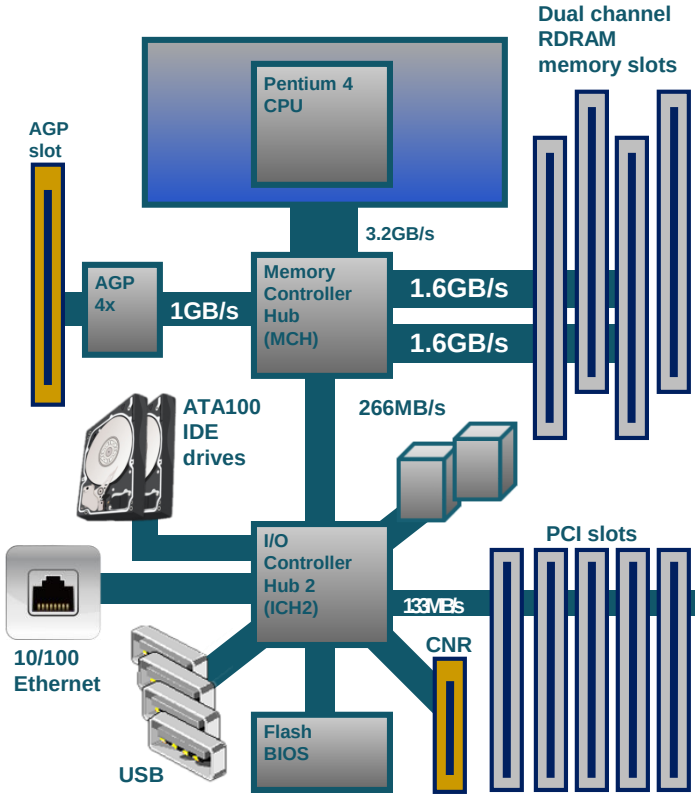
- 端口：设备与计算机的通信经由的连接点
- 总线：一组线路和通过线路传输信息的严格定义的一个协议
- 控制器：操作端口、总线或设备的一组电子器件
- 控制器具有一个或多个寄存器，用于数据和控制信号。处理器通过读写这些寄存器的位模式来于控制器通信
  - ✓ 一种方式是，通过特殊I/O指令针对I/O端口地址传输一个字节或字。I/O指令触发总线线路，选择适当设备，并将位移入或移除设备寄存器
  - ✓ 另一种是，设备控制器支持内存映射I/O。此时，设备控制器被映射到处理器的地址空间。处理器执行I/O请求是通过标准数据传输指令读写映射到物理内存的设备控制器
- 有些系统使用上述两种技术。如，PC使用I/O指令来控制一些设备，而使用内存映射I/O来控制其他设备



I/O端口

- **I/O端口通常由四个寄存器组成，包括状态、控制、数据输入和数据输出寄存器**
  - ✓ 数据输入寄存器被主机读出以获取数据
  - ✓ 数据输出寄存器被主机写入以发送数据
  - ✓ 状态寄存器包含一些主机可以读取的位，例如当前命令是否完成、数据输入寄存器中是否有数据可以读取、是否出现设备故障等
  - ✓ 控制寄存器可由主机写入，以便启动命令或更改设备模式。例如串口控制寄存器中的一位选择全工通信或单工通信，另一位控制启动奇偶校验检查，第三位设置字长为7或8位，其他位选择串口通信支持的速度等
- **数据寄存器的大小通常为1-4字节。有些控制器有FIFO芯片，可以保留多个输入或输出字节，以便在数据寄存器大小的基础上拓展控制器的容量**

I/O结构



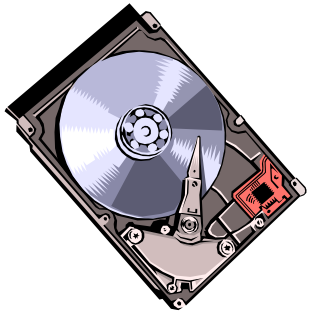
北桥

- 内存
- AGP/PCI-Express
- Built-in display

南桥

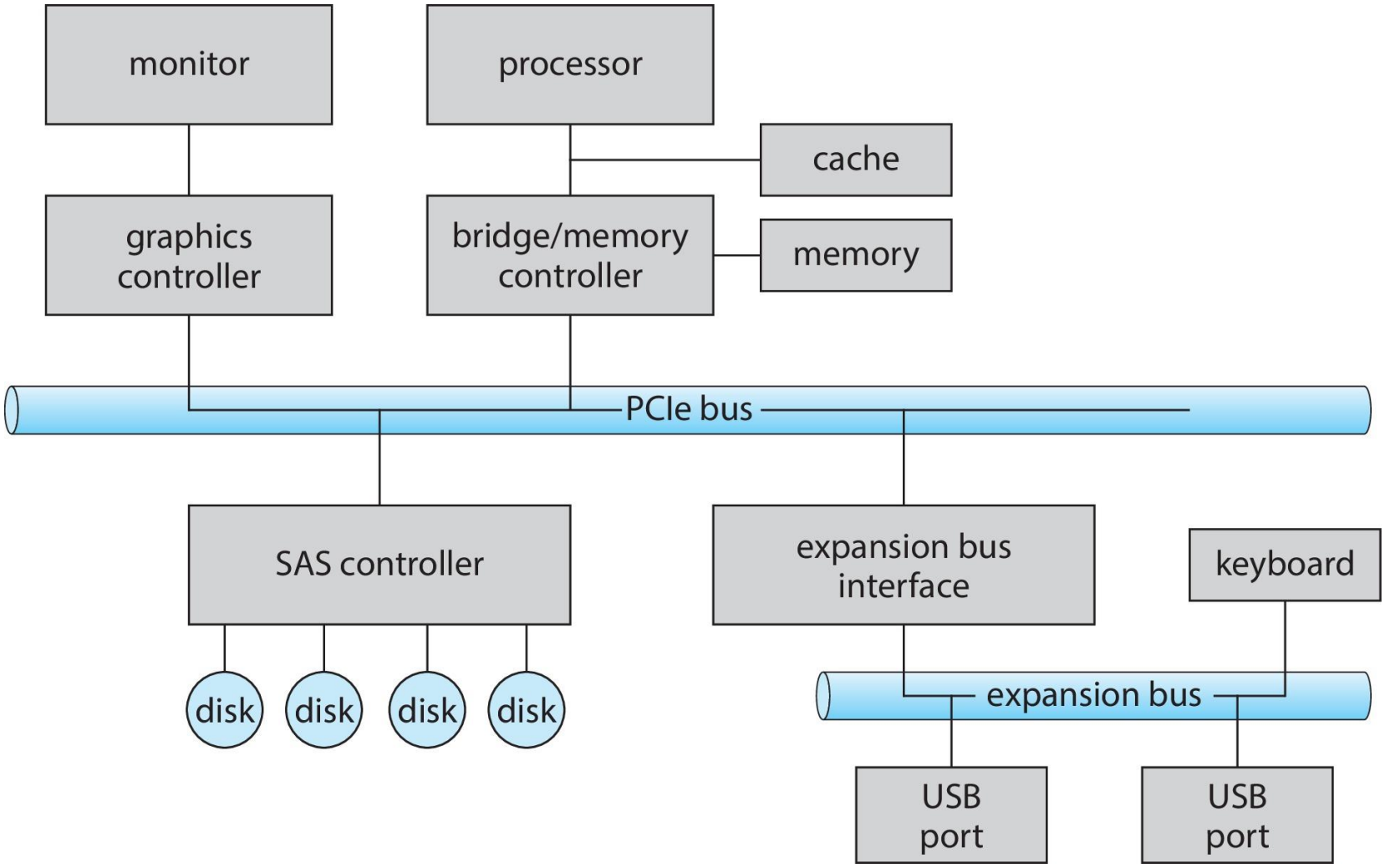
- ATA/IDE
- PCI总线
- USB/Firewire总线
- Serial/Parallel接口
- DMA 控制器
- Interrupt控制器
- RTC, ACPI, BIOS, …

I/O设备分类



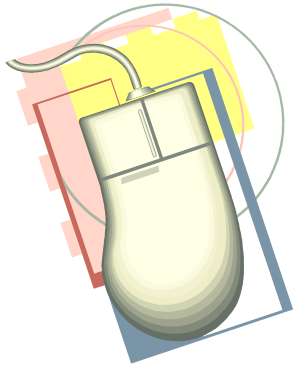
- 计算机系统中参与I/O的外设大体上分为3类:
  - 种类繁多的I/O设备
    - 存储
    - 传输
    - 人机交互
  - 通用概念－I/O设备与计算机接口的信号
    - 端口－设备的连接点
    - 总线--一组线路和通过线路传输信息的严格定义的一个协议；
      - PC和服务器中常见的PCI总线，PCI Express（PCIe）
      - 扩展总线连接相对较慢的设备
      - 串行连接SCSI（SAS）公共磁盘接口

I/O 总线


monitor
processor
cache
graphics controller
bridge/memory controller
memory
PCIe bus
SAS controller
expansion bus interface
keyboard
disk
disk
disk
disk
expansion bus
USB port
USB port

I/O设备之间的差别

- 各类别设备之间的差别很大，同一类别不同设备差别也很大



Data Rate

- there may be differences of magnitude between the data transfer rates

Application

- the use to which a device is put has an influence on the software

Complexity of Control

- the effect on the operating system is filtered by the complexity of the I/O module that controls the device

Unit of Transfer

- data may be transferred as a stream of bytes or characters or in larger blocks

Data Representation

- different data encoding schemes are used by different devices

Error Conditions

- the nature of errors, the way in which they are reported, their consequences, and the available range of responses differs from one device to another

典型I/O设备的速率

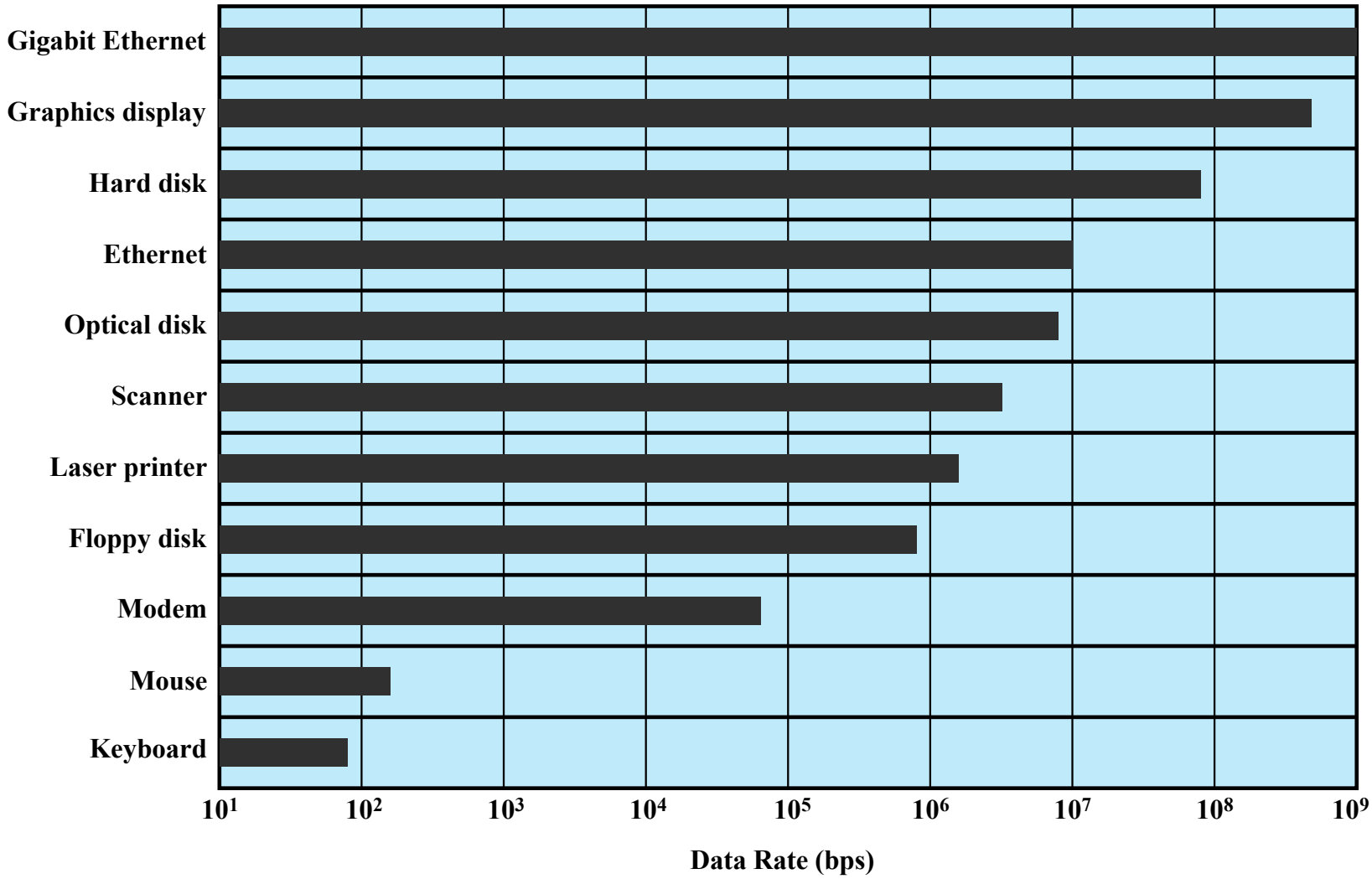


Figure 11.1 Typical I/O Device Data Rates

I/O 控制器

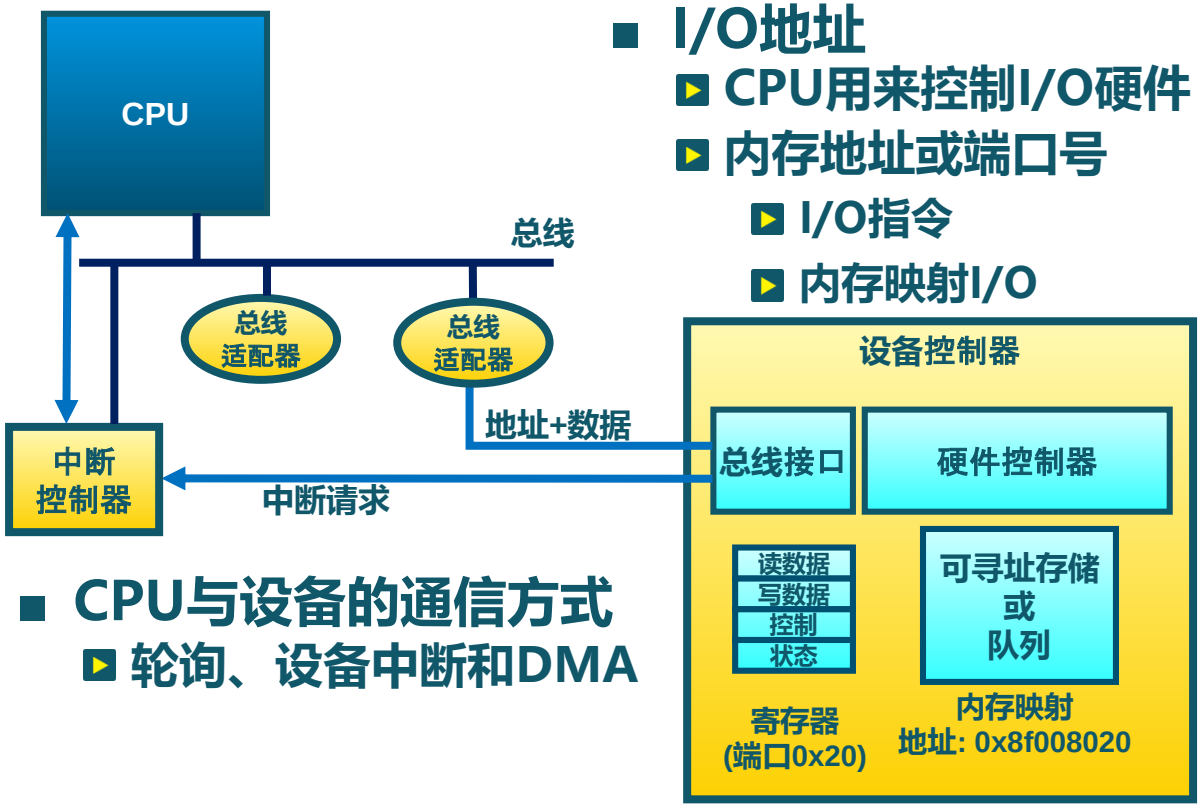
- **Controller (host adapter)** – electronics that operate port, bus, device （可以操作端口、总线和设备的一组电子器件）
  - Sometimes integrated （集成）;
  - Sometimes separate circuit board (host adapter)
  - Contains processor, microcode, private memory, bus controller, etc.
    - Some talk to per-device controller with bus controller, microcode, memory, etc.

I/O硬件

- **Fibre channel (FC) is complex controller, usually separate circuit board (host-bus adapter, HBA) plugging into bus**
- **I/O instructions control devices**
- **Devices usually have registers where device driver places commands, addresses, and data to write, or read data from registers after command execution**
  - Data-in register, data-out register, status register, control register
  - Typically 1-4 bytes, or FIFO buffer

I/O结构

- 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
- I/O地址
  - CPU用来控制I/O硬件
  - 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

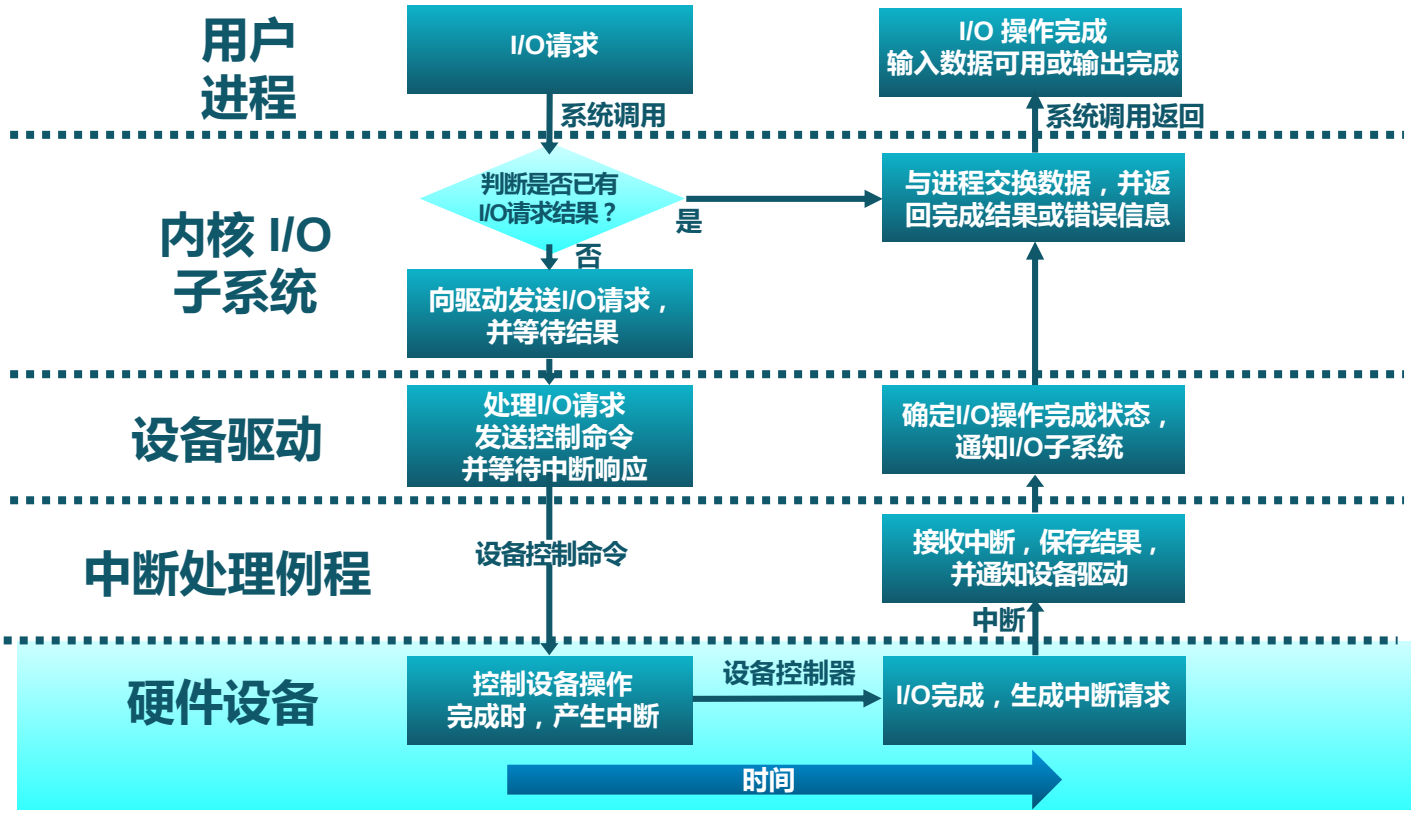
I/O指令和内存映射I/O

- I/O指令
  - 通过I/O端口号访问设备寄存器
  - 特殊的CPU指令
    - out 0x21,AL
- 内存映射I/O
  - 设备的寄存器/存储被映射到内存物理地址空间中；
  - 通过内存load/store指令完成I/O操作；
  - MMU设置映射，硬件跳线或程序在启动时设置地址；

PC设备中的I/O端口位置

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

I/O请求生存周期


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 技术

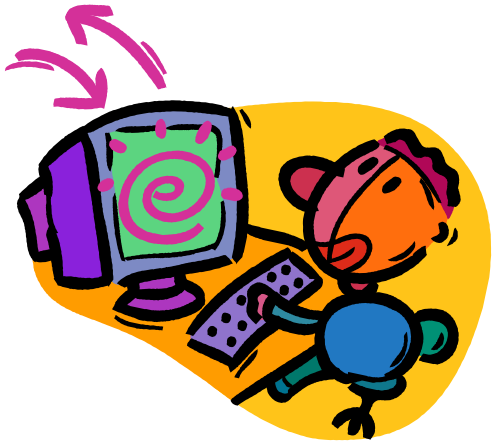


Table 11.1 I/O Techniques

	No Interrupts	Use of Interrupts
I/O-to-memory transfer through processor	Programmed I/O 轮询（polling）	Interrupt-driven I/O
Direct I/O-to-memory transfer		Direct memory access (DMA)

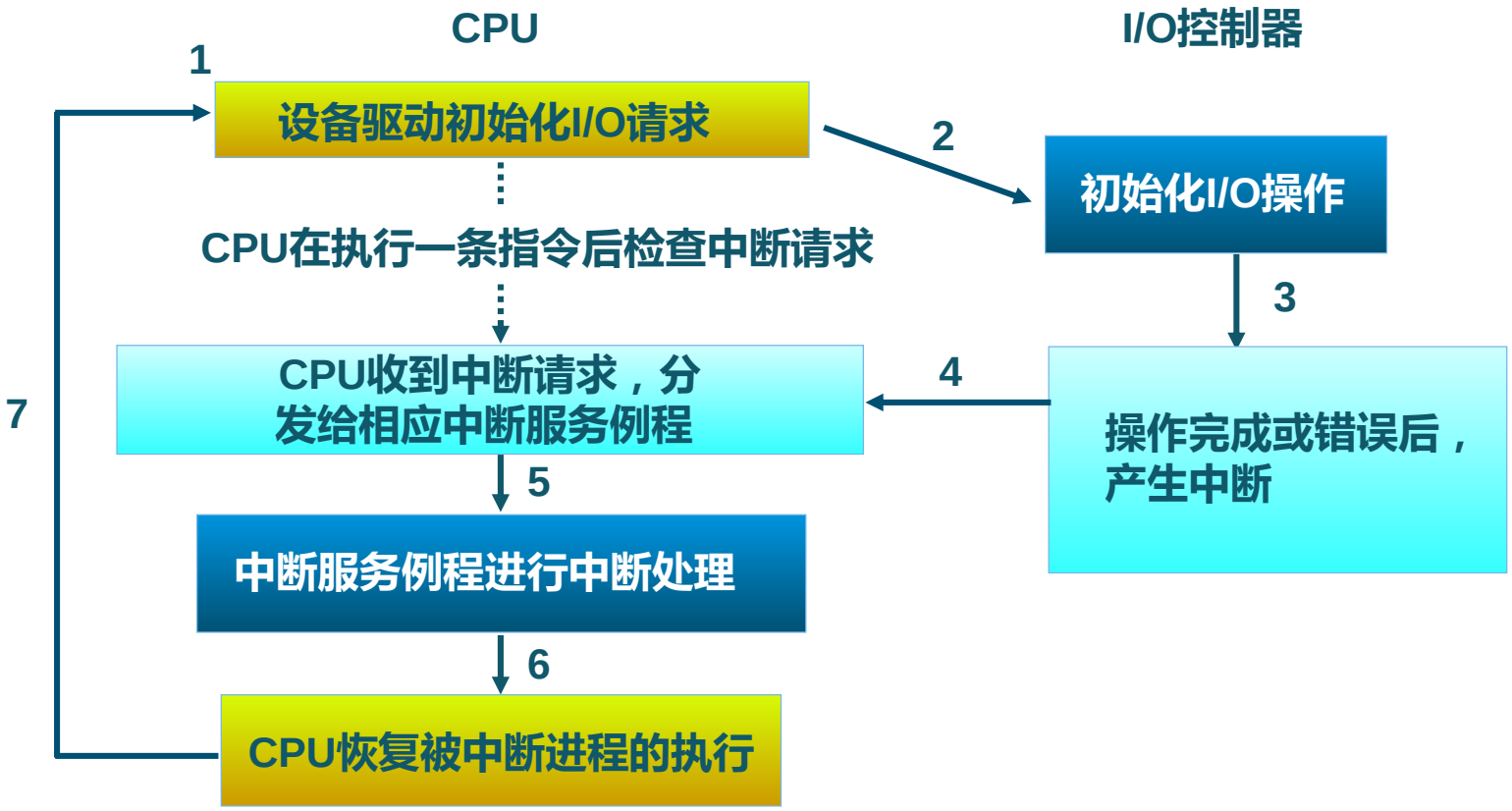
轮询（polling）

- **当主机需要通过端口来输出数据时，主机与控制器之间的握手协调如下：**
  1. 传输主机重复从状态寄存器读取忙位（busy bit），直到该位清零
  2. 主机设置命令寄存器的写位，并写出一个字节到数据输出寄存器
  3. 主机设置命令就绪位
  4. 当控制器注意到命令就绪位已设置，则设置忙位
  5. 控制器读取命令寄存器，并看到写命令。它从数据输出寄存器中读取一个字节，并向设备执行I/O操作
  6. 控制器清除命令就绪位，清楚状态寄存器的故障位表示设备I/O成功，清除忙位表示完成；
- **步骤1是等待设备I/O的忙等待周期或轮询**
  - 循环读取状态寄存器，直到忙位被清除
  - 如果设备速度快，则合理
  - 如果等待时间太长，主机可能切换到另一任务
  - 当数据是来自串口或键盘的数据流时，如果主机等待太久再来读取数据，其小缓冲器可能会因溢出而出现数据被覆盖/丢失

中断

- 轮询可以在3个指令周期内发生
  - 读取设备寄存器，逻辑AND以提取状态位，根据是否为0进行跳转。基本轮询操作是高效的
  - 如不断重复轮询，主机很少发现就绪设备，同时其他需要使用处理器处理的工作又不能完成，轮询就低效了
- 当设备准备好服务时直接通知处理器，而不是要求CPU重复轮询I/O完成，效率会更高。这种让设备通知CPU的硬件机制称为中断
- CPU有中断请求线。中断机制接受一个地址，根据这个数字从一个集合选择一个特定中断处理程序，这个地址称为中断向量的表中一个偏移量
- 中断处理程序接收中断
  - 可以忽略或延迟某些中断
- 中断向量将中断分派给正确的处理程序
  - 开始和结束时的上下文切换
  - 基于优先级
  - 两条中断请求线：非屏蔽中断（用于诸如不可恢复的内存错误等事件）和可屏蔽中断（可由CPU关闭，它由设备控制器用来请求服务）；
  - 如果多个设备处于同一中断编号，则形成中断链；

中断I/O过程


CPU
I/O控制器
1
设备驱动初始化I/O请求
2
初始化I/O操作
CPU在执行一条指令后检查中断请求
3
CPU收到中断请求，分
发给相应中断服务例程
4
操作完成或错误后，
产生中断
7
5
中断服务例程进行中断处理
6
CPU恢复被中断进程的执行

中断的其他用处

- 中断机制也可以用于异常
  - 例如除以0，访问保护的或不存在的内存地址，或尝试执行源自用户模式的特权指令
  - 触发中断的事件有一个共同点：这些事件导致操作系统执行进击的自包含的程序
- 许多操作系统采用中断机制来进行虚拟内存分页，页面错误是引发中断的异常
- 中断可以用于系统调用的实现
- 中断也可用来管理内核的控制流
- 多CPU系统可以并发处理中断
- 多线程内核架构非常适合实现多优先级中断
- 中断大量用于时间敏感的处理，所以高性能系统要求高效的中断

中断向量（Intel Pentium）

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

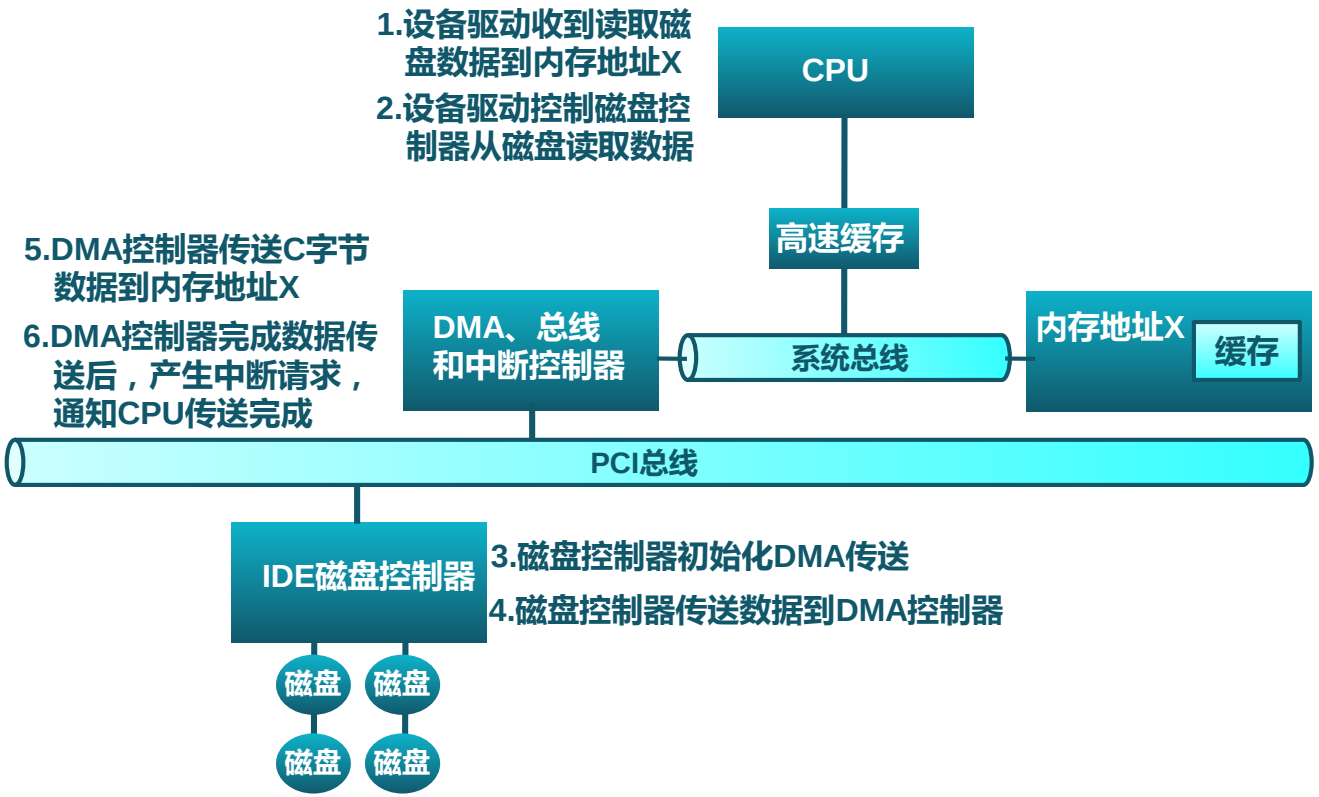
0-31为非屏蔽中断，用于表示各种错误条件的信号

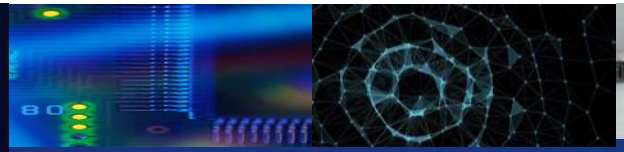
32-255为可屏蔽中断，用于设备产生的中断

DMA

- 对于执行大容量传输的设备，如磁盘驱动器，如果通过昂贵的通用处理器来观察状态位并且按字节来发送数据到控制寄存器（称为程序控制I/O, PIO），存在大量性能浪费
- 为避免因PIO而增加CPU负担，将一部分任务交给直接内存访问（DMA）控制器，绕过 CPU，直接在I/O设备和内存之间传输数据
- 启动DMA传输时，主机将 DMA 命令块写入内存
  - 块包含传输来源和目标地址的指针、传输的字节数
  - CPU将命令块的地址写到DMA控制器
  - DMA控制器直接操作内存总线，将地址放到总线，在没有主CPU的帮助下执行传输。当DMA控制器占用内存总线时，CPU被暂时阻止访问内存，但仍可以访问主缓存或辅助缓存，这种从 CPU 窃取周期虽然减慢CPU计算，但效率更高
  - 当完成整个传输时，DMA控制器中断CPU
  - DMA控制器与设备控制器之间的握手，由一对成为DMA请求和DMA确认的线路来进行。有数据需传输，设备控制器发送信号到DMA请求线路，并占用内存总线，随后发送信号到DMA确认线路，并开始传输数据到内存
- 直接虚拟内存访问（DVMA）更高效

DMA过程举例

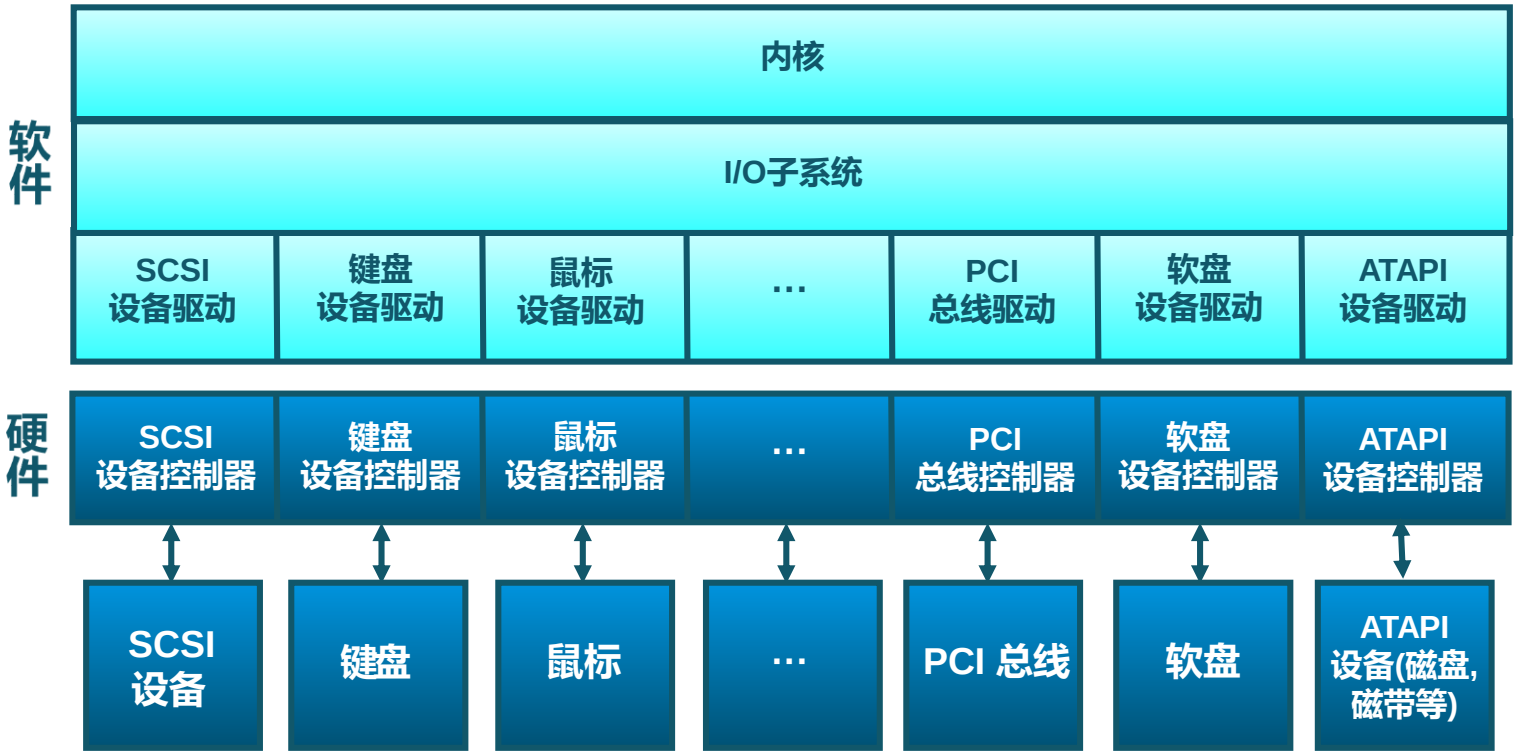

1.设备驱动收到读取磁盘数据到内存地址X
2.设备驱动控制磁盘控制器从磁盘读取数据
CPU
高速缓存
5.DMA控制器传送C字节数据到内存地址X
6.DMA控制器完成数据传送后，产生中断请求，通知CPU传送完成
DMA、总线和中断控制器
系统总线
内存地址X
缓存
PCI总线
IDE磁盘控制器
3.磁盘控制器初始化DMA传送
4.磁盘控制器传送数据到DMA控制器
磁盘
磁盘
磁盘
磁盘



应用程序I/O接口

- 将设备行为封装在泛型类中，以便按统一的标准方式来处理I/O设备
  - 如应用程序打开磁盘上的文件而不必知道它在什么磁盘，新的磁盘和其他设备如何添加到计算机而不必中断操作系统
- 设备驱动层为内核I/O子系统隐藏设备控制器之间的差异
- I/O子系统与硬件的分离简化了操作系统开发人员的工作
- 设备在多个维度存在差异
  - 字符流或块
  - 顺序访问或随机访问
  - 同步或异步
  - 共享或专用
  - 操作速度
  - 读写、只读、只写

内核I/O结构


软件
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
硬件
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)

I/O设备的特点

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read–write	CD-ROM graphics controller disk

I/O设备的特点

- 每种操作系统都有自己的设备驱动接口标准。每个给定设备可能有多个设备驱动程序
- 大体上，I/O设备可以被操作系统分组为
  - 块I/O
  - 字符I/O（流）
  - 内存映射文件访问；
  - 网络套接字；

```
brw-rw---- 1 root disk 8, 0 Mar 16 09:18 /dev/sda
brw-rw---- 1 root disk 8, 1 Mar 16 09:18 /dev/sda1
brw-rw---- 1 root disk 8, 2 Mar 16 09:18 /dev/sda2
brw-rw---- 1 root disk 8, 3 Mar 16 09:18 /dev/sda3
```

- 对于I/O设备特定特性的直接操作，通常为逃生/后门，它允许应用程序透明传递任何命令到设备控制器
  - Unix ioctl（）调用将任意位发送到设备控制寄存器，并将数据发送到设备数据寄存器
- UNIX和Linux使用“主要”和“次要”设备编号的元组来标识设备的类型和实例（此处主要8和次要0-4）%ls -l/dev/sda*

块与字符设备

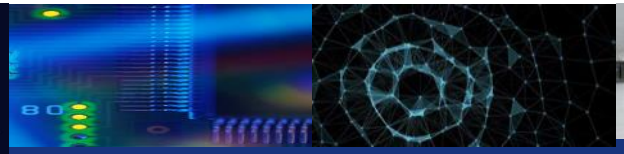
- **块设备接口为磁盘驱动器和其他基于块设备的访问，规定了所需的各个方面，它使应用程序不必关注设备底层差异**
  - 控制命令read()、write()以及seek()—用于指定下一个传输块
  - 原始I/O（将块设备作为简单的线性的块数组来访问），直接I/O（禁止缓冲和锁定的一种文件操作）
  - 内存映射文件的访问可以在块设备驱动程序之上
    - 传输采用与按需分页虚拟内存访问相同的机制来处理
- **字符流接口设备包括键盘、鼠标、串口等**
  - **接口的基本系统调用可以实现get(), put()**
  - 可以构建库以便提供按行访问，并且具有缓冲和编辑功能（例如，当用户输入了一个退格键，可以从输入流中删除前一个字符）

网络设备

- **网络I/O的性能和寻址的特点明显不同于磁盘I/O，大多数操作系统提供的网络I/O接口不同于磁盘的read()-write()-seek()接口**
- **Linux, Unix, Windows等操作系统使用套接字(socket)接口**
  - 将网络协议与网络操作分离
  - 提供函数select()，以便管理一组套接字
- **方法差异很大(pipes, FIFOs, streams, queues, mailboxes)**

时钟和定时器

- 大多数计算机都有硬件时钟和定时器，以便提供三种基本功能：
  - ✓ 获取当前时间
  - ✓ 获取经过的时间
  - ✓ 设置定时器，以便在T时触发操作X
- 测量经过时间和触发操作的硬件称为可编程间隔定时器，它可以设置成等待一定的时间，然后触发中断
- 这些功能大量用于操作系统和时间敏感的应用程序。但实现这些函数的系统调用不属于操作系统标准
- 多数计算机硬件时钟生成的中断率为每秒18-60次
- 一般硬件时钟是由高频计数器来构造的，为获得高精度的时钟可挺过设置寄存器来读取

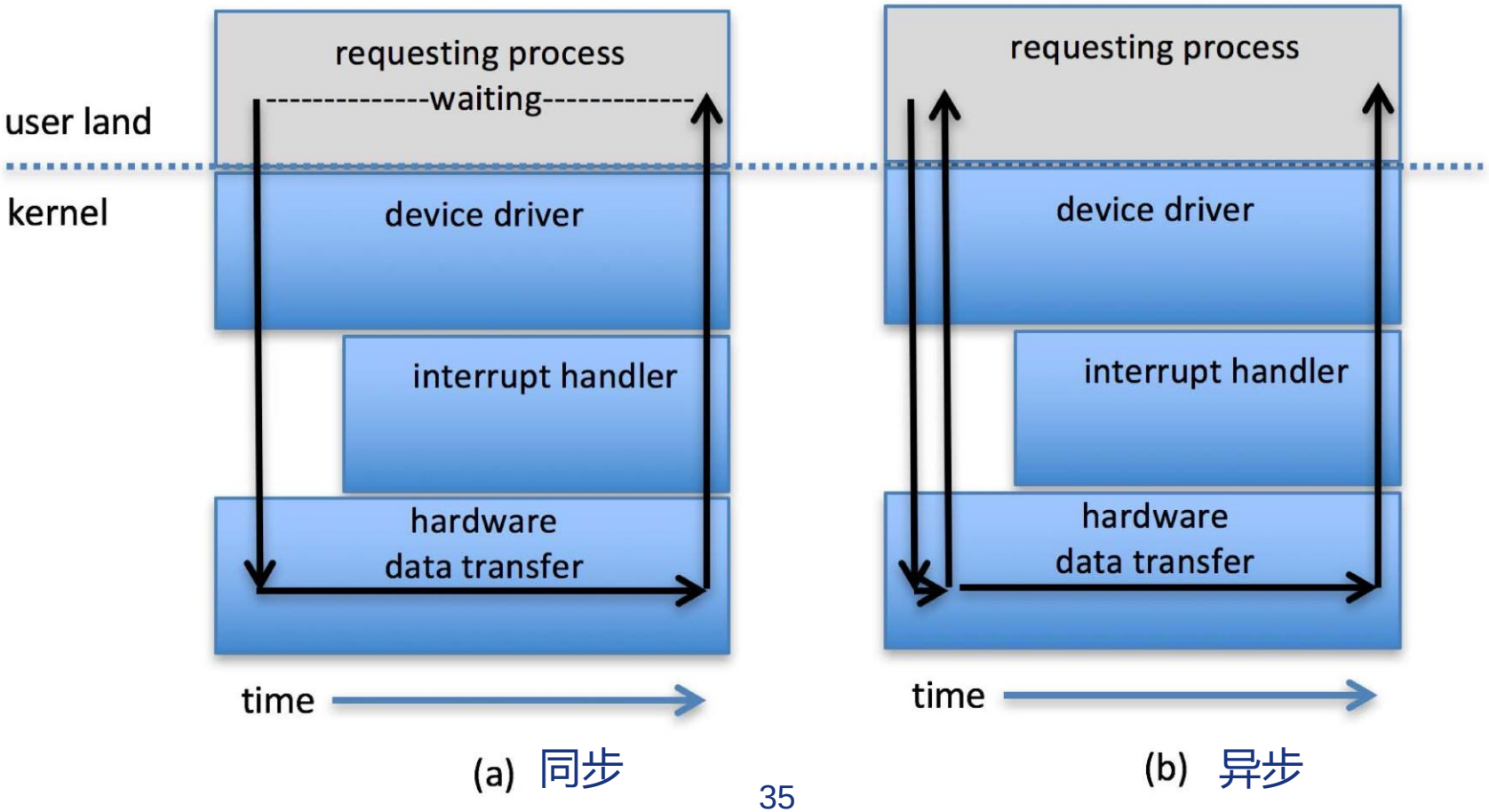


非阻塞和异步I/O

- **阻塞 - 进程挂起直到 I/O 完成**
  - 易于使用和理解
  - 不足以满足某些需求
- **非阻塞 - I/O 调用尽可能多地返回**
  - 如用来接收键盘和鼠标输入，同时处理数据并显示到屏幕
  - 用户界面、数据复制（缓冲 I/O）
  - 通过多线程实现，某些线程可以执行阻塞系统调用，而其他线程继续执行
  - 快速返回读取或写入的字节数
  - select() 查找数据是否准备好然后 read() 或 write()
- **异步 - 进程在 I/O 执行时运行（非阻塞系统调用的替代方法）**
  - 难以使用
  - 当 I/O 完成时，I/O 子系统向进程发出信号

两种I/O方式

非阻塞和异步系统调用区别：非阻塞调用read()立即返回任何可用的数据，读取的数据等于或少于请求的字节数，或为零。异步调用read()要求的输出会完整执行，但是完成是在将来的某个特定时间



向量I/O

- 向量 I/O 允许一个系统调用执行涉及多个位置的多个I/O操作
- 例如Unix系统调用ready接收多缓冲区的一个向量，并且从源读取到向量或将向量写入到目的
- 同一传输可以由多个系统调用的引用来产生
- 这种分散收集的方法比多个单独的I/O调用更好
  - 减少了上下文切换和系统调用开销。如果没有向量I/O，数据可能首先需要按正确顺序传输到较大的缓冲区，然后发送，因此效率低下
  - 程序员应尽可能地利用分散收集I/O功能，以便增加吞吐量并降低系统开销
  - 有些版本分散收集提供原子性，确保所有I/O操作能无间断完成
    - 避免多个线程在读/写时出现数据损坏

内核I/O子系统

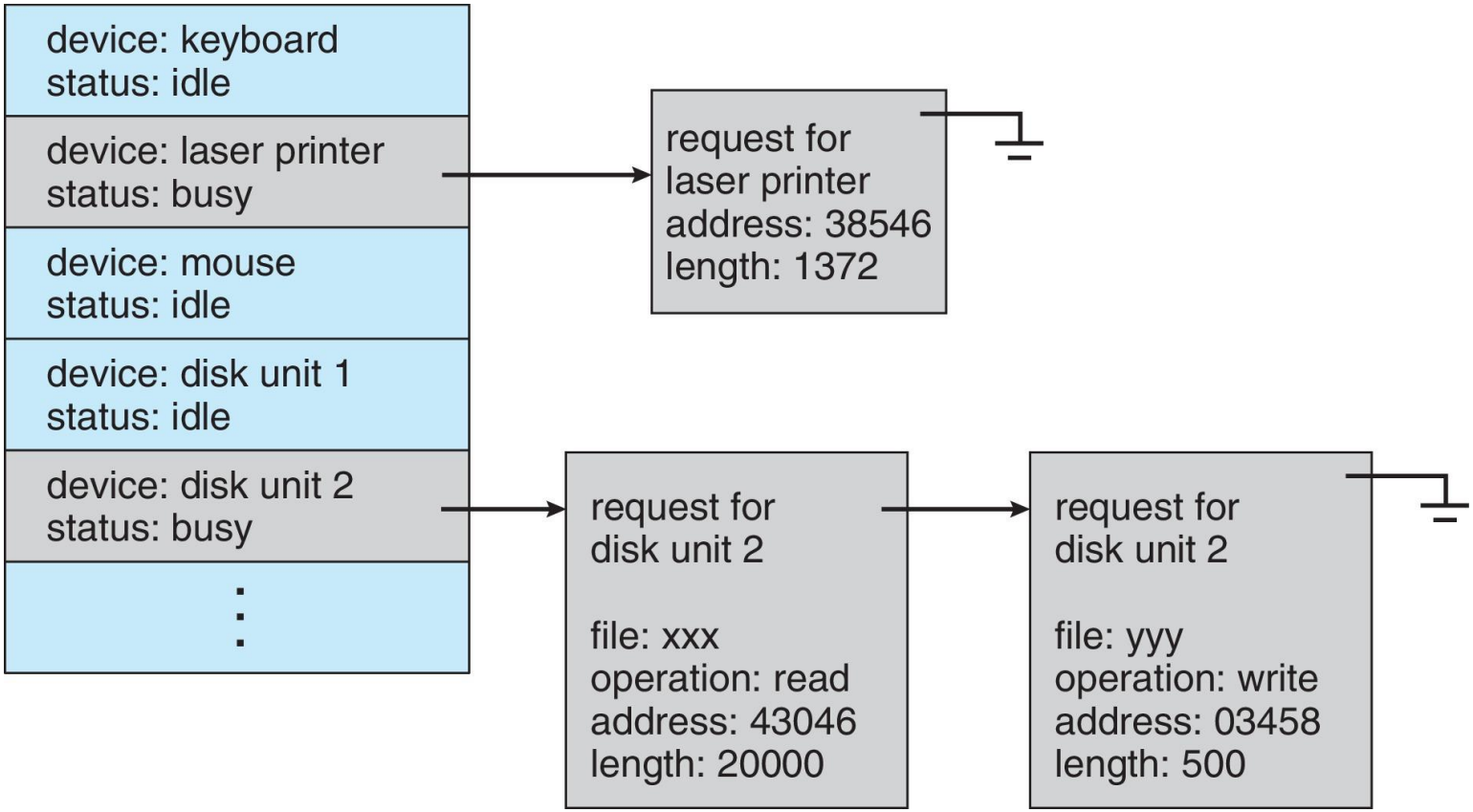
- 内核提供与I/O相关的许多服务：
  - 调度
  - 缓冲
  - 缓存
  - 假脱机
  - 设备预留
  - 错误处理
- 这些服务由内核I/O子系统提供，并建立在硬件和设备驱动程序的基础设施之上
- I/O子系统也负责保护自己免受错误进程和恶意用户的侵扰

内核I/O子系统

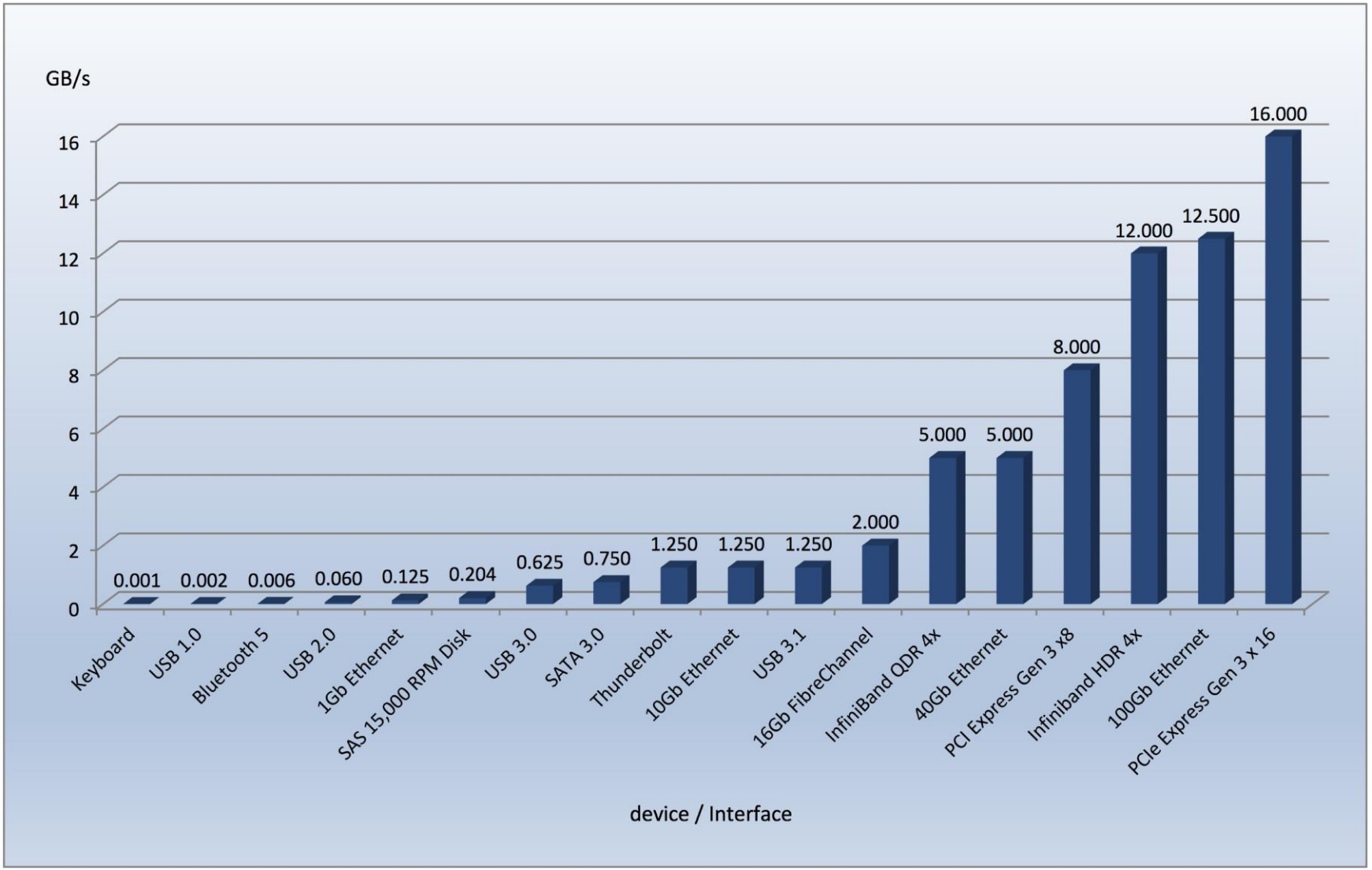
- **Scheduling（确定一组I/O请求的执行顺序）**
  - Some I/O request ordering via per-device queue
  - Some OSs try fairness
  - Some implement Quality Of Service (i.e. IPQOS)
- **Buffering - store data in memory while transferring between devices**
  - To cope with device speed mismatch
  - To cope with device transfer size mismatch
  - To maintain “copy semantics”（写到磁盘的数据版本保证是应用程序系统调用时的版本，而与应用程序缓冲区的更改无关）
  - **Double buffering** – two copies of the data
    - Kernel and user
    - Varying sizes
    - Full / being processed and not-full / being used
    - Copy-on-write can be used for efficiency in some cases

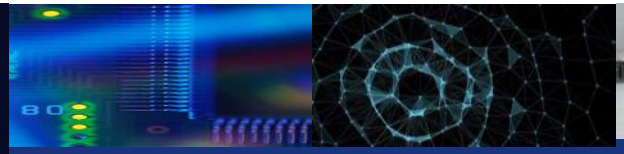
设备状态表

当内核支持异步I/O时，它必须能够同时跟踪许多I/O请求。为此操作系统需要将等待队列附加到设备状态表。内核管理此表，其中每个条目对应每个I/O设备。每个表条目表明设备的状态、地址和状态（不能工作、空闲或忙），以及当前忙于的请求



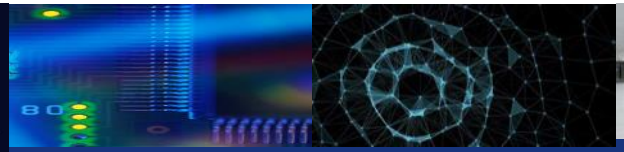
PC和数据中心I/O设备和接口速度





内核I/O子系统

- **Caching - faster device holding copy of data**
  - Always just a copy
  - Key to performance
  - Sometimes combined with buffering
- **Spooling(假脱机) –保存设备输出的缓冲区**
  - If device can serve only one request at a time
  - 比如打印机一次只能打印一个任务，但多个应用程序可能并发打印输出，此时会将所有打印输出先假脱机到一个单独的磁盘文件，之后按假脱机系统排序执行打印
- **Device reservation （设备预留）- provides exclusive access to a device（提供设备的独占访问）**
  - System calls for allocation and de-allocation
  - Watch out for deadlock



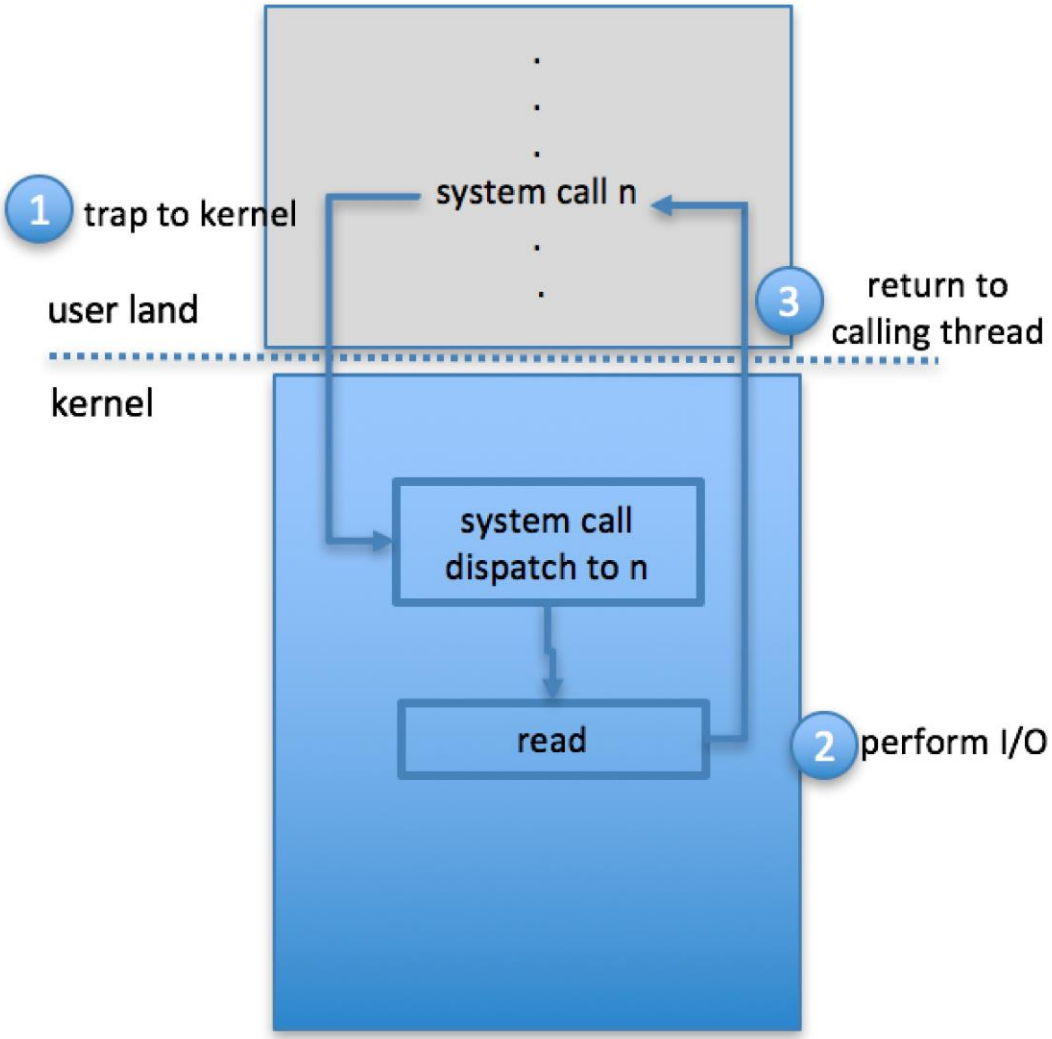
错误处理

- **OS can recover from disk read, device unavailable, transient write failures（瞬时写入失败）**
  - Retry a read or write, for example
  - Some systems more advanced – Solaris FMA, AIX
    - Track error frequencies, stop using device with increasing frequency of retry-able errors
- **Most return an error number or code when I/O request fails**
- **System error logs hold problem reports**

I/O保护

- **User process may accidentally or purposefully attempt to disrupt normal operation via illegal I/O instructions**
  - All I/O instructions defined to be privileged （特权指令）—防止用户自行执行非法I/O操作
  - I/O must be performed via system calls
    - Memory-mapped and I/O port memory locations must be protected too

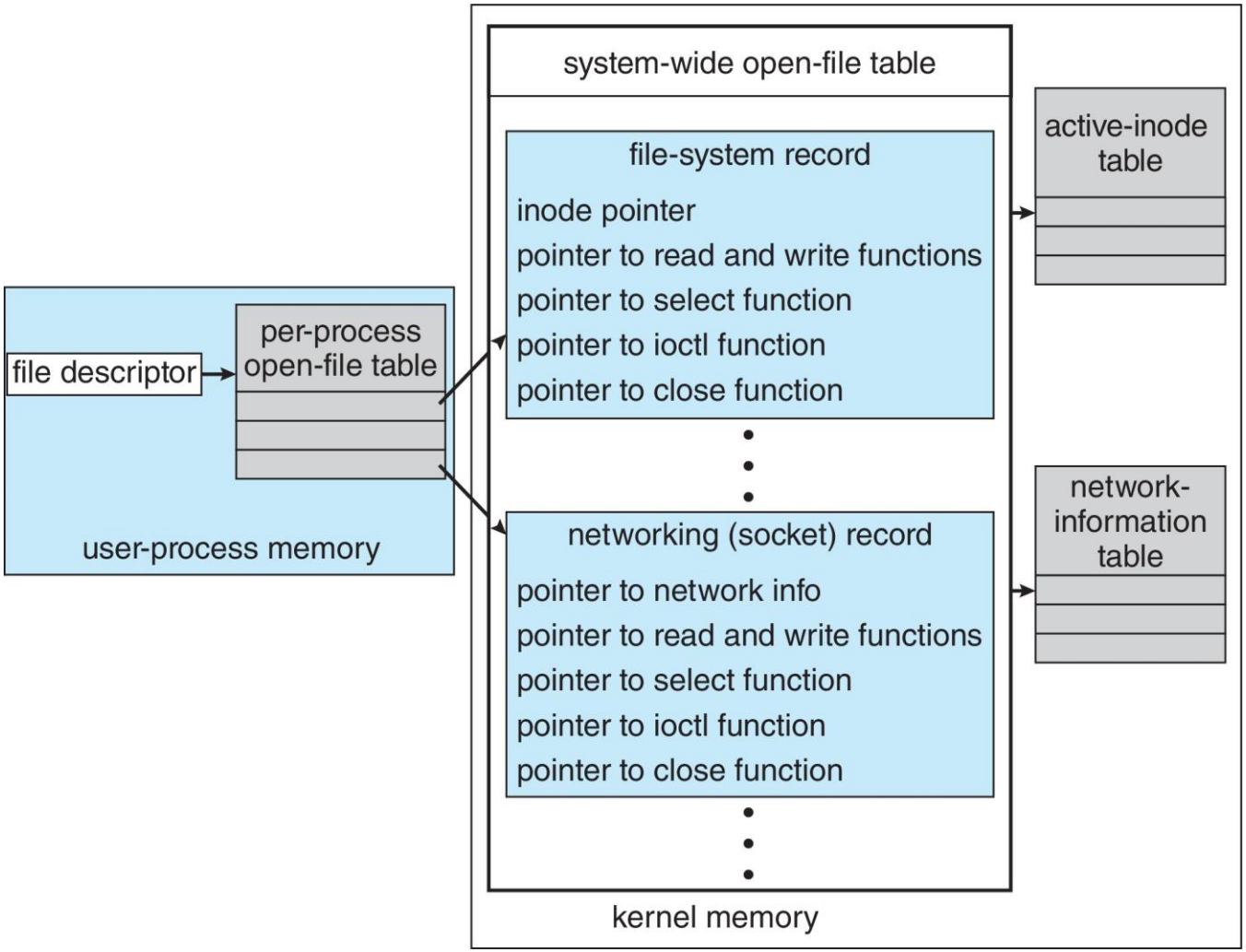
利用系统调用执行I/O

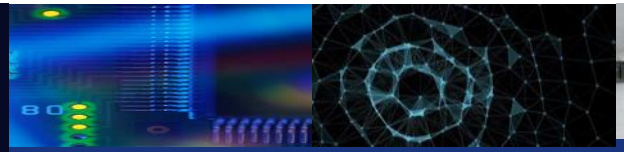

system call n
1
trap to kernel
user land
3
return to
calling thread
kernel
system call
dispatch to n
read
2
perform I/O

内核数据结构

- 内核需要保存I/O组件的状态信息，包括打开的文件表、网络连接、字符设备状态
- 内核使用许多数据结构来跟踪网络连接、字符设备通信和其他I/O活动
- Unix提供各种实体的文件系统访问，如用户文件、原始设备和进程的地址空间
  - 各实体都支持read()但语义不同，采用面向对象技术封装差异
- 有些使用面向对象技术和消息传递来实现I/O
  - Windows使用消息传递
    - 带有从用户模式传递到内核的I/O信息的消息
    - 消息在流向设备驱动程序并返回到进程时被修改
  - 消息传递方法，与采用共享数据结构的程序调用技术相比，可能增加开销，但是简化了I/O系统的结构和设计，增加了灵活性

Unix I/O结构


system-wide open-file table
file-system record
inode pointer
pointer to read and write functions
pointer to select function
pointer to ioctl function
pointer to close function
active-inode table
file descriptor
per-process open-file table
user-process memory
networking (socket) record
pointer to network info
pointer to read and write functions
pointer to select function
pointer to ioctl function
pointer to close function
network-information table
kernel memory



电源管理

- 不是严格意义上的I/O领域，但很多是与I/O相关的
- 计算机和设备使用电，产生热量，经常需要冷却
- 操作系统可以帮助管理和改进使用
  - 云计算环境在服务器之间移动虚拟机
    - 最终可能会分散整个系统并关闭它们
- 移动计算将电源管理作为第一类操作系统

电源管理

- 例如，Android实现了
  - 组件级电源管理
    - 了解组件之间的关系
    - 构建表示物理设备拓扑的设备树
    - 系统总线->I/O子系统->闪存，USB存储}
    - 设备驱动程序跟踪设备的状态，无论是否在使用中
    - 未使用的组件 - 将其关闭
    - 树分支中的所有设备未使用 - 关闭分支

电源管理

- 例如，Android实现（续）
  - 唤醒锁 – 与其他锁一样，但在锁定时防止设备睡眠
  - 电源崩溃 – 使设备进入深度睡眠
    - 边际用电
    - 只有足够清醒才能对外部刺激做出反应（按键、来电）
- **现代系统使用高级配置和电源接口（ACPI）固件，提供作为内核调用的例程运行的代码，用于设备发现、管理、错误和电源管理**

内核I/O子系统总结

- 总之，I/O子系统协调应用程序和内核其他部分可用的大量服务
  - 管理文件和设备的名称空间
  - 对文件和设备的访问控制
  - 操作控制（例如，调制解调器无法查找（）
  - 文件系统空间分配
  - 设备分配
  - 缓冲、缓存和假脱机
  - I/O调度
  - 设备状态监视、错误处理和故障恢复
  - 设备驱动程序配置和初始化
  - I/O设备的电源管理
- I/O子系统的上层通过设备驱动程序提供的统一接口访问设备

I/O请求转成硬件操作

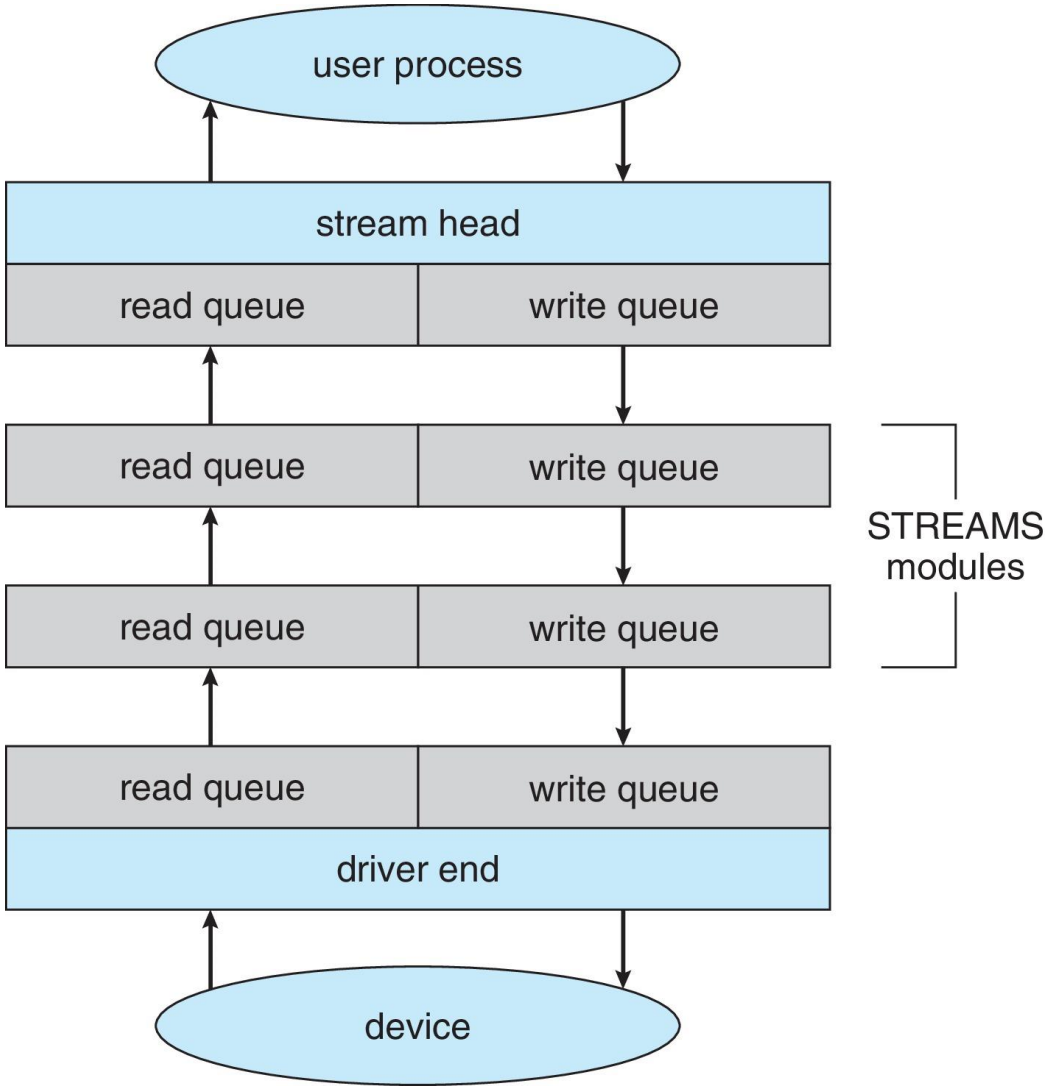
- 没有解释操作系统如何将应用程序请求连到网络线路或特定的磁盘扇区。如何建立从文件名称到磁盘控制器的连接（硬件端口地址或内存映射控制器寄存器）？
- MS-DOS方案：在文件名称中使用冒号，冒号之前表示特定硬件设备的字符串，如C:是主硬盘的每个文件名称的第一部分
- Unix通过常规文件系统名称空间来表示设备名称，与具有冒号分隔符的MS-DOS文件名称不同，Unix路径名称没有明确区分设备部分
  - ✓ Unix有一个安装表，以便将路径名称的前缀与特定设备名称关联。为了解析路径名，Unix检查安装表内的名称，以查找最长的匹配前缀；安装表内的相应条目给出了设备名称
- 现代操作系统在请求和物理设备控制器之间的路径中，具有多个阶段的查找表，具有高度灵活性

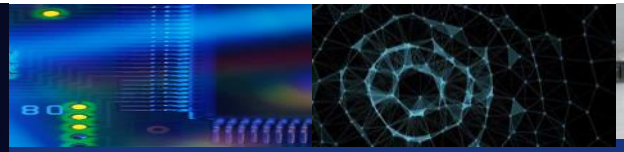
流式I/O

- STREAM－Unix System V及更高版本中用户级进程与设备之间的全双工通信通道
- 流包括：
  - 流头与用户进程接口
  - 驱动程序端与设备接口
  - 它们之间有零个或多个流模块
- 每个模块包含一个读队列和一个写队列

- 消息传递用于队列之间的通信
  - 流量控制选项，指示可用或忙
- 内部异步，用户进程与流头通信的同步

流式I/O

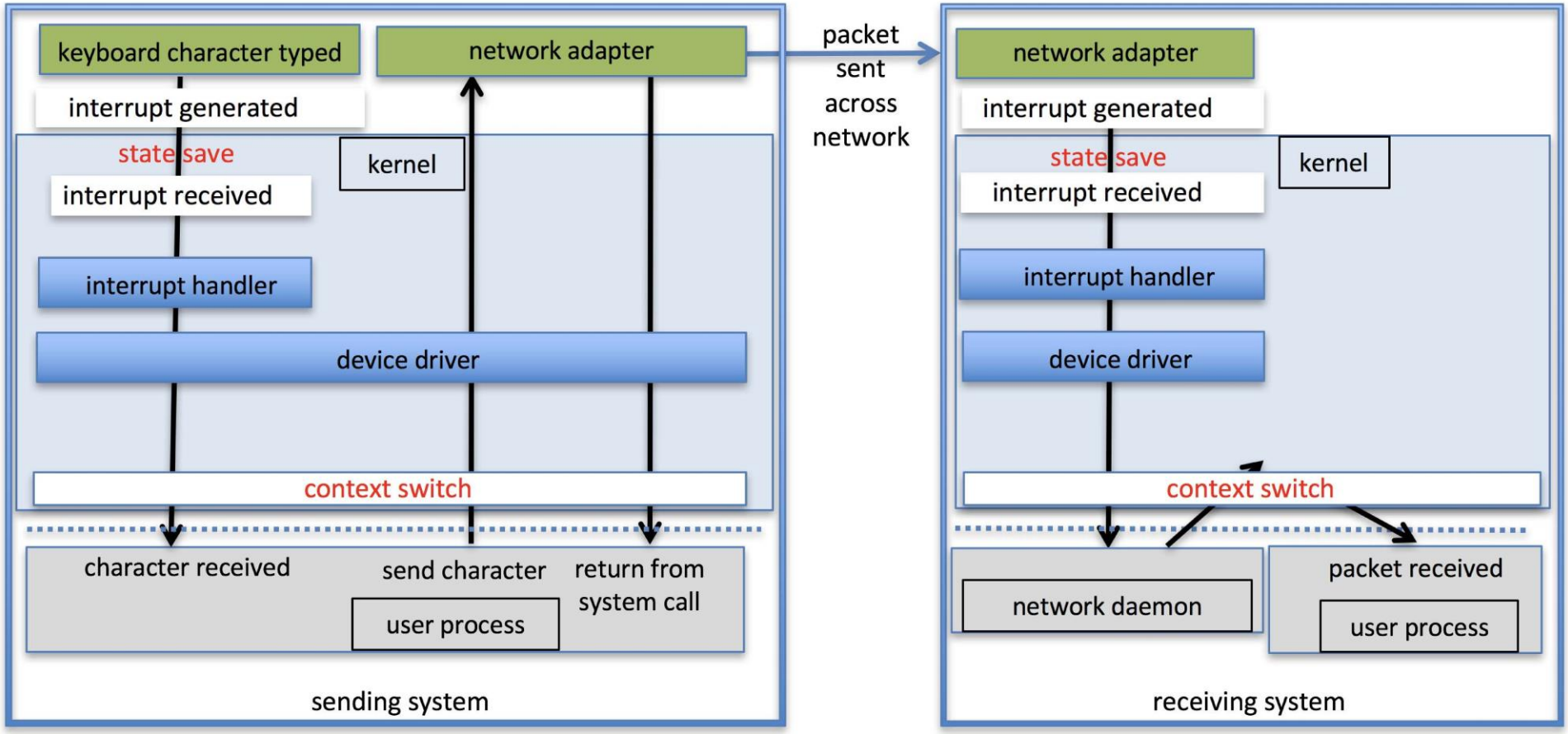

user process
stream head
read queue
write queue
read queue
write queue
read queue
write queue
read queue
write queue
driver end
device
STREAMS
modules



性能

- I/O是影响系统性能的主要因素：
  - 要求CPU执行设备驱动程序、内核I/O代码
  - 中断导致的上下文切换
  - 数据复制
  - 网络流量密度

内部计算机通信


keyboard character typed
network adapter
interrupt generated
state save
kernel
interrupt received
interrupt handler
device driver
context switch
character received
send character
return from system call
user process
sending system
packet sent across network
network adapter
interrupt generated
state save
kernel
interrupt received
interrupt handler
device driver
context switch
network daemon
packet received
user process
receiving system

性能改进

- 减少上下文切换的数量
- 减少数据复制
- 通过使用大型传输、智能控制器和轮询减少中断
- 使用DMA
- 使用更智能的硬件设备
- 平衡CPU、内存、总线和I/O性能以获得最高吞吐量
- 将用户模式进程/守护进程移动到内核线程

小结

- I/O硬件基本要素：总线、设备控制器、设备驱动程序和设备本身
- 设备与主机之间的数据移动方式：轮询、中断、直接内存访问（DMA）
- 应用程序的系统调用接口类型：块设备、字符设备、内存映射文件、网络套接字、可编程间隔定时器，非阻塞与异步I/O
- 内核的I/O子系统服务：I/O调度、缓冲、缓存、假脱机、设备预留以及错误处理
- I/O请求转硬件操作、I/O流架构以及性能

谢谢