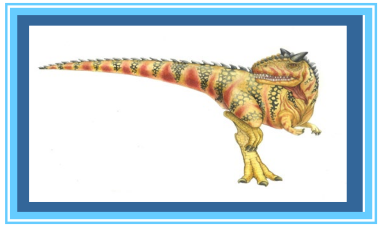
Chapter 13: I/O Systems



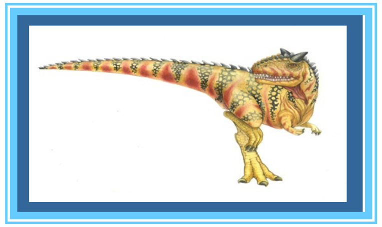
Chapter 13: I/O Systems

- 13.1 Overview
- 13.2 I/O Hardware
- 13.3 Application I/O Interface
- 13.4 Kernel I/O Subsystem
- 13.5 Transforming I/O Requests to Hardware Operations
- 13.6 STREAMS
- 13.7 Performance

Objectives

- Explore the structure of an operating system’s I/O subsystem
- Discuss the principles of I/O hardware and its complexity
- Provide details of the performance aspects of I/O hardware and software

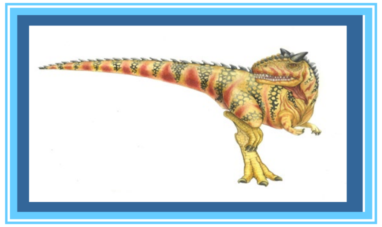
13.1 Overview



Overview

- The two main jobs of a computer:
  - I/O (Input/Output)
  - processing
- The control of devices connneted to the computer is a major concern of operating-system designers.
- I/O设备技术出现两个相矛盾的趋势：
  - 硬件和软件接口日益增长的标准化。
  - I/O设备日益增长的多样性。
- 操作系统内核设计成使用设备驱动程序模块的结构。
- 设备驱动程序为I/O子系统提供了统一接口。

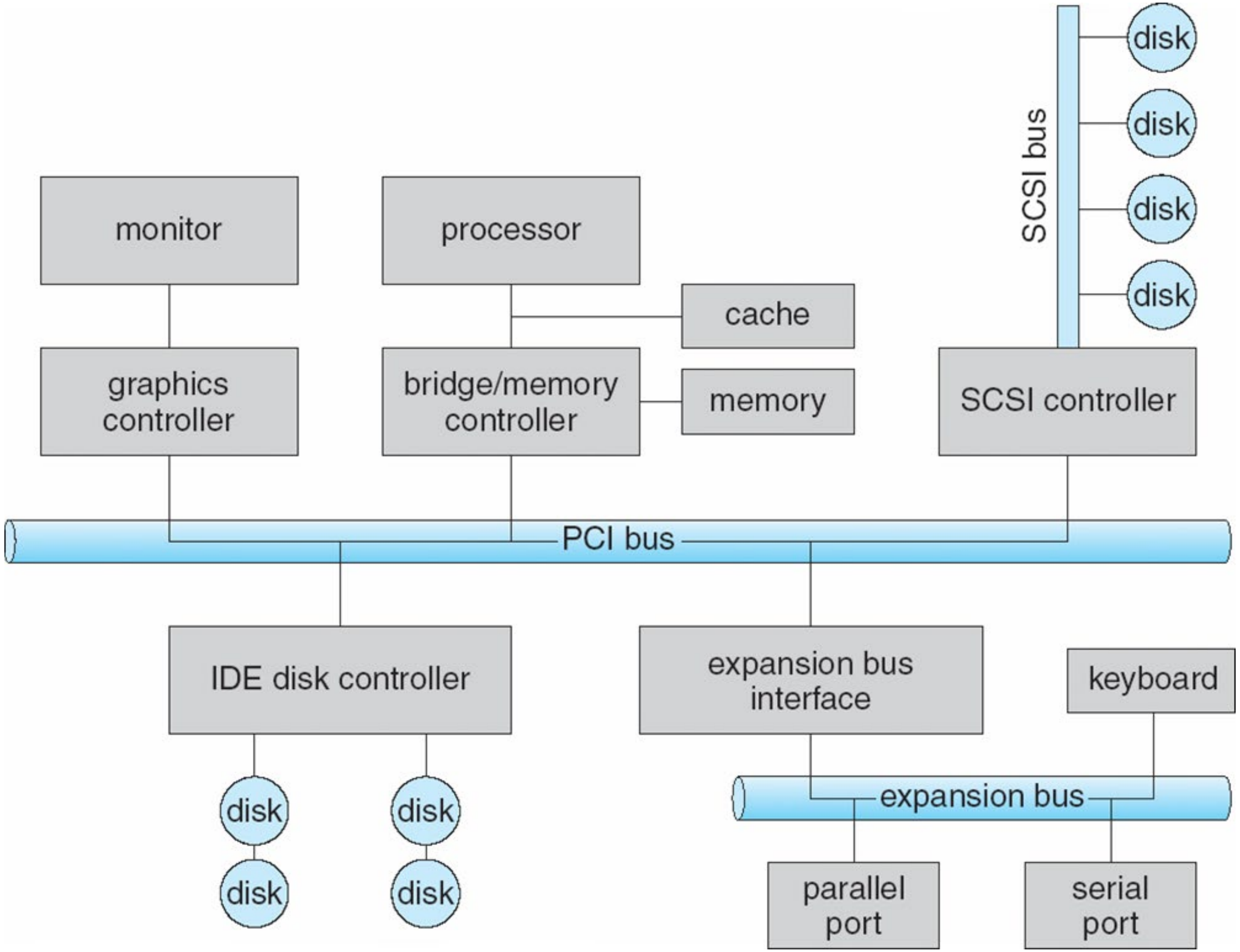
12.2 I/O Hardware



I/O Hardware

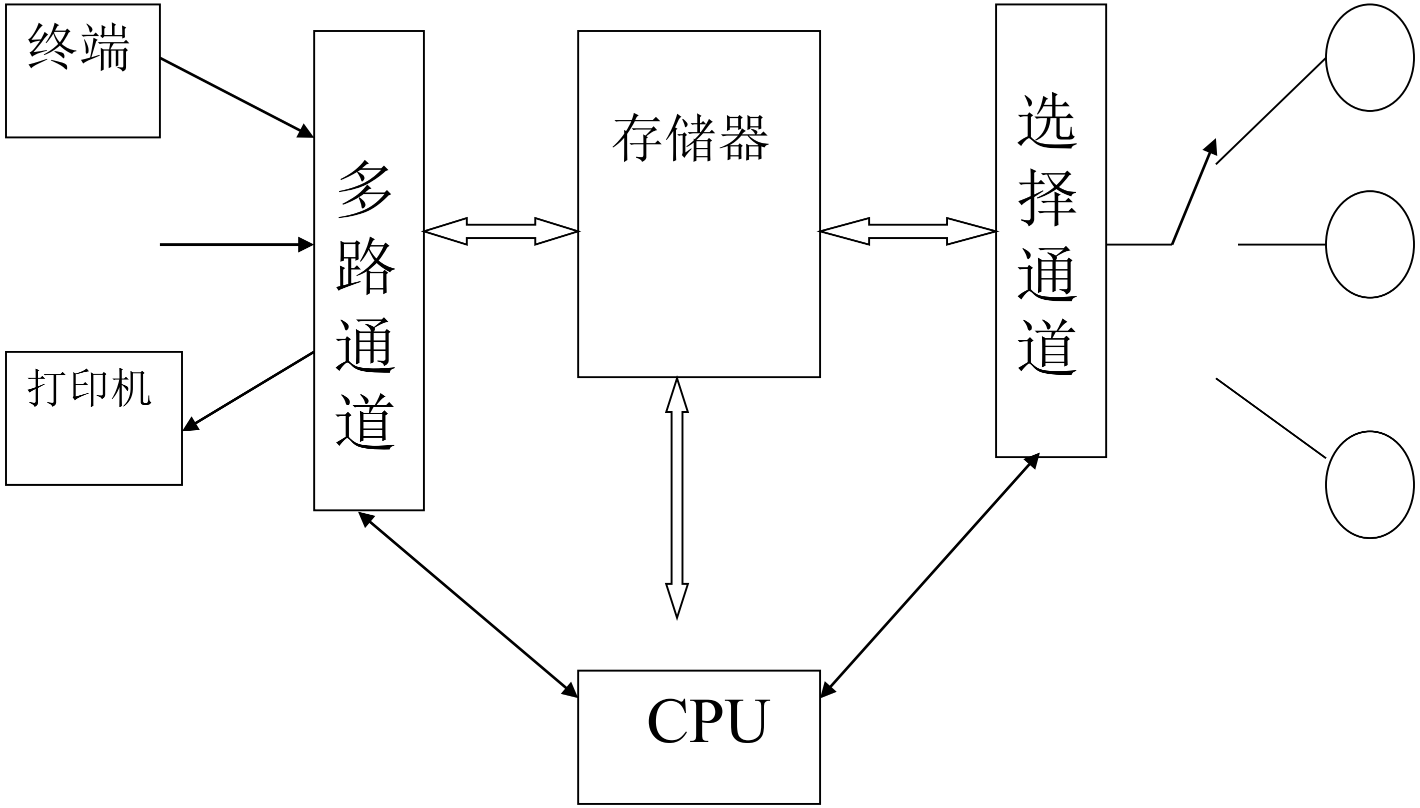
- **I/O系统的组成：**
  - **PC BUS I/O系统 （Fig）**
  - **主机I/O系统（Fig）**
- Common concepts
  - Port， 端口
  - Bus (daisy chain or shared direct access)， 总线
  - Controller (host adapter)， 控制器
- I/O instructions control devices
- Devices have addresses（寻址方式）, used by
  - Direct I/O instructions
  - Memory-mapped I/O

A Typical PC Bus Structure



Mainframe Systems 大型机（主机）系统

- 这类计算机以存储器为中心，CPU和各种通道都与存储器相连。



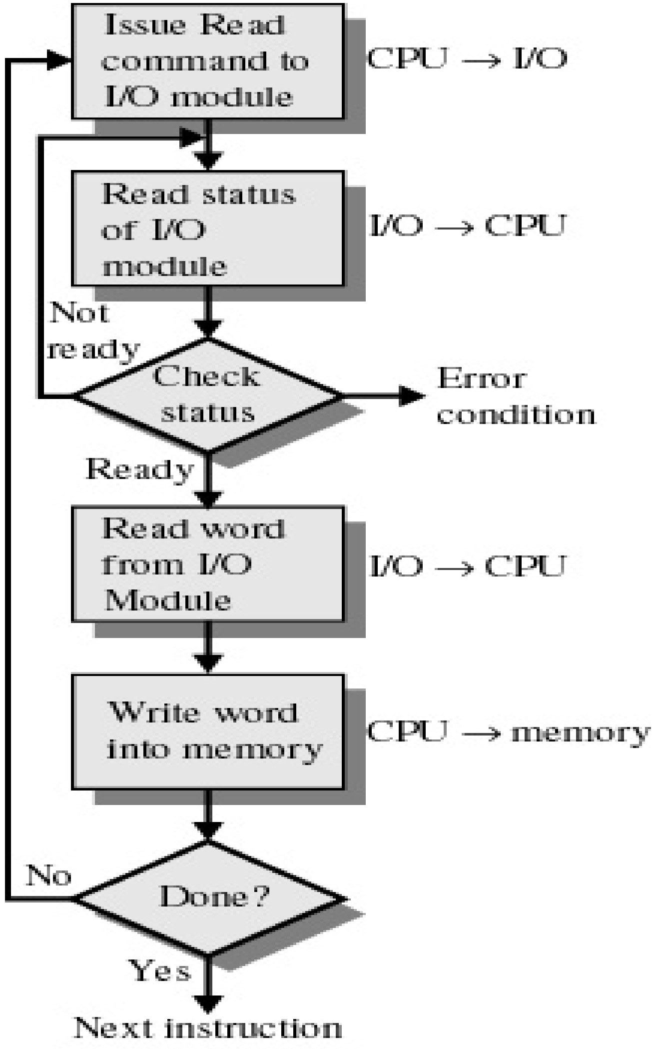
Device I/O Port Locations on PCs (partial)

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方式

一、Polling轮询

- Determines state of device
  - command-ready
  - busy
  - Error
- Busy-wait cycle to wait for I/O from device

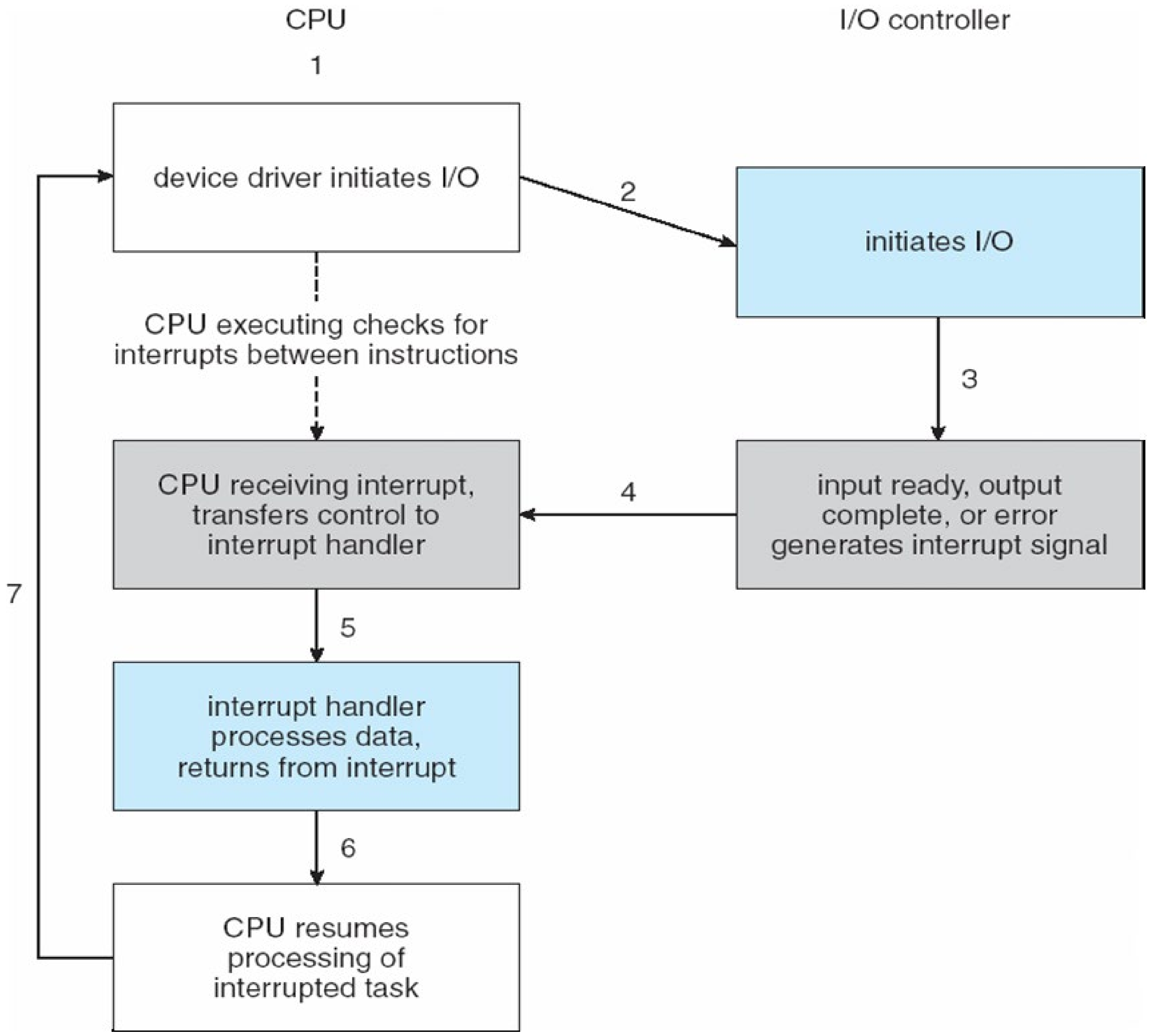


(a) Programmed I/O

二、Interrupts中断

- CPU硬件有一条中断请求线（interrupt-request line, IRL），由I/O设备触发
  - 设备控制器通过中断请求线发送信号而引起中断，CPU捕获中断并派遣到中断处理程序，中断处理程序通过处理设备来清除中断。
- 两种中断请求
  - **非屏蔽中断**：主要用来处理如不可恢复内存错误等事件
  - **可屏蔽中断**：由CPU在执行关键的不可中断的指令序列前加以屏蔽
- **中断向量**
- **中断优先级**：能够使CPU延迟处理低优先级中断而不屏蔽所有中断，这也可以让高优先级中断抢占低优先级中断处理。
- 中断的用途
  - 中断机制用于处理各种异常，如被零除，访问一个受保护的或不存在的内存地址
  - 系统调用的实现需要用到中断（软中断）
  - 中断也可以用来管理内核的控制流

Interrupt-Driven I/O Cycle



Intel Pentium Processor Event-Vector Table

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

Interrupt vectors in Linux

Vector range	Use
0-19	Nonmaskable interrupts and exceptions
20-31	Intel-reserved
32-127	External interrupts (IRQs)
128 (0x80)	Programmed exception for system calls
129-238	External interrupts (IRQs)
239	Local APIC timer interrupt
240	Local APIC thermal interrupt (introduced in the Pentium 4 models)
241-250	Reserved by Linux for future use
251-253	Interprocessor interrupts
254	Local APIC error interrupt (generated when the local APIC detects an erroneous condition)
255	Local APIC spurious interrupt (generated if the CPU masks an interrupt while the hardware device raises it)

An example of IRQ(interrupts Request) assignment to I/O devices

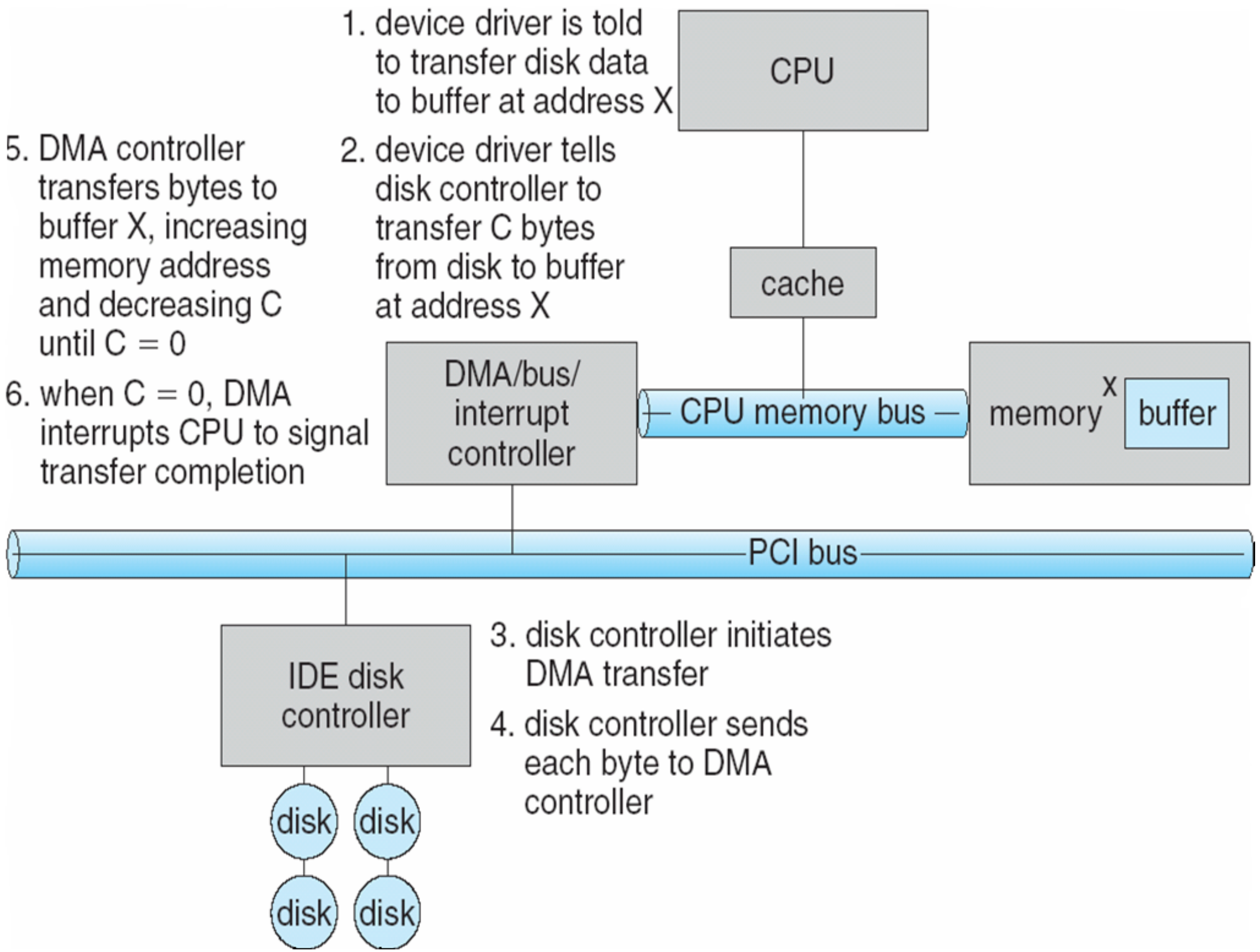
中断向量

IRQ	INT	Hardware device
0	32	Timer
1	33	Keyboard
2	34	PIC cascading
3	35	Second serial port
4	36	First serial port
6	38	Floppy disk
8	40	System clock
10	42	Network interface
11	43	USB port, sound card
12	44	PS/2 mouse
13	45	Mathematical coprocessor
14	46	EIDE disk controller's first chain
15	47	EIDE disk controller's second chain

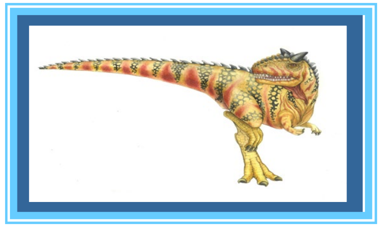
三、Direct Memory Access（DMA）

- Used to avoid programmed I/O for large data movement
- Requires DMA controller
- Bypasses CPU to transfer data directly between I/O device and memory

Six Step Process to Perform DMA Transfer



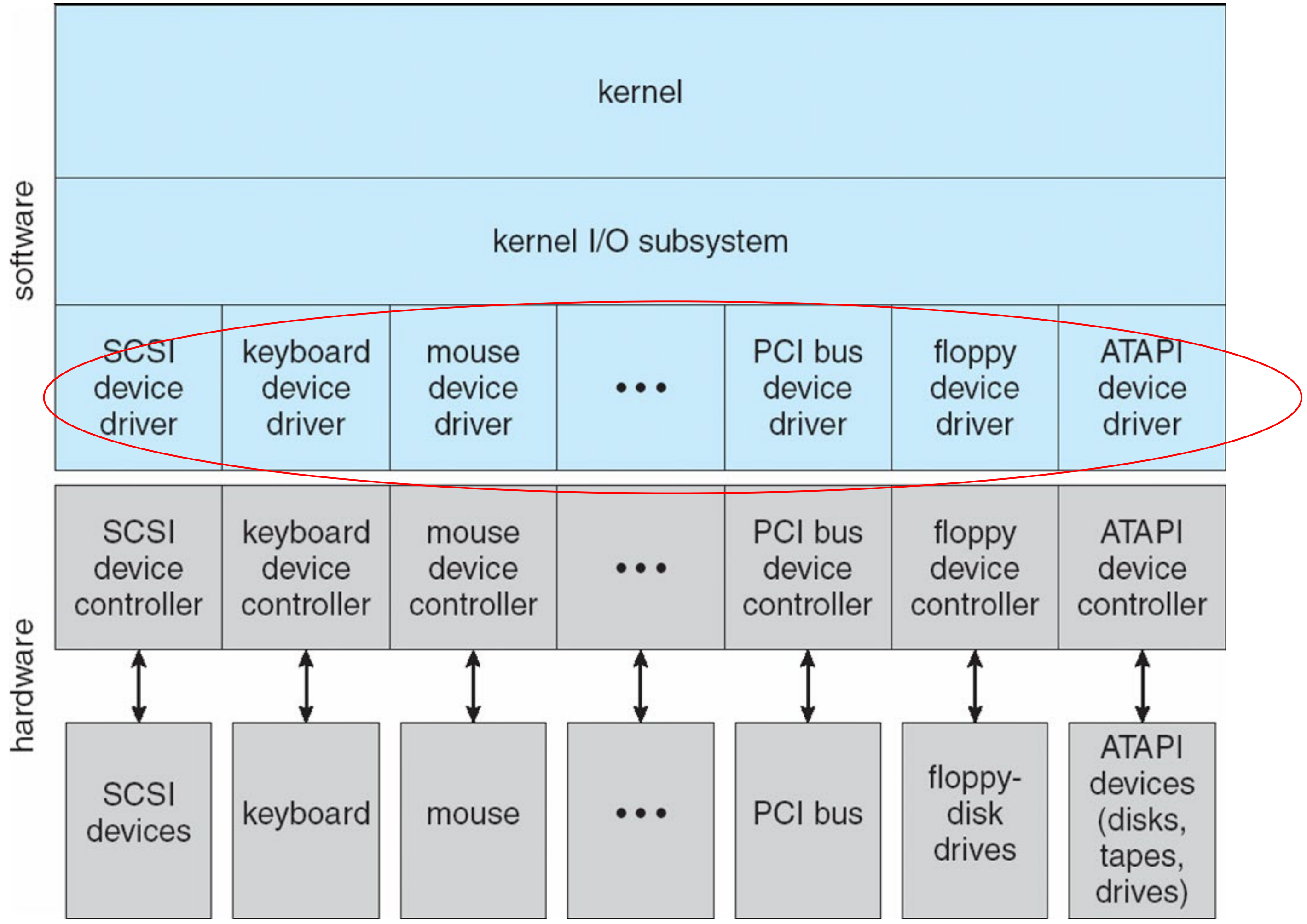
13.3 Application I/O Interface



Application I/O Interface

- **I/O系统调用--实现统一的I/O接口**
- I/O system calls encapsulate device behaviors in generic classes，**如块设备I/O系统调用包括磁盘、磁带、光盘等一系列块设备的read、write、seek。**
- Device-driver(**设备驱动**)layer hides differences among I/O controllers from kernel
- Devices vary in many dimensions
  - **Character-stream or block** 字符流或者块设备
  - **Sequential or random-access** 顺序或随机访问设备
  - **Synchronous or a Synchronous** 同步或异步
  - **Sharable or dedicated** 共享或独占设备
  - **Speed of operation** 操作速度（快速、中速、慢速）
  - **read-write, read only, or write only** 读写、只读、只写设备

A Kernel I/O Structure



Characteristics of I/O Devices

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

Block and Character Devices（块和字符设备）

- **Block devices（块设备）** include disk drives
  - Commands include read, write, seek
  - Raw I/O or file-system access
  - Memory-mapped file access possible

- **Character devices （字符设备）** include keyboards, mice, serial ports
  - Commands include `get()`, `put()`
  - Libraries layered on top allow line editing

Network Devices（网络设备）

- Varying enough from block and character to have own interface

- Unix and Windows NT/9*x*/2000 include socket interface
  - Separates network protocol from network operation
  - Includes `select()` functionality

- Approaches vary widely (pipes, FIFOs, streams, queues, mailboxes)

- Linux：
  - 块设备、字符设备、网卡设备

Clocks and Timers（时钟和定时器）

- 提供以下三个基本函数
  - 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
- 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（programmable interval timer）
  - 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

Blocking and Nonblocking I/O（阻塞和非阻塞I/O）

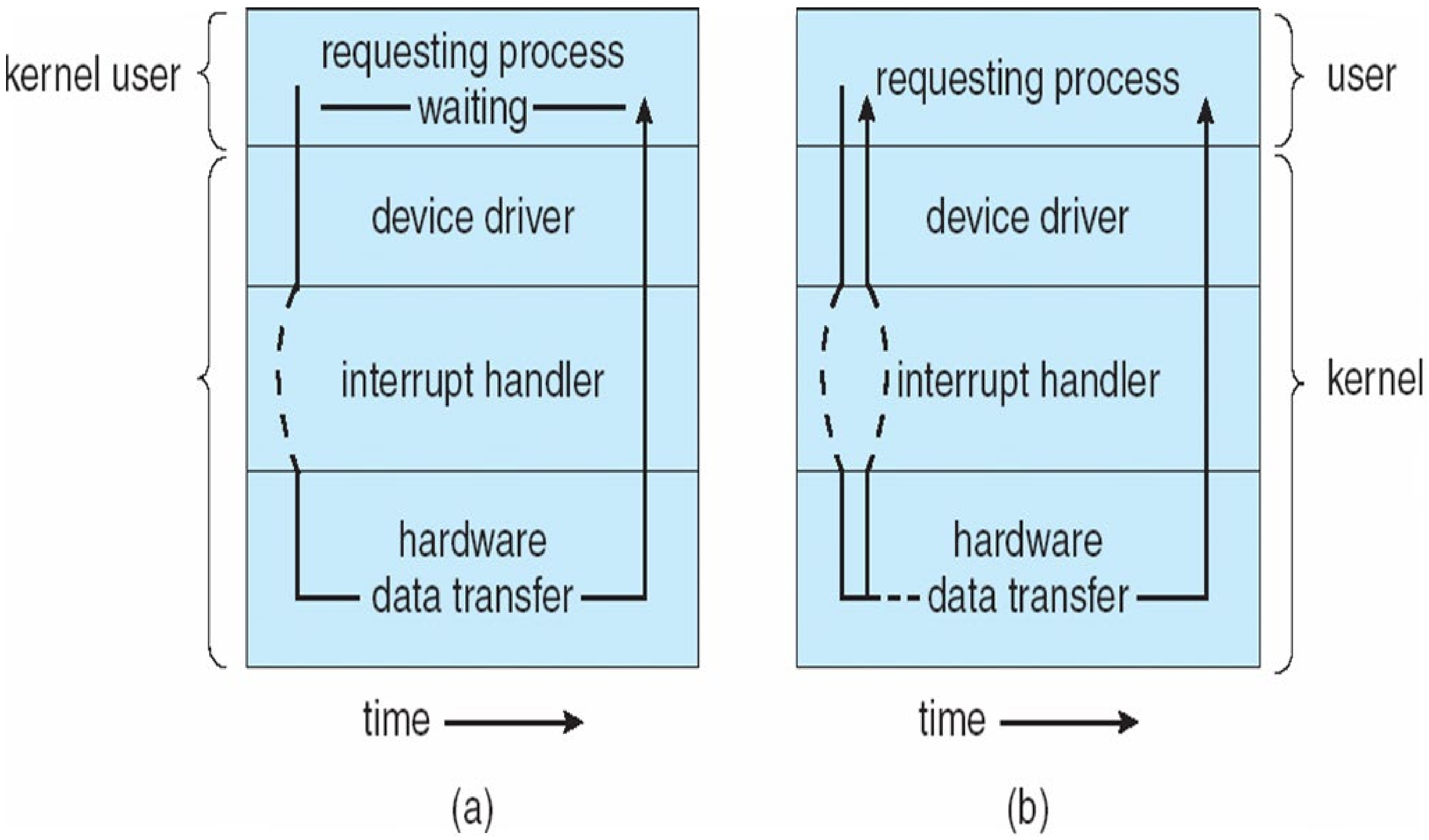
- **Blocking** - process suspended until I/O completed，进程挂起直到I/O完成为止
  - Easy to use and understand
  - Insufficient for some needs
- **Nonblocking** - I/O call returns as much as available，I/O调用立刻返回
  - User interface, data copy (buffered I/O)
  - Implemented via multi-threading
  - Returns quickly with count of bytes read or written

OAsynchronous（异步）

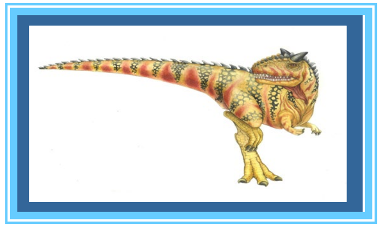
- **Asynchronous（异步）** - process runs while I/O executes，进程与I/0同时运行
  - Difficult to use
  - I/O subsystem signals process when I/O completed

- 非阻塞与异步系统调用的差别**是：**
  - **非阻塞read调用会马上返回**，其所读取的数据可以等于或少于所要求的，或为零；
  - **异步read调用所要求的传输应完整地执行，其具体执行可以是将来某个特定时间。**

Two I/O Methods



13.4 Kernel I/O Subsystem



Kernel I/O Subsystem

- **内核与I/O有关服务：I/O scheduling、buffering、caching、spooling(虚拟化）、device reservation、and error handling.**
- **内核I/O子系统负责：**
  - 文件和设备命名空间的管理
  - 文件和设备访问控制
  - 操作控制（for example,a moderm cannot seek()）
  - 文件系统空间的分配
  - 设备分配
  - 缓冲、高速缓存、假脱机
  - I/O调度
  - 设备状态监控、错误处理、失败恢复
  - 设备驱动程序的配置和初始化

I/O调度

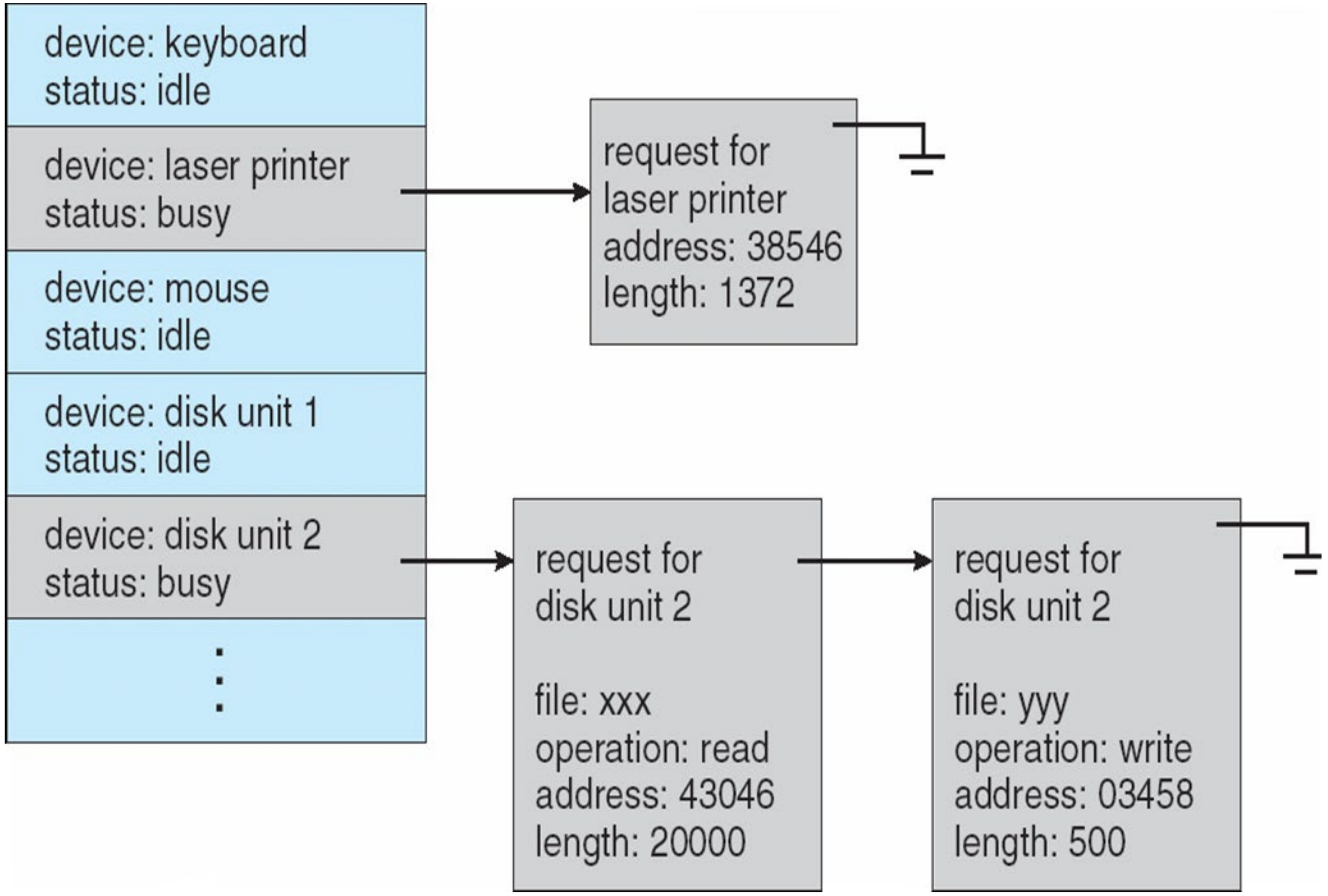
- **Scheduling（I/O调度）调度一组I/O请求就是确定一个好的顺序来执行这些请求。**
  - 某些I/O需要按设备队列的顺序--**先来先服务**
  - 某些操作系统尝试着公平--**优先级高者优先**
  - 磁盘I/O调度

- 实现
  - OS通过为每个设备维护一个请求队列来实现调度。
  - 可以试图公平，也可以根据不同的优先级进行I/O调度。
  - 其他方法：缓冲、高速缓冲、假脱机

缓冲buffer

- 缓冲 Buffering—-store data in memory while transferring between devices，用来保存在两设备之间或在设备和应用程序之间所传输数据的内存区域。。
- 缓冲作用：
  - **解决设备速度不匹配**
  - **解决设备传输块的大小不匹配**
  - **为了维持拷贝语义“copy semantics”要求**
- 缓冲区管理：为了解决**CPU与I/O之间速度不匹配的矛盾**，在它们之间配置了缓冲区。这样设备管理程序又要负责管理缓冲区的建立、分配和释放。
- **单缓冲、双缓冲、多缓冲、缓冲池**

Device-status Table



Kernel I/O Subsystem

- **Caching （高速缓存）** - fast memory holding copy of data
  - 缓冲与高速缓存的差别是缓冲只是保留数据仅有的一个现存拷贝，而高速缓存只是提供了一个驻留在其他地方的数据的一个高速拷贝。
  - 高速缓存和缓冲是两个不同的功能，但有时一块内存区域也可以同时用于两个目的。
  - 当内核接收到I/O请求时，内核首先检查高速缓存以确定相应文件的内容是否在内存中。如果是，物理磁盘I/O就可以避免或延迟。

假脱机技术

- SPOOLing(Simultaneous Peripheral Operation On Line)称为假脱机技术。：用来保存设备输出的缓冲，这些设备如打印机不能接收交叉的数据流。
  - 操作系统通过截取对打印机的输出来解决这一问题。应用程序的输出先是假脱机到一个独立的磁盘文件上。当应用程序完成打印时，假脱机系统将相应的待送打印机的假脱机文件进行排队
- Printing：打印机虽然是独享设备，通过SPOOLing技术，可以将它改造为一台可供多个用户共享的设备。

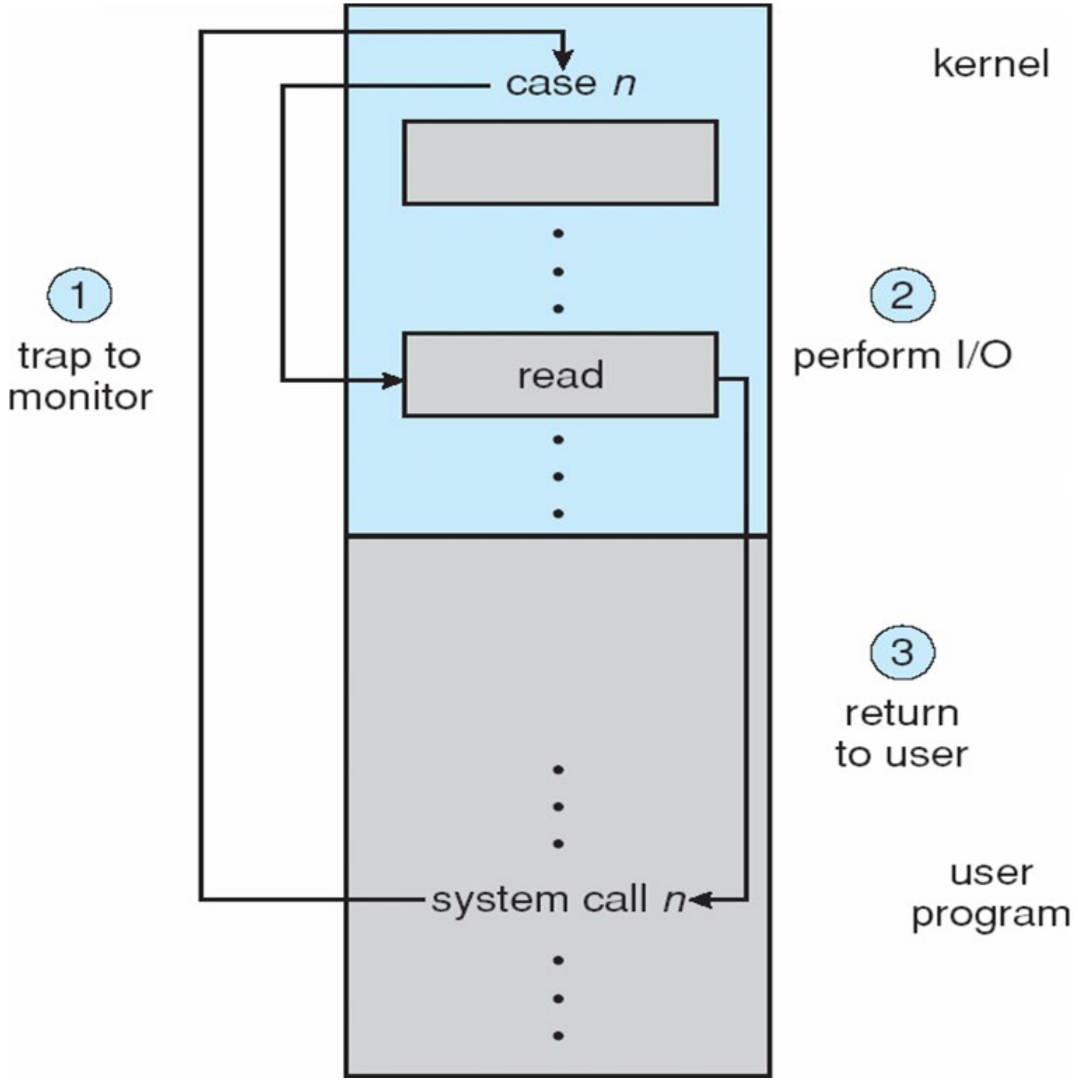
Device reservation & Error Handling

- **Device reservation（设备预定）** - provides exclusive access to a device提供对设备的独占访问
  - System calls for allocation and deallocation分配和再分配的系统调用
  - Watch out for deadlock有可能产生死锁
- **Error Handling（错误处理）**
  - OS can recover from disk read, device unavailable, transient write failures。操作系统可以恢复磁盘读，设备无效，暂时的失败
  - Most return an error number or code when I/O request fails当I/O失败时，大多数返回一个错误码
  - System error logs hold problem reports系统日志记录了出错报告

I/O Protection

- User process may accidentally or purposefully attempt to disrupt normal operation via illegal I/O instructions
  - All I/O instructions defined to be privileged
  - I/O must be performed via system calls
    - Memory-mapped and I/O port memory locations must be protected too

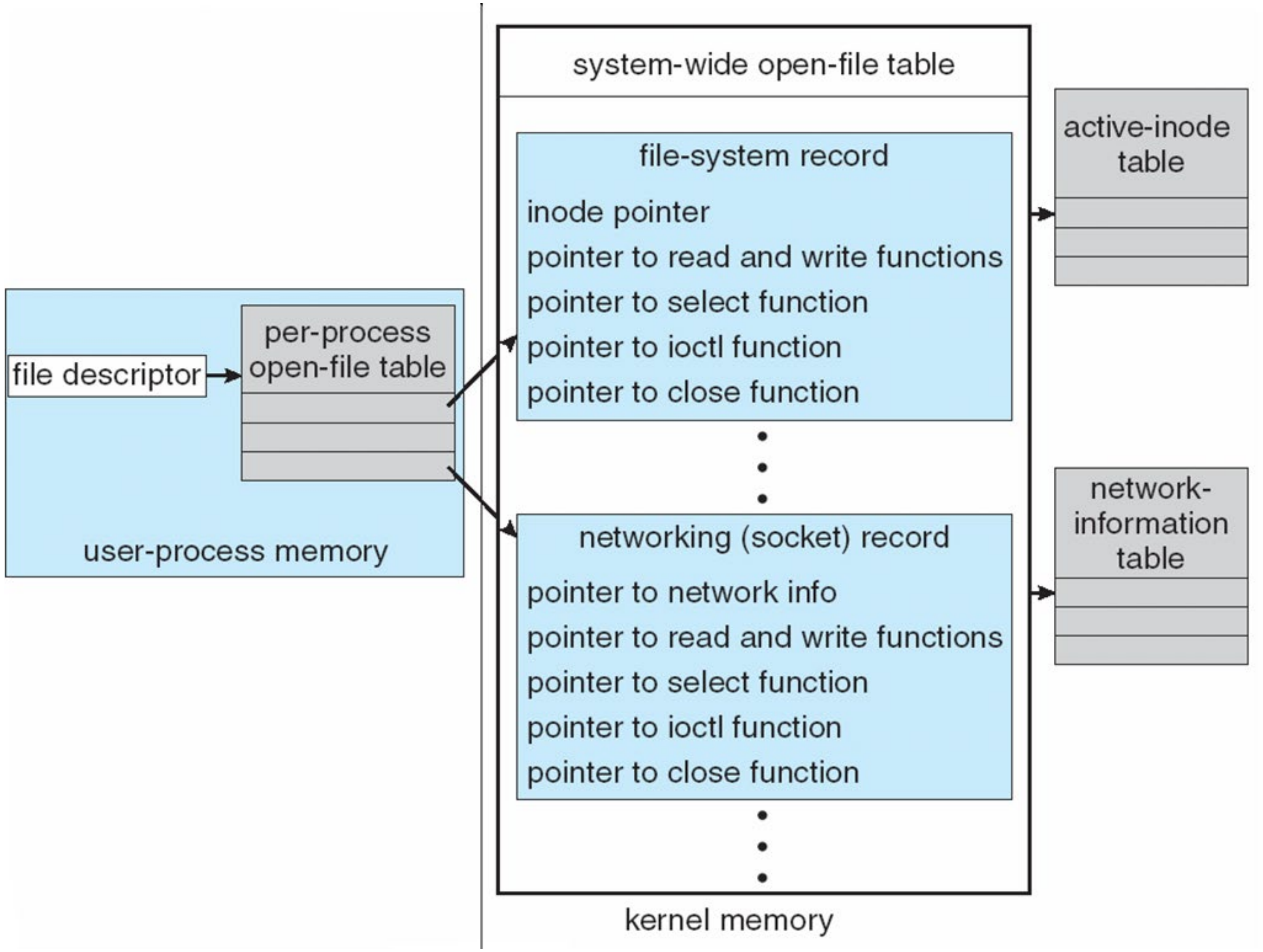
Use of a System Call to Perform I/O



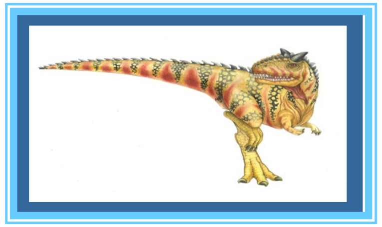
Kernel Data Structures

- Kernel keeps state info for I/O components, including open file tables, network connections, character device state。内核需要保存留I/O组件使用的状态信息，包括打开文件表，网络连接，字符设备状态等
- Many, many complex data structures to track buffers, memory allocation, “dirty” blocks。许多复杂的数据结构用来跟踪缓冲，内存分配，及“脏”块
- Some use object-oriented methods and message passing to implement I/O。某些OS用面向对象的方法和消息传递的方法来实现I/O

UNIX I/O Kernel Structure



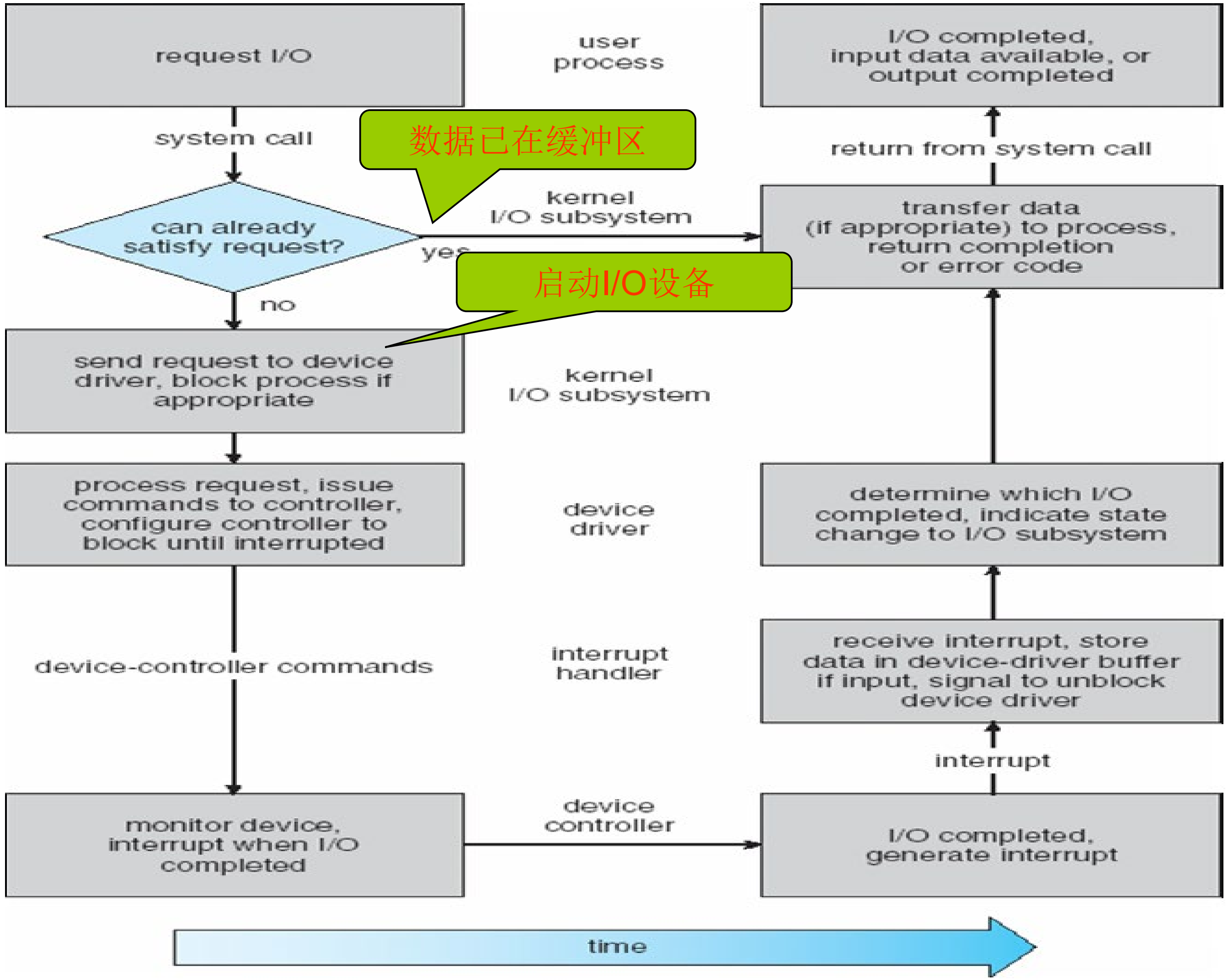
13.5 I/O Requests to Hardware Operations



I/O Requests to Hardware Operations

- Consider reading a file from disk for a process:
  - 确定保存文件的设备
  - 转换名字到设备的表示法
  - 把数据从磁盘读到缓冲区中
  - 通知请求进程数据现在是有效的
  - 把控制权返回给进程

Life Cycle of An I/O Request



13.6 STREAMS

- STREAM – a full-duplex communication channel between a user-level process and a device in Unix System V and beyond

- A STREAM consists of:

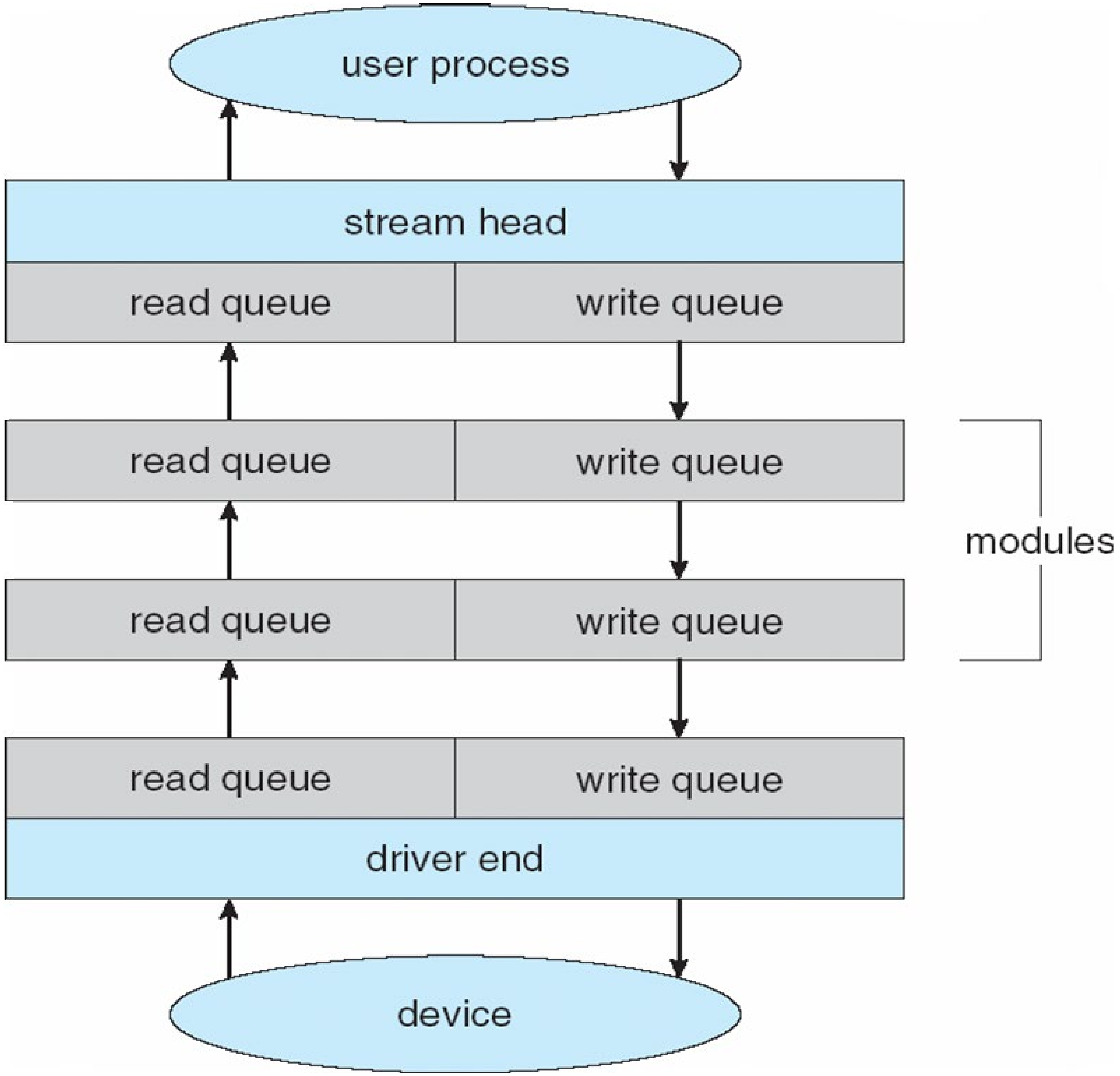
  - STREAM head interfaces with the user process

  - driver end interfaces with the device
  - zero or more STREAM modules between them.

- Each module contains a read queue and a write queue

- Message passing is used to communicate between queues

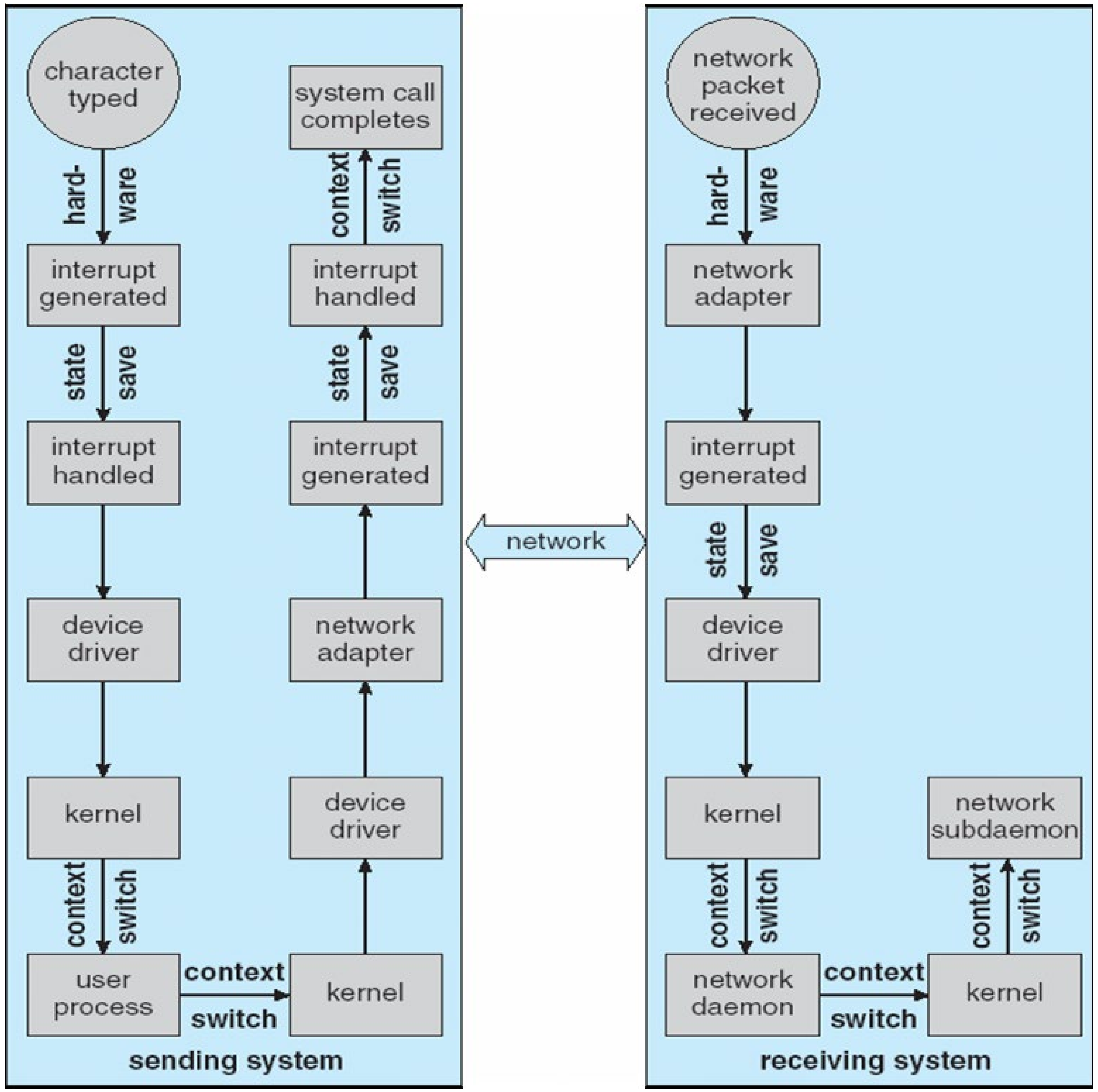
The STREAMS Structure



13.7 Performance

- I/O a major factor in system performance:
  - Demands CPU to execute device driver, kernel I/O code
  - Context switches due to interrupts
  - Data copying
  - Network traffic especially stressful

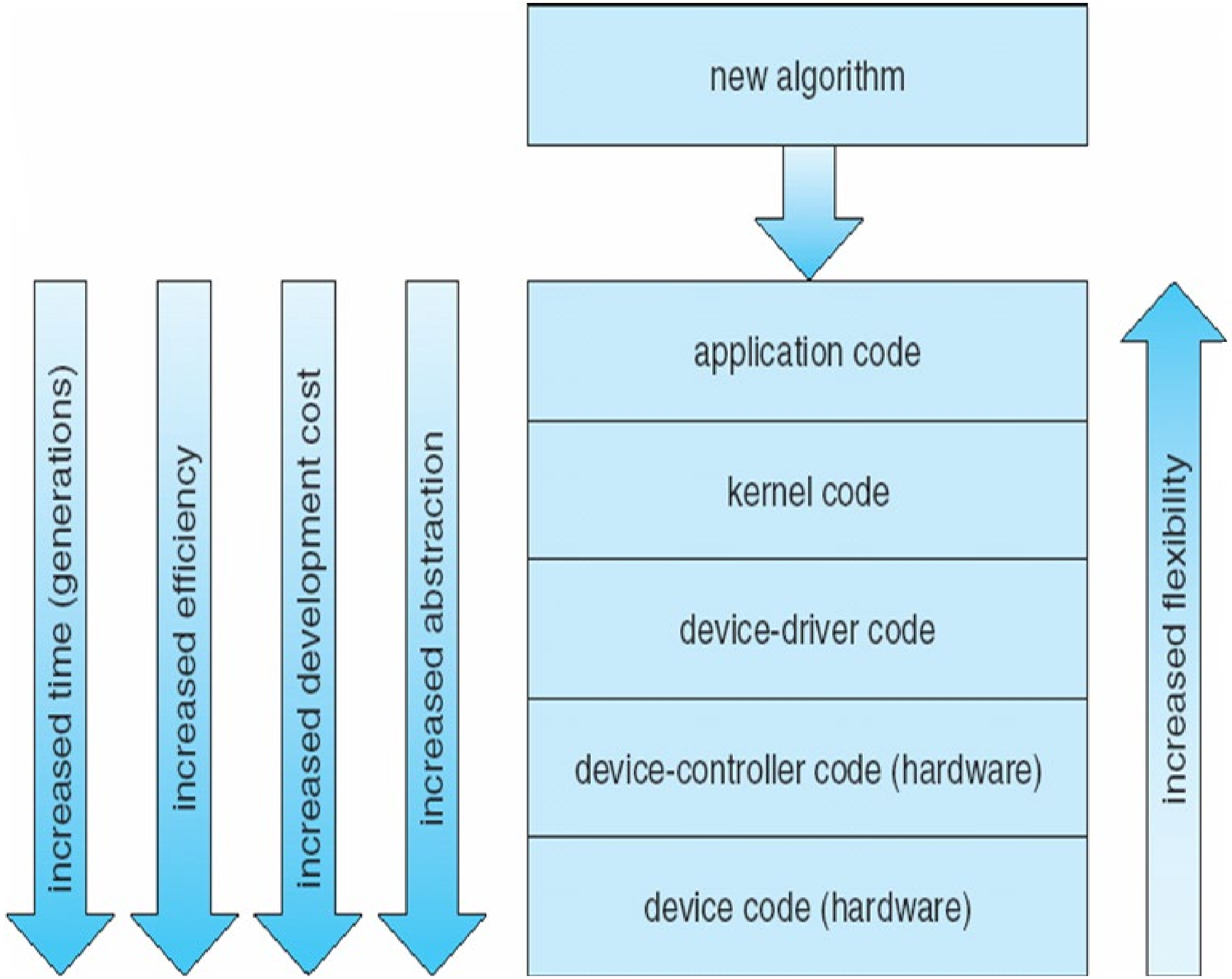
Intercomputer Communications



Improving Performance

- Reduce number of context switches
- Reduce data copying
- Reduce interrupts by using large transfers, smart controllers, polling
- Use DMA
- Balance CPU, memory, bus, and I/O performance for highest throughput

Device-Functionality Progression



作业

- 学在浙大
- 习题分析

End of Chapter 13