北京邮电大学软件学院
School Of software Engineering Of BUPT

Operating Systems

Lecture 12 IO Systems

Jinpengchen
Email: jpchen@bupt.edu.cn

Catalog Description

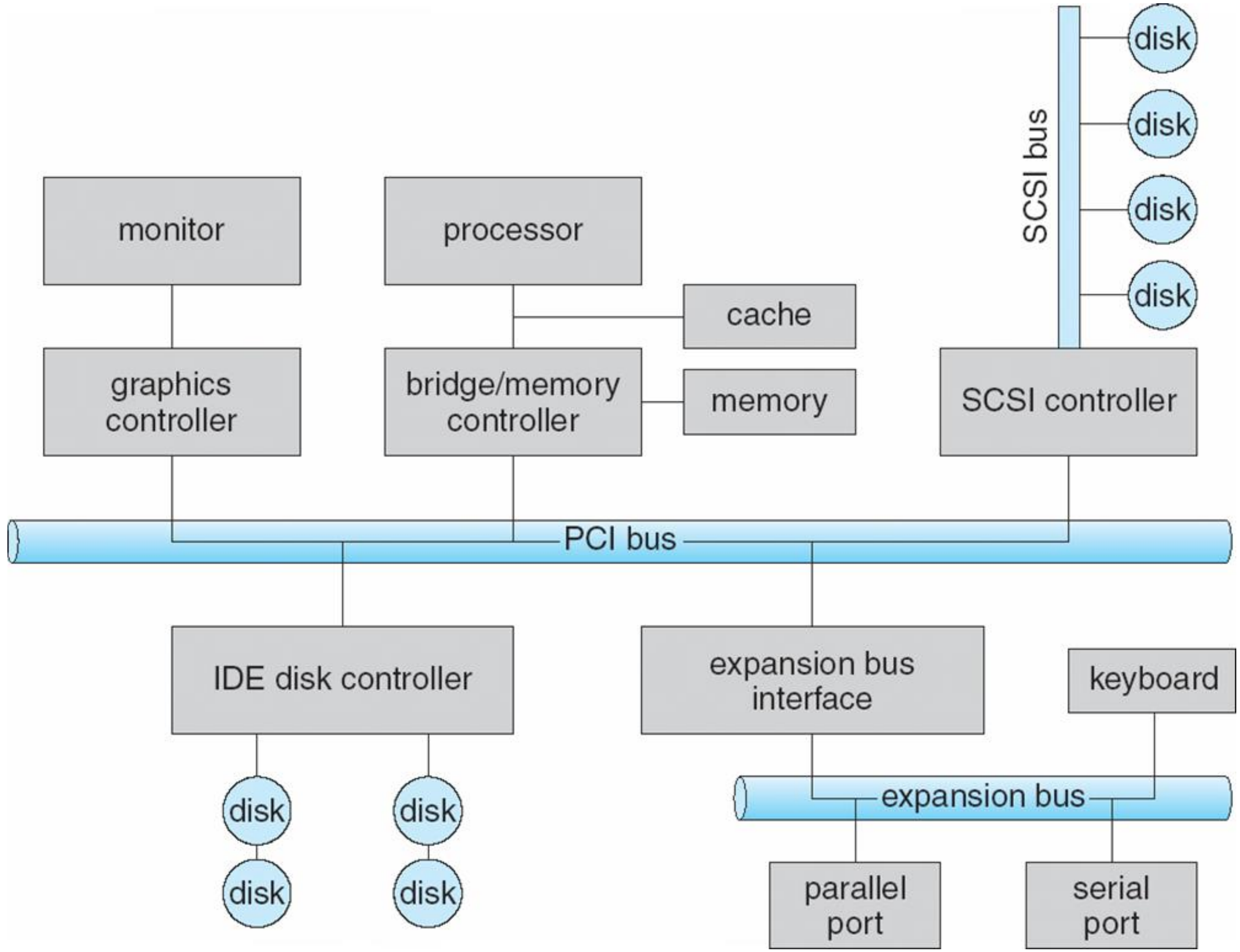
- I/O Hardware
- Application I/O Interface
- Kernel I/O Subsystem
- Transforming I/O Requests to Hardware Operations
- Performance

Overview

- I/O devices
  - vary widely
- The control of devices connected to the computer is a major concern of OS designers.
- How OS manages and controls various peripherals(外设)？

I/O Hardware

- Incredible variety of I/O devices



I/O Hardware

- Common concepts : CPU→PORT→BUS→Controller
  - Port (端口)
  - Bus (总线) (daisy chain(菊花链) or shared direct access)
    - ✓ PCI (Peripheral Component Interconnect(外部器件互连))
    - ✓ SCSI (Small computer systems interface)
    - ✓ Expansion bus
  - Controller (控制器) (host adapter)
- How can the processor command controller?
  - Controller has one or more registers for data and control signals.
  - The processor communicates with the controller by reading and writing bit patterns in the registers.

I/O Hardware

- Devices have addresses， used by two communication techniques:
  - Direct I/O instructions
    - ✓ Access the port address
    - ✓ Each port typically contains of four registers, i.e., status, control, data-in and data-out.
    - ✓ Instructions: In, out
  - Memory-mapped I/O
    - ✓ Example: 0xa0000 ~ 0xfffff are reserved to ISA graphics cards and BIOS routines
- Some systems use both techniques.

I/O Hardware

- I/O address range
  - 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 Hardware

- I/O Hardware
  - Polling (轮询方式)
  - Interrupts (中断方式)
  - Direct Memory Access (DMA方式)
  - I/O hardware summary

Polling（轮询方式）

- Need handshaking (握手)
- Determines state of device
  - Command-ready
    - ✓ In command register
    - ✓ 1: a command is available for the controller
  - Busy
    - ✓ In status register
    - ✓ 0: ready for the next command; 1: busy
  - Error
    - ✓ To indicate whether an I/O is ok

Polling（轮询方式）

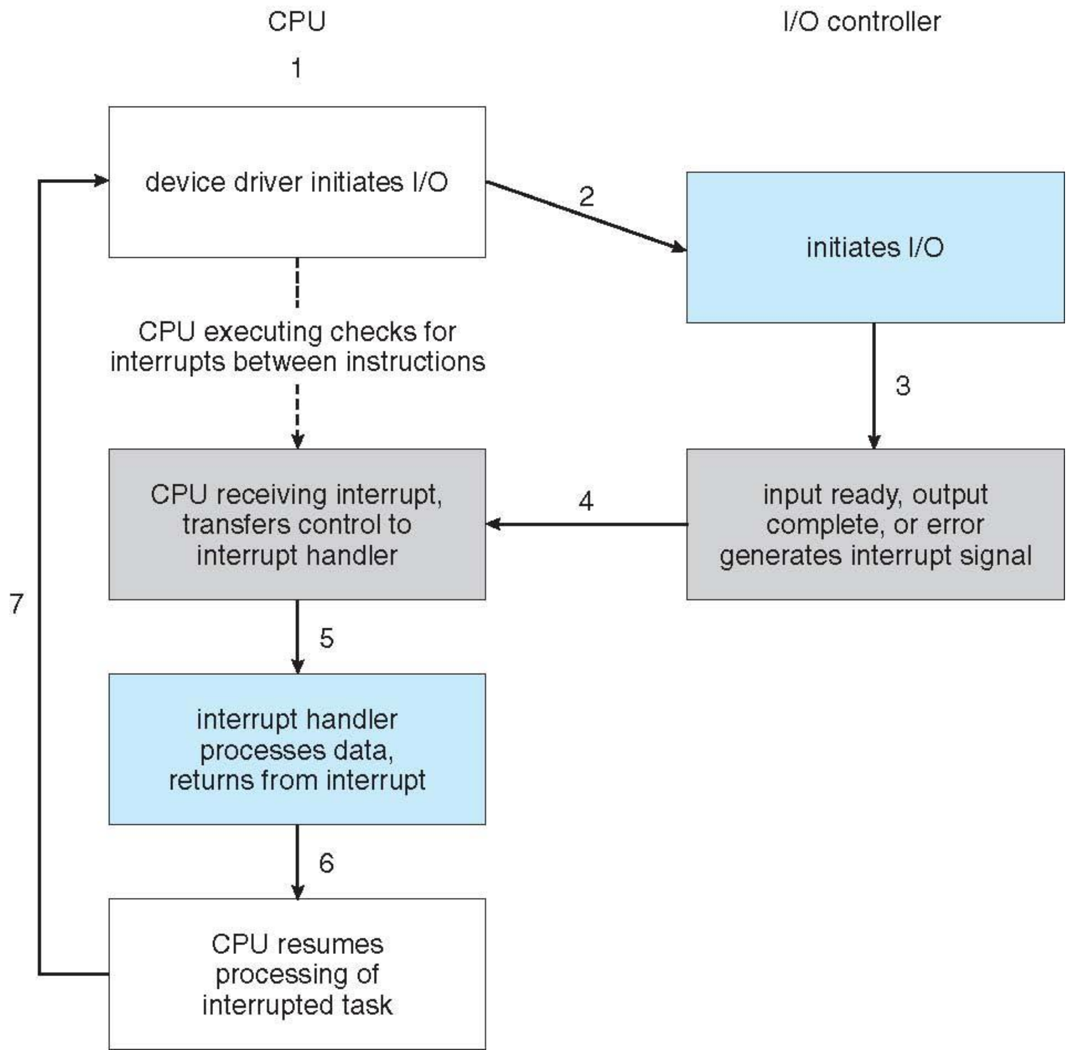
- Basic handshaking notion for writing output
  - 1. Host repeatedly reads the busy bit until it is 0
  - 2. Host sets write bit in the command register and writes a byte into the data-out register
  - 3. Host sets the command-ready bit
  - 4. When the controller notices that the command-ready is set, it sets the busy bit
  - 5. Controller gets the write command and data, and works
  - 6. Controller clears command-ready bit, error bit and busy bit
- In Step1: Busy-wait cycle to wait for I/O from device ＝polling

Interrupts（中断方式）

- The CPU hardware has Interrupt-request line triggered by I/O device
- Interrupt handler receives interrupts
- Basic interrupt scheme
  - Raise → Catch → Dispatch → Clear

Interrupts（中断方式）

- Interrupt-Driven I/O Cycle



Interrupts（中断方式）

- More sophisticated interrupt-handling features:

  Most CPU have two interrupt request line.
  - Nonmaskable
  - Maskable to ignore or delay some interrupts
- Efficient dispatching without polling the devices
  - Interrupt vector: to dispatch interrupt to correct handler
  - Interrupt chaining: to allow more device & more interrupt handlers
- Distinguish between high- and low-priority interrupts:
  - Interrupt priority: the handling of low-priority interrupts is deferred without masking, even preempted.
- Interrupt mechanism is also used for exceptions

Direct Memory Access (DMA方式)

- Direct Memory Access (DMA方式):

  Used to avoid programmed I/O for large data movement, and bypasses CPU to transfer data directly between I/O device and memory

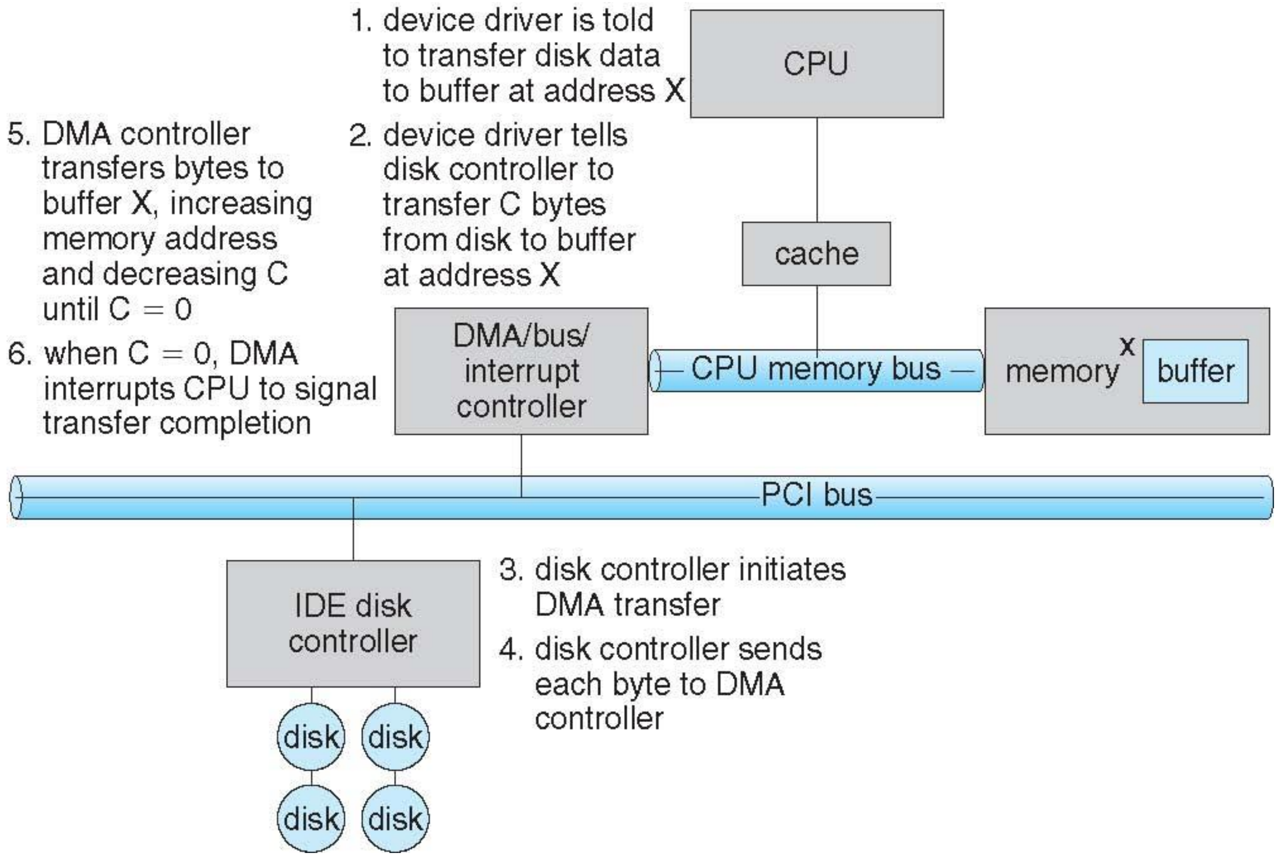
- Requires DMA controller
  - the host prepares a DMA command block in memory
    - ✓ a pointer to the source of a transfer
    - ✓ a pointer to the destination of the transfer
    - ✓ a count of the number of bytes to be transferred
  - CPU writes the address of the DMA command block to DMA controller, and then goes on with other work.

Direct Memory Access (DMA方式)

- Handshaking between DMA controller & device controller
  - Device controller raises DMA-request when one word is available
  - DMA controller seizes memory bus, places the desired address on memory-address wires, and raises DMA-acknowledge
  - Device controller transfers the word to memory, and removes the DMA-request signal. Goto 1
  - DMA controller interrupts the CPU.

Direct Memory Access (DMA方式)

- Six Step Process to Perform DMA Transfer



- Cycle stealing: when DMA seizes the memory bus, CPU is momentarily prevented from accessing main memory

I/O hardware summary

- A bus
- A controller
- An I/O port and its registers
- The handshaking relationship between the host and a device controller
- The execution of this handshaking in a pooling loop via interrupts
- the offloading of this work to a DMA controller for large transfer

Catalog Description

- I/O Hardware
- Application I/O Interface
- Kernel I/O Subsystem
- Transforming I/O Requests to Hardware Operations
- Performance

Application I/O Interface

- Block and Character Devices
- Network Devices
- Clocks and Timers
- Blocking (阻塞) and Nonblocking (非阻塞) I/O

I/O control challenges

- Wide variety of devices
- Two challenges

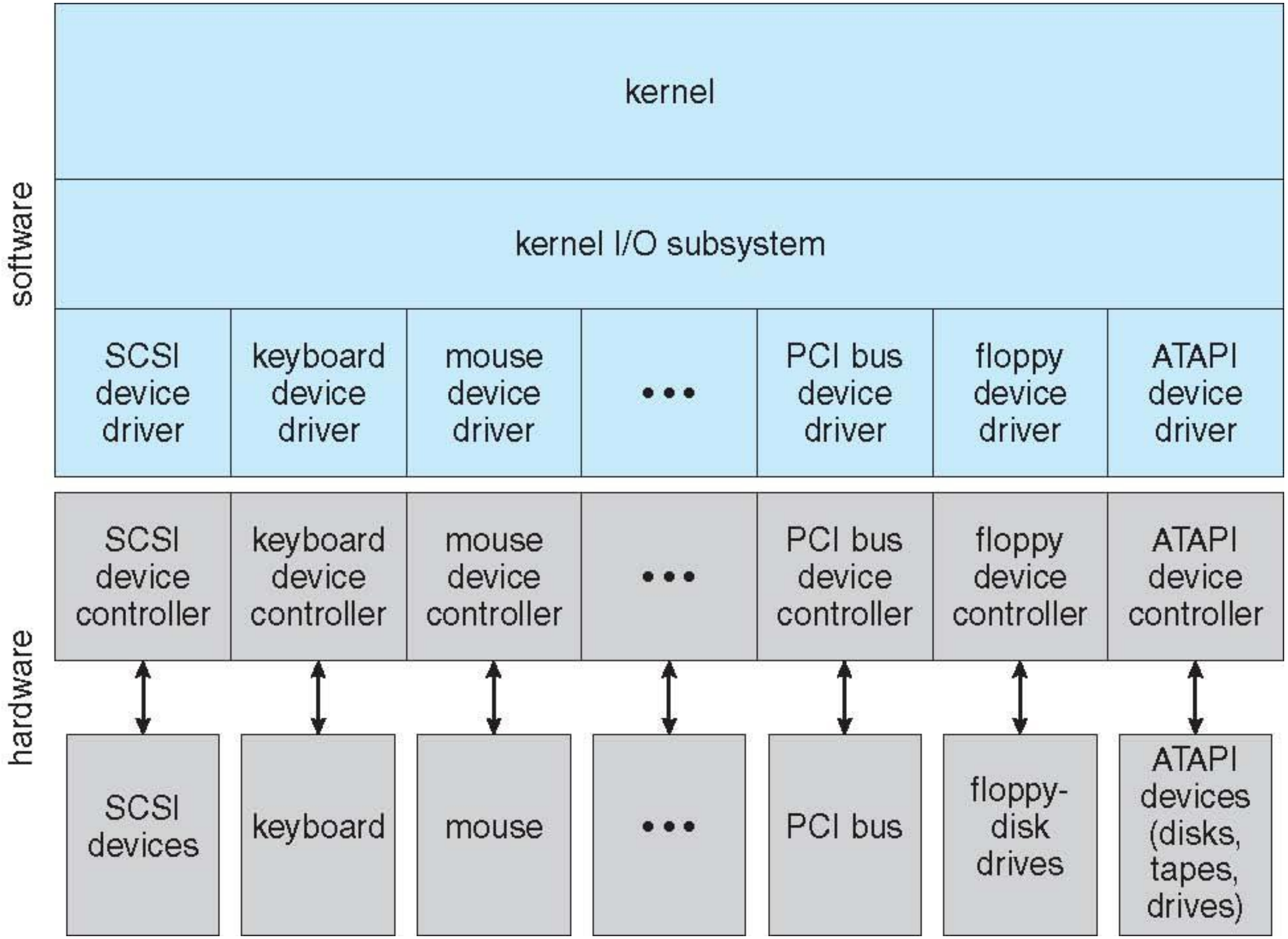
  Applications → OS ← Devices

  - How can the OS give a convenient, uniform I/O interface to applications?
  - How can the OS be designed such that new devices can be attached to the computer without the OS being rewritten?
- For device manufacturers, device-driver layer hides

differences among I/O controllers from kernel

I/O control challenges

- A Kernel I/O Structure



Application I/O Interface

- For applications, I/O system calls encapsulate device behaviors in generic classes
- 设备独立性：应用程序与具体的物理设备无关。
- Device-driver layer hides differences among I/O controllers from kernel
- Devices vary in many dimensions
  - Character-stream or block
  - Sequential or random-access
  - Sharable or dedicated
  - Speed of operation
  - read-write, read only, or write only

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

- Block devices include disk drives
  - Commands include read, write, seek
  - Raw I/O or file-system access
  - Memory-mapped file access possible
- Character devices include keyboards, mice, serial ports
  - Commands include get(), put()
  - Libraries layered on top allow line editing

Network Devices

- Varying enough from block and character to have own interface
- Unix and Windows NT/9x/2000 include socket interface
  - Separates network protocol from network operation
  - Server – socket, bind, listen, accept
  - Client – socket, connect
  - Includes select() functionality
- Approaches vary widely (pipes, FIFOs, streams, queues, mailboxes)

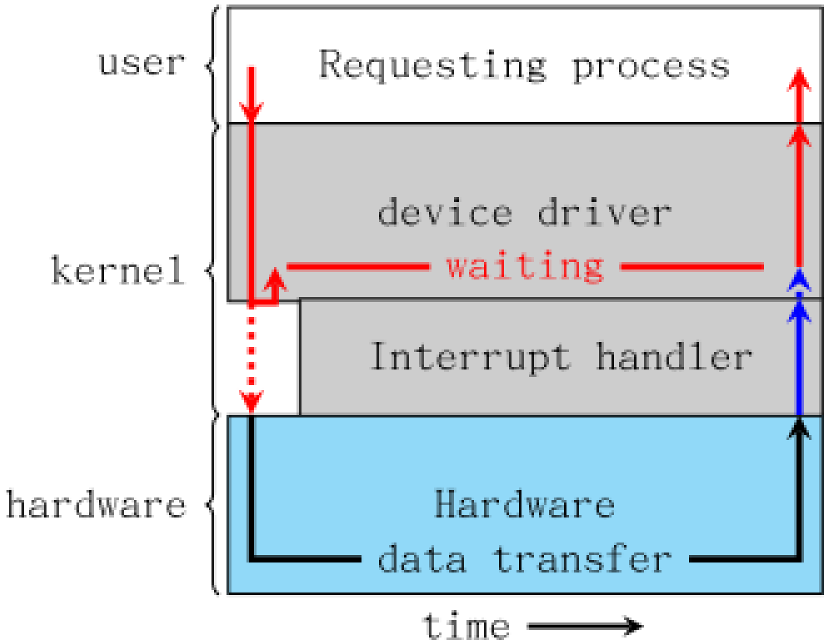
Clocks and Timers

- Provide current time, elapsed time, timer
- Hardware clocks
  - Real Time Clock (RTC, 实时时钟)
  - Time Stamp Counter (TSC, 时间戳计数器)
  - Programmable Interval Timer (PIT, 可编程间隔定时器)
    - ✓ used for timings, periodic interrupts
- Ioctl() (on UNIX) covers odd aspects of I/O such as clocks and timers

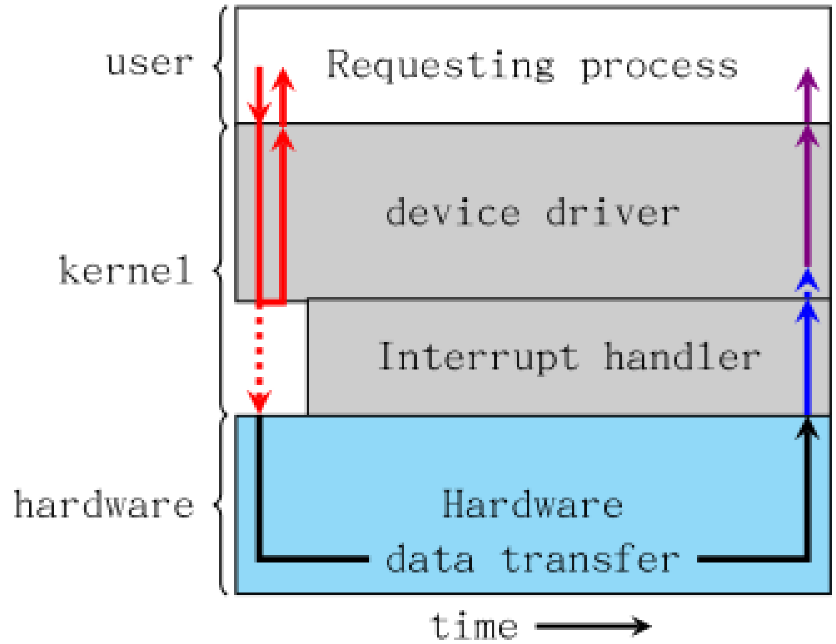
Blocking(阻塞)and Nonblocking(非阻塞) I/O

- Blocking (阻塞) - process suspended until I/O completed
  - Easy to use and understand
  - Insufficient for some needs
- Nonblocking (非阻塞) - I/O call returns as much as available
  - User interface, data copy (buffered I/O)
  - Implemented via multi-threading
  - Returns quickly with count of bytes read or written
  - Asynchronous (异步) - process runs while I/O executes
    - ✓ Difficult to use
    - ✓ I/O subsystem signals process when I/O completed

Two I/O Methods



(a)
Synchronous



(b)
Asynchronous

Catalog Description

- I/O Hardware
- Application I/O Interface
- Kernel I/O Subsystem
- Transforming I/O Requests to Hardware Operations
- Performance

Kernel I/O Subsystem

- I/O Scheduling
- Buffering (缓冲机制)
- Caching, Spooling & device reservation
- Error Handling
- I/O Protection
- Kernel Data Structures

I/O scheduling

- I/O scheduling: To schedule a set of I/O requests means to determine a good order in which to execute them
  - Origin order: the order in which applications issue system calls: May NOT the best order!
  - Scheduling can
    - ✓ Improve overall system performance
    - ✓ Share device access fairly among processes
    - ✓ Reduce the average waiting time for I/O to complete
  - Example: Disk read request from Apps.

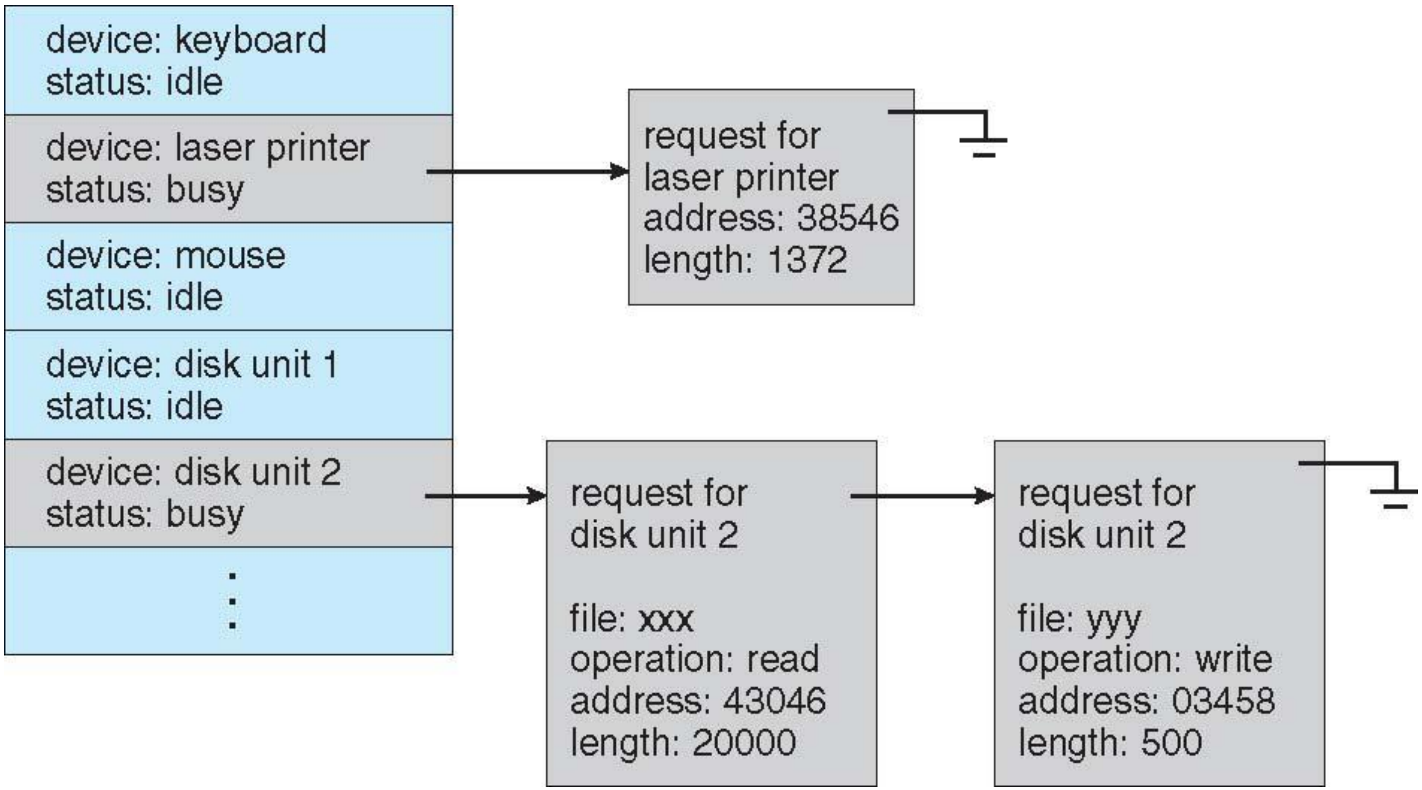
    App1: 0; App2: 100; App3: 50;

    Now at 100;

    The OS may serve the applications in the order App2, App3, App1.

I/O scheduling

- OS maintains a wait queue of request for each device
  - Device-status Table



  - I/O scheduling, some OSes try fairness, some not

I/O scheduling

- Another way to improve performance is by using storage space in main memory or on disk
  - Buffering （缓冲机制）
  - Caching
  - Spooling

Buffering(缓冲机制)

- Buffer – A memory area that stores data while they are transferred between two devices or between a device and an application
- Store data in memory while transferring between devices
- Why buffering?
  - To cope with device speed mismatch.

    Example: Receive a file via modem and store the file to local hard disk.
    - ✓ Speed: The modem is about a thousand times slower than the hard disk.
    - ✓ Two buffers are used.

Buffering(缓冲机制)

- Why buffering?
  - To cope with device transfer size mismatch.

    Example: Send/receive a large message via network.
    - ✓ At sending side: the large message is fragmented into small network packets.
    - ✓ At receiving side: the network packets are placed in a reassembly buffer.
  - To maintain "copy semantics"

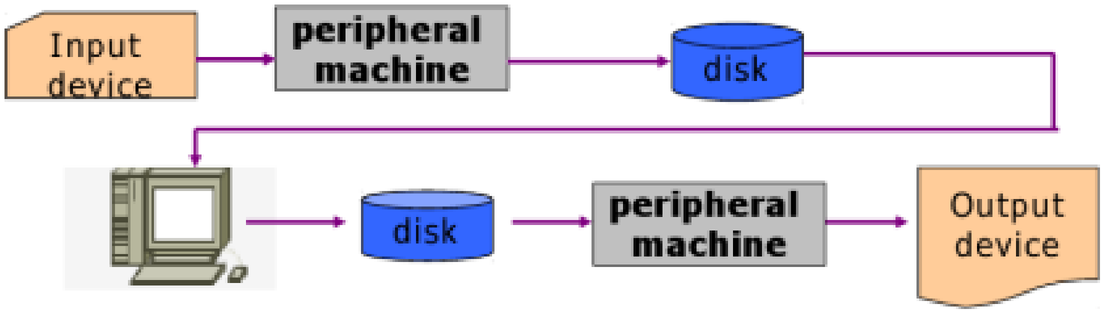
    Example: When write() data to disk, it first copies the data from application's buffer to a kernel buffer.

Caching, Spooling & device reservation

- Caching - fast memory holding copy of data
  - Always just a copy
  - Key to performance
- Spooling - hold output for a device
  - Dedicated device can serve only one request at a time
  - Spooling is a way of dealing with I/O devices in a multiprogramming system
  - Example: Printing
- Device reservation - provides exclusive access to a device
  - System calls for allocation and deallocation
  - Watch out for deadlock

Spooling

- Out-line I/O (脱机I/O)，使用外围机（peripheral machine）

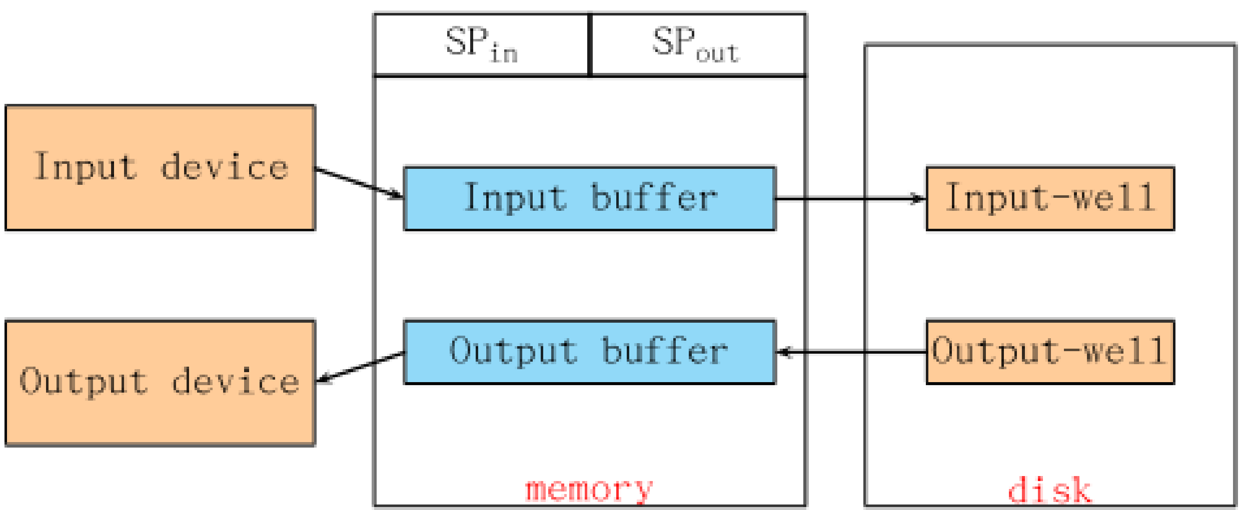


- SPOOL:

  Simultaneous Peripheral Operation On-Line
（外部设备联机并行操作，假脱机）
  - Dedicated device → sharable device
  - Using processes of multiprogramming system

Spooling

- SPOOL:
  Simultaneous Peripheral Operation On-Line
(外部设备联机并行操作，假脱机)
  - Structure
    - ✓ Input-well (输入井), output-well (输出井)
    - ✓ Input-buffer, output-buffer
    - ✓ Input-process SP in , output-process SP out
    - ✓ Requested-queue

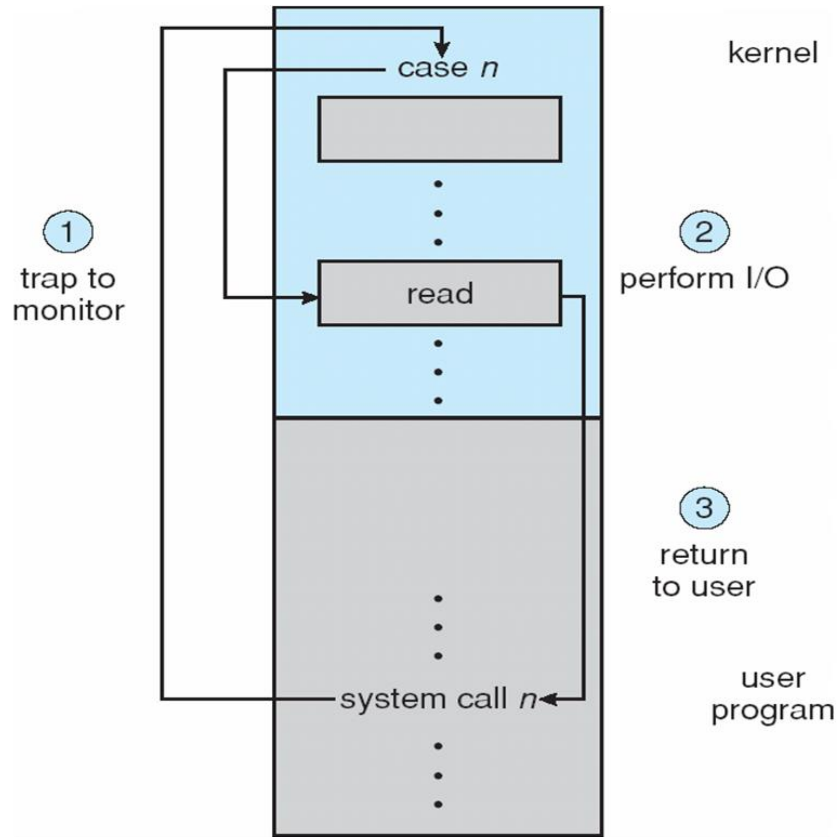


Error Handling

- OS can recover from disk read, device unavailable, transient write failures
  - Example: read() again, resend(), . . . , according to some specified rules
- Most return an error number or code when I/O request fails
- System error logs hold problem reports

I/O Protection

- User process may accidentally or purposefully attempt to disrupt normal operation via illegal I/O instructions
- To prevent users from performing illegal I/O
  - 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
- Many, many complex data structures to track buffers, memory allocation, "dirty" blocks
- Some use object-oriented methods and message passing to implement I/O

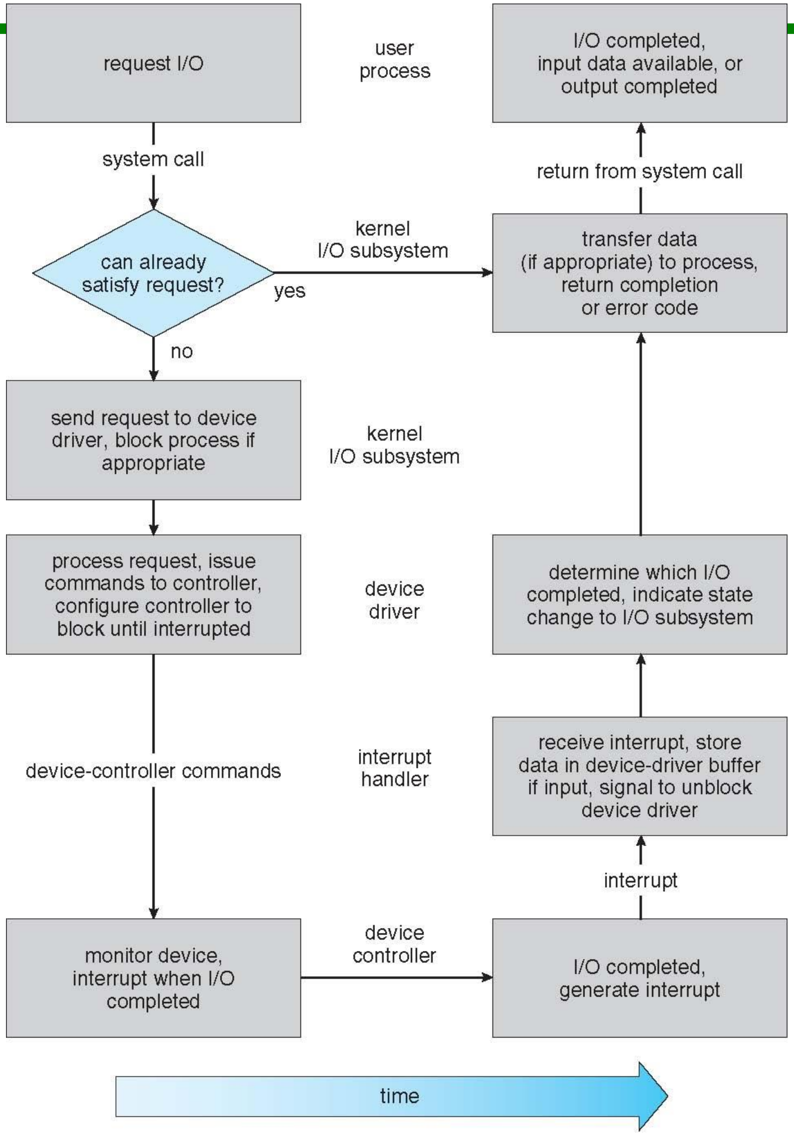
Catalog Description

- I/O Hardware
- Application I/O Interface
- Kernel I/O Subsystem
- Transforming I/O Requests to Hardware Operations
- Performance

I/O Requests to Hardware Operations

- Consider reading a file from disk for a process:
  - Determine device holding file
  - Translate name to device representation
  - Physically read data from disk into buffer
  - Make data available to requesting process
  - Return control to process

Life Cycle of An I/O Request



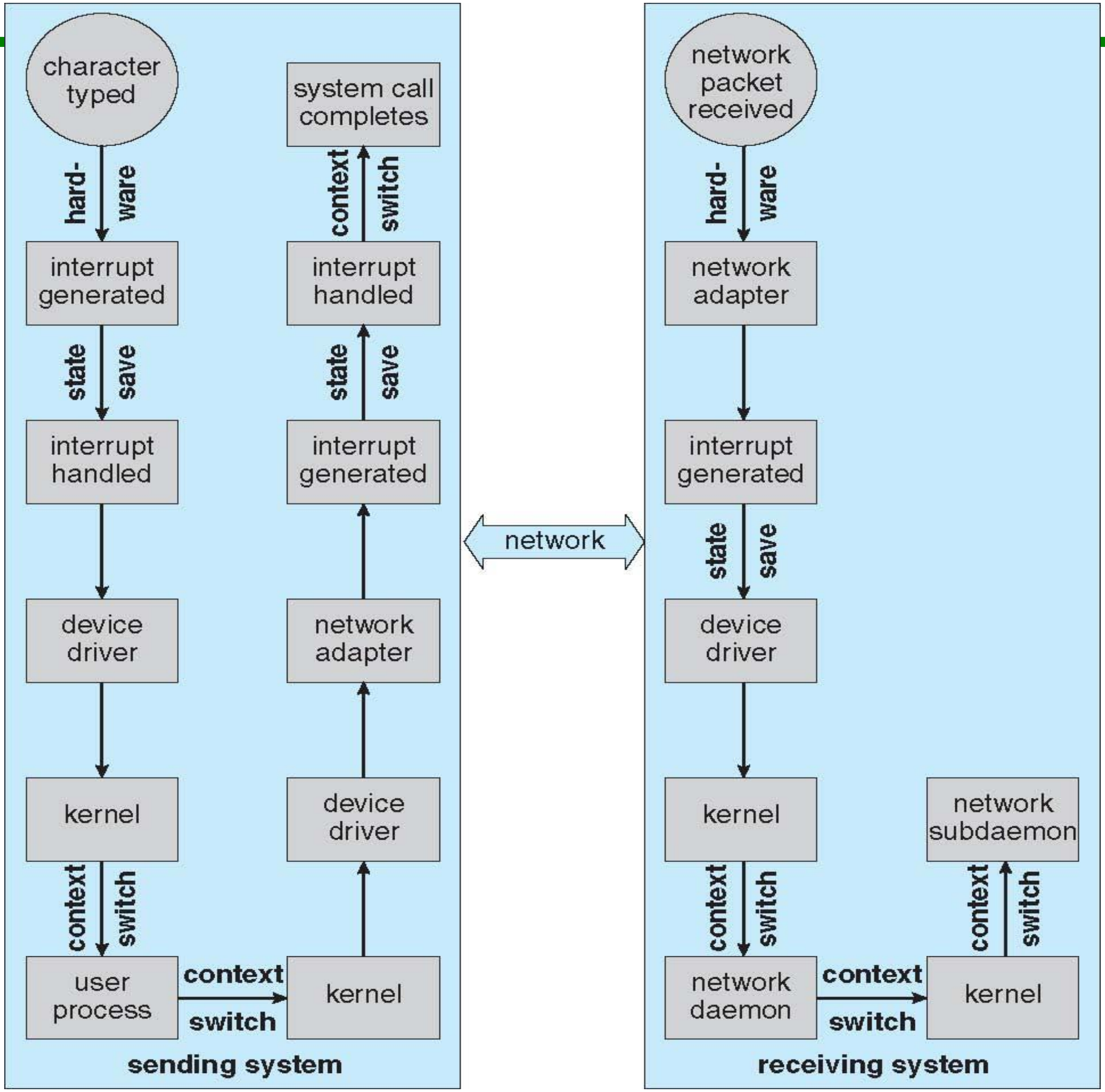
Catalog Description

- I/O Hardware
- Application I/O Interface
- Kernel I/O Subsystem
- Transforming I/O Requests to Hardware Operations
- Performance

Performance

- I/O is 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

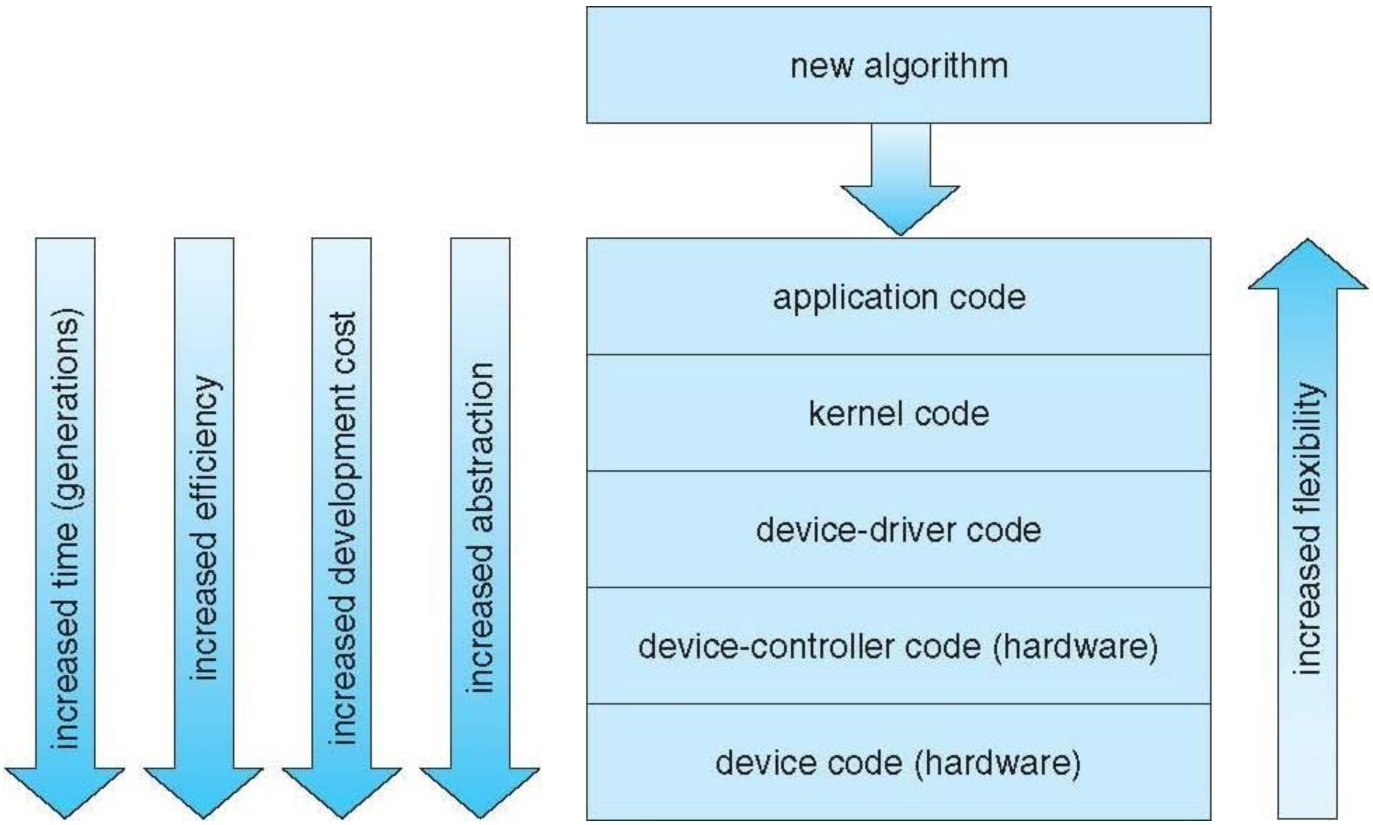


Network traffic can also cause a high context-switch rate

Improving Performance

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

Device-Functionality Progression



End of Chapter 12