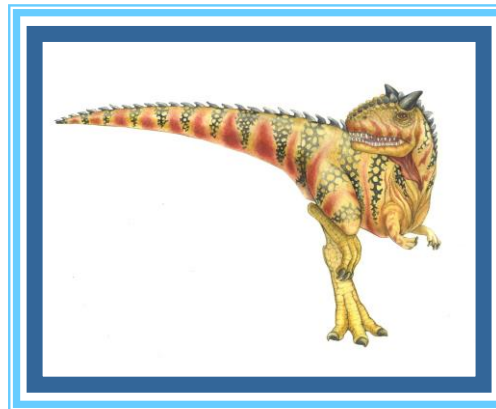
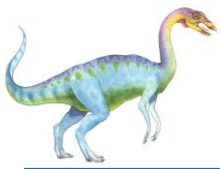


# Ch11: I/O Systems

---





# Overview

---

- I/O management is a major component of operating system design
  - Important aspect of computer operation
  - I/O devices vary greatly
  - Various methods to control them
  - Performance management
  
- Ports, busses, device controllers connect to various devices
  
- **Device drivers** encapsulate device details
  - Present uniform device-access interface to I/O subsystem





# I/O Hardware

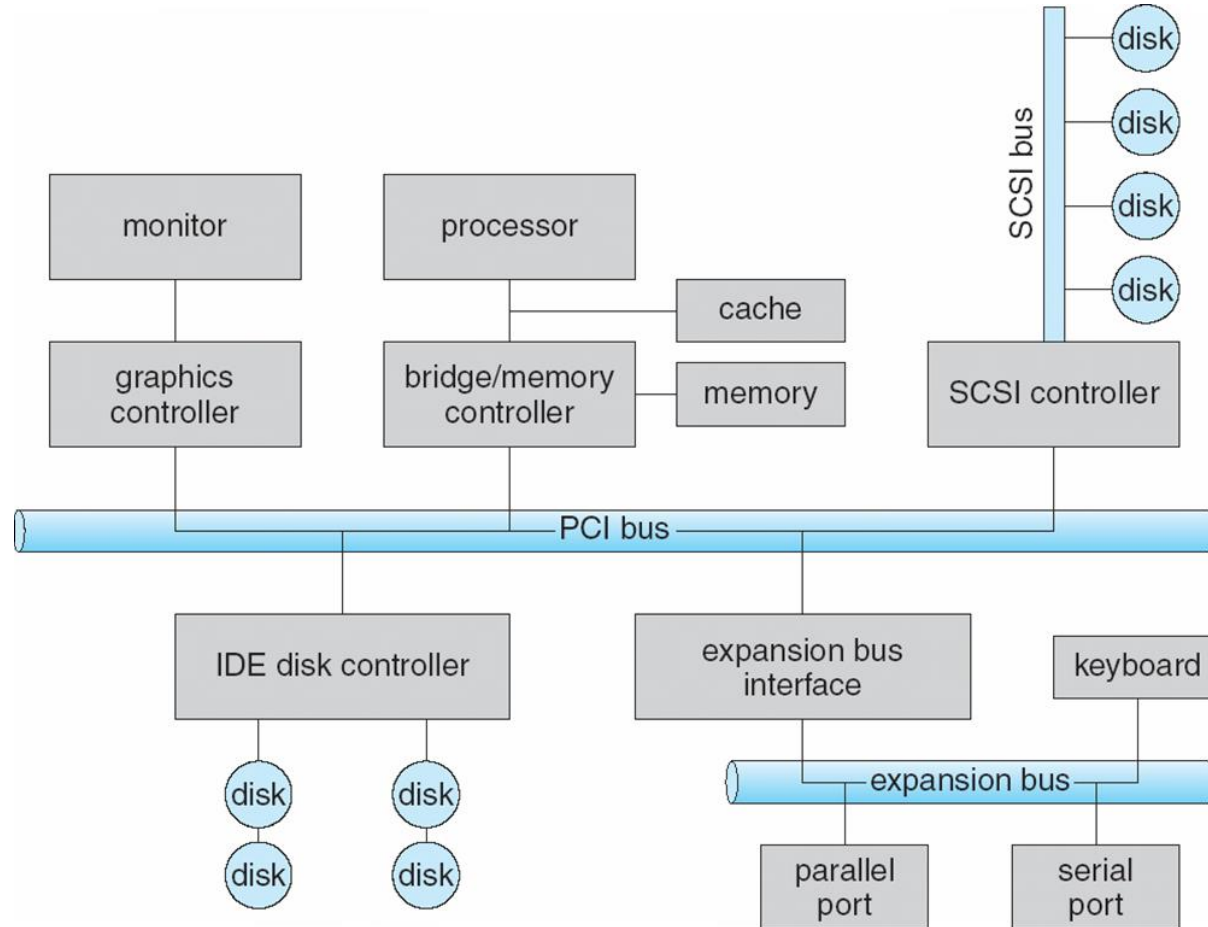
---

- Incredible variety of I/O devices
  - Storage
  - Transmission
  - Human-interface
  
- Common concepts
  - **Port** – connection point for device
  - **Bus - daisy chain** or shared direct access
    - ▶ **PCI** bus common in PCs and servers, PCI Express (**PCIe**)
    - ▶ **expansion bus** connects relatively slow devices
  - **Controller (host adapter)** – electronics that operate port, bus, device
    - ▶ Sometimes integrated
    - ▶ Sometimes separate circuit board (host adapter)





# A Typical PC Bus Structure



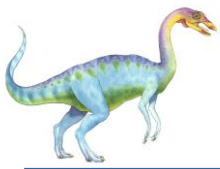


# I/O Hardware

---

- How to 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**
  
- How to communicate with controller?
  - Devices have addresses, used by
    - ▶ Direct I/O instructions
    - ▶ **Memory-mapped I/O**
      - Device data and command registers mapped to processor address space





# 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)     |





# Polling (轮询)

- For **each byte** of I/O
  1. Read busy bit from status register until 0
  2. Host sets read or write bit and if write copies data into data-out register
  3. Host sets command-ready bit
  4. Controller sets busy bit, executes transfer
  5. Controller clears busy bit, error bit, command-ready bit when transfer done
  
- Step 1 is **busy-wait** cycle to wait for I/O from device
  - Reasonable if device is fast
  - But inefficient if device is slow



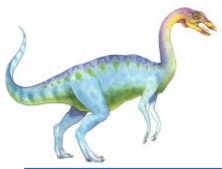


# Interrupts (中断)

- CPU **Interrupt-request line** triggered by I/O device
  - Two lines:
    - ▶ **Maskable (可屏蔽)** and **nonmaskable (非屏蔽) interrupt**
  - Checked by processor after each instruction
  
- **Interrupt handler** receives interrupts
  
- **Interrupt vector (中断向量)** to dispatch interrupt to correct handler
  - Context switch at start and end
  - Based on priority, some are **nonmaskable**
  - Interrupt chaining if more than one device at same interrupt number







# 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                    |

非屏蔽中断





# Interrupts (Cont.)

---

- Interrupt mechanism also used for **exceptions (异常)**
  - Terminate process, crash system due to hardware error
  
- Page fault
  - executes when memory access error
  
- System call
  - executes via **software interrupt** or **trap** to trigger kernel to execute request





# Direct Memory Access

- Used to avoid **programmed I/O** (one byte at a time) (**程序控制I/O**) for large data movement
  - Requires **DMA** controller
  - Bypasses CPU to transfer data directly between device & memory
- How to work?
  - OS writes DMA command block into memory
    - ▶ Source and destination addresses
    - ▶ Read or write mode
    - ▶ Count of bytes
  - Writes location of command block to DMA controller, then CPU can continue to execute other tasks
  - DMA controller masters bus and does the transmission without CPU
    - ▶ DMA-request and DMA acknowledge between DMA controller and device controller





# Application I/O Interface

- Devices vary in many dimensions
  - **Character-stream** or **block**
  - **Sequential** or **random-access**
  - **Synchronous** or **asynchronous**
  - **Sharable** or **dedicated**
  - **Speed of operation**
  - **read-write, read only, write only**

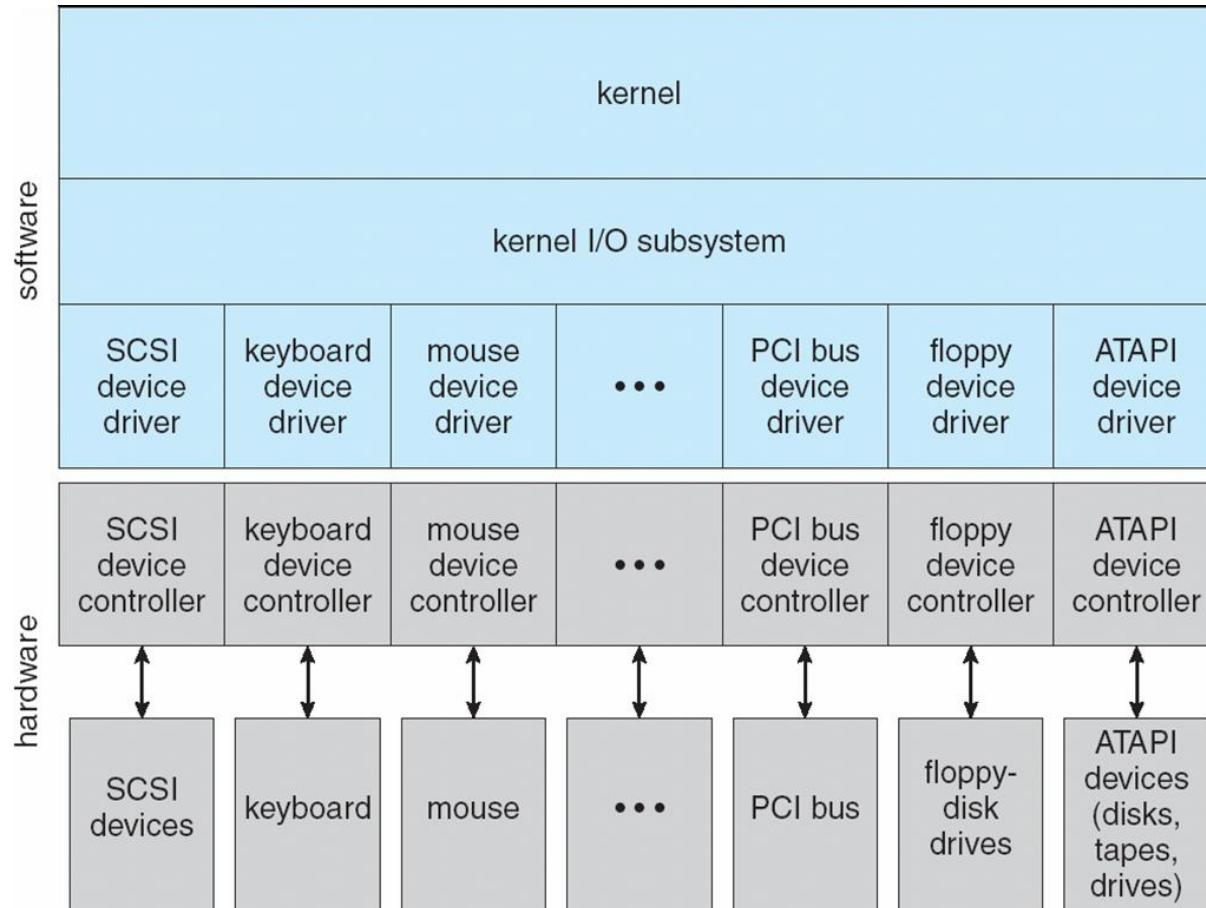
| aspect             | variation                                                         | example                               |
|--------------------|-------------------------------------------------------------------|---------------------------------------|
| data-transfer mode | character<br>block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

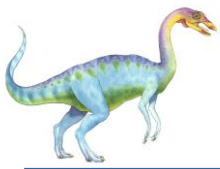
- How to provide a standard and uniform I/O interface?
  - **Abstraction, encapsulation, layering (抽象, 封装, 分层)**





# A Kernel I/O Structure





# I/O Devices

---

- **Block devices** include disk drives
  - Commands include **read**, **write**, **seek**
  - **Raw I/O**, **direct I/O**, or file-system access
  - Memory-mapped file access possible
    - ▶ File mapped to virtual memory and clusters brought via demand paging
  - DMA
  
- **Character devices** include keyboards, mice, serial ports
  - Commands include `get()`, `put()`
  
- **Network devices**
  - **socket** interface





# Clocks and Timers

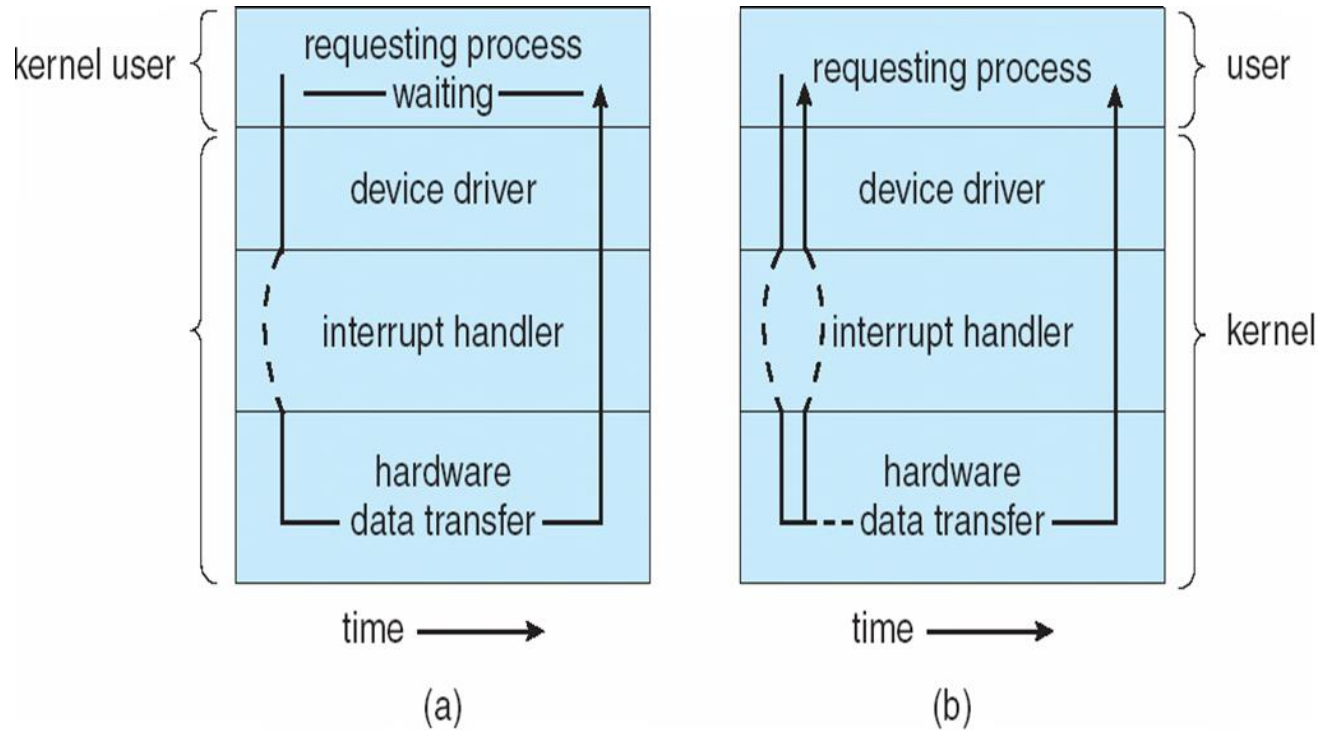
---

- Functionalities of hardware clock and timer
  - Get current time
  - Get elapsed time
  - Timer
  
- **Programmable interval timer (可编程间隔定时器)** used for timings, periodic interrupts
  - Process scheduler: interrupt when time quantum is zero
  - I/O subsystem: periodic flush





# Two I/O Methods



Synchronous

Asynchronous





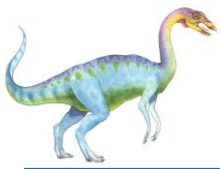


# Kernel I/O Subsystem

---

- **Kernel I/O subsystem provides many services**
  
- **I/O scheduling**
  - Maintain a per-device queue
  - Re-ordering the requests
  - Average waiting time, fairness, etc.
  
- **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” (e.g., copy from application’s buffer to kernel buffer)





# Kernel I/O Subsystem

---

- **Caching** - faster device holding copy of data
  - Always just a copy
  - Key to performance
  - Sometimes combined with buffering
- **Spooling** - hold output for a device
  - If device can serve only one request at a time, e.g., Printing
- **Error handling and I/O protection**
  - OS can recover from disk read error, device unavailable, transient write failures
  - All I/O instructions defined to be privileged
- **Power management, etc.**





# Summary

---

- I/O hardware
  - Port, bus, controller
  - Polling, interrupt, DMA
  
- Application I/O interface
  - block devices, character devices, network devices, clock and timer
  
- Kernel I/O subsystem
  - Services

