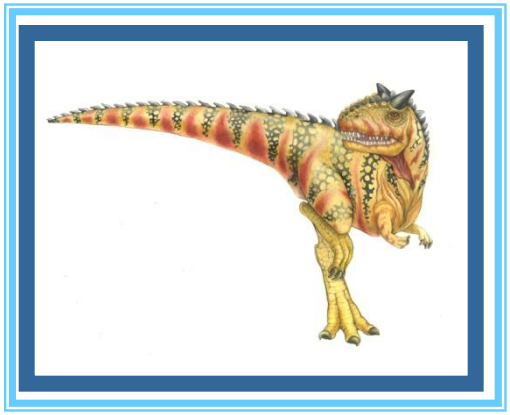
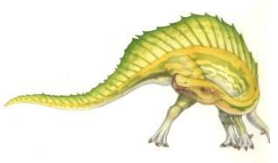
Chapter 13: I/O Systems



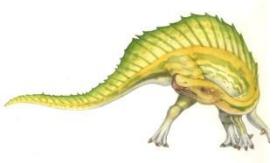
Chapter 13: I/O Systems

- I/O Hardware
- Application I/O Interface
- Kernel I/O Subsystem



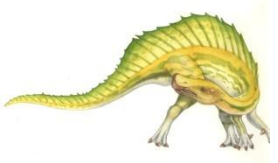
Objectives

- Explore the structure of an operating system's I/O subsystem
- Discuss the principles of I/O hardware and its complexity



I/O Hardware

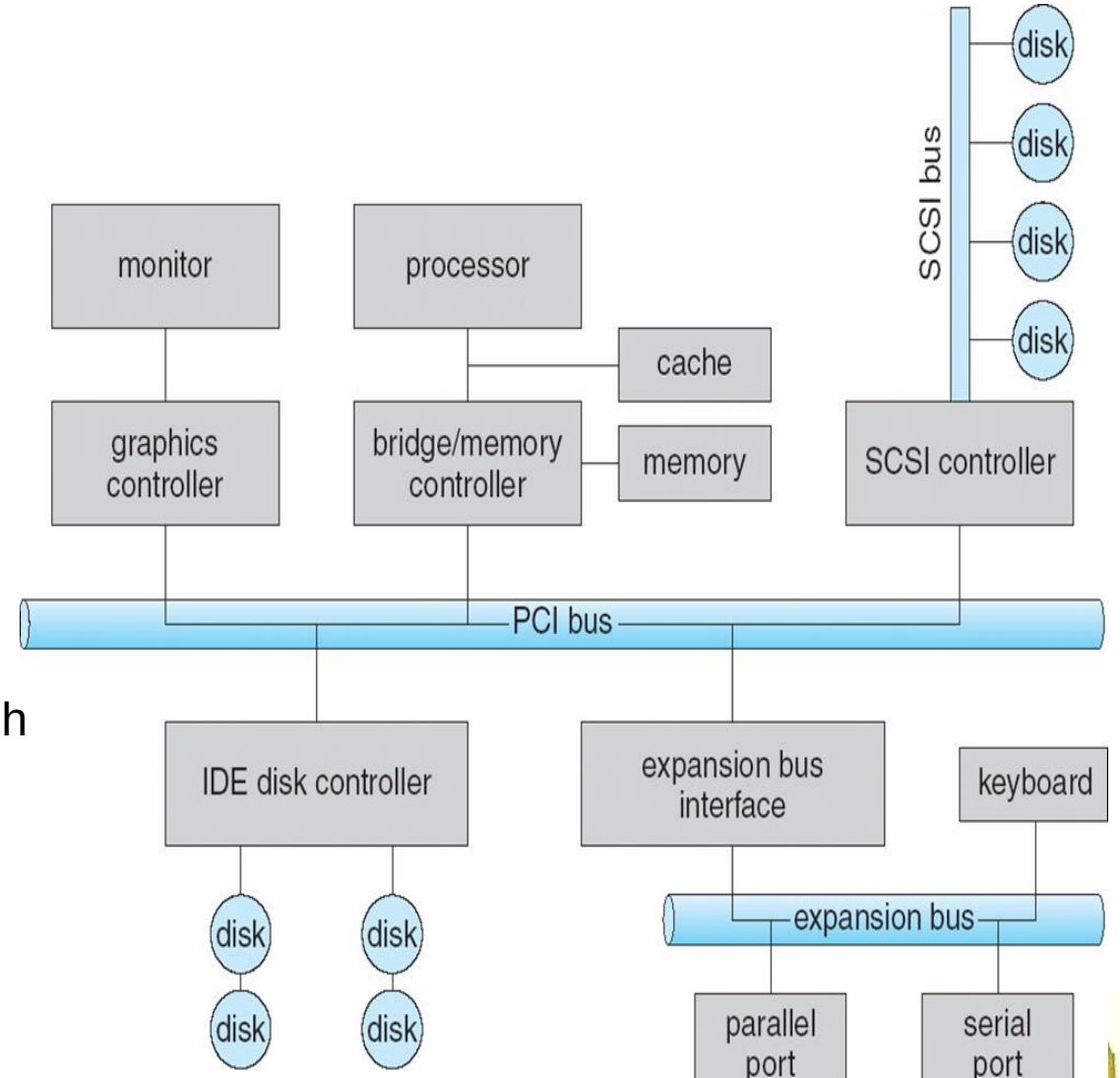
- Computers operate a great many kinds of devices, general and special types.
  - Storage devices (disks, tapes)
  - Transmission devices (network cards, modems)
  - Human-interface devices (screen, keyboard, mouse)
- A device communicate with a computer via:
  - **Port** (connection point)
  - **Bus** (set of wires , **daisy chain**)
  - **Controller** (collection of electronics operate a port, a bus, and a device) implement as **host adapter.**



A Typical PC Bus Structure

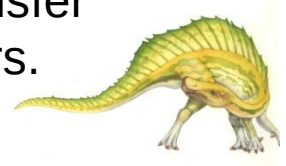
The figure shows :

- **PCI bus** (the common PC system bus) : connect the processor-memory subsystem to the fast devices.
- **Expansion bus:** connects relatively slow devices, such as keyboard and serial and USB ports.



I/O Hardware (cont.)

- How can the processor give the commands and data to a controller to accomplish an I/O transfer?
  - The controller has one or more registers for data and control signals.
  - The processor communicates with the controller by reading and writing bit patterns in these registers.
  - Using one of the following ways:
    - **Direct I/O instructions:**
      - a special I/O instructions specify the transfer of a byte or word to an I/O port address.
      - Triggers bus lines to select the proper device and to move bits into or out of a device register.
    - **Memory-mapped I/O**
      - The device –control registers mapping into the address space of the processor.
      - The CPU executes I/O request using standard data transfer instructions to read and write the device-control registers.

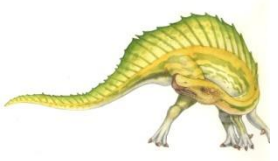


Device I/O Port Locations on PCs (partial)

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)

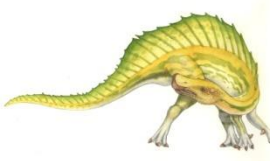
I/O Hardware (cont.)

- An I/O port consists of four registers:
  - **Data-in register** : read by the host to get input
  - **Data-out register:** written by the host to send output
  - **Statues register:** contain bits indicate states that can be read by the host, e.g. the current command has completed or a device error has occurred.
  - **Control register:** written by the host to start a command or to change the mode of a device.



Polling

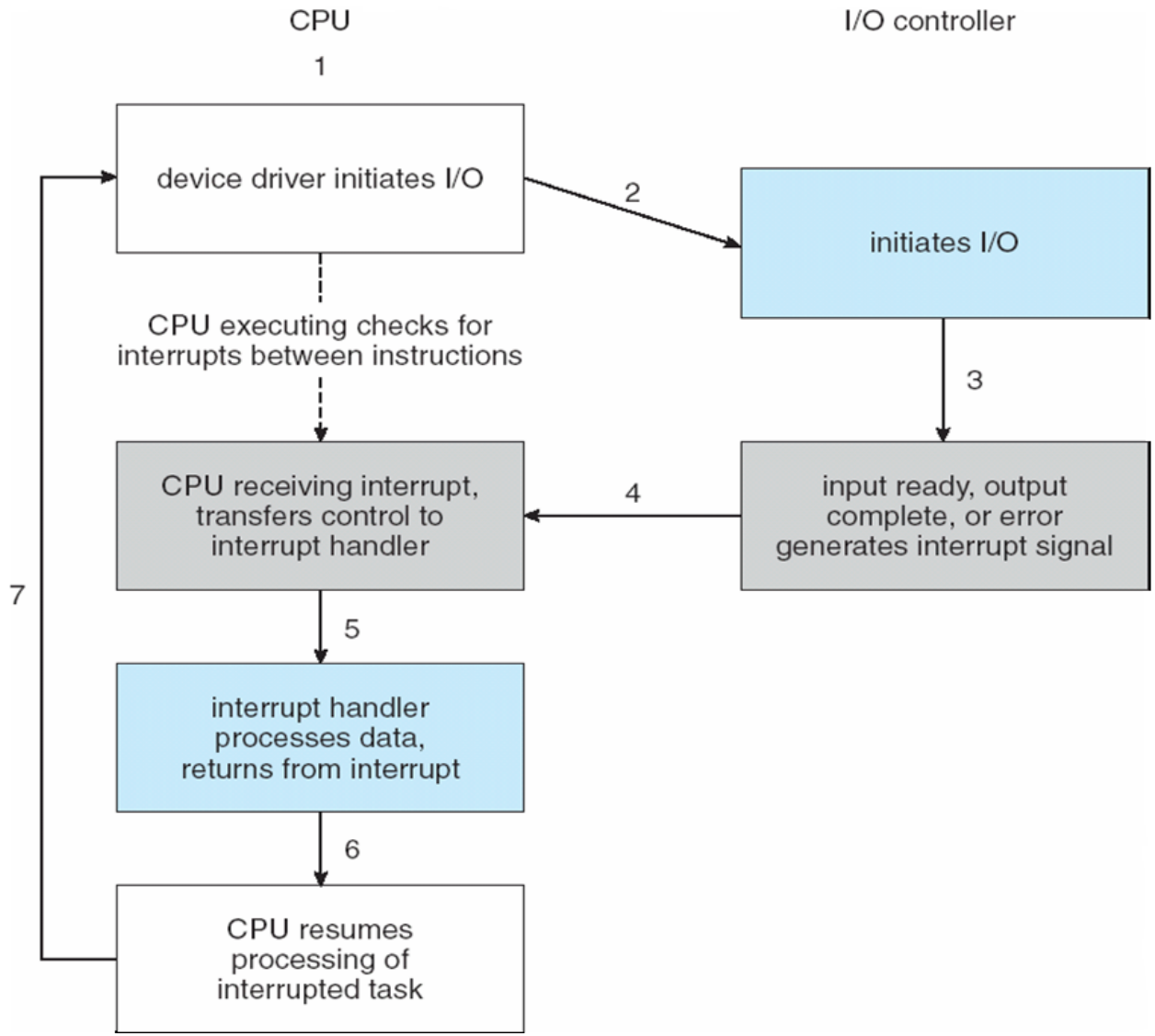
- The basic handshaking notion between the host and a controller is simple
- For example: the host writes output through a port, coordinating with the controller by handshaking as follows:
  1. The host repeatedly reads the **busy bit** until it become clear.
     - When the controller is
       - busy working ➔ sets the busy bit
       - ready to accept the next command ➔ clears the busy bit
  2. The host sets the **write bit** in the *command* register and writes a byte into the *data-out* register.
  3. The host sets the **command-ready bit**.
  4. When the controller notices that the command-ready bit is set, it sets the busy bit.
  5. The controller reads the *command register* and sees the write command. It reads the *data-out register* to get the byte and does the I/O to the device.
  6. The controller clears the command-ready bit, clears the error bit in the *statues register* to indicate that the device I/O succeeded, and clears the busy bit to indicate that it is finished.
- **In step 1 , the host is busy-waiting or *polling*.**

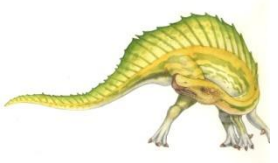


Interrupts

- The polling becomes inefficient when it is attempted repeatedly yet rarely finds a device to be ready for service, while other useful CPU processing remains undone.
- It may be more efficient to arrange for the hardware controller to notify the CPU when the device becomes ready for service, rather than polling.
- The hardware mechanism that enables a device to notify the CPU is called **an interrupt**.
- The CPU has a wire called **Interrupt-request line** triggered by I/O device
- **Interrupt handler** receives interrupts

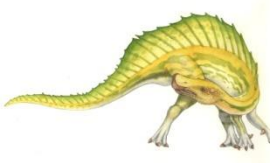
Interrupt-Driven I/O Cycle



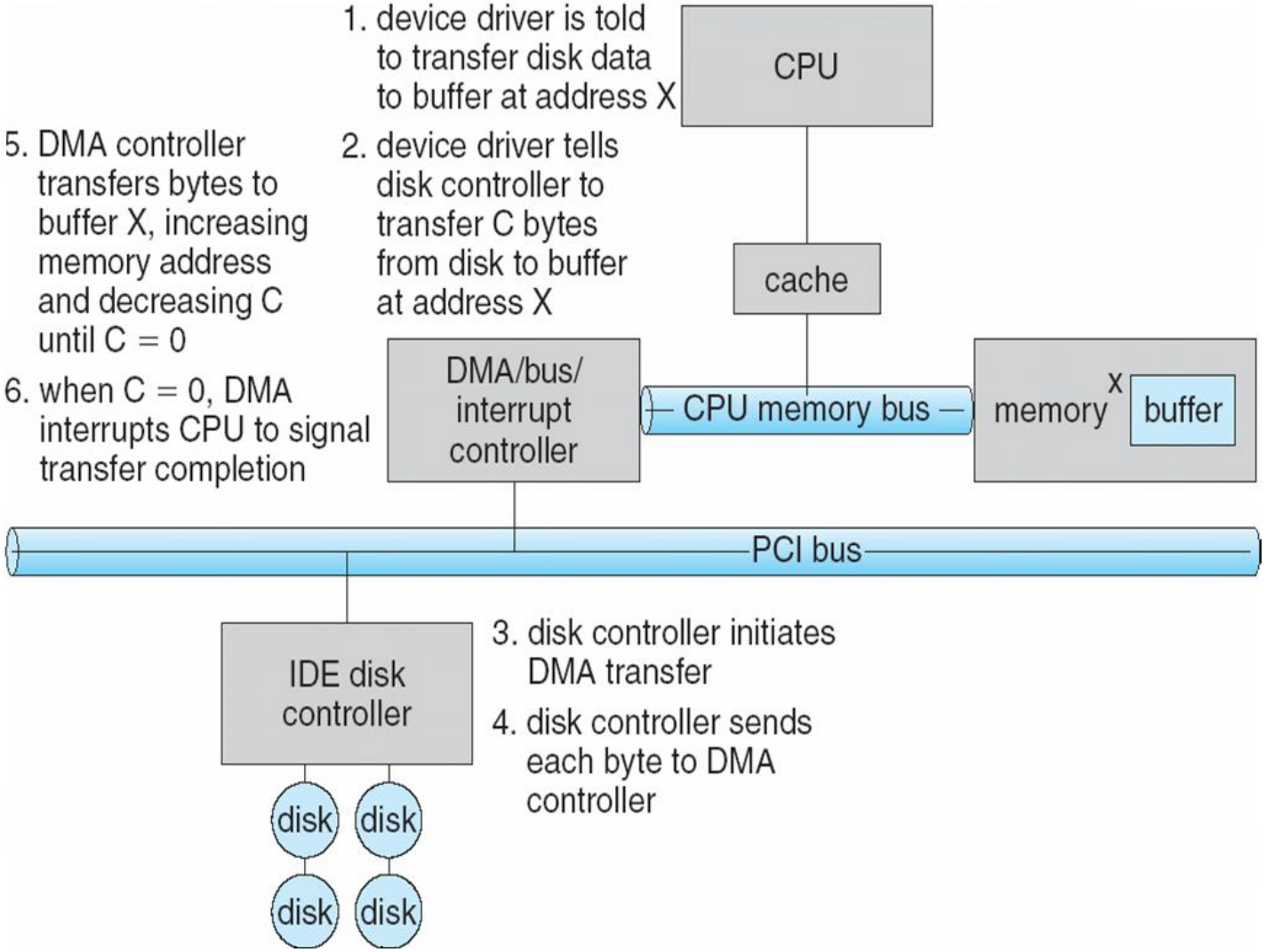


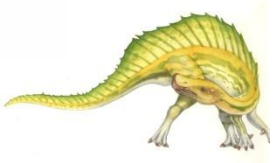
Direct Memory Access

- 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

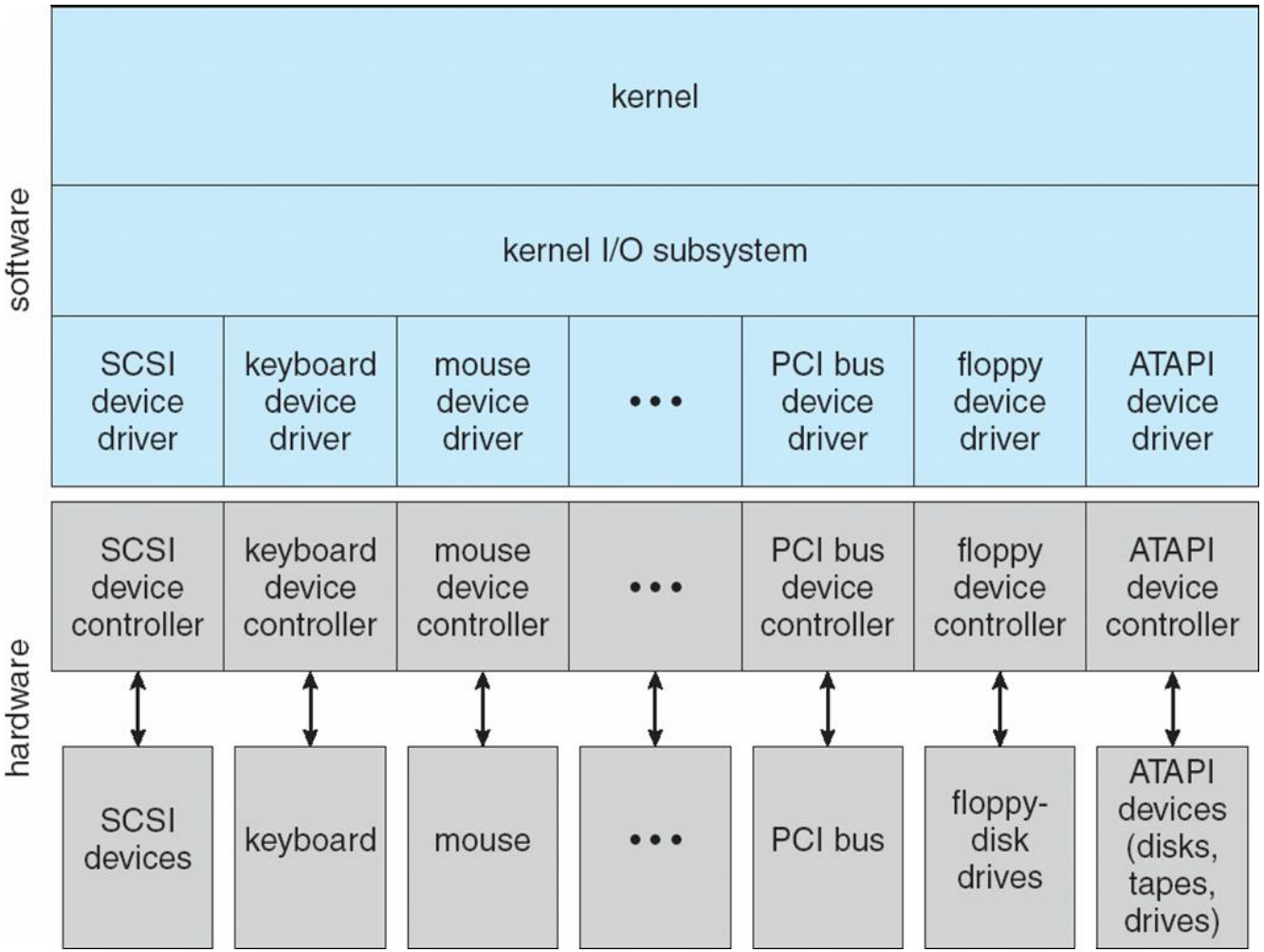




Application I/O Interface

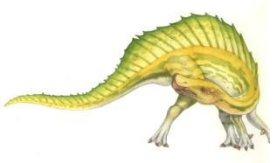
- I/O system calls encapsulate device behaviors in generic classes
- Device-driver layer hides differences among I/O controllers from kernel
- Devices vary in many dimensions
  - **Character-stream** or **block**
  - **Sequential or random-access**
  - **Sharable or dedicated**
  - **Speed of operation**
  - **read-write, read only,** or **write only**

A Kernel I/O Structure



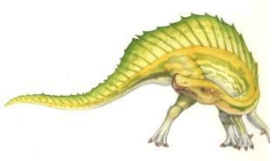
Characteristics of I/O Devices

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



Kernel I/O Subsystem

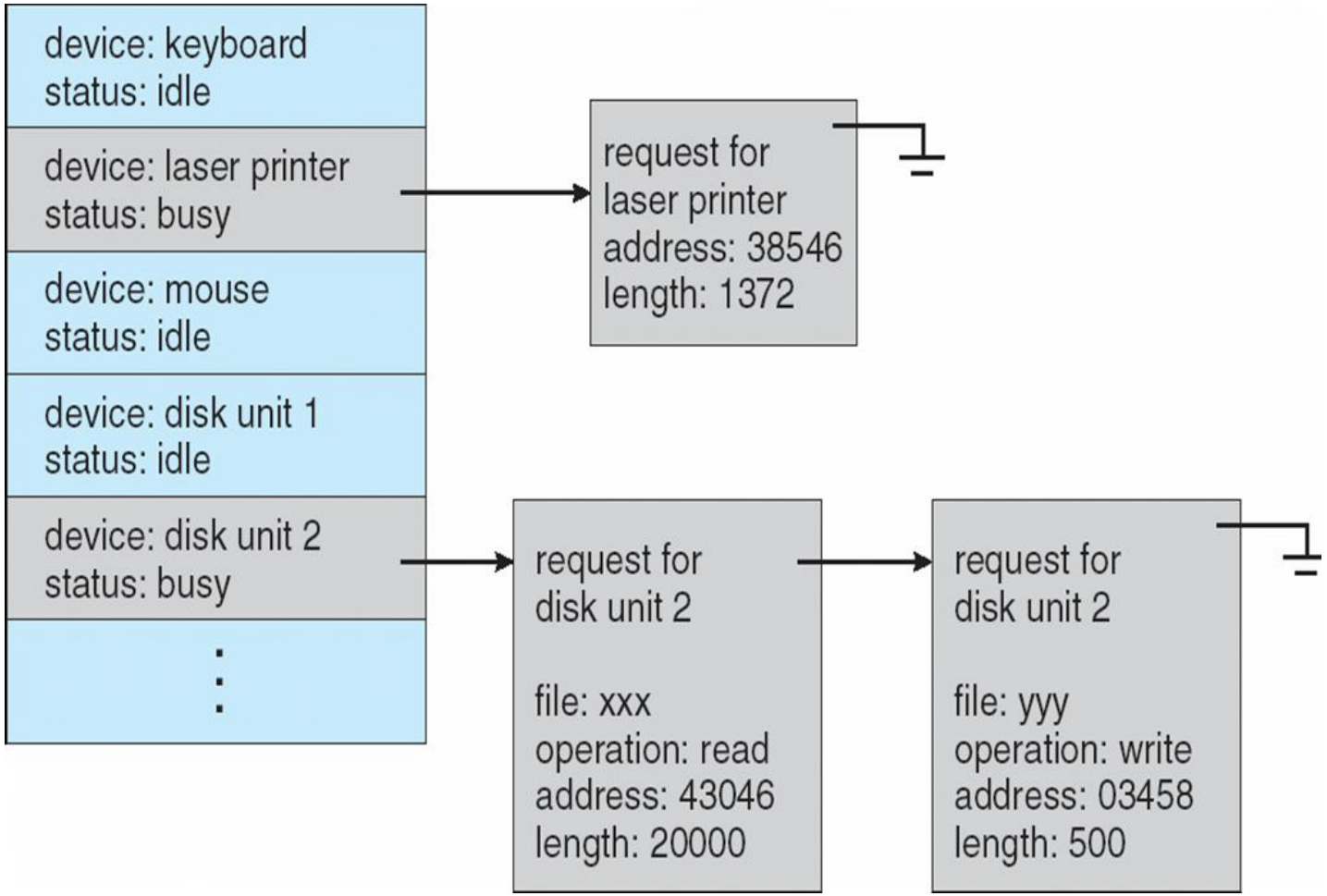
- Kernels provide many services related to I/O
- The kernel's I/O subsystem services are :
  - Scheduling
  - Buffering
  - Caching
  - Spooling
  - Device reservation
  - Error handling



Kernel I/O Subsystem services

- **Scheduling**
  - Some I/O request ordering via per-device queue (disk scheduling)
  - Some OSs try fairness (no one receiving poor service)
  - Some OS support asynchronous I/O by using device-status table
  - The kernel manages device-status table, which contains an entry for each I/O devices.
    - Each table entry indicates the device's type, address, and state
    - If the device is busy with the request , the type of request and other parameters will be stored in the table entry for that device.

Device-status Table



Kernel I/O Subsystem services (cont.)

- **Buffering** - store data in memory while transferring between devices
  - To cope with device speed mismatch
  - To cope with device transfer size mismatch
- **Caching** - fast memory holding copy of data
  - Always just a copy
  - Key to performance

Kernel I/O Subsystem services (cont.)

- **Spooling** - hold output for a device
  - If device can serve only one request at a time
  - i.e., Printing
- **Device reservation** - provides exclusive access to a device
  - System calls for allocation and deallocation
  - Watch out for deadlock
- **Error Handling**
  - OS can recover from disk read, device unavailable, transient write failures
  - Most return an error number or code when I/O request fails

End of Chapter 13