CSI 4500 Operating Systems

I/O Systems

The Requirements of I/O

- What about I/O?
  - Without I/O, computers are useless (why we say so?)
  - But… thousands of devices, each slightly different
    - How can we standardize the interfaces to these devices?
  - Devices unreliable: media failures and transmission errors
    - How can we make them reliable?
  - Devices unpredictable and/or slow
    - How can we manage them if we don't know what they will do or how they will perform?
    - How to use the I/O devices efficiently?

The Requirements of I/O (Cont.)

- Some operational parameters:
  - Byte/Block
    - Some devices provide single byte at a time (*e.g.* keyboard)
    - Others provide whole blocks (*e.g.* disks, networks, etc)
  - Sequential/Random
    - Some devices must be accessed sequentially (*e.g.* tape)
    - Others can be accessed randomly (*e.g.* disk, cd, etc.)
  - Polling/Interrupts
    - Some devices require continual monitoring
    - Others generate interrupts when they need service

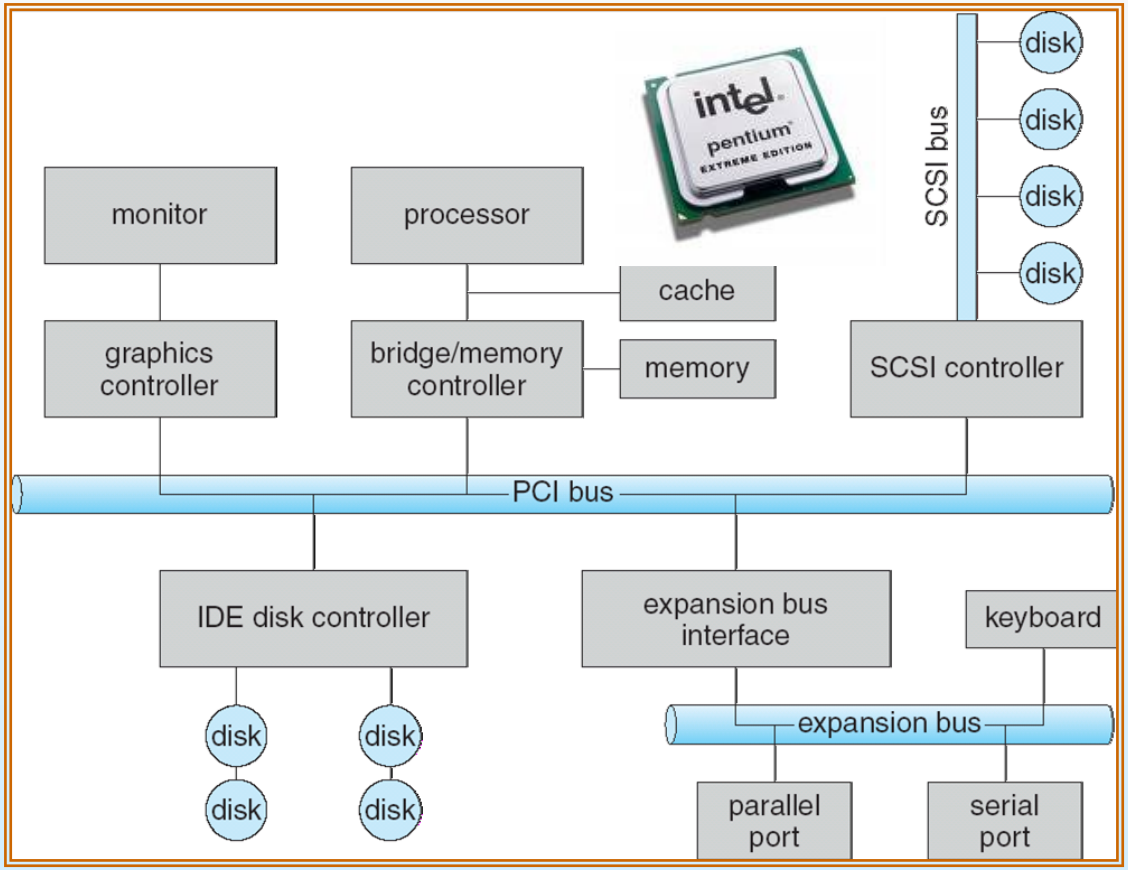
The Goal of the I/O Subsystem

- Provide Uniform Interfaces, Despite a wide range of different devices
  - This code works on many different devices:

```
FILE fd = fopen("/dev/devicename","rw");
for (int i = 0; i < 10; i++) {
   fprintf(fd,"Count %d\n",i);
}
close(fd);
```

  - Why? Because code that controls devices ("device driver") implements standard interface (***fopen***).
- We will study what is involved in actually controlling devices in rest of lecture
  - Can only touch the surface!

Modern I/O Systems

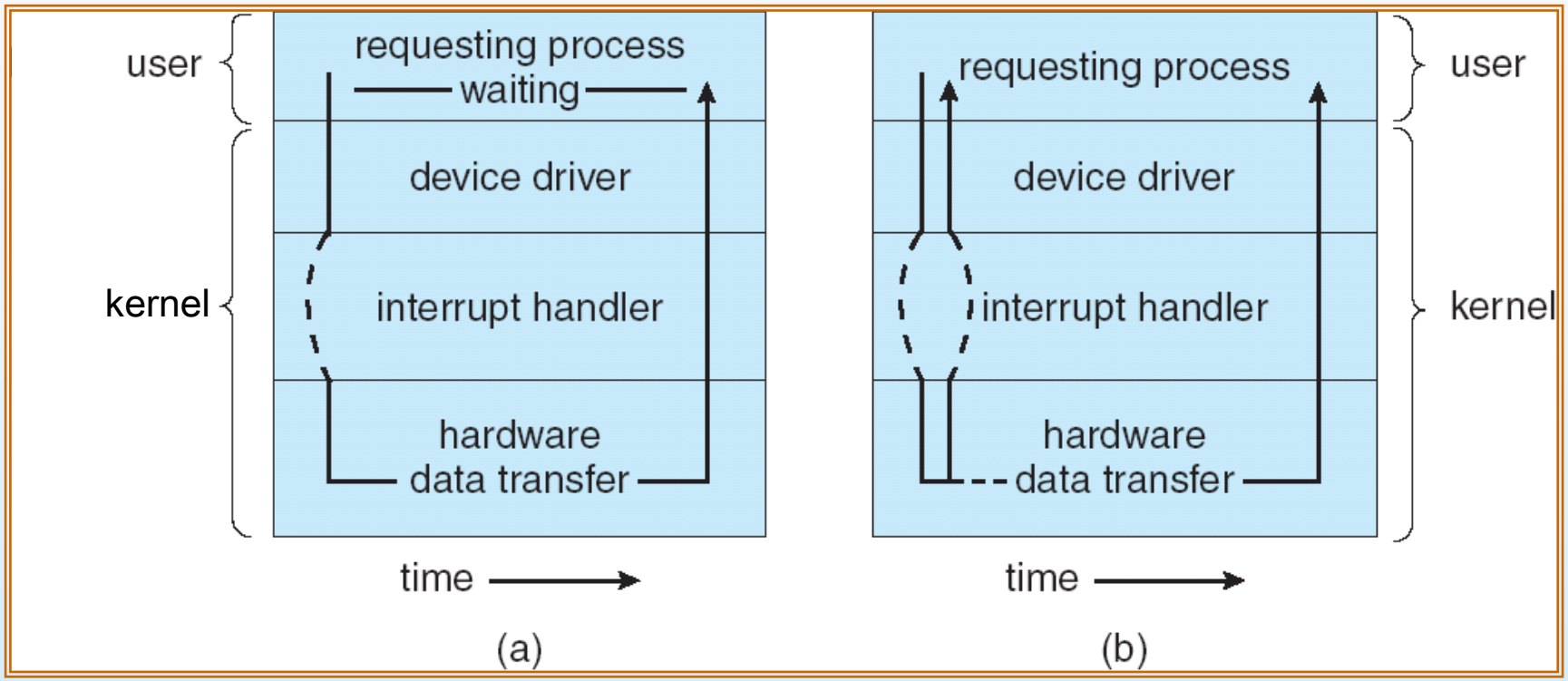


Oakland UNIVERSITY

Devices Controller

- Converts between serial bit stream and a block of bytes
- Performs error correction if necessary
- Components:
  - Device registers to communicate with the CPU
  - Data buffer that an OS can read or write

Two I/O Methods

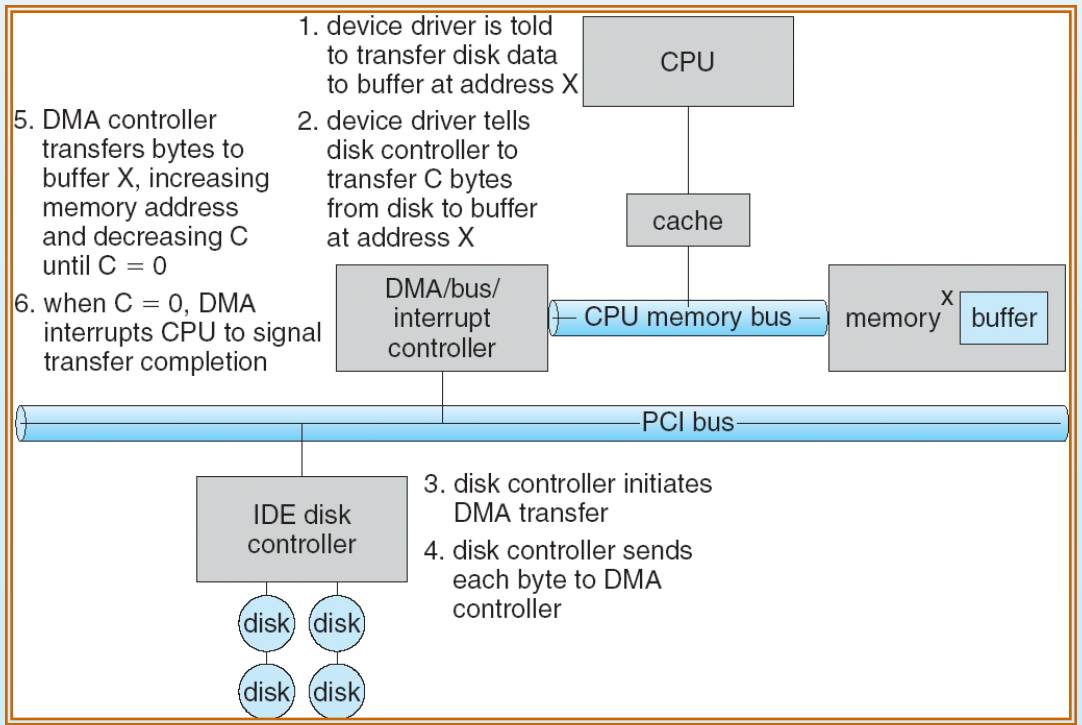


Synchronous

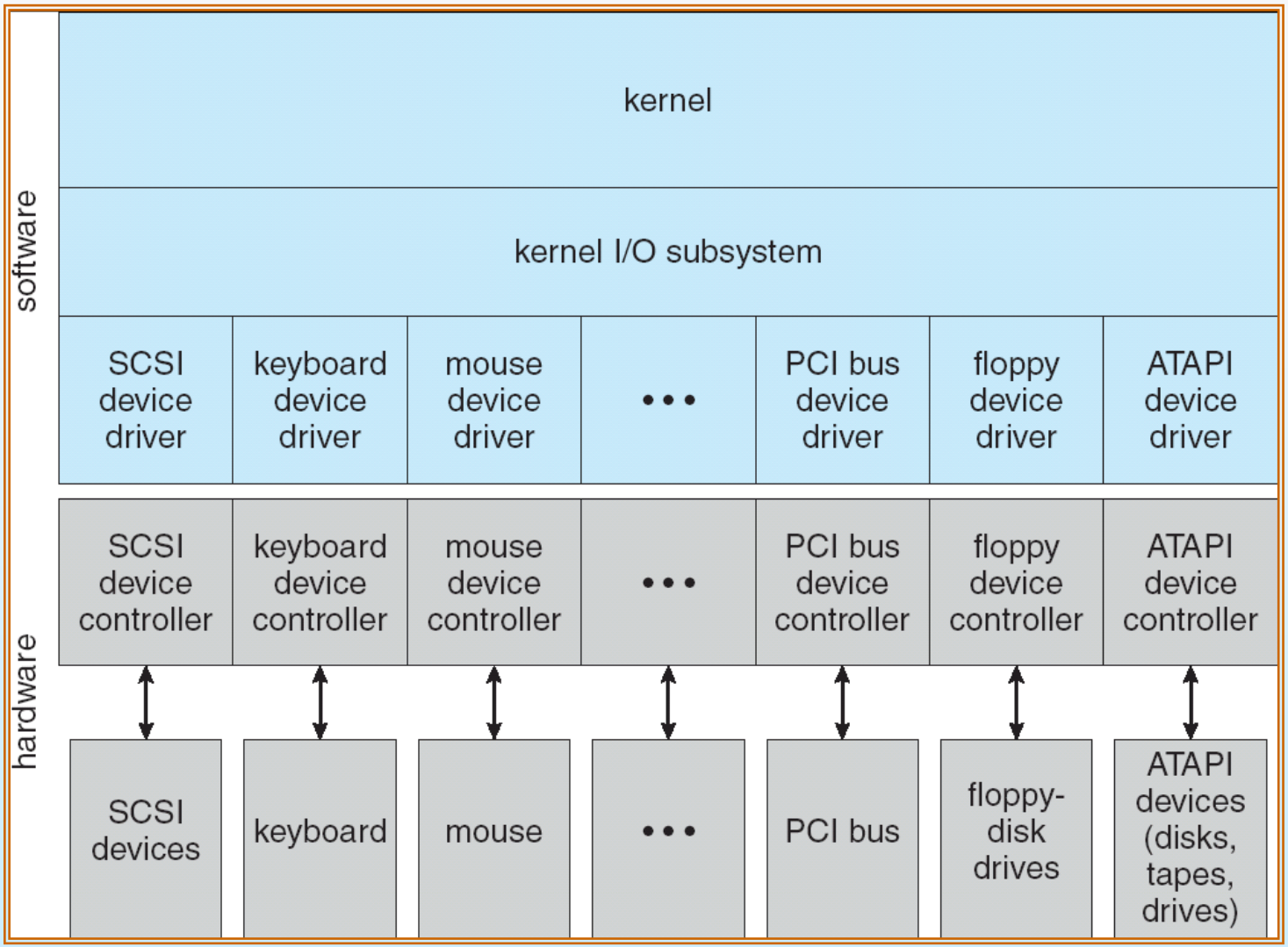
Asynchronous

Transferring Data To/From Controller

- Programmed I/O:
  - Each byte transferred via processor in/out or load/store
  - Pro: Simple hardware, easy to program
  - Con: Consumes processor cycles proportional to data size
- Direct Memory Access:
  - Give controller (DMA controller) access to memory bus
  - Ask it to transfer data to/from memory directly while bypassing CPU
- Sample interaction with DMA controller (from book)



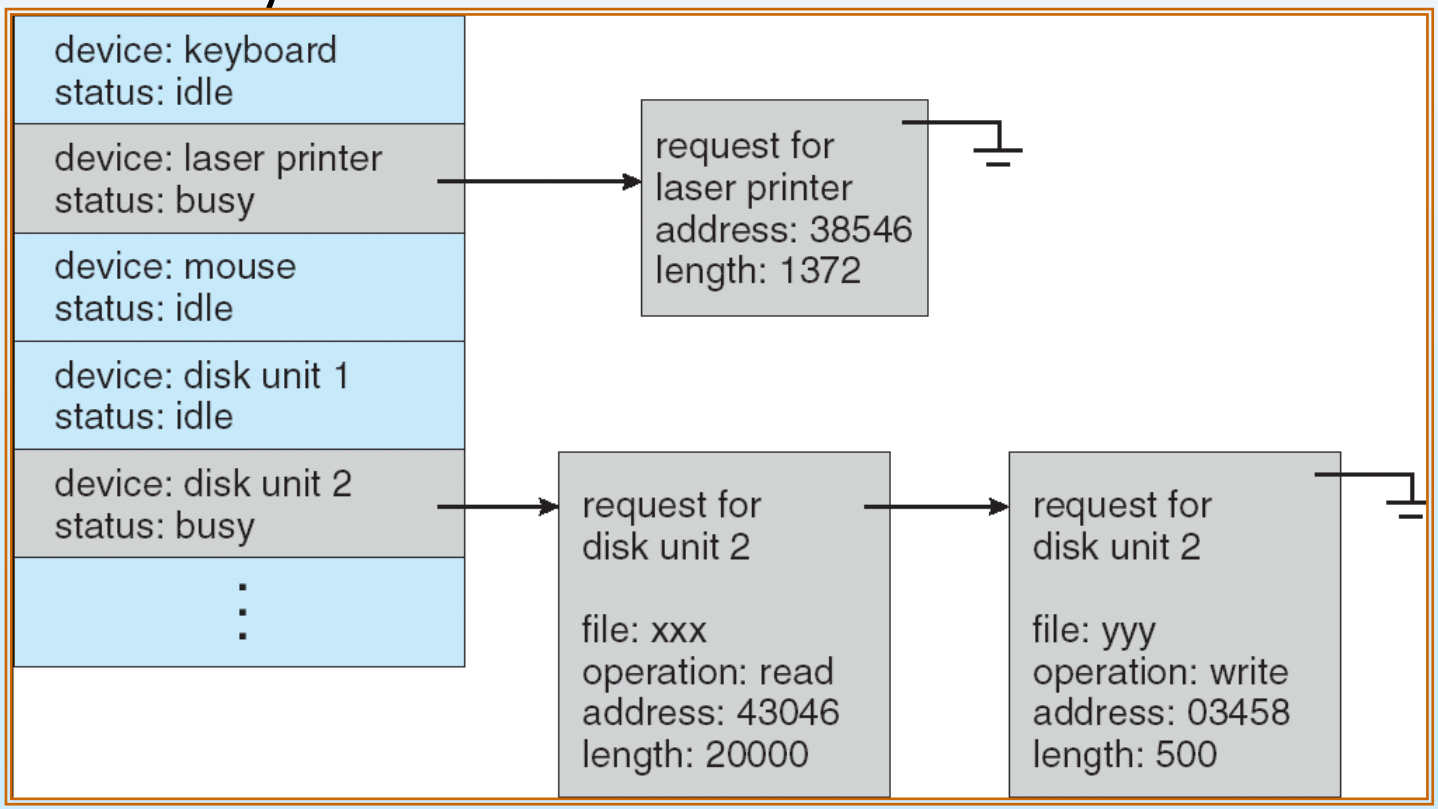
A Kernel I/O Structure



Kernel I/O Subsystem (1)

I/O Scheduling

- Some I/O request ordering via per-device queue
- Some OSs try fairness



Kernel I/O Subsystem (2)

- **I/O 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”

Kernel I/O Subsystem (3)

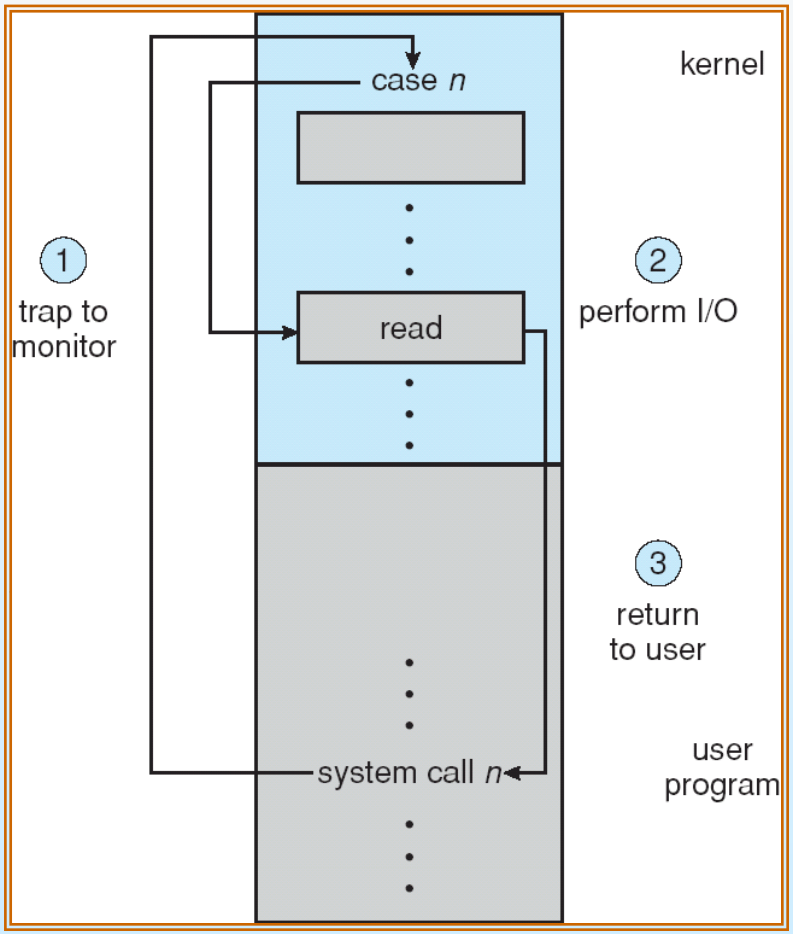
- **Caching** - fast memory holding copy of data
  - Always just a copy
  - Key to performance
- **Spooling** - hold output for a device
  - If device can serve only one request at a time, i.e., Printing
- **Device reservation** - provides exclusive access to a device
  - System calls for allocation 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
  - In UNIX OSes, *errno* is used to return an error code
- System error logs hold problem reports

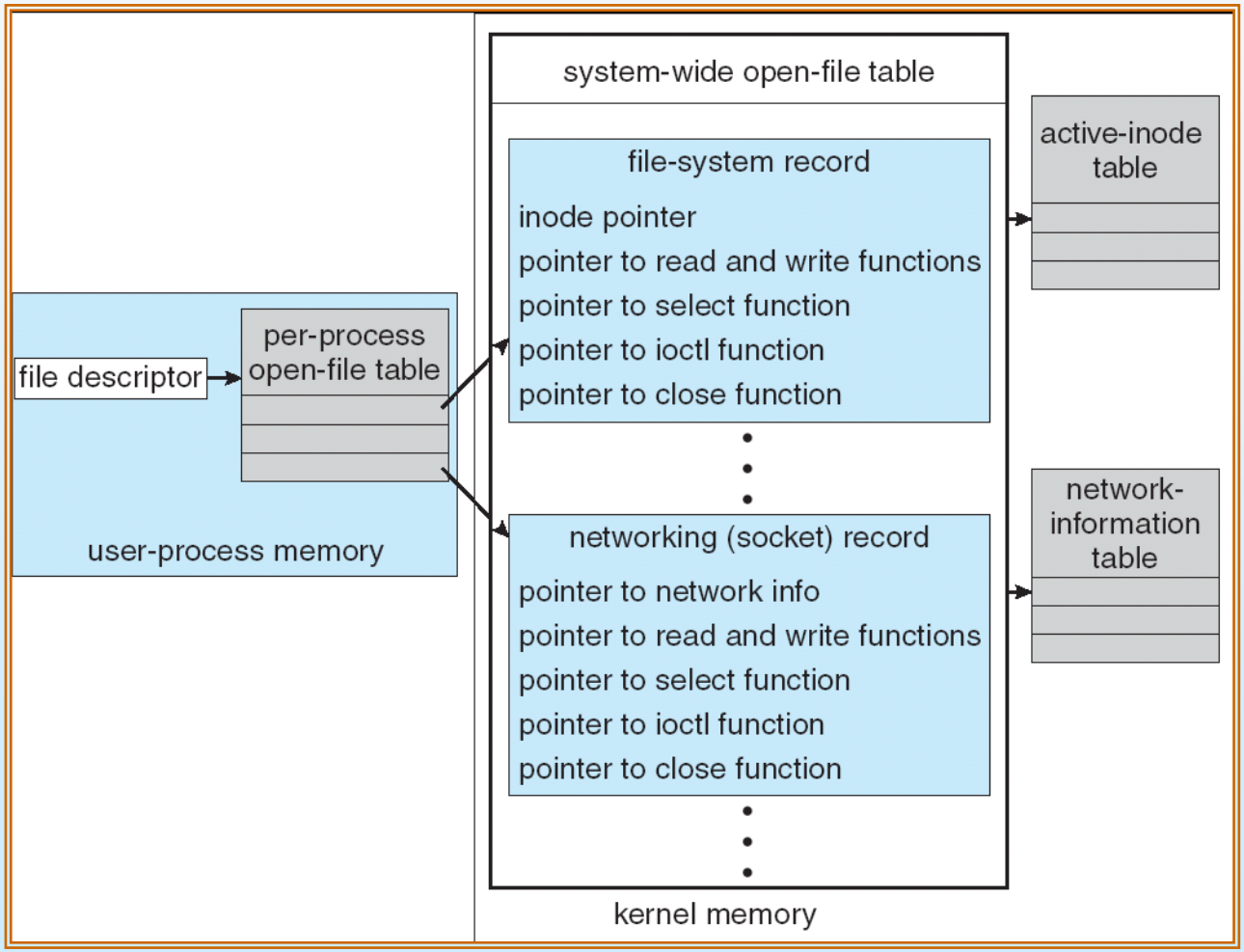
I/O Protection

- User process may accidentally or purposefully attempt to disrupt normal operation via illegal I/O instructions
  - 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



Kernel Data Structures

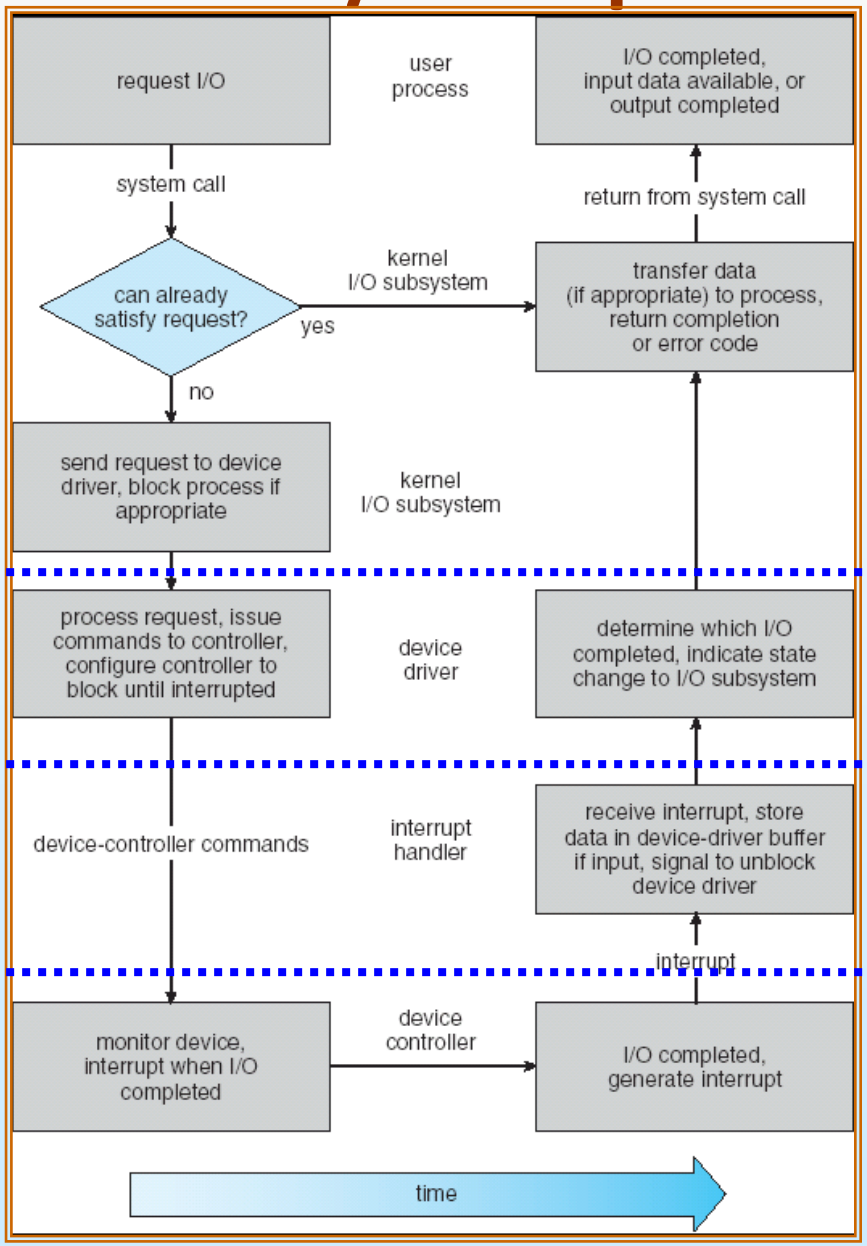
- Kernel keeps state info for I/O components, including open file tables, network connections, character device state



Device Drivers

- Device Driver: Device-specific code in the kernel that interacts directly with the device hardware
  - Supports a standard, internal interface
  - Same kernel I/O system can interact easily with different device drivers
  - Special device-specific configuration supported with the `ioctl()` system call
- Device Drivers typically divided into two pieces:
  - Top half: accessed in call path from system calls
    - Implements a set of standard, cross-device calls like `open()`, `close()`, `read()`, `write()`, `ioctl()`
    - This is the kernel's interface to the device driver
    - Top half will *start* I/O to device, may put thread to sleep until finished
  - Bottom half: run as interrupt routine
    - Gets input or transfers next block of output
    - May wake sleeping threads if I/O now complete

Life Cycle of An I/O Request



Device Driver Top Half

Device Driver Bottom Half

I/O Device Notifying the OS

- The OS needs to know when:
  - The I/O device has completed an operation
  - The I/O operation has encountered an error

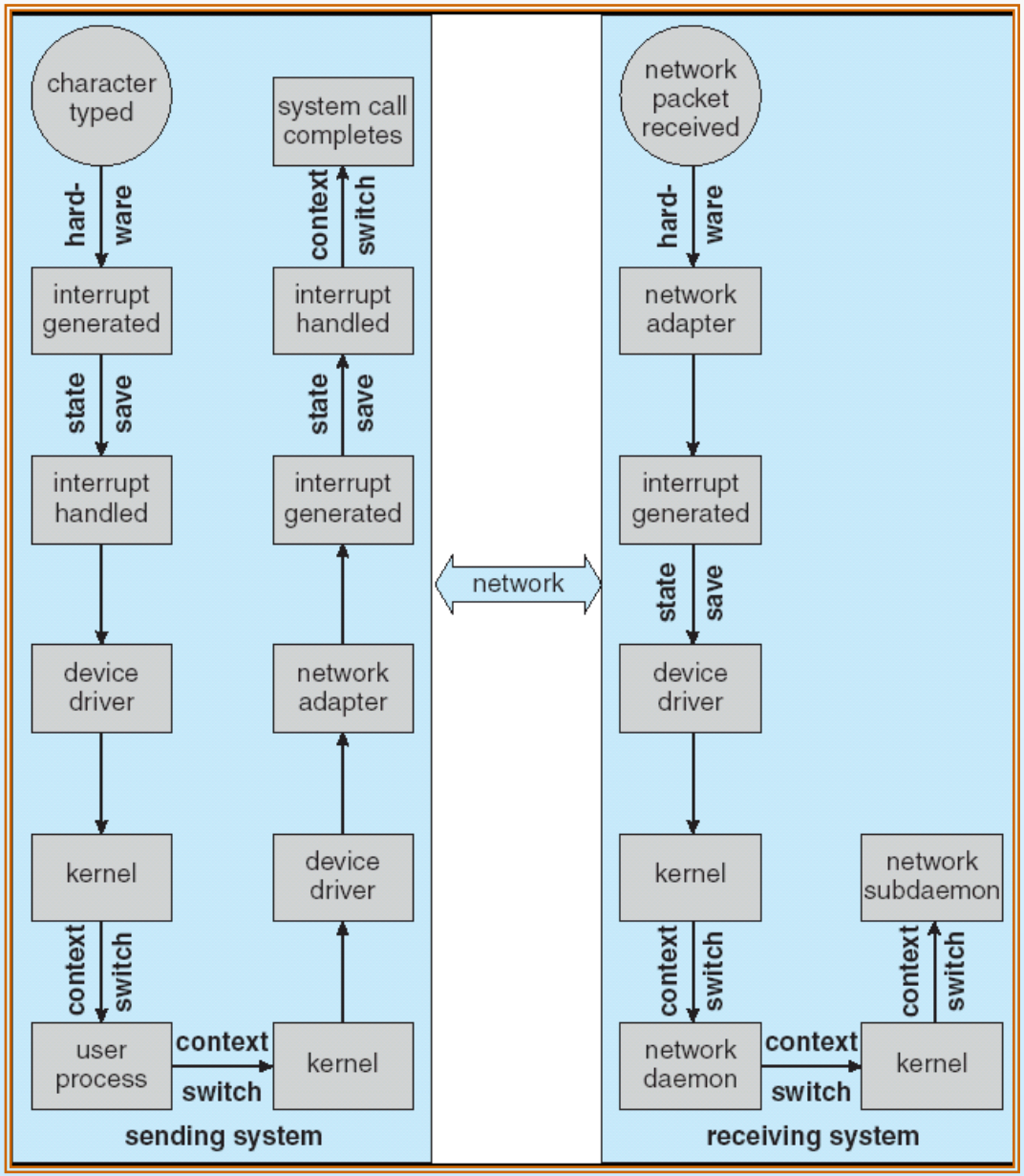
I/O Interrupt	Polling
■Device generates an interrupt whenever it needs service ■Handled in bottom half of device driver ●Often run on special kernel-level stack	■OS periodically checks a device-specific status register ●I/O device puts completion information in status register ●Could use timer to invoke lower half of drivers occasionally
Pro: handles unpredictable events well Con: interrupts relatively high overhead	Pro: low overhead Con: may waste many cycles on polling if infrequent or unpredictable I/O operations

- Actual devices combine both polling and interrupts
  - For instance: High-bandwidth network device:
    - Interrupt for first incoming packet
    - Poll for following packets until hardware empty

Performance

- Balance CPU, memory, bus, and I/O performance for highest throughput
- I/O a major factor in system performance:
  - Demands CPU to execute device driver, kernel I/O code
  - Context switches due to interrupts
    - Reduce number of context switches
  - Data copying
    - Reduce data copying while passing between devices and processes
    - Use DMA-knowledgeable controllers to increase concurrency
  - Network traffic especially stressful
    - Reduce the frequency of interrupts by using large transfers, smart controllers, polling

Intercomputer Communications

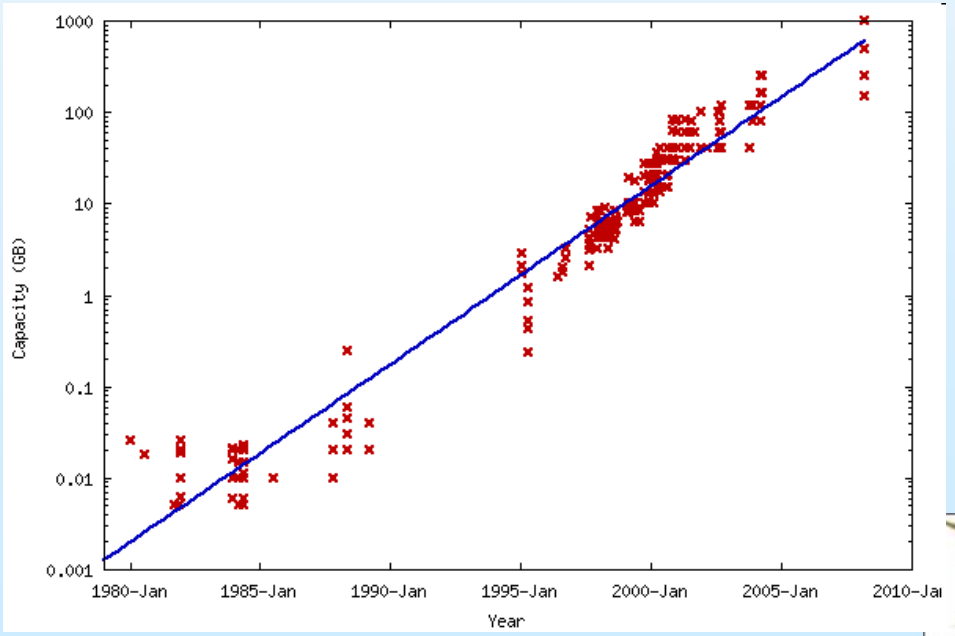


Summary

- I/O Devices Types:
  - Many different speeds (0.1 bytes/sec to GBytes/sec)
  - Different Access Patterns:
    - Block Devices, Character Devices, Network Devices
  - Different Access Timing:
    - Blocking, Non-blocking, Asynchronous
- I/O Controllers: Hardware that controls actual device
  - Processor Accesses through I/O instructions, load/store to special physical memory
  - Report their results through either interrupts or a status register that processor looks at occasionally (polling)
- Device Driver: Device-specific code in kernel

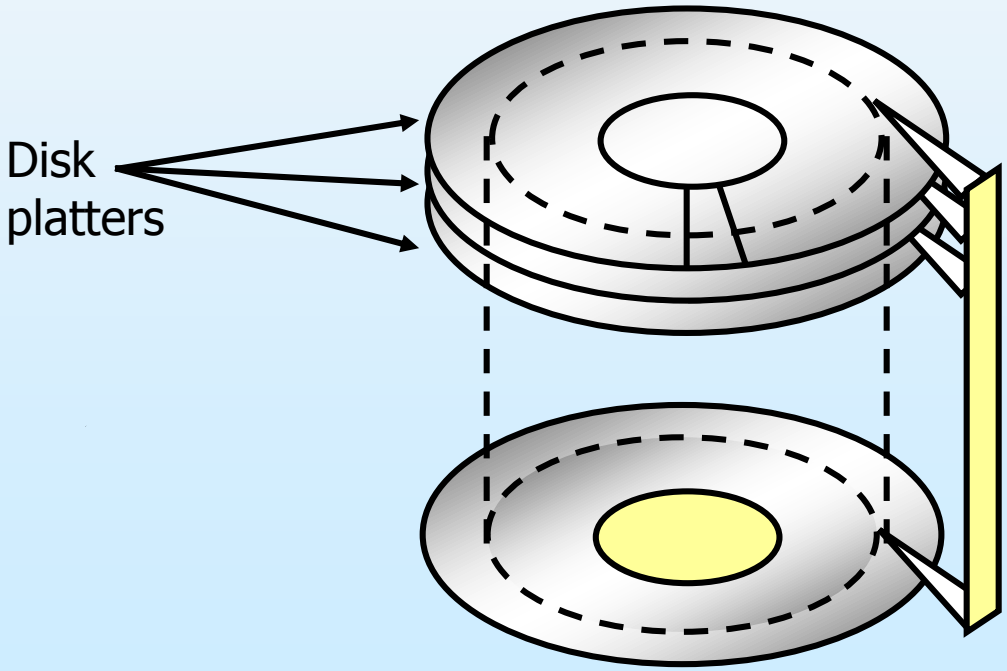
Disk as An Example Device of Storage

- Non-volatile storage
- More than 60 years old (was born on Sep.13th,1956)
- Incredibly complicated
- A modern drive
  - 250,000 lines of micro code



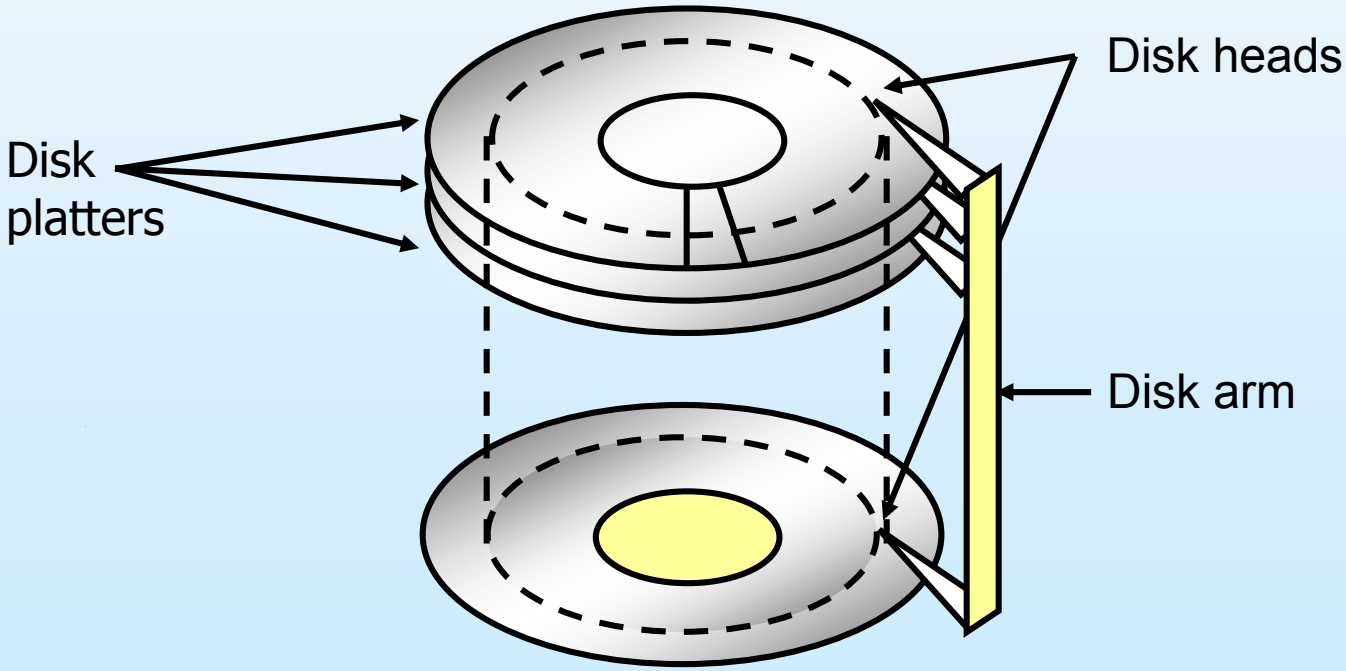
Disk Characteristics

- **Disk platters:** coated with magnetic materials for recording data.



Disk Characteristics

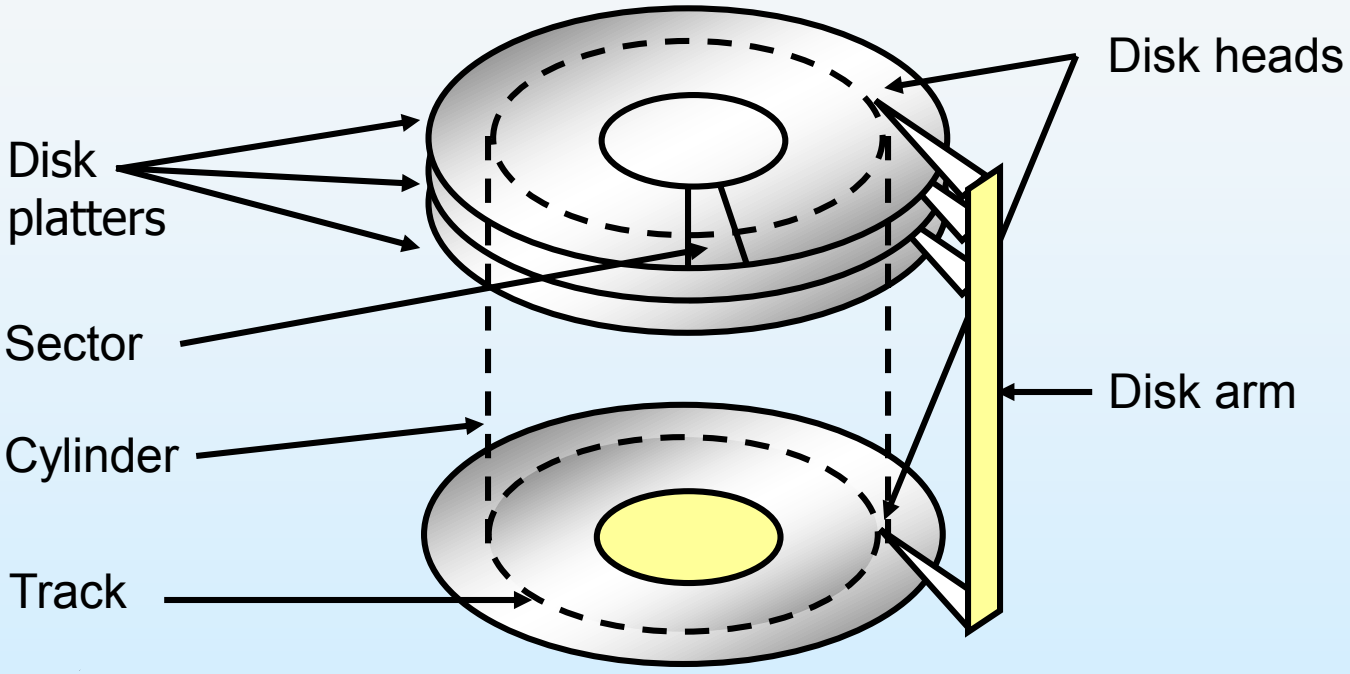
- **Disk arm:** moves a comb of **disk heads**
  - Only one disk head is active for reading/writing



Hard Disk Trivia…

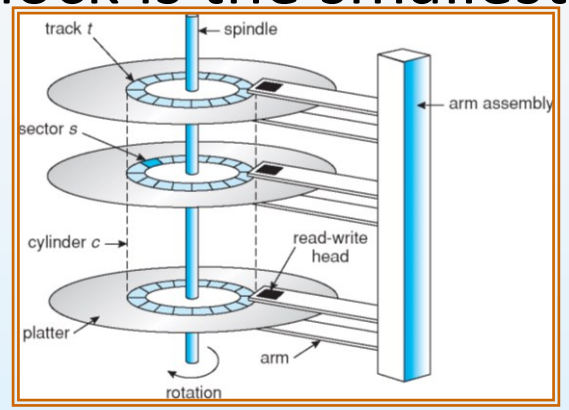
- Aerodynamically designed to fly
  - As close to the surface as possible
  - No room for air molecules
- Therefore, hard drives are filled with special inert gas
- If head touches the surface
  - Head crash
  - Scrapes off magnetic information

Disk Characteristics



Disk Structure

- **Disk drives** are addressed as large 1-dimensional arrays of *logical blocks*, where the logical block is the smallest unit of transfer.



- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially.
  - Sector 0 is the first sector of the first track on the outermost cylinder.
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.

Disk Performance Metrics

- Access time has two major components
  - **Seek time** is the time for the disk are to move the heads to the cylinder containing the desired sector.
  - **Rotational latency** is the additional time waiting for the disk to rotate the desired sector to the disk head.
    - Assume 7,200 rotations per minute (RPM)
    - 7,200 / 60 = 120 rotations per second
    - 1/120 = ~8 msec per rotation
    - Average rotational delay is ~4 msec

Disk Performance Metrics

- **Transfer time:** the time to transfer bytes
  - Assumptions:
    - 58 Mbytes/sec
    - 4-Kbyte disk blocks
  - Time to transfer a block takes 0.07 msec

- **Disk access time**
  - Seek time + rotational delay + transfer time

Disk Performance Metrics

- **Latency**
  - Seek time + rotational delay
- **Bandwidth** B = D/T
  - Bytes transferred / disk access time
  - D is the total number of bytes transferred
  - T = the completion of the last transfer - first request for service.

Examples of Disk Access Times

- If disk blocks are randomly accessed
  - Average disk access time = ~8 msec
  - Assume 4-Kbyte blocks
  - 4 Kbyte / 8 msec = ~500 Kbyte/sec
- If disk blocks of the same cylinder are randomly accessed without disk seeks
  - Average disk access time = ~4 msec
  - 4 Kbyte / 4 msec = ~ 1 Mbyte/sec

Examples of Disk Access Times

- If disk blocks are accessed sequentially
  - Without seeks and rotational delays
  - Bandwidth: 58 Mbytes/sec
- Key to good disk performance
  - Minimize seek time and rotational latency

Disk Controller

- Few popular standards
  - IDE (integrated device electronics)
  - ATA (advanced technology attachment interface)
  - SCSI (small computer systems interface)
  - SATA (serial ATA)
- Differences
  - Performance
  - Parallelism

Disk Device Driver

- The operating system is responsible for using hardware efficiently
  - Fast access time
  - Large disk bandwidth.
- Major goal: reduce seek time for disk accesses
  - Schedule disk requests to minimize disk arm movements

Disk Scheduling (Cont.)

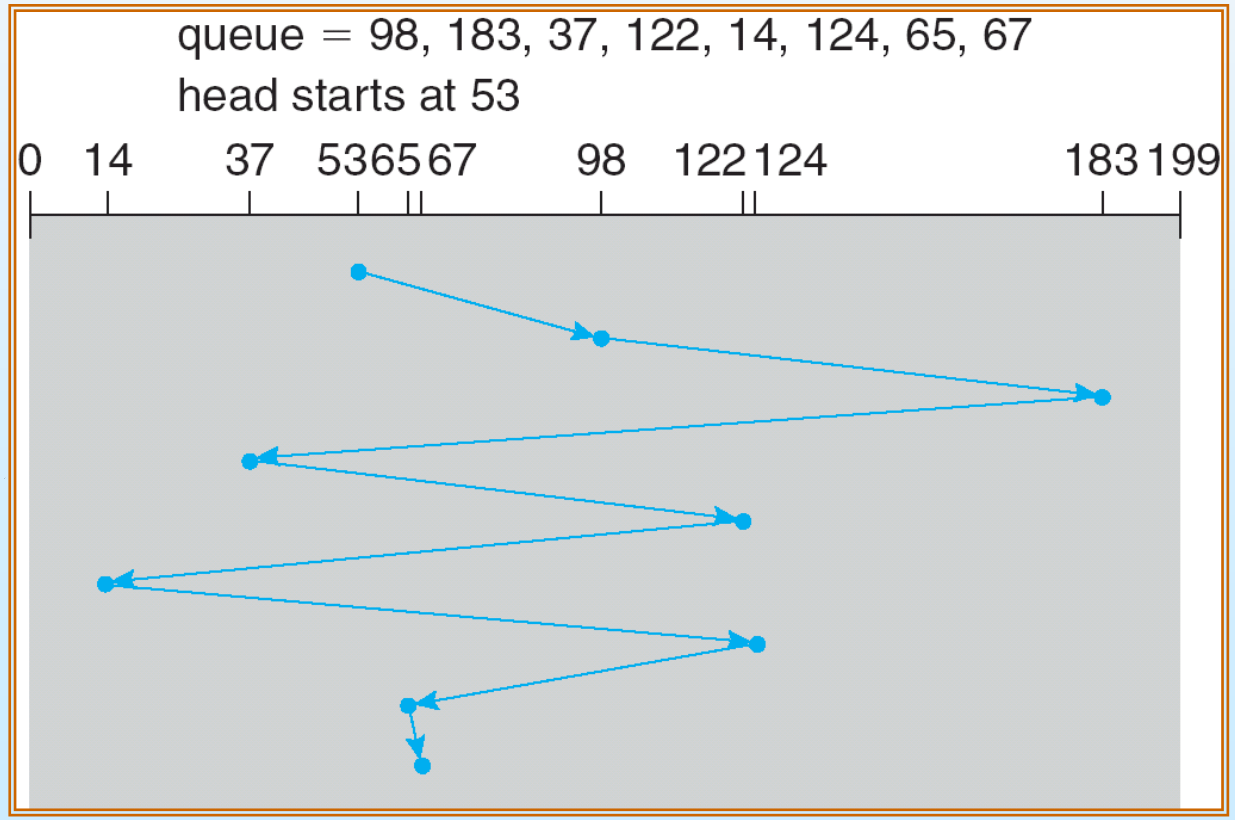
- Several algorithms exist to schedule the servicing of disk I/O requests.
- We illustrate them with a request queue (0-199).

Target Cylinders: 98, 183, 37, 122, 14, 124, 65, 67

Head pointer currently at 53

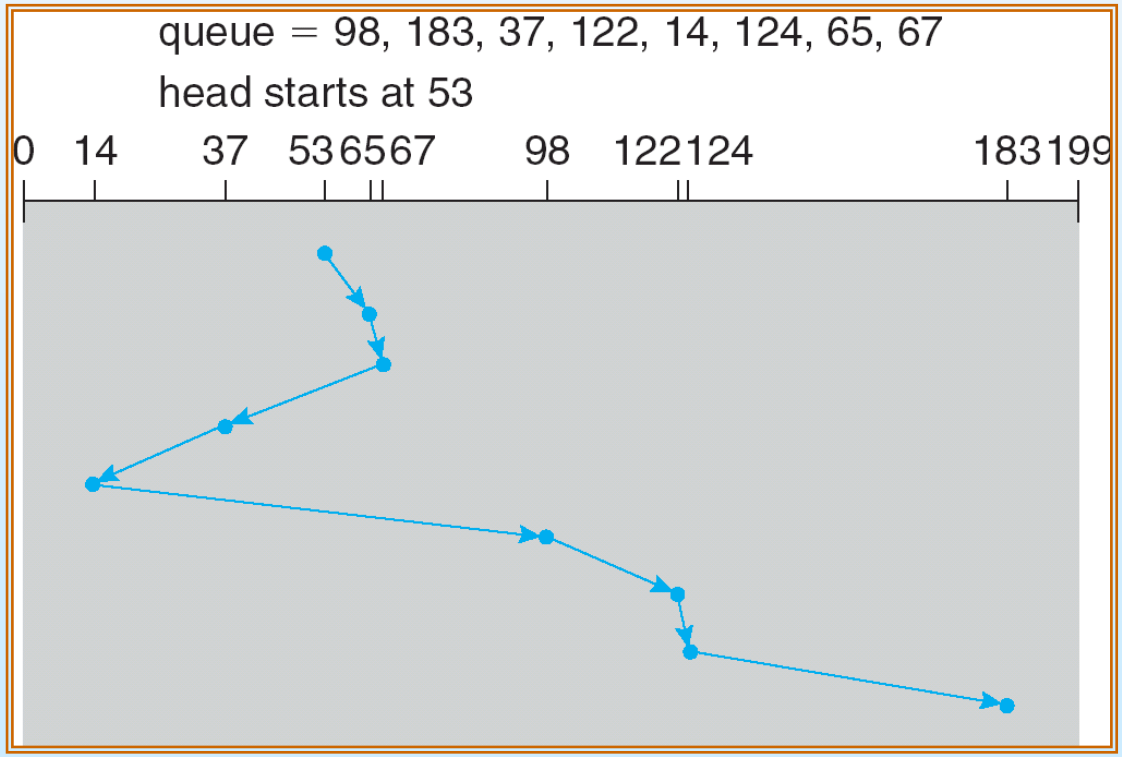
First Come First Serve

- FCFS: requests are served in the order of arrival
  - Fair among requesters
  - Poor for accesses to random disk blocks
- Illustration shows total head movement of 640 cylinders



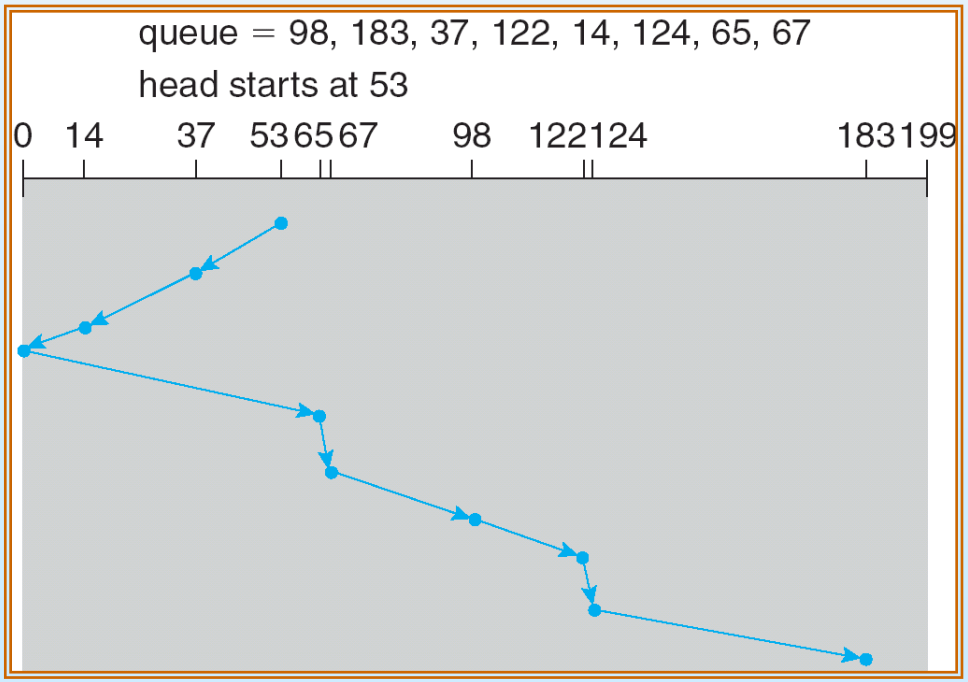
Shorted Seek Time First

- Selects the request with the minimum seek time from the current head position.
- SSTF scheduling is a form of SJF scheduling
  - may cause starvation of some requests.
- Illustration shows total head movement of 236 cylinders.



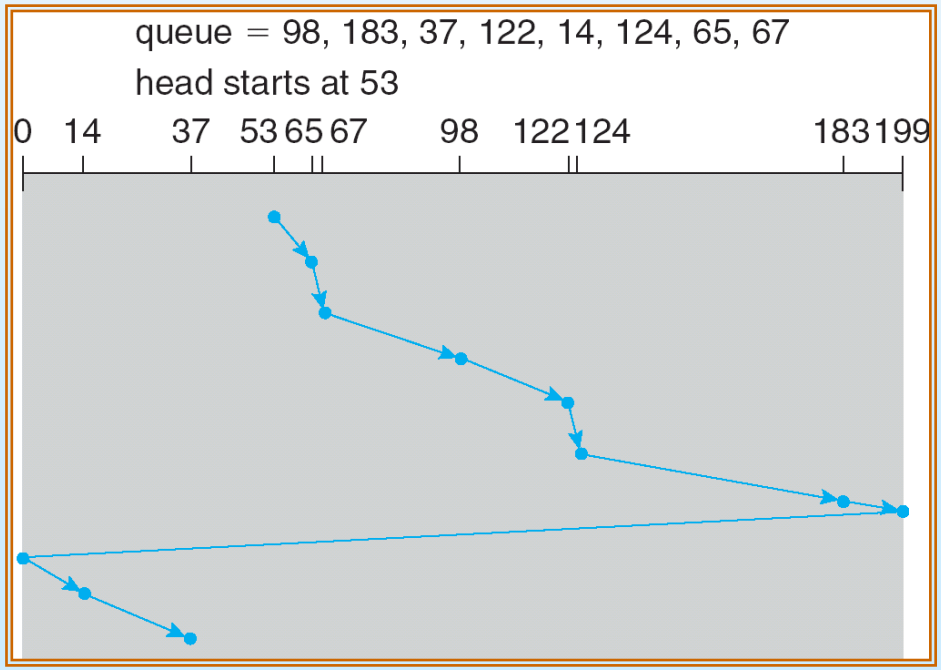
SCAN

- **Elevator algorithm**: takes the closest request in the direction of travel (an example of elevator algorithm)
  - No starvation
  - A new request can wait for almost two full scans of the disk
- Illustration shows total head movement of 208 cylinders.



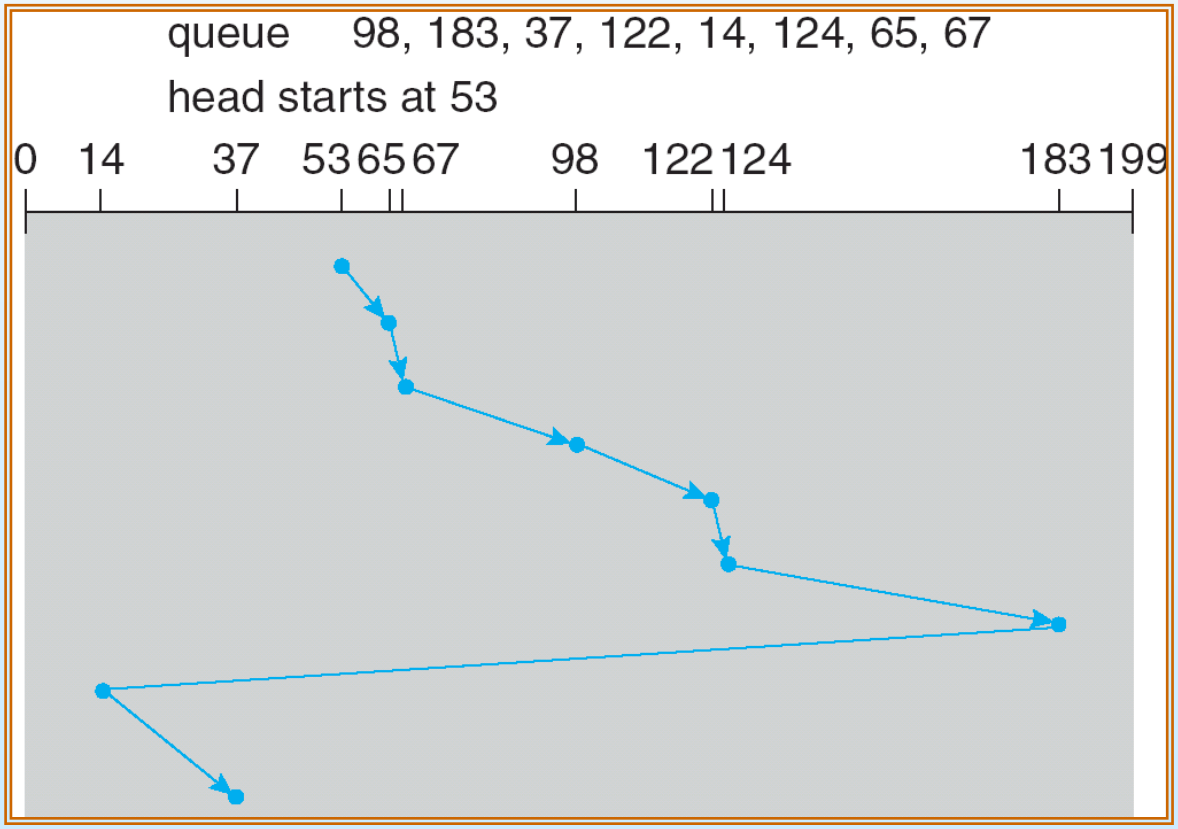
Circular-SCAN

- Provides a more uniform wait time than SCAN.
- The head moves from one end of the disk to the other. servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one.



C-LOOK

- Variation of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.



Selecting a Disk-Scheduling Algorithm

- Performance depends on the number and types of requests.
- Requests for disk service can be influenced by the file-allocation method.
- The location of the directories and index blocks is also important.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.
  - SSTF is common and has a natural appeal
  - SCAN and C-SCAN perform better for systems that place a heavy load on the disk.
  - Either SSTF or LOOK is a reasonable choice for the default algorithm.

Summary

- Disk drives are the major secondary-storage I/O devices
- Requests for disk I/O are generated by the file system and by the virtual memory system.
  - Each request has a logical block number
  - Disk scheduling algorithms can improve the effective bandwidth, the average response time, the variance in response time.