OPERATING SYSTEMS II

DPL. ING. CIPRIAN PUNGILĂ, PHD.

I/O. Storage Devices.

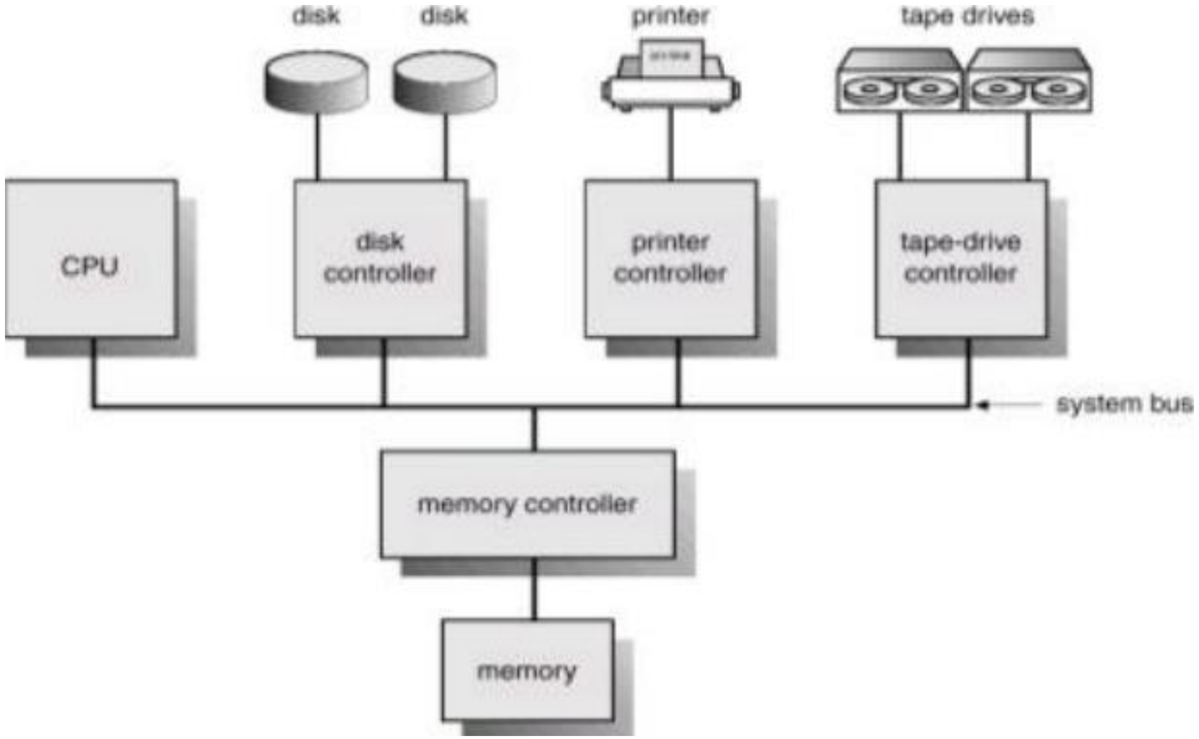
A LOOK INSIDE THE MACHINE

BIBLIOGRAPHY

1. A. TANNENBAUM, “MODERN OPERATING SYSTEMS”, 3RD EDITION, 2008, PRENTICE HALL
2. SILBERSCHATZ, GALVIN, AND GAGNE, "OPERATING SYSTEM CONCEPTS", 8TH EDITION, 2009, WILEY

Input/Output (I/O) Principles
Hardware I/O. Controller. DMA.

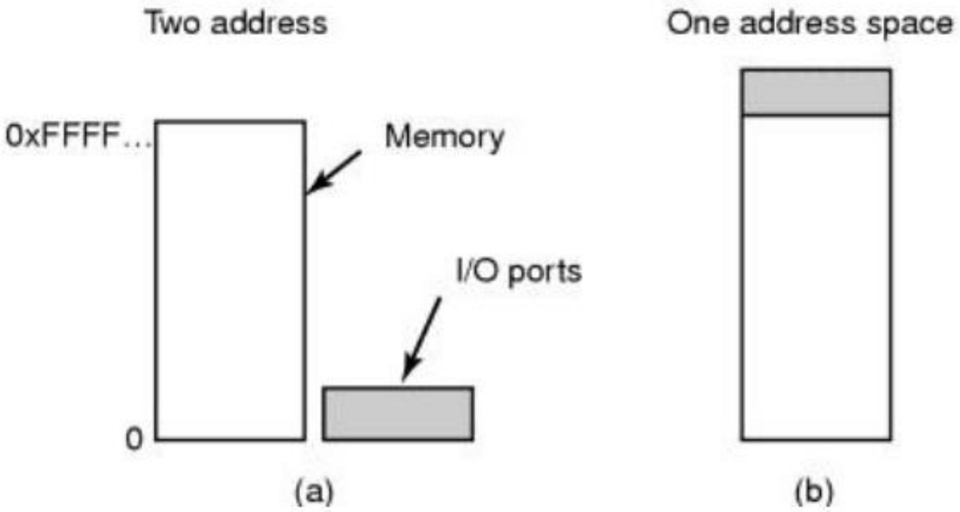
- ❑I/O devices have both a mechanical and an electronic component (device controller)
- ❑Device controller
  - ❑Control logic
  - ❑Command registers
  - ❑Status registers
  - ❑On-board buffer space



Input/Output (I/O) Principles

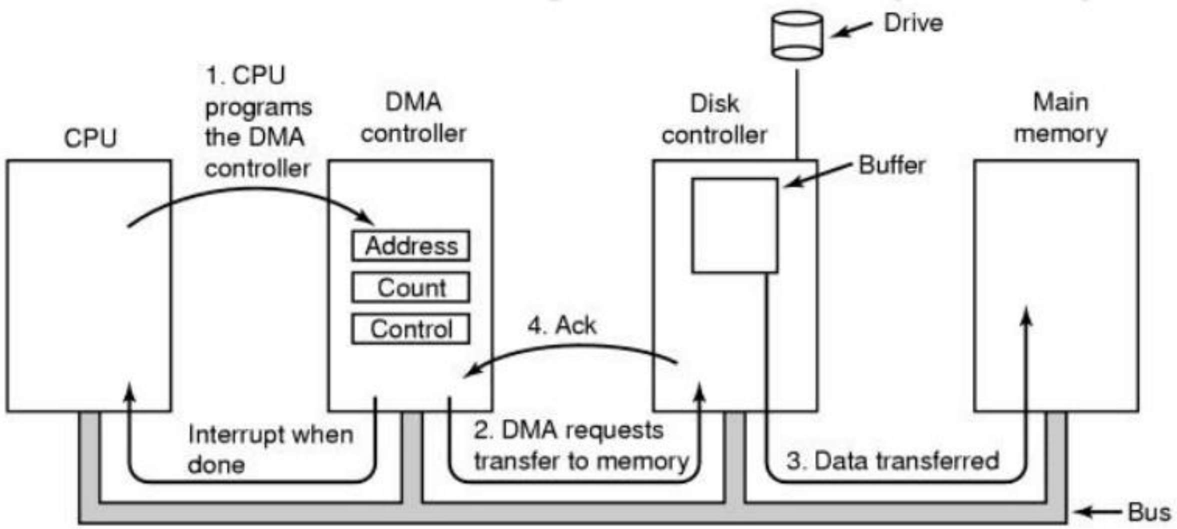
I/O Ports & Memory Mapped I/O

- ❑I/O approaches
  - ❑Separate I/O and memory space
    - ❑Special I/O commands (IN/OUT)
  - ❑Memory-mapped I/O
- ❑Known issues
  - ❑Convenience/efficiency when using a high-level language
  - ❑Protection mechanisms
  - ❑Special data access scheme: TEST
  - ❑Caching



Input/Output (I/O) Principles
Direct Memory Access (DMA)

- ❑Are the addresses the CPU sends to the DMA controller virtual or physical addresses?
- ❑Can the disk controller directly read data into the main memory (bypassing the controller buffer)?

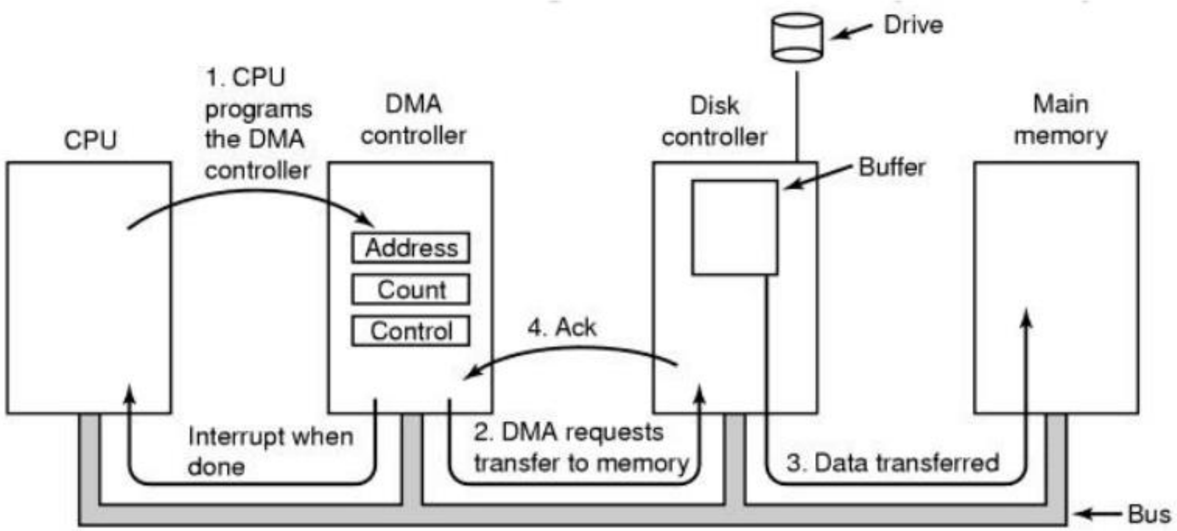


Input/Output (I/O) Principles
Direct Memory Access (DMA)

- A device driver using DMA has to talk to hardware connected to the interface bus, which uses physical addresses, whereas program code uses virtual addresses

- DMA-based hardware uses *bus*, rather than *physical*, addresses

- Linux kernel support: <asm/io.h>
  - `unsigned long virt_to_bus(volatile void *address);`
  - `void *bus_to_virt(unsigned long address);`



Input/Output (I/O) Principles

How to accomplish I/O?

- ❑Polling-based
  - ❑CPU spins and polls the I/O until finished
- ❑Periodic polling
  - ❑Continous polling consumes too much CPU
  - ❑Saves CPU overhead
  - ❑May not react immediately to hadware events
- ❑Interrupt-driven
  - ❑CPU initiates I/O then focuses on something else
  - ❑We get notifications when the I/O is done (interrupts)

Input/Output (I/O) Principles
Interrupt Handlers

- Save registers of old process
- Setup context for interrupt service procedure
  - Switch from user-space to kernel-space
  - MMU
  - Stack
  - …
- Run service procedure
  - When safe, re-enable interrupts
- Run scheduler to choose new process to run next
- Setup context (MMU, registers) for process to run next
- Start running the new process
- How costly is this?
  - Gigabit Ethernet: each packet arrives every 12us

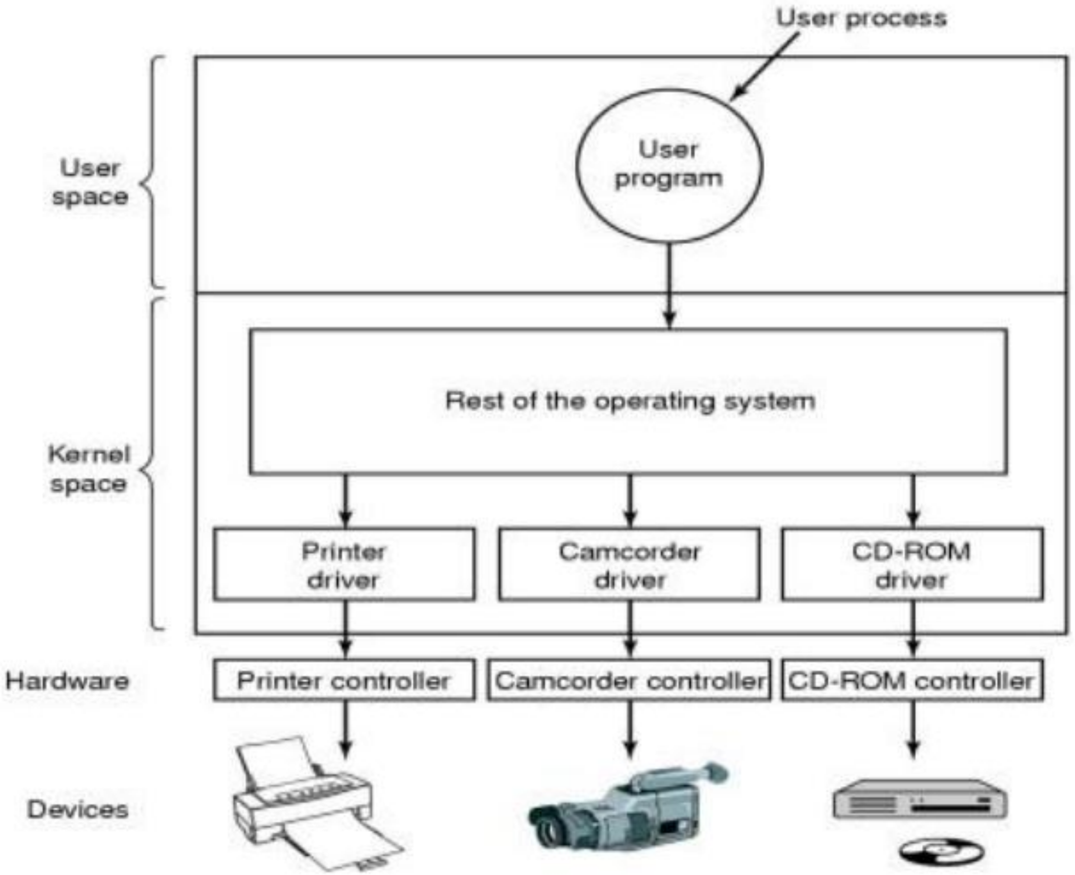
Input/Output (I/O) Principles

Interrupt Vectors

- A **non-maskable interrupt** (**NMI**) is an interrupt that cannot be ignored by standard interrupt masking techniques in the system
  - It is typically used to signal attention for non-recoverable hardware errors
- Intel Pentium
  - 0: divide by zero
  - 6: invalid opcode
  - 11: segment not present
  - 12:stack fault
  - 14: page fault
  - …31: non-maskable
  - 32-255: maskable interrupts

Input/Output (I/O) Principles
I/O Software Layers

- ❑Device-dependent OS I/O software
  - ❑Directly interacts with controller hardware
- ❑Interface to upper-layer OS code is standardized

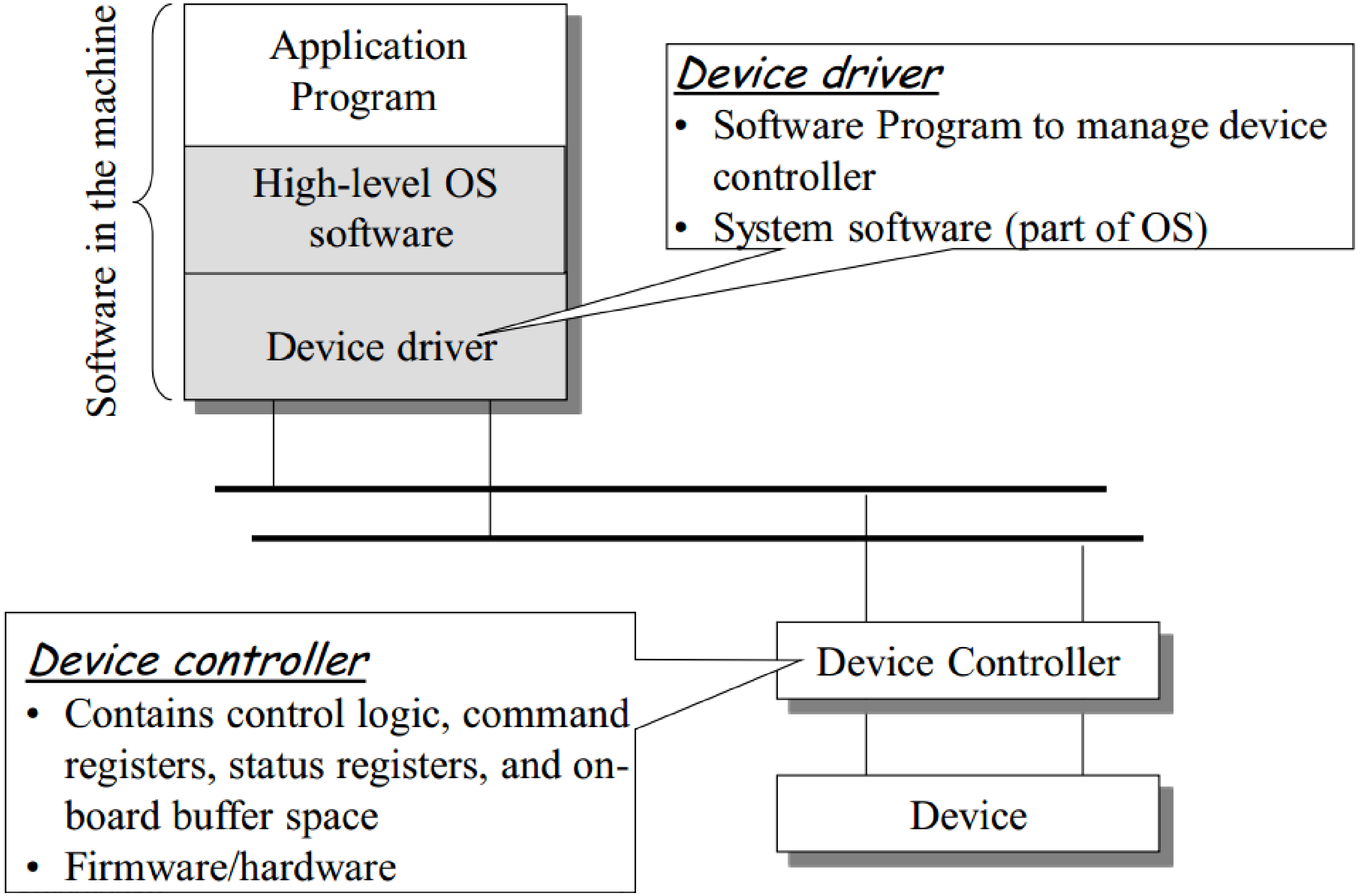


Input/Output (I/O) Principles
Device Driver Reliability

- A device drive is a device-specific part of the kernel-space I/O software
  - Includes interrupt handlers
  - Must run in kernel mode
    - Crashing often brings down the entire system
  - The buggiest part of an OS
- How to make the system more stable by isolating faults in device drivers?
  - Run most of the device driver code at user level
  - Restrict and limit device driver operations in the kernel

Input/Output (I/O) Principles
High-Level I/O Software

- Device independence
  - Reuse software as much as possible across different types of devices
- Buffering
  - Data coming off a device is stored in an intermediate buffer
  - Access speed/granularity matching with I/O devices
- Caching
- Speculative I/O

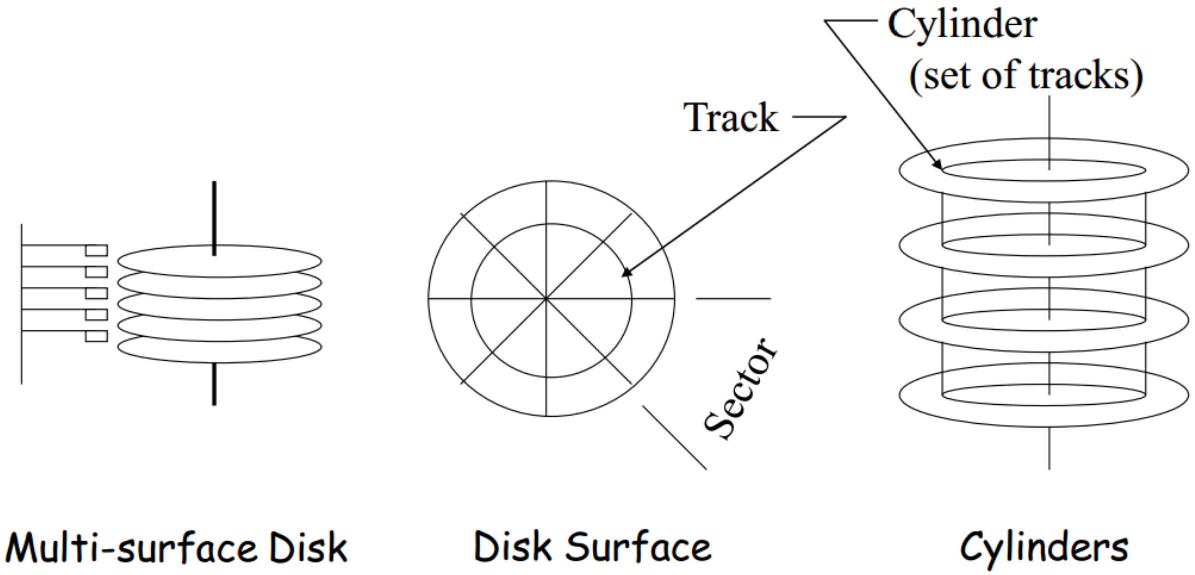


Device driver
- Software Program to manage device controller
- System software (part of OS)

Device controller
- Contains control logic, command registers, status registers, and on-board buffer space
- Firmware/hardware

Input/Output (I/O) Principles
Disk Drives – Mechanical Components

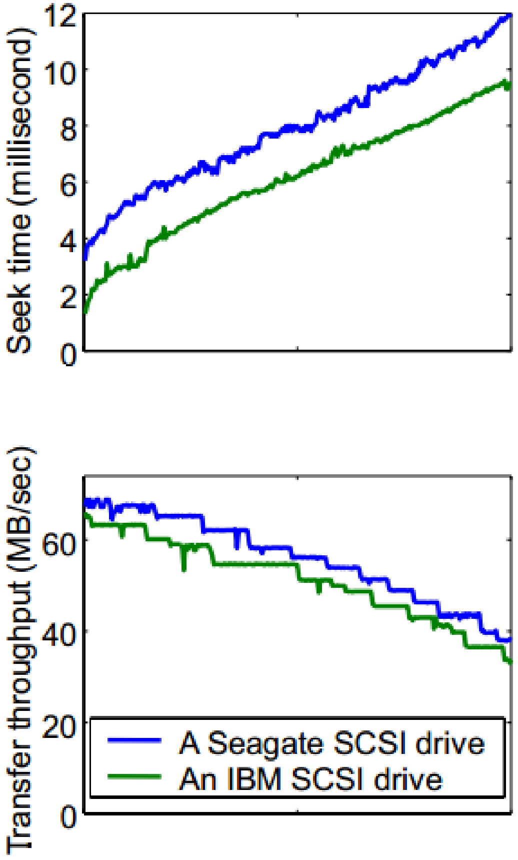
- ❑Disk drives are addressed as large 1-dimensional arrays of *logical blocks*, where the logical block is the smallest unit of transfer
- ❑The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially
    - ❑First sector: sector 0 (first track on the outermost cylinder)
    - ❑Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, then through the rest of cylinders (from outermost to innermost)



Input/Output (I/O) Principles

Disk Performance Characteristics

- Three major components
  - Seek – moving the heads to the cylinder containing the desired sector
  - Rotation – rotating the desired sector to the disk head
  - Transfer – sequentially moving data to or from disk



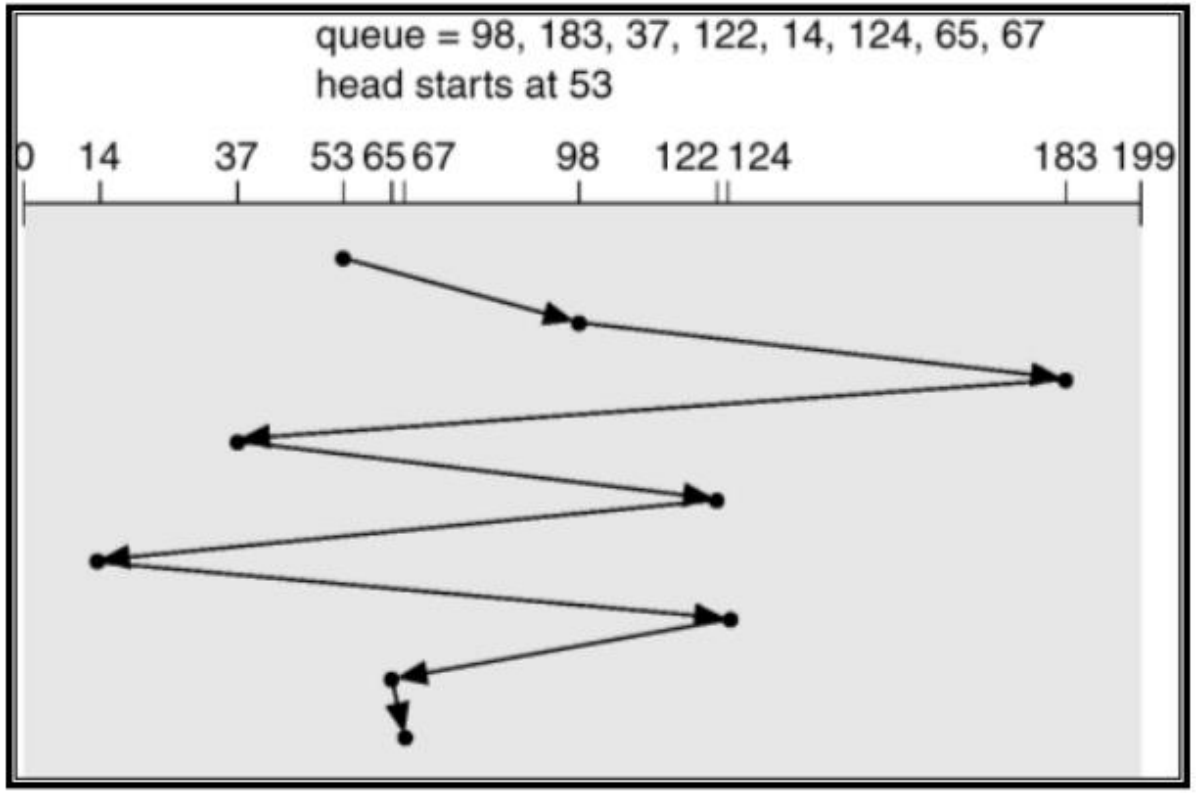
Input/Output (I/O) Principles

Disk Scheduling

- Choose from outstanding disk requests when the disk is ready for a new request
  - Can be done in both disk controller and the operating system
  - Disk scheduling is non-preemptible
- Goals of disk scheduling
  - Overall efficiency
    - Small resource consumption for competing I/O disk workload
  - Fairness
    - Prevent starvation

Input/Output (I/O) Principles
Disk Scheduling - FCFS

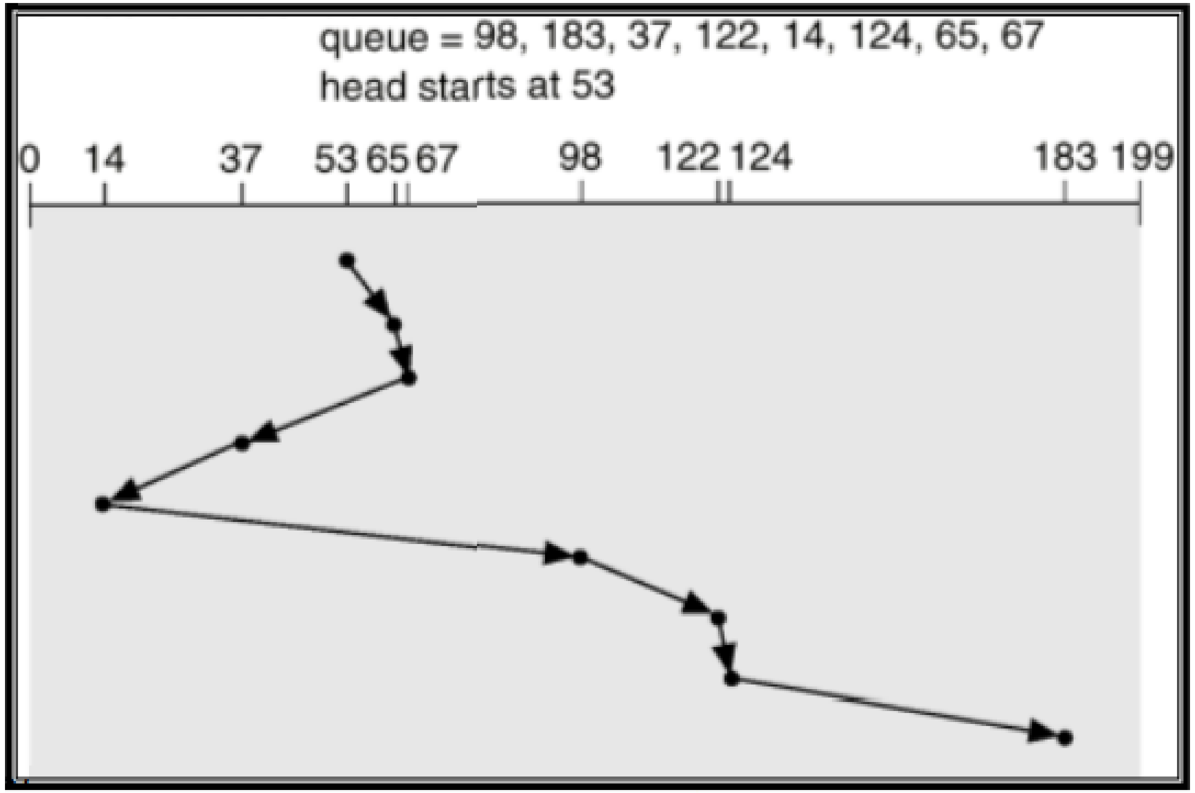
- ❑First Come First Serve
  - ❑Total head movement: 640
  - ❑Starvation?



Input/Output (I/O) Principles

Disk Scheduling - SSTF

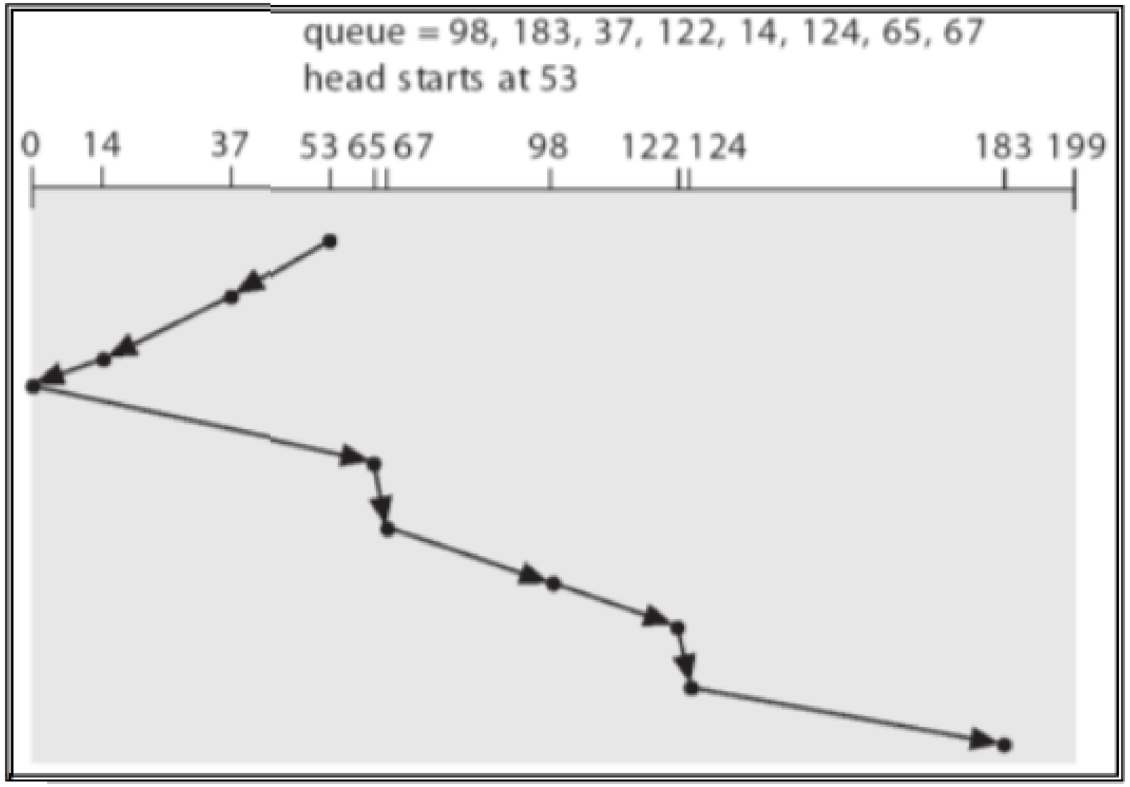
- ❑ Shortest Seek Time First
  - ❑ Select request with minimum seek time from the current head position
  - ❑ SSTF scheduling is a form of SJF scheduling
  - ❑ Total head movement: 236
- ❑ Starvation?



Input/Output (I/O) Principles

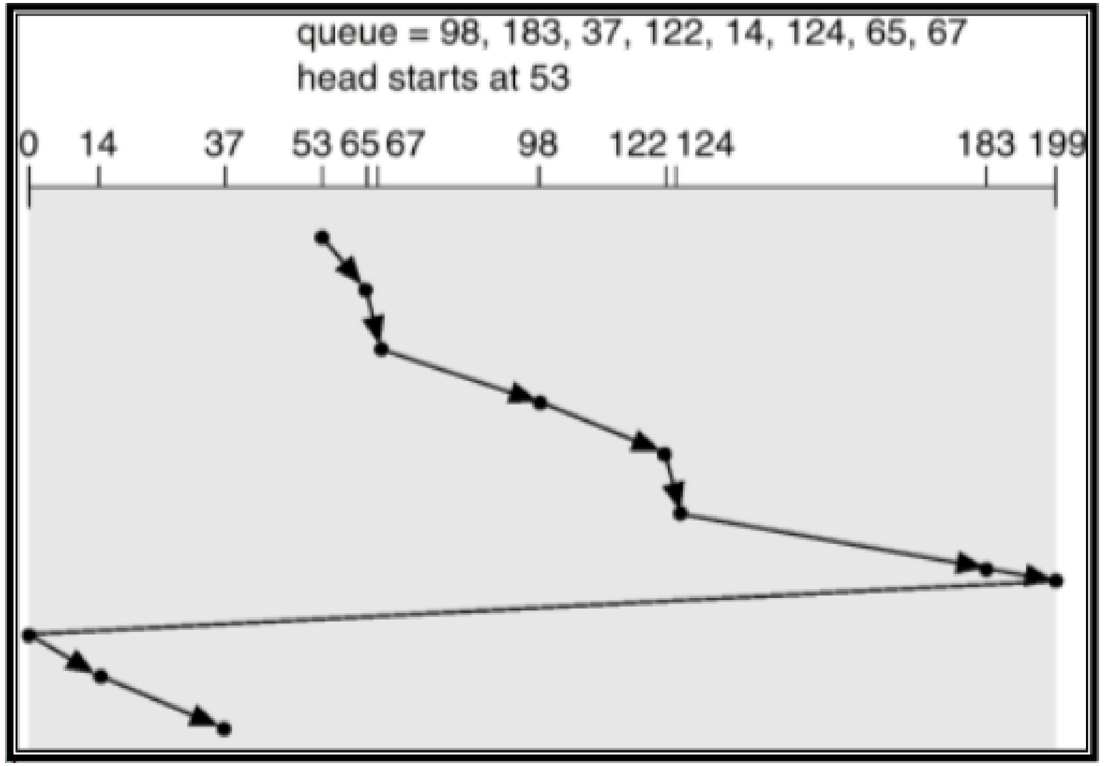
Disk Scheduling - SCAN

- ❑ Disk arm starts at one end of the disk, moves toward the other end, servicing requests until it gets to the other end, where the head movement is reversed and servicing continues
- ❑ *Elevator* algorithm
- ❑ Total head movement: 208
- ❑ Starvation?



Input/Output (I/O) Principles
Disk Scheduling – C-SCAN

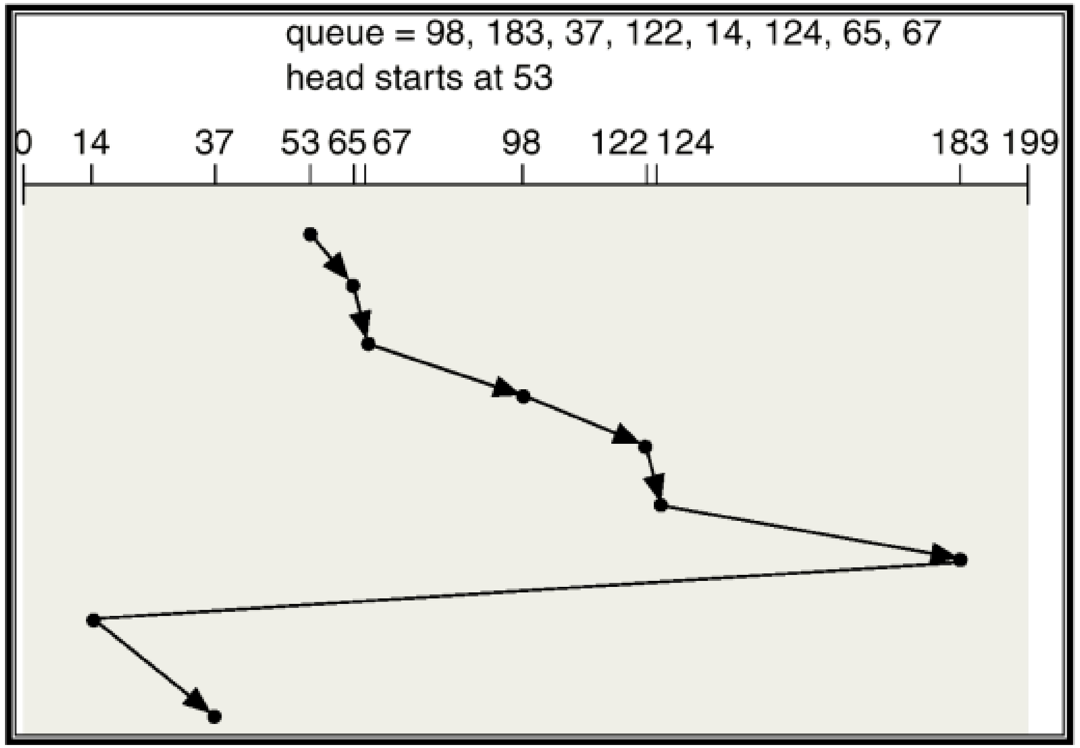
- ❑ Circular SCAN
  - ❑ Provides a more uniform wait time than SCAN
  - ❑ Head moves from one end of the disk to the other
  - ❑ Servicing requests as it goes
  - ❑ When it reaches the other end, immediately returns to the beginning of the disk, without servicing any requests on the return trip
- ❑ Starvation?



Input/Output (I/O) Principles

Disk Scheduling – C-LOOK

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



Input/Output (I/O) Principles

Deadline Scheduling In Linux

❑Regular elevator-style scheduling similar to C-LOOK

❑Additionally, all I/O requests are put into the FIFO queue with an expiration time (e.g. 500ms)

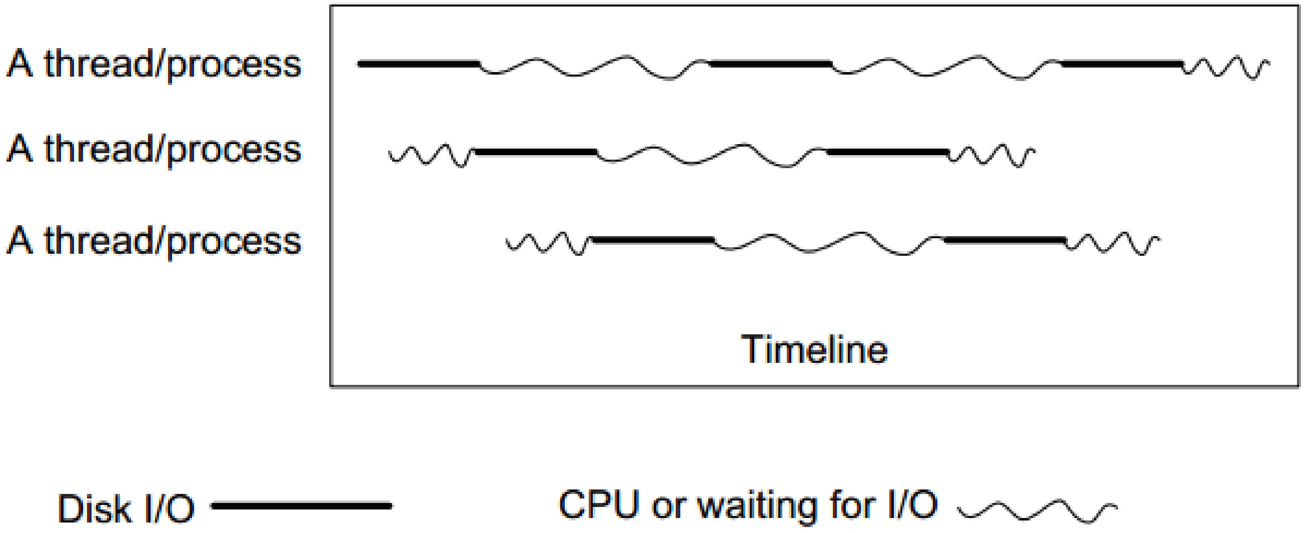
❑When the head request in the FIFO queue expires, it will be executed next (even if it is not next in line according to C-LOOK)

❑A mix of performance and fairness

Input/Output (I/O) Principles

Concurrent I/O

- Consider two request handlers in a Web server
  - Each accesses a different stream of sequential data (file) on disk
  - Each reads a chunk (the buffer size) at a time, does a little CPU processing, and reads the next chunk
- What happens?



Input/Output (I/O) Principles

Concurrent I/O - Implementations

- Aggressive prefetching
- Anticipatory scheduling
  - At the completion of an I/O request, disk scheduler waits a bit (despite the fact that there is other work to do), in anticipation that a new request with strong locality will be issued
  - Schedule another request if no such new request appears before timeout
  - Included in Linux kernel 2.6

Input/Output (I/O) Principles
Concurrent I/O – Exploiting Concurrency

- ❑RAID – Redundant Array of Inexpensive Disks
  - ❑RAID 0: data stripping at block level, no redundancy
  - ❑RAID 1: mirrored disks (100% overhead)
  - ❑RAID 2: bit-level stripping with parity bits, synchronized writes
  - ❑RAID 3: data stripping at the bit level with parity disk, synchronized writes
  - ❑RAID 4: data stripping at block level with parity disk
  - ❑RAID 5: scattered parity
  - ❑RAID 6: handles multiple disk failures

Input/Output (I/O) Principles
Disk Management

- ❑Formatting
  - ❑Header: sector number, etc.
  - ❑Footer/tail: ECC codes
  - ❑Gap
  - ❑Initialize mapping from logical block number to defect-free sectors
- ❑Logical disk partitioning
  - ❑One or more groups of cylinders
  - ❑Sector 0: master boot record loaded by BIOS firmware, which contains partition information
  - ❑Boot record points to boot partition

Input/Output (I/O) Principles

Swap File Management

- Part of the file system?
  - Requires navigating directory structure
  - Disk allocation data structures
- Separate disk partition
  - No file system or directory structure
  - Optimize for speed rather than storage efficiency
  - When do we create swap space?