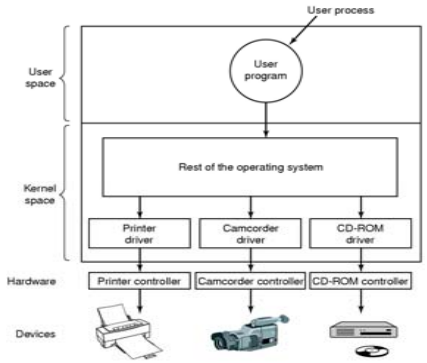
Storage Systems

CS 256/456

Dept. of Computer Science, University of Rochester

I/O Software Layers



- Device driver: device-dependent OS I/O software; directly interacts with controller hardware
- Interface to upper-layer OS code is standardized.

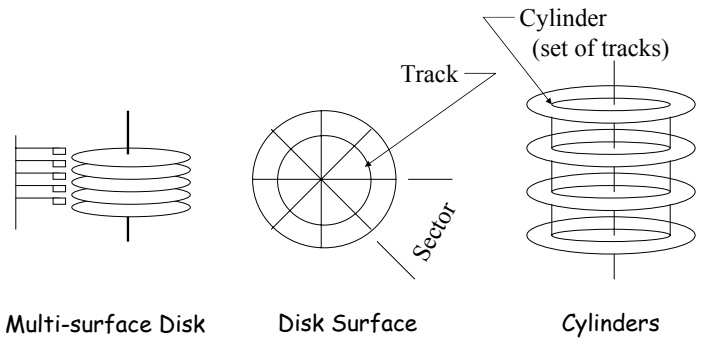
Device Drivers

- Device driver is the device-specific part of the kernel-space I/O software. It also includes interrupt handlers.
- Device drivers must run in kernel mode. Why?
- ⇒ The crash of a device driver brings down the whole system.
- Device drivers are probably the buggiest part of the OS. Why?
- How to make the system more reliable by isolating the faults of device drivers?
  - Run most of the device driver code at user level.
  - Restrict and limit device driver operations in the kernel.

Upper-level I/O Software

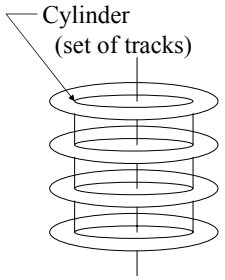
- Device independence
  - reuse software as much as possible across different types of devices
- Buffering
  - data coming off a device is stored in intermediate buffer
- Need for buffering
  - access speed/granularity matching with I/O devices
  - caching
  - speculative I/O

Disk Drive – Mechanical Parts



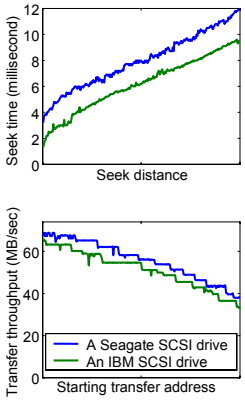
Disk Structure

- 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 Characteristics

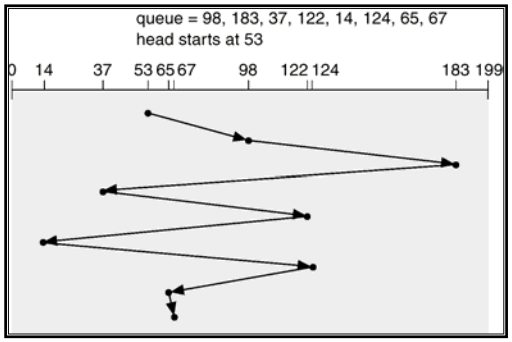
- A disk operation has three major components
  - *Seek* – moving the heads to the cylinder containing the desired sector
  - *Rotation* – rotating the desired sector to the disk head
  - *Transfer* – sequentially moving data to or from disk



Disk Scheduling

- 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 non-preemptible
- Goals of disk scheduling
  - overall efficiency – small resource consumption for completing disk I/O workload
  - fairness – prevent starvation

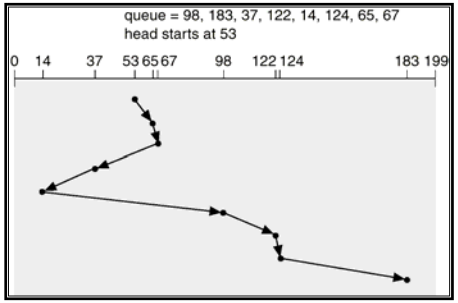
FCFS (First-Come-First-Serve)



- Illustration shows the total head movement is 640.
- Starvation?

SSTF (Shortest-Seek-Time-First)

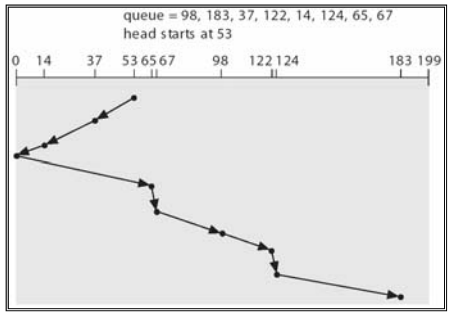
- Selects the request with the minimum seek time from the current head position.
- SSTF scheduling is a form of SJF scheduling.
- Illustration shows the total head movement is 236.



Starvation?

SCAN

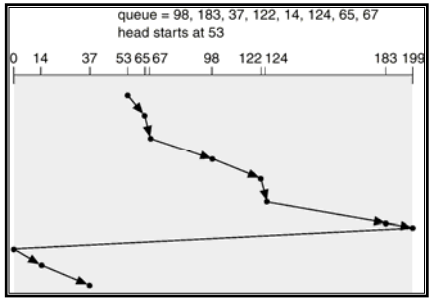
- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end, where the head movement is reversed and servicing continues.
- Sometimes called the *elevator algorithm*.
- Illustration shows the total head movement is 208.



Starvation?

C-SCAN (Circular-SCAN)

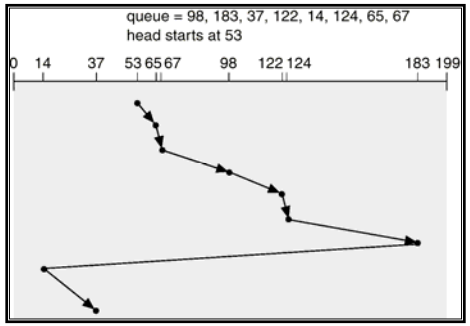
- Provides a more uniform wait time than SCAN.
- The head moves from one end of the disk to the other. servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.



Starvation?

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.



Deadline Scheduling in Linux

- A regular elevator-style scheduler similar to C-LOOK
- Additionally, all I/O requests are put into a 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.

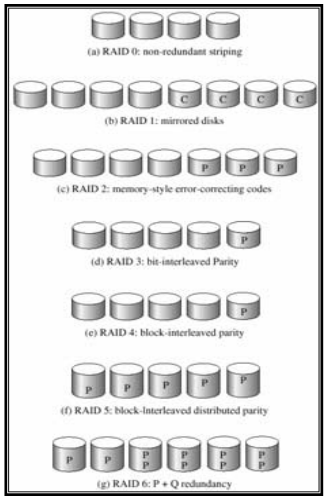
Concurrent I/O

- Consider two request handlers in a Web server
  - each accesses a different stream of sequential data (a 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?
- How to deal with it?
- Anticipatory scheduling [Iyer & Druschel, SOSP 2001]
  - at the completion of an I/O request, the disk scheduler will wait a bit (despite there is other work to do), in anticipation that a new request with strong locality will be issued. go ahead to schedule another request if no such new request appears before timeout.
  - default I/O scheduler in Linux 2.6

RAID

RAID – redundant array of inexpensive disks.

- improve performance through stripping; parallel I/O.
- improve reliability via redundancy.



Disclaimer

- Parts of the lecture slides contain original work of Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Andrew S. Tanenbaum, and Gary Nutt. The slides are intended for the sole purpose of instruction of operating systems at the University of Rochester. All copyrighted materials belong to their original owner(s).