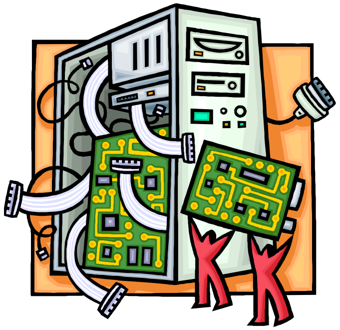
I/O = Input/Output Devices



How to converse with devices

- Polling
  - CPU issues I/O command
  - CPU directly writes instructions into device's registers
  - CPU busy waits for completion
- Interrupt-driven I/O
  - CPU issues I/O command
  - CPU directly writes instructions into device's registers
  - CPU continues operation until interrupt
- This time: Direct Memory Access (DMA)
  - CPU asks DMA controller to perform device-to-memory transfer
  - DMA issues I/O command and transfers new item into memory
  - CPU module is interrupted after completion

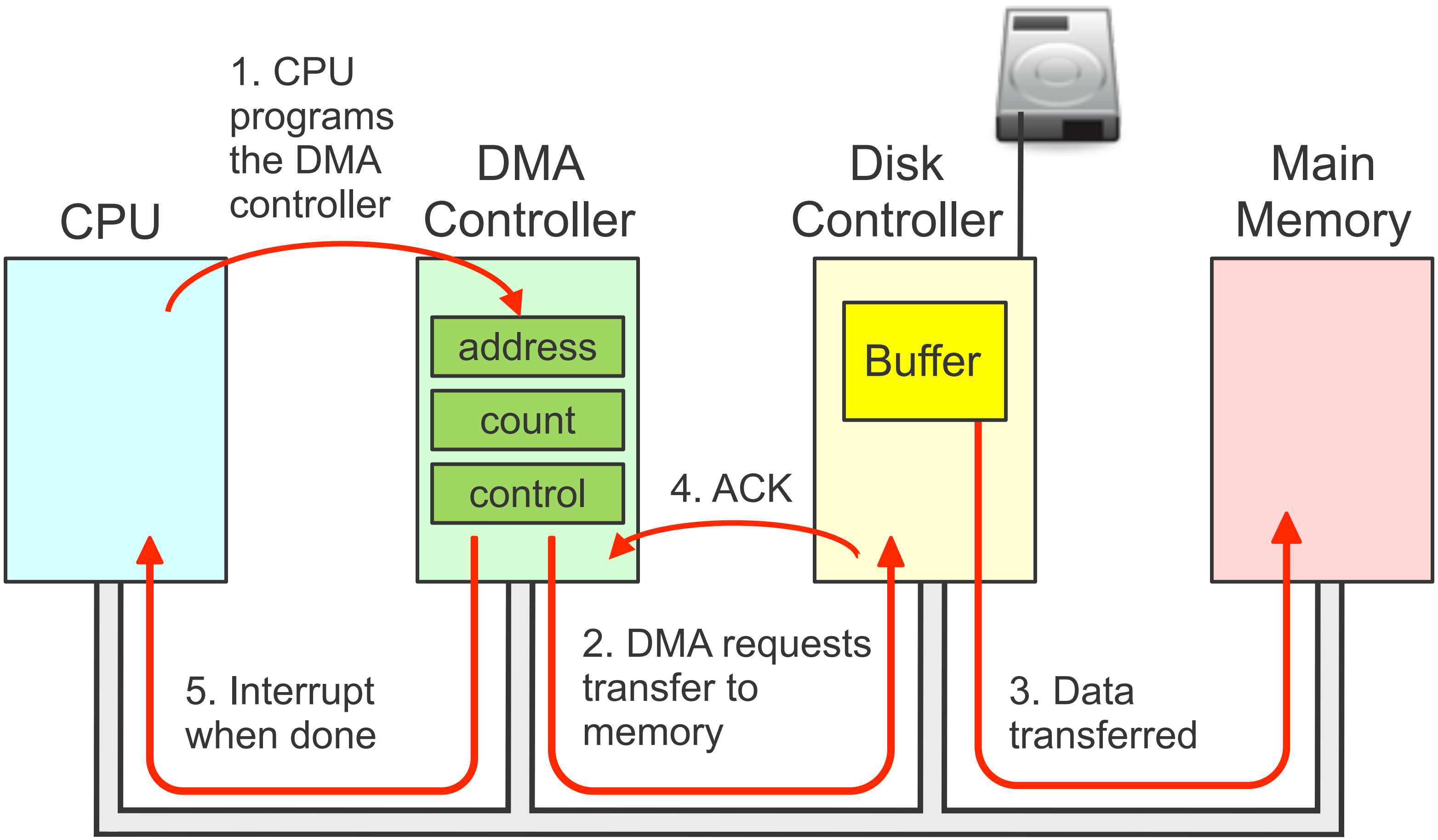
Direct Memory Access (DMA)

- Means what it says!
- Involves special hardware element: DMA controller
- Assists in direct exchange of data between main memory and I/O controller
- More efficient than CPU requesting data from I/O controller byte by byte, e.g., for block devices like disks

The DMA-CPU Protocol

1. Device driver (on CPU) programs DMA controller
   - Sets registers to specify source/destination addresses, byte count and control information (e.g., read/write). Then continues with other work.
2. DMA controller tells I/O controller to directly move data to memory via bus, without involving the CPU
3. Disk controller transfers data to main memory
4. Disk controller ACKs transfer to DMA controller
5. DMA controller sends interrupt back to CPU

Direct Memory Access (DMA)



DMA

- Driver operation to input sequence of chars

```
write_reg(mm_buf, m);
write_reg(count, n);
write_reg(opcode, read);
block to wait for interrupt;
```

- Writing opcode triggers DMA controller
- DMA controller issues interrupt after n chars in memory

What's the catch?

- Handshaking between DMA controller and the device controller
- Can the CPU execute as normal during transfer?
  - Bus is shared by DMA controller and CPU
  - DMA controller takes away CPU cycles when it uses bus, hence blocks CPU from accessing memory
  - Not an interrupt: CPU does not switch context
  - Causes the CPU to execute more slowly: "cycle stealing"
- But in general DMA controller improves the total system performance

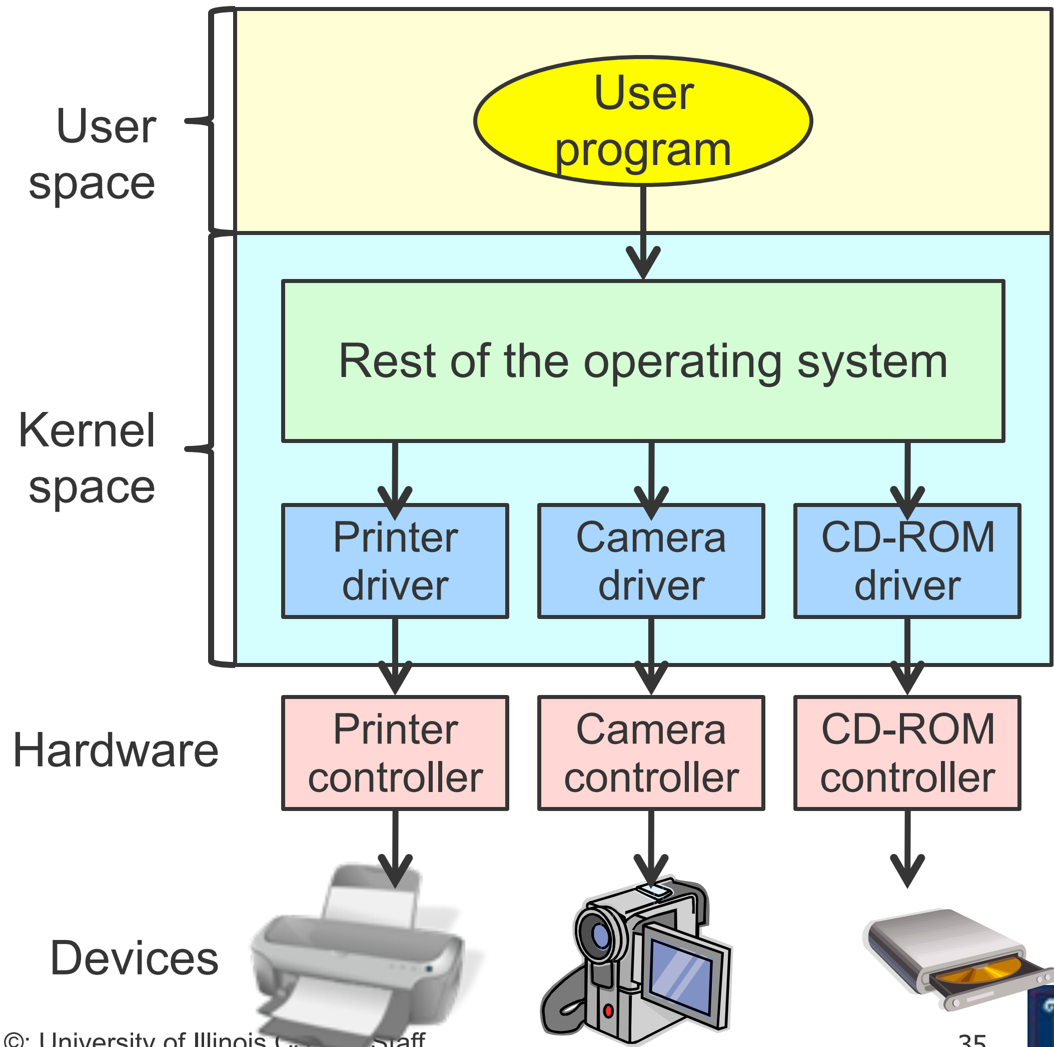
Discussion

- Tradeoffs between
  - Polling I/O
  - Interrupt-driven I/O
  - I/O using DMA

- Which is fastest for a single I/O request that takes a very short time?
- Which is fastest for a single I/O request that takes a very long time?
- Which one gives the highest throughput?

Device Drivers

- Logical position of device drivers
- Communications between device driver and device controllers goes over the bus



Device Drivers

- Device-specific code to control an IO device
  - Typically written by device's manufacturer
  - Controller has some device registers used to give it commands.
  - Number of device registers and the nature of commands vary from device to device
    - Mouse driver accepts information from the mouse about how far it has moved
    - Disk driver has to know about sectors, tracks, heads, etc).

Device Drivers

- Typically part of the OS kernel
  - Compiled with the OS
  - Dynamically loaded into the OS during execution
  - But in microkernel, drivers can run in user space
- Each device driver handles
  - One device type (mouse, disk, etc.)
  - Or one class of closely related devices
    - SCSI disk driver to handle multiple disks of different sizes and different speeds
- Categories
  - Block devices
  - Character devices

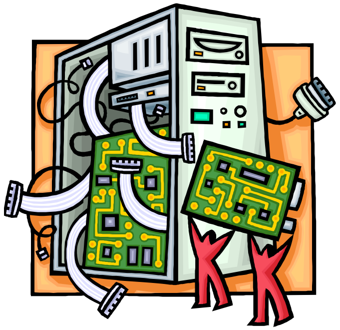
Functions of Device Drivers

- Initialize the device
- Accept abstract read and write requests from the device-independent layer above
- Manage power requirements and log events
- Check to ensure input parameters are valid
- Translate valid input from abstract to concrete terms
  - e.g., convert linear block number into the head, track, sector and cylinder number for disk access
- Check the device if it is in use (i.e., check the status bit)
- Control the device by issuing a sequence of commands. The driver determines what commands will be issued.

Error Reporting

- Programming I/O Errors
  - occur when a process asks for something impossible
  - e.g., write to an input device such as keyboard, or read from output device such as printer
  - Report back an error to the caller
- Actual I/O Errors
  - occur at the device level
  - e.g., read disk block that has been damaged, or try to read from video camera which is switched off
  - Report back to the device-independent software
- The device-independent I/O software detects these errors and responds to them by reporting to the process

Disks



What we'll cover: Bottom-up view

- The device: a disk
- Disk scheduling
- Filesystem structures
- User-level: using a filesystem

A disk

http://www.youtube.com/watch?v=9eMWG3fwiEU

Disk Examples (Summarized Specs)

	Seagate Barracuda	IBM Ultrastar 72ZX
Capacity, Interface & Configuration		
Formatted Gbytes	28	73.4
Interface	Ultra ATA/66	Ultra160 SCSI
Platters / Heads	4 / 8	11/22
Bytes per sector	512	512-528
Performance		
Max Internal transfer rate (Mbytes/sec)	40	53
Max external transfer rate (Mbytes/sec)	66.6	160
Avg Transfer rate(Mbytes/sec)	> 15	22.1-37.4
Multisegmented cache (Kbytes)	512	16,384
Average seek, read/write (msec)	8	5.3
Average rotational latency (msec)	4.16	2.99
Spindle speed (RPM)	7,200	10,000

Detailed view of a disk

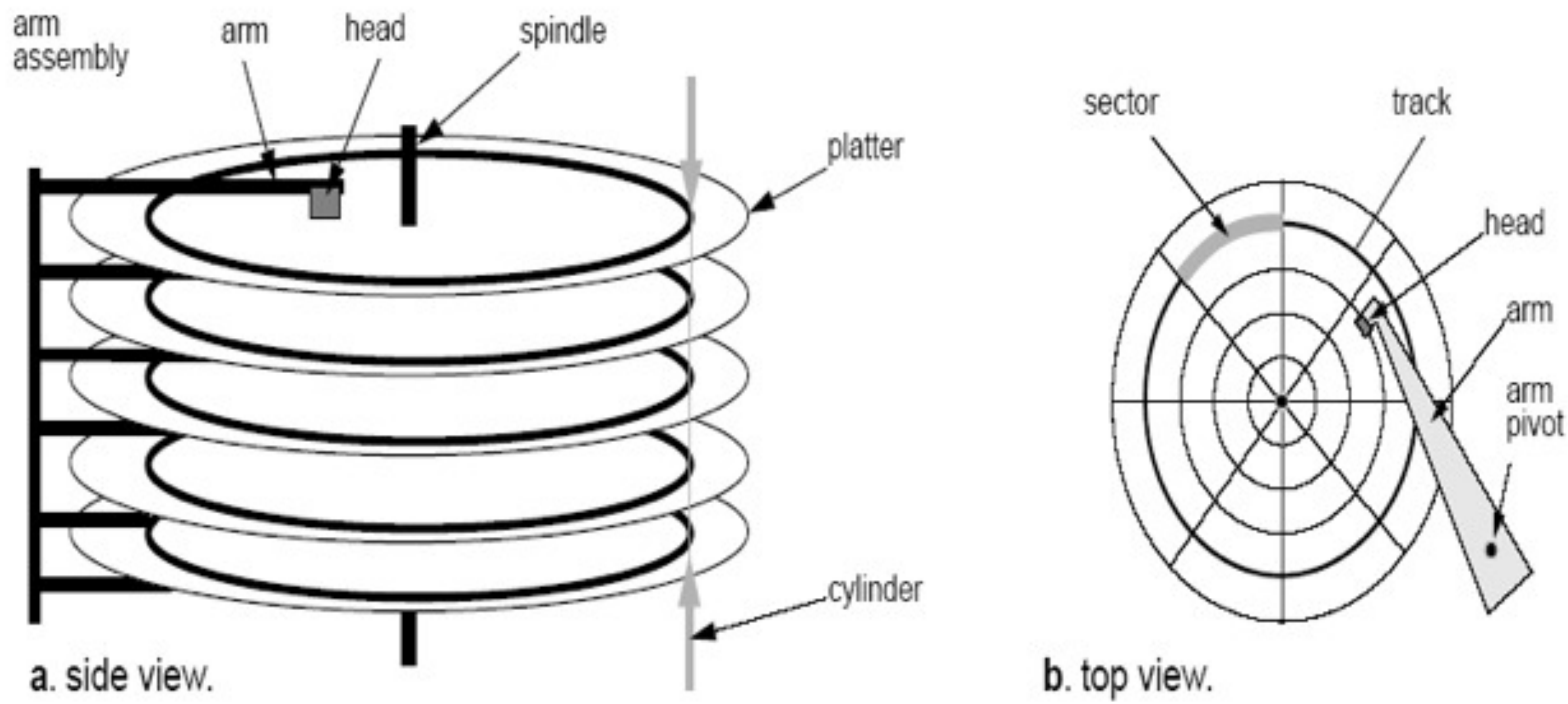
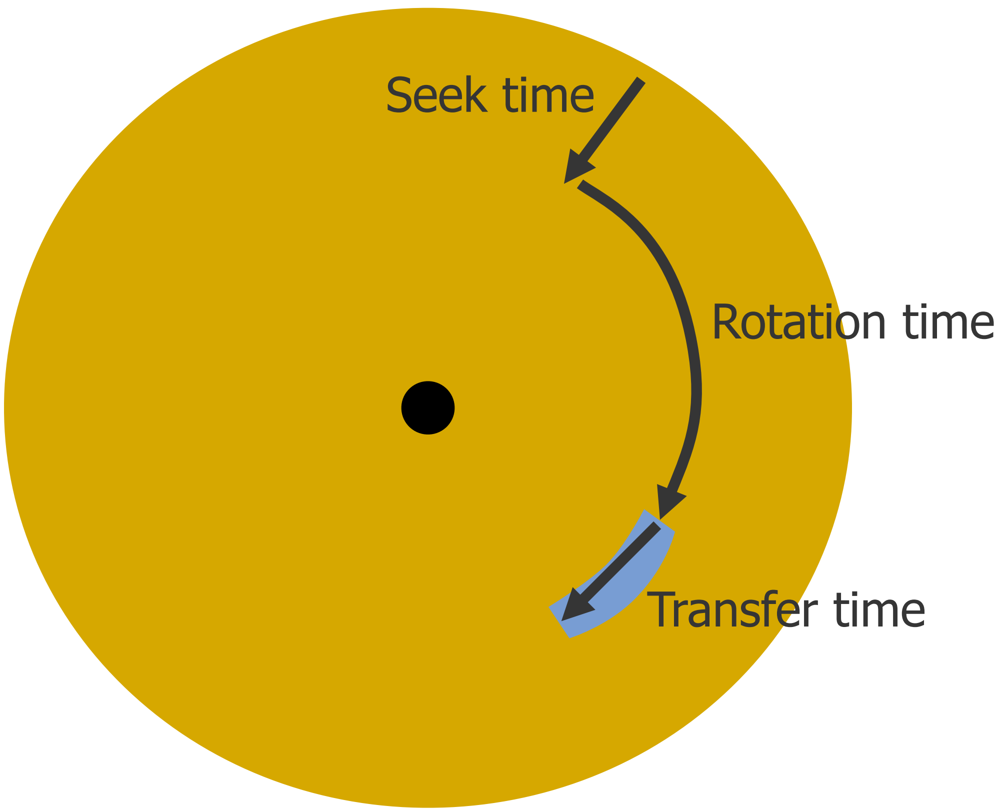


Figure 1: the mechanical components of a disk drive.

Disk Scheduling

Three steps in reading a sector:



Avg rotation time =
½ rotation time

Transfer time =
(sector size)/(transfer speed)

Disk Performance Factor: Seeking

- Seeking: position the head to the desired cylinder
  - Takes roughly 2-5ms
- Seeking speed depends on:
  - The power available for the pivot motor
    - halving the seek time requires quadrupling the power
  - The arm's stiffness
    - Accelerations of 30-40g are required to achieve good seek times, and too flexible an arm can twist and bring the head into contact with the platter surface.
- A seek is composed of
    - A speedup, a coast, a slowdown, a settle
    - For very short seeks, the settle time dominates (1-3ms)

Disk Performance: Other Factors

- Rotational delay
  - Wait for a sector to rotate underneath the heads
  - Typically 8.3 – 6.0ms (7,200 – 10,000RPM) or ½ rotation takes 4.15-3ms
- Transfer bytes
  - Average transfer bandwidth (15-37 MB/sec)
- Suppose: seek=5.3ms, rotational delay=6ms, Transfer speed = 25MBps
- What is the effective bandwidth for transferring sector of 1 Kbytes?
  - Seek (5.3 ms) + half rotational delay (3ms) + transfer (1KB/ 25MBps=0.04 ms)
  - Total time is 8.34ms. Effective BW = 1KB/8.34ms=120 KB/sec!
- What block size can get 90% of the disk transfer bandwidth?

Disk Behaviors

Block Size	% of Disk Transfer Bandwidth
1 KB	0.5%
8 KB	3.7%
256 KB	55%
1 MB	83%
2 MB	90%

- Seek time and rotational latency dominates the cost of small reads
- There are more sectors on outer tracks than inner tracks
  - Read outer tracks: 37.4MB/sec
  - Read inner tracks: 22MB/sec

So, how do you speed up disk transfer speed?

- Increase block size
- What else?

Disk Scheduling!

- Given a queue of waiting requests for disk accesses (from various processes)
- Which disk request is serviced first?
  - FCFS
  - Shortest seek time first
  - Elevator (SCAN)
  - C-SCAN (Circular SCAN)
- Implemented inside device driver

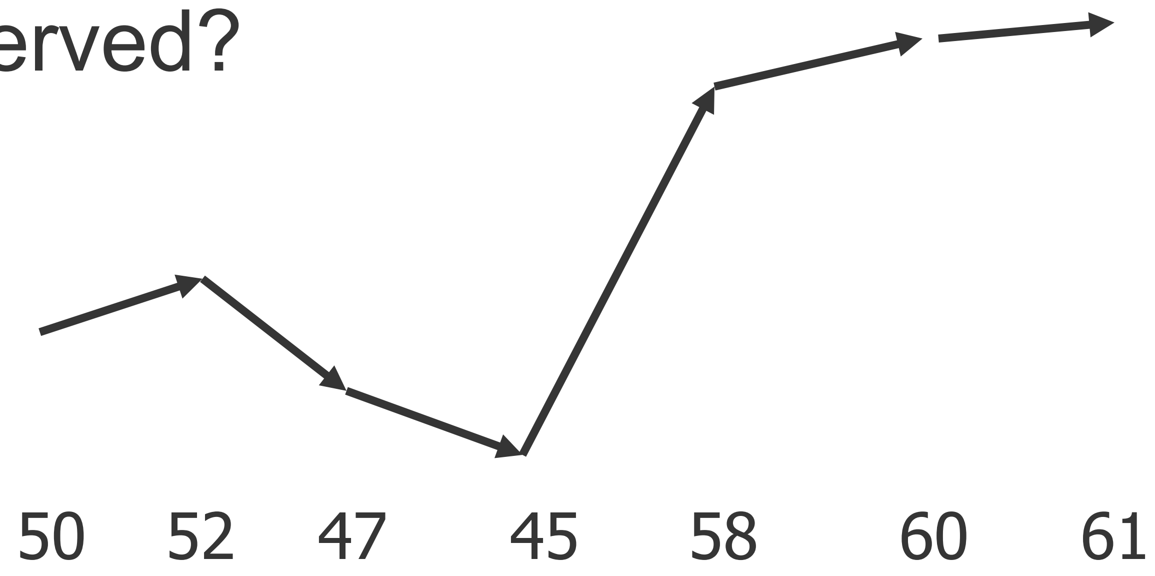
Disk Scheduling - FIFO

FIFO (FCFS) order

- Method
  - Queue of IO requests held by device driver
  - Dispatched in First come first serve order
- Pros
  - Fairness among requests
  - In the order applications expect
- Cons
  - Arrival may be on random spots on the disk (long seeks)
  - Wild swings can happen
- Analogy: What would FCFS elevator scheduling look like?

Scheduling – Shortest Seek Time First (SSTF)

- Example: Start from track 49, serve requests at tracks 45, 47, 50, 52, 58, 60, and 61. In what order are they served?

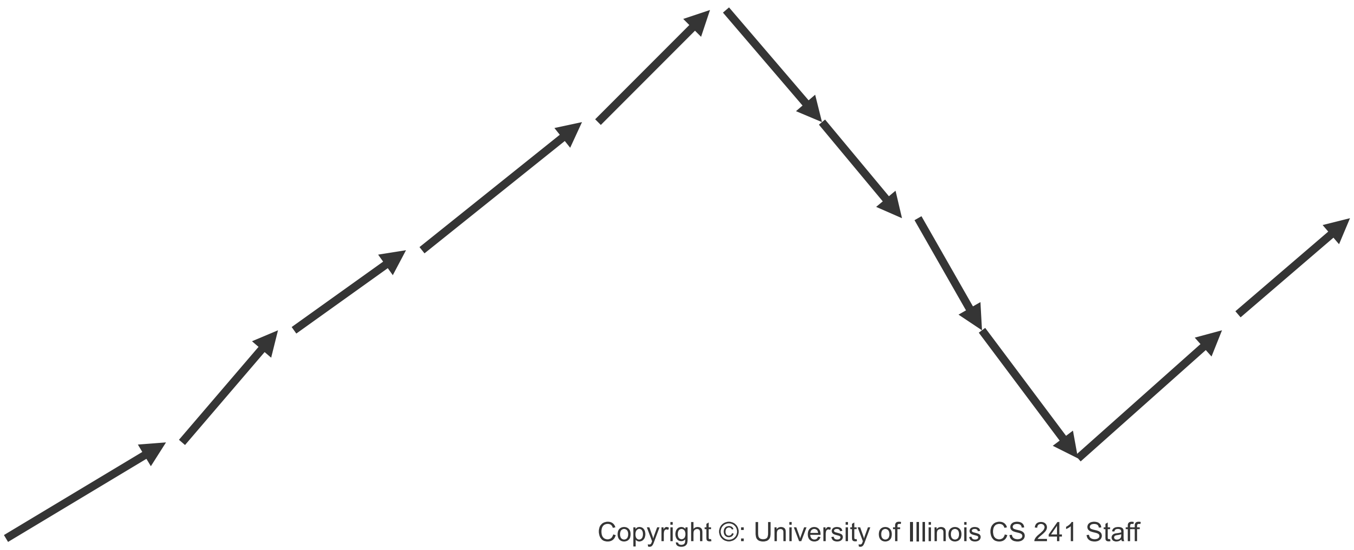


SSTF (Shortest Seek Time First)

- Method
  - Pick the request closest on disk to current position of head
- Pros
  - Tries to minimize seek time
- Cons
  - Starvation – why?
- Question
  - Is SSTF optimal?
  - Can we avoid starvation?

Scheduling - Scan

- Move head from one side to other side, serving all requests on the way. Then, go reverse way doing the same.
- Requests that arrived too late while scanning in one direction will be served in reverse direction

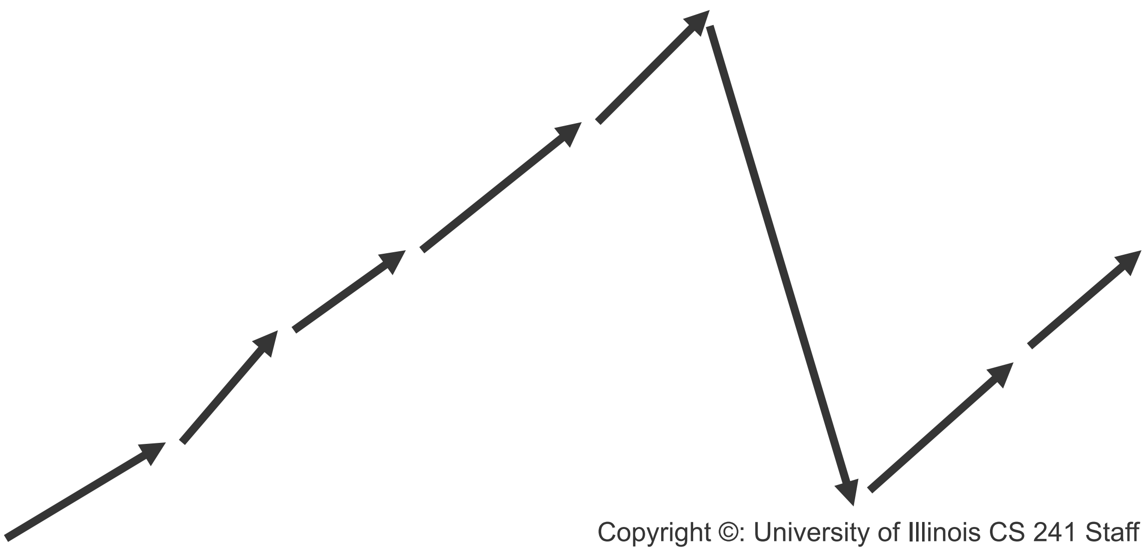


Elevator (SCAN)

- Method
  - Take the closest request in the direction of travel
  - Reverse direction at end of disk
- Pros
  - Bounded time for each request
- Cons
  - Request at the other end may take a while to get to
  - Requests near the center tend to have lower delay
    - Why?

Scheduling – Circular Scan (C-SCAN)

- Move head from side to side serving all requests in one direction only
- Requests that arrived too late for one scan will be served on the next



C-SCAN

- Method
  - Like SCAN
  - But wrap around
- Pros
  - Uniform service time
- Cons
  - Do nothing on the return