

## Contents

---

- Overview
- I/O Data Transfer
- Memory-Mapped I/O
- Device Driver
  
- Summary

## How to Perform I/O Operations?

---

- Programmed I/O
- Interrupt-driven I/O
- I/O using DMA

## Programmed I/O

---

- Every I/O operation is programmed and controlled.
- Example:
  - printing a file from user program to the printer means that data is first copied to the kernel, then the OS enters a tight loop outputting the characters one at a time.
- Essential aspect
  - the CPU continuously polls the device to see if it is ready to accept another character. It uses **polling** (or busy waiting).
- Cons & Pros?
  - Pros: simple
  - Cons: required CPU full time until all the I/O is done.

## Interrupt-Driven I/O

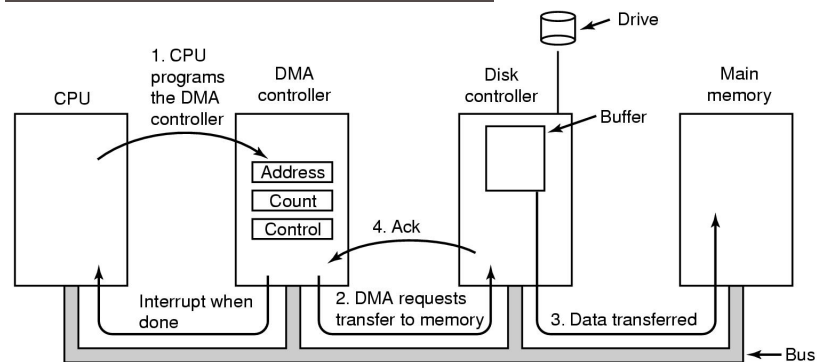
---

- CPU hardware has the interrupt report line that the CPU senses after executing every instruction.
  - Device raises an interrupt
  - CPU catches the interrupt and saves the state (e.g., instruction pointer)
  - CPU dispatches the interrupt handler
  - Interrupt handler determines the cause of the interrupt, services the device and clears the interrupt.

## Direct Memory Access (DMA)

- Use a special purpose processor, called a DMA controller
- Host writes a DMA command block into memory
  - pointer to source, pointer to destination, count of bytes to be transferred
- CPU writes the address of this command block to the DMA controller and goes on with other work
- DMA controller proceeds to operate the memory bus directly without help of main CPU
- Cons & Pros?

## Direct Memory Access (DMA)



DMA controller feeds the characters to the printer one at the time, without CPU being bothered. DMA is actually the programmed IO, only with DMA controller doing the work.

## Tradeoffs

### □ Programmed I/O

- Pro: require no interrupt or DMA support
- Con: waste CPU time

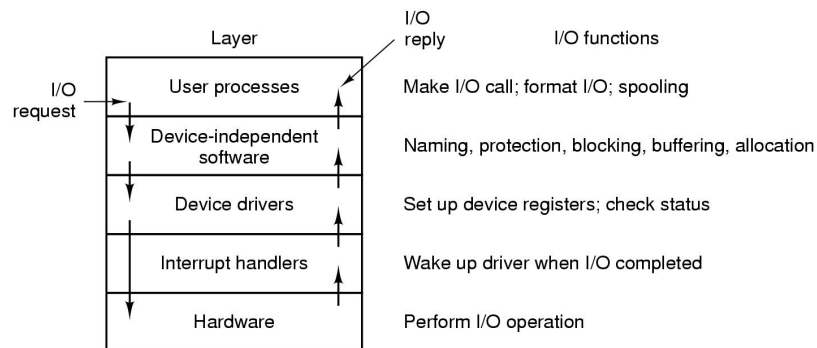
### □ Interrupt-drive IO:

- Pro: save CPU time for busy polling
- Con: triggering interrupt takes time, too.

### □ I/O using DMA:

- Pro: relieve CPU from I/O operation
- Con: DMA is slower than CPU

## I/O Software



Layers of the I/O system and the main functions of each layer

## Device Drivers

---

- Device-specific code to control an IO device, is usually written by device's manufacturer
  - Each controller has some device registers used to give it commands. The number of device registers and the nature of commands vary from device to device (e.g., mouse driver accepts information from the mouse how far it has moved, disk driver has to know about sectors, tracks, heads, etc).
- A device driver is usually part of the OS kernel
  - Compiled with the OS
  - Dynamically loaded into the OS during execution
- Each device driver handles
  - one device type (e.g., mouse)
  - one class of closely related devices (e.g., SCSI disk driver to handle multiple disks of different sizes and different speeds.).
- Categories:
  - Block devices
  - Character devices

## Functions in Device Drivers

---

- Accept abstract read and write requests from the device-independent layer above;
- Initialize the device;
- Manage power requirements and log events
- Check whether 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
- Control the device by issuing a sequence of commands. The driver determines what commands will be issued.

## Persistent Storage Devices

---

- Hard disk (Magnetic disks)
  - IDE disks (ATA disks): cheaper, lower performance
  - SCSI disks: high-performance, expensive
- Floppy disk
- Optical disk: DVD, CD
- Flash Memory/USB memory

## Disk Drive

---



Form factor:  
.5-1" × 4" × 5.7"  
Storage:  
120-500GB

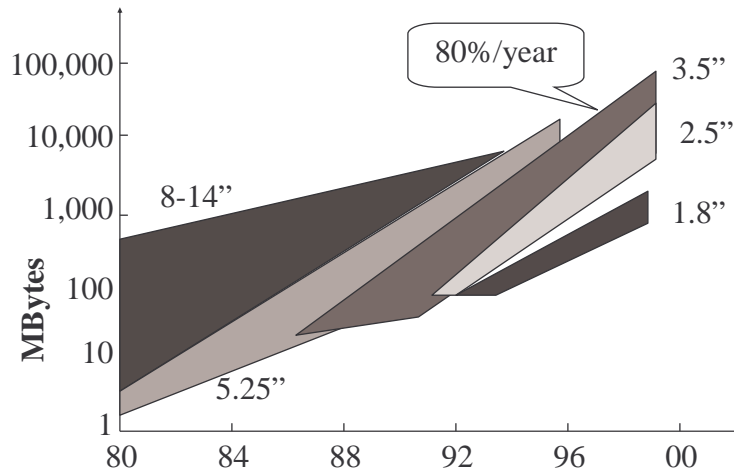


Form factor:  
.4-.7" × 2.7" × 3.9"  
Storage:  
20-120GB



Form factor:  
.2-.4" × 2.1" × 3.4"  
Storage:  
512MB-4GB

## Magnetic Disk Capacity



CSE660: Introduction to Operating Systems

13

## Disk Technology Trends

- Disks are getting smaller for similar capacity
  - Spin faster, less rotational delay, higher bandwidth
  - Less distance for head to travel (faster seeks)
  - Lighter weight (for portables)
- Disk data is getting denser
  - More bits/square inch
  - Tracks are closer together
  - Doubles density every 18 months
- Disks are getting cheaper (\$/GB)
  - Factor of ~2 per year since 1991
  - Head close to surface

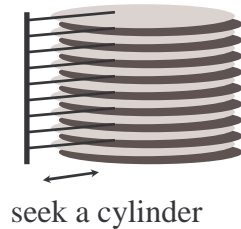
CSE660: Introduction to Operating Systems

14

## Disk Internal

---

- Platter
  - 1-12 platters /disk
- Cylinder
  - Certain track of the platters
  - 3.5-inch disk has about 2,000 cylinders.
- Disk arm
  - Seek the right cylinder



## Disk Performance Factor: Seeking

---

- Seeking: position the head to the desired cylinder
  - Roughly takes 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 Examples (Summarized Specs)

	Seagate Barracuda	IBM Ultrastar 72ZX
<b>Capacity, Interface &amp; Configuration</b>		
Formatted Gbytes	28	73.4
Interface	Ultra ATA/66	Ultra160 SCSI
Platters / Heads	4 / 8	11/22
Bytes per sector	512	512-528
<b>Performance</b>		
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

## Observations

- Getting first byte from disk read is slow
  - high latency
- Peak bandwidth high, but rarely achieved
- Need to mitigate disk performance impact
  - Do extra calculations to speed up disk access
    - Schedule requests to shorten seeks
  - Move some disk data into main memory – file system caching