**Principles of I/O Hardware and Software**

**Principles of I/O Hardware**

I/O Hardware

I/O Devices

Block Devices

* Information is stored and accessed in fixed-size blocks
* Block addressable, not byte addressable
* Common block sizes: 512 – 32,768 bytes

Character Devices

* Send or receive streams of characters
* NOT byte or block addressable

Device Controllers

* Most I/O devices are electro-mechanical.
* The electrical component that interfaces with the CPU (actually the OS) is called the device controller or adapter.
* The Controller is the go-between for the OS and the device
* Controllers for PCs and embedded devices are implemented as daughter cards and inserted into the backplane of the parentboard (or motherboard)

Memory-mapped I/O

* Memory-mapped I/O uses the same address bus to address both memory and I/O devices
* The memory and registers of the I/O devices are mapped to (associate with) address values.
* When an address is accessed by the CPU, it may refer to a portion of physical RAM, but it can also refer to memory of the I/O device.
* The CPU instructions used to access the memory can also be used for accessing devices.
* Each I/O device monitors the CPU's address bus and responds to any CPU access of an address assigned to that device, connecting the data bus to the desired device's hardware register.
* To accommodate the I/O devices, areas of the addresses used by the CPU must be reserved for I/O and must not be available for normal physical memory

I/O Ports

* Port-mapped I/O often uses a special class of CPU instructions specifically for performing I/O
* This is found on Intel microprocessors, with the IN and OUT instructions.
* These instructions can read and write one to four bytes (outb, outw, outl) to an I/O device.
* I/O devices have a separate address space from general memory, either accomplished by an extra "I/O" pin on the CPU's physical interface, or an entire bus dedicated to I/O.
* Because the address space for I/O is isolated from that for main memory, this is sometimes referred to as isolated I/O.

Interrupt Request Line (IRQ)

* An IRQ is a hardware line used in a PC by (ISA bus) devices like keyboards, modems, sound cards, etc., to send interrupt signals to the processor to tell it that the device is ready to send or accept data.
* There are only sixteen IRQ's (0-15) available in the i386 (PC) architecture for sharing among the various ISA devices.





Direct Memory Access (DMA)

* Feature of modern computers that allow certain hardware subsystems within the computer to access system memory independently of the central processing unit (CPU)
* The CPU initiates the transfer, does other operations while the transfer is in progress, and receives an interrupt from the DMA controller when the operation is done
* Many hardware systems use DMA, including disk drive controllers, graphics cards, network cards and sound cards
* Computers that have DMA channels can transfer data to and from devices with much less CPU overhead than computers without a DMA channel
* Used mainly in Block devices

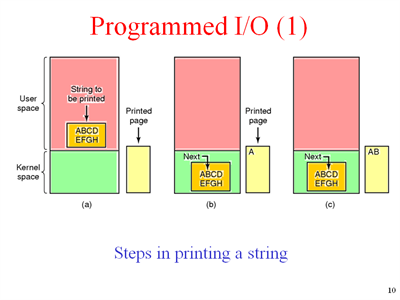


**Principal of I/O Software**

1. Goals of I/O Software
   * Device independence-
     + Read and Write from Floppy, Disk, CD-ROM without modifying programs
   * Uniform naming
     + Names are not dependent on the specific device.
       - Different devices of same type have similar name
     + In UNIX, all I/O is integrated with the file system
   * Error handling done as close to HW as possible
     + Propagate errors up only when lower layer cannot handle it.
     + Hide errors as much as possible—many HW errors are transient
   * Synchronous read/write at application level
     + Most I/O hardware operates asynchronously
     + Synchronous (blocking) read/write easier to program
     + Buffering
     + data coming off a device cannot be stored in final destination
     + Sharable vs. dedicated devices
       - disks are sharable
       - tape drives would not be

**Programmed I/O**

* Also called polling, or busy waiting
* I/O module (controller) performs the action, not the processor
* Sets appropriate bits in the I/O status register
* No interrupts occur
* Processor checks status until operation is complete; Wastes CPU cycles



**Interrupt-Driven I/O**

* Processor is interrupted when I/O module (controller) ready to exchange data
* Processor is free to do other work
* No needless waiting
* Consumes a lot of processor time because every word read or written passes through the processor

**Direct Memory Access**

* Transfers a block of data directly to or from memory
* An interrupt is sent when the task is complete
* The processor is only involved at the beginning and end of the transfer

1. **Structured I/O Software**

****

**Interrupt Handlers**

* Hidden from applications
* Used to bridge gap between asynchronous I/O hardware and synchronous read/write semantics
* Often implemented as top-half and bottom-half handlers
  + Top-half
    - does as little as possible
    - not scheduled
  + Bottom-half
    - closely related to (if not exactly) the device driver
    - scheduled

**Device Drivers**

* Device dependent code
* Each device driver handles (at most) one class of devices
* Device drivers communicate with the device controllers
  + Only part that knows the details of the device.
  + Hence, device dependent
* Translate device-independent (abstract) requests to device-specific commands

**Device-Independent I/O Software**

* Functions of device-independent I/O SW
  + Uniform interfacing for device drivers
  + Device naming
  + Mnemonic names mapped to Major and Minor device numbers
  + Device protection
  + Providing a device-independent block size
  + Buffering
  + Storage allocation on block devices
  + Allocation and releasing dedicated devices
  + Error Reporting

**User-Space I/O SW**

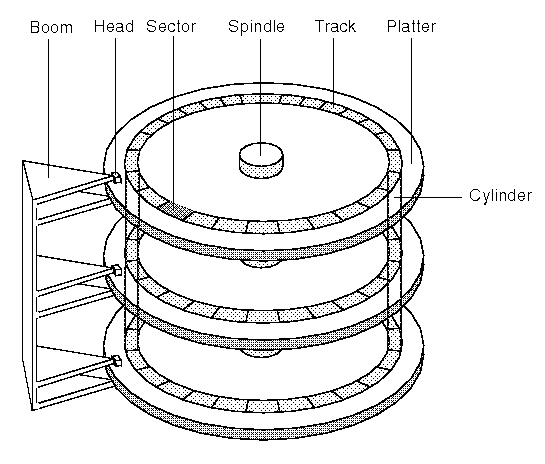
* I/O Libraries (e.g.,stdio) are in user-space to provide an interface to the OS resident device-independent I/O SW
  + These routines do the formatting for the user that is such a pain to do, but everyone wants it
  + Simultaneous Peripheral Operations On-Line (Spooling)
  + A user-space print command puts a file in the spooling directory and then asks a daemon process to execute the I/O request
  + Printing is one use of spooling

**Disk**

* Management and ordering of disk access requests is important:
  + Huge speed gap between memory and disk
  + Disk throughput is extremely sensitive to
    - Request order $\Longrightarrow$Disk Scheduling
    - Placement of data on the disk $\Longrightarrow$file system design
  + Disk scheduler must be aware of disk geometry
* Disk management issues
  + Formatting
    - Physical: divide the blank slate into sectors identified by headers containing such information as sector number; sector interleaving
    - Logical: marking bad blocks; partitioning (optional) and writing a blank directory on disk; installing file allocation tables, and other relevant information (file system initialization)
  + Reliability
    - disk interleaving or striping
    - RAIDs (Redundant Array of Inexpensive Disks): various levels, e.g., level 0 is disk striping)
  + Controller caches newer disks have on-disk caches (128KB 512KB)

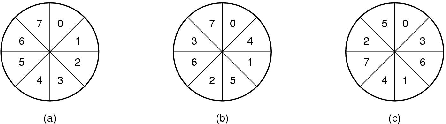
**Disk Storage**

* Disk storage or disc storage is a general category of storage mechanisms, in which data are digitally recorded by various electronic, magnetic, optical, or mechanical methods on a surface layer deposited of one or more planar, round and rotating disks (or discs) (also referred to as the media).
* A disk drive is a device implementing such a storage mechanism with fixed or removable media; with removable media the device is usually distinguished from the media as in compact disc drive and the compact disc.
* Notable types are the hard disk drive (HDD) containing a non-removable disk, the floppy disk drive (FDD) and its removable floppy disk, and various optical disc drives and associated optical disc media.



**Disk Interleaving**

Blocks are often placed on the disk in nonsequential order to allow time for the DMA buffer to be transferred to main memory.



**Disk Hardware**

* Disk drives addressed as large 1-dimensional arrays of logical blocks (smallest transfer unit)
* 1-dimensional array of logical blocks mapped onto sectors of disk sequentially
  + sector 0: 1st sector of 1st track on outermost cylinder
  + mapping in order through that track, then rest of tracks in that cylinder, then through rest of cylinders from outermost to innermost
* Outer tracks can store more sectors than inner without exceed max information density (see Fig. [5.14](http://siber.cankaya.edu.tr/OperatingSystems/spring2004/ceng328/node110.html" \l "9.6) Left)
* Evolution of Disk Hardware (see Fig. [5.14](http://siber.cankaya.edu.tr/OperatingSystems/spring2004/ceng328/node110.html#9.6) Right)

|  |
| --- |
| \includegraphics[scale=0.13]{figures/5-18.ps} |
| \includegraphics[scale=0.13]{figures/5-17.ps} |

|  |
| --- |
| **Figure 5.14:** Left: (a) Physical geometry of a disk with two zones (b) A possible virtual geometry for this disk, Right: Disk parameters for the original IBM PC floppy disk and a Western Digital WD 18300 hard disk. |

* + Average seek time is approx 12 times better
  + Rotation time is 24 times faster
  + Transfer time is 1300 times faster
  + Most of this gain is due to increase in density
  + Represents a gradual engineering improvement
* Disk Performance (see Fig. [5.15](http://siber.cankaya.edu.tr/OperatingSystems/spring2004/ceng328/node110.html#9.7))
  + Disk is a moving device; must be positioned correctly for I/O
  + Execution of a disk operation involves
    - Wait time: the process waits to be granted device access
      * Wait for device: time the request spend in wait queue
      * Wait for channel: time until a shared I/O channel is available
    - Access time: time hardware need to position the head
      * Seek time: position the head at the desire track
      * Rotational delay (latency): spin disk to the desired sector
    - Transfer time: sectors to be read/written rotate below head

|  |
| --- |
| \includegraphics[scale=1]{figures/diskperformance.ps} |
| **Figure 5.15:** Disk Performance. |

* Estimating Access Time;
  + Seek Time $T_s$: Moving the head to the required tgrack not linear in the number of tracks to traverse: startup time, settling time. Typical avearge seek time: a few milliseconds
  + Rotational delay: rotational speed, $r$, of 5000 to 10000 rpm. At 10000 rpm, one revolution per 6ms $\Rightarrow$average delay 3ms
  + Transfer time: to transfer $b$bytes, with $N$bytes per track;

$\displaystyle T={b \over rN}
$

Total average access time:

$\displaystyle T_a=T_s+{1 \over 2r}+{b \over rN}
$

* A Timing Comparison
  + $T_s=2$ms, $r=10000$rpm, 512B sect, 320 sect/track
  + read a file with 2560 sectors (=1.3MB)
  + file stored compactly (8 adjacent tracks): Read first track

|  |  |
| --- | --- |
| Average seek | 2ms |
| Rot. Delay | 3ms |
| Read 320 sectors | 6ms |
| Total | 11ms |
| All sectors | 11+7\*9=74ms |

* + Sectors distributed randomly over the disk: Read any sector

|  |  |
| --- | --- |
| Average seek | 2ms |
| Rot. Delay | 3ms |
| Read 1 sectors | 0.01875ms |
| Total | 5.01875ms |
| All | 2560\*5.01875=20,328ms |

* Disk Performance is Entirely Dominated by Seek and Rotational Delays
  + Will only get worse as capacity increases much faster than increase in seek time and rotation speed (it has been easier to spin the disk faster than improve seek time)
  + Operating System should minimise mechanical delays as much as possible

**Disk Formatting**

A new magnetic disk is a blank slate: It is just a platter of a magnetic recording material.

Types of Disk Formatting

1. Low level Formatting( Physical Formatting)
2. Logical Formatting( High Level Formatting)

**Low Level Formatting**

* Before a disk can store data, it must be divided into sectors that the disk controller can read and write. This process is called low-level formatting, or physical formatting.
* Low-level formatting fills the disk with a special data structure for each sector.
  + - The data structure for a sector typically consists of a header, a data area (usually 512 bytes in size), and a trailer.
    - The header and trailer contain information used by the disk controller, such as a sector number and an **error-correcting code** (ECC).
* When the controller writes a sector of data during normal I/O, the ECC is updated with a value calculated from all the bytes in the data area.
* When the sector is read, the ECC is recalculated and is compared with the stored value. If the stored and calculated numbers are different, this mismatch indicates that the data area of the sector has become corrupted and that the disk sector may be bad.
* The controller automatically does the ECC processing whenever a sector is read or written.

**Logical Formatting**

To use a disk to hold files, the OS still needs to record its own data structures on the disk. It does so in two steps.

* + The first step is to partition the disk into one or more groups of cylinders.
    - The OS can treat each partition as though it were a separate disk.
    - For instance, one partition can hold a copy of the OS's executable code, while another holds user files.
  + After partitioning, the second step is logical formatting (or creation of a file system).
    - In this step, the OS stores the initial file-system data structures onto the disk.
    - These data structures may include maps of free and allocated space (a FAT or inodes) and an initial empty directory.

1. When reading sequential blocks, the seek time can result in missing block 0 in the next track. Disk can be formatted using a cylinder skew to avoid this.

**Disk ARM Scheduling**

1. **First-in, First-out (FIFO)**

* Process requests as they come
* Fair (no starvation)
* Good for a few processes with clustered requests
* Deteriorates to random if there are many processes

1. **Shortest Seek Time First (SSTF)**

* Select request that minimizes the seek time
* Generally performs much better than FIFO
* May lead to starvation

1. **Elevator Algorithm (SCAN)**

* Move head in one direction; Services requests in track order until it reaches the last track, then reverses direction
* Better than FIFO, usually worse than SSTF
* Avoids starvation
* Makes poor use of sequential reads (on down-scan)

1. **Modified Elevator (Circular SCAN, C-SCAN)**

* Like elevator, but reads sectors in only one direction; When reaching last track, go back to first track non-stop
* Better locality on sequential reads
* Better use of read ahead cache on controller
* Reduces max delay to read a particular sector

**Selecting a Disk-Scheduling Algorithm**

* SSTF common, natural appeal
* SCAN and C-SCAN perform better if heavy load on disk
* Performance depends on number and types of requests
* Requests for disk service influenced by file-allocation method
* Disk-scheduling should be separate module of OS, allowing replacement with different algorithm if necessary

Example:

Suppose that the head of a moving head disk has 200 tracks numbers 0-199, is currently serving the request at track 143 and has just finished a request at track 125. The queue of requests is kept in FIFO order 86, 147, 91, 177, 94, 150, 102, 175, 130.

What is the total number of head movements needed to satisfy these requests for the following disk scheduling algorithm?

FCFS, SSTF, SCAN, LOOK, C-SCAN.

Solution:

**FIFO (FCFS) scheduling algorithm**

Here the head is move in the order 143, 86, 147, 91, 177, 94, 150, 102, 175, 130

## = |86-143|+|147-86|+|91-147|+|177-91|+|94-177|+|150-94|+|102-150|+|175-102|+|130-175|

## = 565 cylinders

## Average head movements =565/9=62.77 cylinders

## SSTF scheduling algorithm

## The head will move in the following order

## 143-147-150-130-102-94-91-86-175-177

## Total number of head movement is:

## = |147-143|+|150-147|+|130-150|+|102-130|+|94-102|+|91-94|+|86-91|+|175-86|+|175-177|

## = 4+3+20+28+8+3+5+89+2

## =162 cylinder

## Average head movements=162/9=18 cylinders

## SCAN Scheduling Algorithm

## The head is move in order

## 143, 147, 150, 175, 177, 199, 130, 102, 94, 91, 86

## Total number of head movements are:

## = |143-147 |+ |150-147| + |175-150| + |177-175| + |199-177| + |130-199| + |102-130| + |94-102| + |91-94| + |86-91|

## =169 cylinders

## Average head movements =169/9=18.77 cylinders

## LOOK Scheduling Algorithm

## The head is move in order

## 143, 147, 150, 175, 177, 130, 102, 94, 91, 86

## Total number of head movements is:

## = |147-147 |+ |150-147|+|175-150|+|177-175|+|130-177| + |102-130| +|94-102| + |91-94| + |86-91|

## =125 cylinders

## Average head movements =125/9=13.88 cylinders

## C-SCAN Scheduling Algorithm

The head is move in order

143, 147, 150, 175, 177, 199, 0, 86, 91, 94, 102, 130

Total number of head movements are:

= |147-147 |+ |150-147| + |175-150| + |177-175| + |199-0| + |0-86| + |91-86| + |94-91| + |102-94| + |130-102|

=385 cylinders

Average head movements =385/9=18.77 cylinders

## Error-Handling

|  |
| --- |
| \includegraphics[scale=0.2]{figures/5-29.ps} |
| a) A disk track with a bad sector  b) Substituting a spare for the bad sector  c) Shifting all the sectors to bypass the bad one. |

* Bad blocks are usually handled transparently by the on-disk controller (see Fig)

**Stable Storage:**

**Stable storage** is a classification of computer data storage technology that guarantees atomicity for any given write operation and allows software to be written that is robust against some hardware and power failures. To be considered atomic, upon reading back a just written-to portion of the disk, the storage subsystem must return either the write data or the data that was on that portion of the disk before the write operation.