Chapter Four

Device Management

Contents

- overview
- Principles of I/O hardware
- I/O devices
- Devices controller
- Memory-mapped I/O
- Direct memory access(DMA)
- Interrupt revisited
- Principle of I/O software
- Goals of I/O and issues of software
- I/O operation
- I/O software layers
- Disks

Overview

- The control of device connected to computer is major concern of an operating system.
- Operating system issue commands to the devices, catch interrupts, and handle errors.
- It should also provide an interface between the devices and the rest of the system that is simple and easy to use.
- Because I/O devices varied so widely in their function and speed, varied method is needed to control them.
- These method form the I/O subsystem of the kernel.

I/O devices

- OS concerns with the way data are handled by the I/O device.
- There are two types of I/O devices: blocked and character.

❖ Block devices
  - Stores information in Fixed-size blocks, each one with its own address.
  - Common block size range from 512bytes to 32,768bytes.
  - All transfers are in units of one or more entire (consecutive) blocks.
  - The essential property of a **block device** is that it is possible to read or write each block independently of all the other ones.

- Example: Hard disks, CD-ROMs, and USB sticks.

I/O devices(con't...)

- Character devices
  - Delivers or accepts stream of character, without regard to any block structure.
  - It is not addressable and does not have any seek operation.
- Example: Printers, network interfaces, mice (for pointing)
- Most other devices that are not disk-like can be seen as character devices.
- Some devices just do not fit into both above classification.
- Clocks, for example, are not block addressable, Nor do they generate or accept character streams.

Device controller

- Is collection of electronics that can operate a port, bus or devices
- For example disk **controller's job is to convert the serial** bit stream into a block of bytes and perform any error correction necessary.
- Controller has a few registers that are used for communicating with the CPU.
- By writing into these registers, the operating system can command the device to deliver data, accept data, switch itself on or off, or otherwise perform some action.
- By reading from these registers, the operating system can learn what the device's state is, whether it is prepared to accept a new command, and so on.

Memory-mapped I/O

- How the CPU communicates with controllers to accomplish I/O transfer? Two alternatives exists.

1. Use of special I/O instruction (IN and OUT):
   - Each register is assigned an I/O port number, an 8- or 16-bit integer.
   - The set of all the I/O ports form the I/O port space
   - Protected ordinary user programs cannot access it (only the operating system can).

   Example:

   IN REG, PORT: the CPU can read in control register PORT and store the result in CPU register REG.

   OUT PORT, REG: the CPU can write the contents of REG to a control register.

2. Mapping controller register to memory space (Memory- Mapped I/O):
   - Each register is assigned a unique memory address to which no memory is assigned.
   - CPU executes I/O requests using the standard data transferring instruction to read and write controller registers.

Direct memory access(DMA)

- Requesting data from an I/O controller one byte at a time wastes the CPU's time.
- So a different scheme, called DMA (Direct Memory Access) chip that can control the flow of bits between memory and some controller without constant **CPU intervention.**
- The CPU sets up the DMA chip, telling it how many bytes to transfer, the device and memory addresses involved, and the direction, and lets it go.
- When the DMA chip is done, it causes an **interrupt.**

Interrupt revisited

- When an i/o device has finished the work given to it, it causes an **interrupt**
- It does this by asserting a signal on a bus line that it has been assigned.
- This signal is detected by the interrupt controller chip on the parent board, which then decides what to do.
- If no other interrupts are pending, the interrupt controller processes the interrupt immediately.
- If another one is in progress, or another device has made a simultaneous request on a higher-priority interrupt request line on the bus, the device is just ignored for the moment.
- In this case it continues to assert an interrupt signal on the bus until it is serviced by the CPU.

Interrupt revisited (cont….)

- To handle the interrupt:
  - The controller puts a number on the address lines specifying which device wants attention and asserts a signal to interrupt the CPU.
  - New program counter is fetched from table called interrupt vector using the number on the address lines is used as an index into this table.
  - Interrupt service procedure is start running and finally acknowledges the interrupt by writing a certain value to one of the interrupt controller's I/0 ports.

Goals and issues of i/o software

- Device Independence:
  - It should be possible to write programs that can access any i/o device without having to specify the device in advance.
  - It is up to the operating system to take care of the problems caused by the fact that these devices are really different.
- **Uniform Naming**:

  The name of a file or a device should simply be a string or an integer and not dependent on the device in any way.
- Error handling:
  - In general, errors should be handled as close to the hardware as possible (at the device controller level).
  - Many errors are transient, such as read errors caused by specks of dust on the read head, and will go away if the operations are repeated.

Goals and issues of i/o software

- Transferª There are two types of transfer modes :

Synchronous (Blocking) and **Asynchronous** (interrupt –driven).

- – Synchronous transfer: the program requesting I/O transfer will be suspended until the transfer is completed.
- – Asynchronous transfer: the CPU starts the transfer and goes off to do something until the interrupt that shows the completion of the transfer arrives.

•Device typesª There are two device types

Shareable (such as disk)

- Used by many users at the same time.
- No problems are caused by multiple users having open files on the same disk at the same time.

Dedicated (tape drives)

- Have to be dedicated to a single user until that user is finished.
- Having two or more users writing blocks intermixed at random to the same tape will definitely not work.

I/O Operation | Programmed I/O

- There are three fundamentally different ways that I/O operation can be performed.
  - Programmed I/O
  - Interrupt-driven I/O
  - I/O using DMA
- Programmed I/O
  - is a method in which the main CPU inputs or outputs each byte or word and sits in a tight loop waiting until it can get or send the next one.
  - The CPU continuously polls the device to see if it is ready or not. This behavior is often called polling or busy waiting.

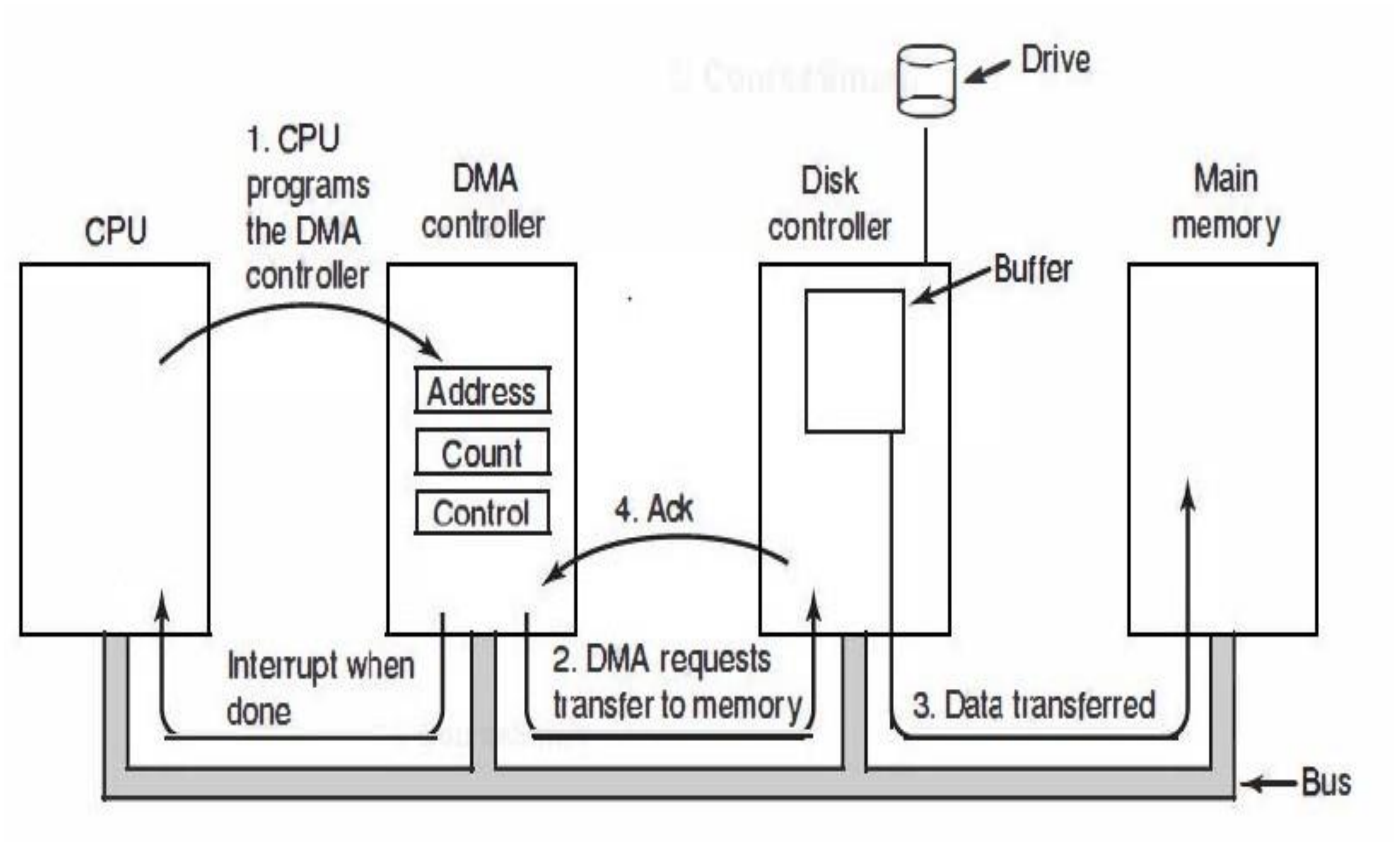
Interrupt-driven I/O

- Interrupt-driven I/O
  - is method in which the CPU starts an I/O transfer for a character or word and **goes off** to do something else until an interrupt arrives signaling completion of the I/O.
  - The advantage of interrupt-driven I/O is allow the CPU to do something else while waiting for the device to become ready for the next work.
  - The disadvantage of interrupt-driven I/0 is that an interrupt occurs on every character or word which Interrupts take time, that wastes a certain amount of CPU time.

Direct memory access(DMA)

- I/O using DMA
  - the way in which a separate chip manages the complete transfer of a block of data, given an interrupt only when the entire block has been transferred.
  - DMA is programmed I/O, only with the DMA controller doing all the work, instead of the main CPU.
  - This strategy requires special hardware (the DMA controller) but frees up the CPU during the I/O to do other work.
  - The big win with DMA is reducing the number of interrupts from one per character to one per buffer.
  - The DMA controller is usually much slower than the main CPU. If the DMA controller is not capable of driving the device at full speed, or the CPU usually has nothing to do any way while waiting for the DMA interrupt, then interrupt-driven I/0 or even programmed I/0 may be better.

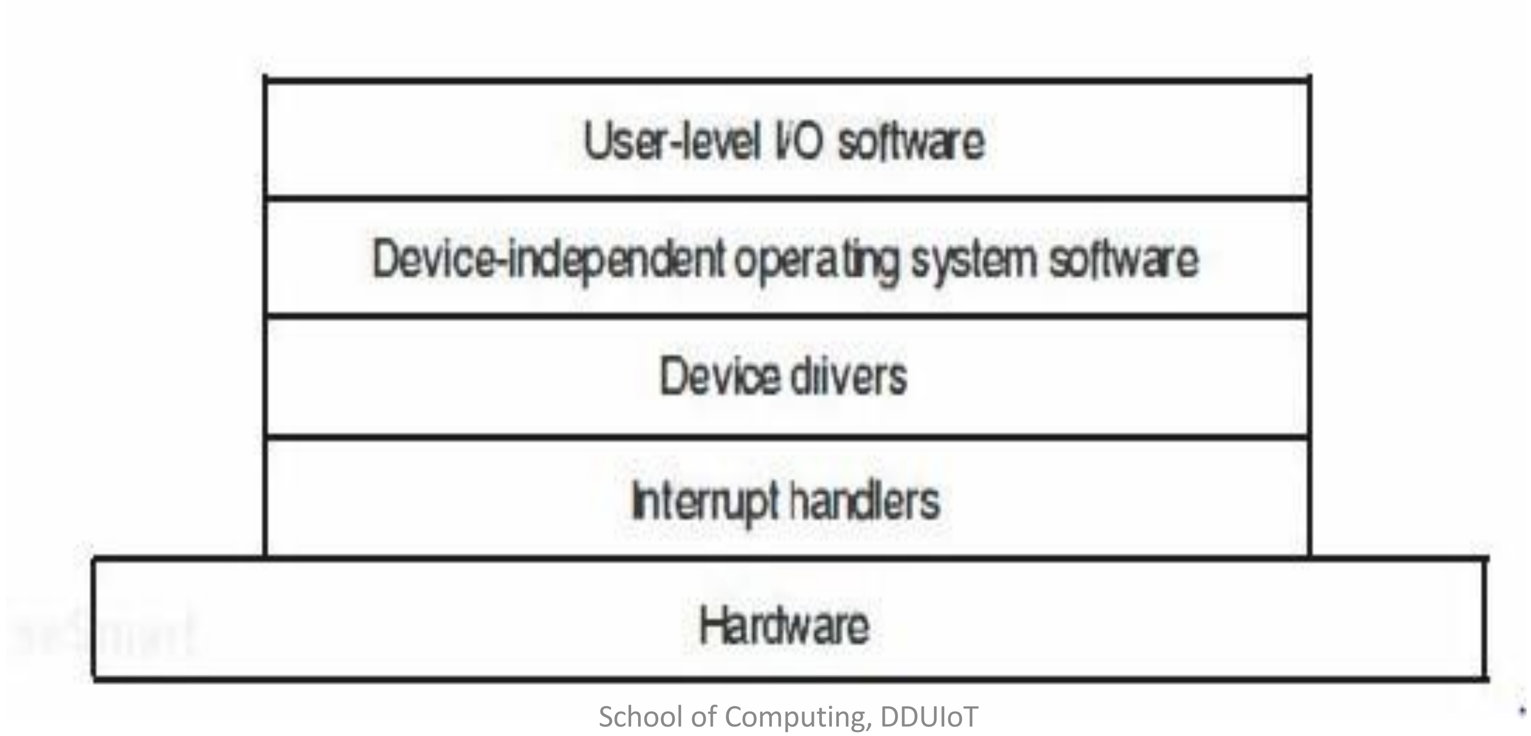
Direct Memory Access(DMA)



Operation of DMA tranfer

I/0 software layers

- I/0 software is typically organized in four layers.
- Each layer has a well-defined function to perform and a well-defined interface to the adjacent layers.
- The functionality and interfaces differ from system to system.



Device driver

- All devices – Dependent code goes in the device driver.
- Each device driver handles one device type, or at most, one class of closely related devices.
- The job of a device driver is to accept requests from the device independent software above it and sees to it that the request is executed.
- Steps in carrying out I/O requests
  - Translate it from abstract to concrete terms.
  - Write into the controller's device registers.
  - The device driver blocks itself until interrupt comes.
  - The blocked driver will be awakened by the interrupt.
  - Device driver checks for errors.
  - If everything is all right, the driver may have data to pass to the device independent software.

If nothing is queued, the driver blocks waiting for the next request

Device independent I/O software

- It is large fraction of I/O software.
- Perform the I/0 functions that are common to all devices and to provide a uniform interface to the user-level software.
- Functions of the device-independent I/O software:

❖ Uniform interfacing for device drivers – Perform I/O function common to all drives.

❖ Device Naming – Responsible for mapping symbolic devices names onto the proper driver.

❖ Device protection – Prevent users from accessing devices that they are not entitled to access.

❖ Providing device-independent block size – Provide uniform block size to higher layers hiding differences in block sizes.

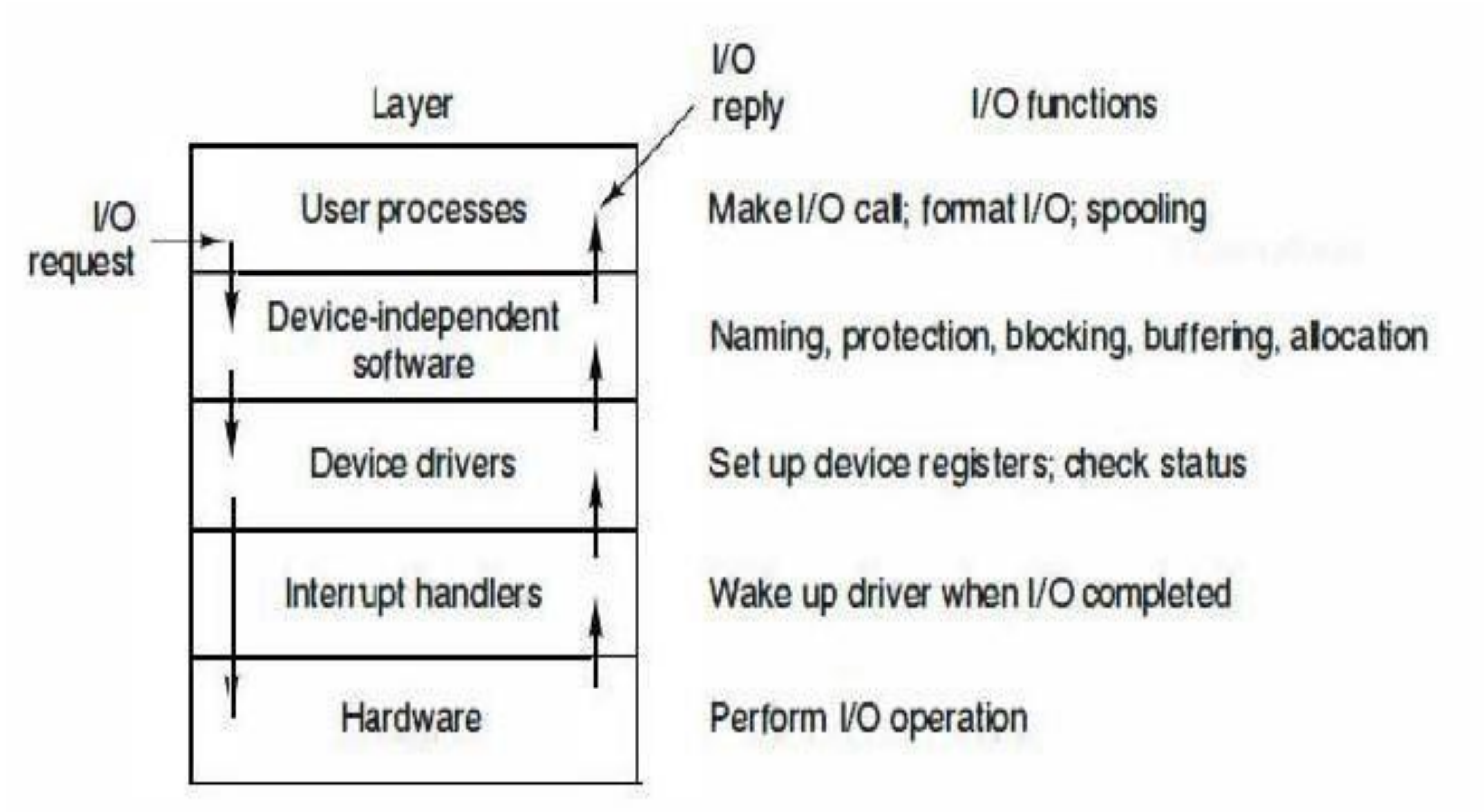
Device independent I/O software

- ❖ **Buffering**: If a user process write half a block, the OS will normally keep the data in buffer until the rest of the data are written. Keyboard inputs that arrive before it is needed also require buffering.
- ❖ **Allocating and releasing dedicated devices**: It is up to the OS to examine requests for devices usage and accept or reject them.
- ❖ **Error reporting**: Errors handling, by and large, is done by drivers. Most errors are device dependent. After the driver tries to read the block a certain number of times, it gives up and informs the device-independent software. It then reports to the caller.

User level I/O software

- Although most of the i/o software is within the operating system, a small portion of it consists of libraries linked together with user programs.
- System calls, including the I/O system calls, are normally made by library procedures.
- The collection of all these library procedures is clearly part of the I/0 system.
- Not all user-level I/0 software consists of library procedures.
- Another important category is the spooling system which is a way of dealing with dedicated I/O devices in a multi-programming system.

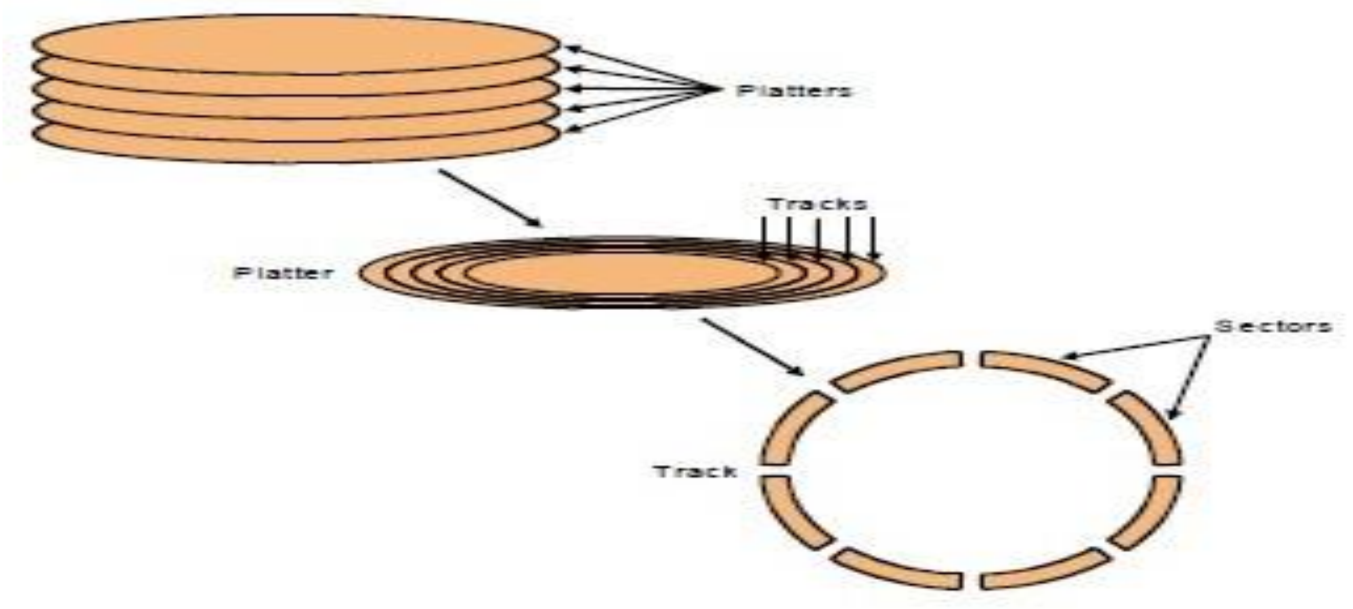
Summary of I/O software layers


Layer
I/O reply
I/O functions
I/O request
User processes
Make I/O call; format I/O; spooling
Device-independent software
Naming, protection, blocking, buffering, allocation
Device drivers
Set up device registers; check status
Interrupt handlers
Wake up driver when I/O completed
Hardware
Perform I/O operation

Disk

- All real disks are organized into cylinders/platter, each platter is arranged like a record, each one containing many tracks.
- Each of the tracks then will be divided into sectors (equal number of sectors or different number of sectors).
- In the case of equal number of sectors
  - The data density as closer to the center (hub) is high.
  - The speed increases as the read/write moves to the outer tracks.
- Modern large hard drives have more sectors per track on outer tracks e.g. IDE drives.
- Many controllers can read or write on one drive while seeking on one or more other drives, but floppy disk controller cannot do that.

Disk (con't…)



The ugly guts of a hard disk.

Disk access time

- The time required to read or write a disk block is determined by three factors
- The seek time: measures the delay for the disk head to reach the right track.
- The rotational delay: accounts for the time to get to the right sector.
- Transfer time: is how long the actual data read or write takes.
- There may be additional over head for the operating system or the controller hardware on the hard disk drive.
- Rotational speed: measured in revolutions per minute or RPM, partially determines the rotational delay and transfer time.

Disk access time(con't…)

- Rotational delay depends partly on how fast the disk platters spin.
- Average rotational delay *= 0.5 x rotations x rotational speed*
- For example, a 5400 RPM disk has an average rotational delay of: 0.5 rotations /(5400 rotations/minute) = 5.55ms.
- The overall response time is the sum of the seek time, rotational delay, transfer time, and overhead.

❖ Example: Assume a disk has the following specifications.
  - An average seek time of 9ms
  - A 5400 RPM rotational speed
  - A 10MB/s average transfer rate
  - 2ms of overheads

Disk access time(con't…)

How long does it take to read a random 1,024 byte sector?

- The average rotational delay is 5.55ms.
- The transfer time will be about (1024 bytes / 10 MB/s) = 0.1ms.

➢ So the response time is then 9ms + 5.55ms + 0.1ms + 2ms = 16.7ms.

➢ That's 16,700,000 cycles for a 1GHz processor!

Disk scheduling algorism

- the seek time dominates the other two times, so reducing the mean seek time can improve system performance substantially.
- There is different disk scheduling algorithm is used to reduce mean seek time.
  - First Come First Served (FCFS)
  - Shortest Seek Time First (SSTF)
  - SCAN (Elevator) Algorithm
  - C-SCAN (Modified Elevator) Algorithm (Circular-SCAN)

First Come First Served (FCFS)

- Accept a request one at a time and carries them out in that order.
- Service requests in the order they are received.
- The simplest and the fairest of all, but it doesn't improve performance

  E.g: Track initial position: 11

✓ Track request: 1,36,16,34,9,12

✓ Service order: 1,36,16,34,9,12

Shortest Seek Time First (SSTF)

- It handles the closest (the least disk arm movement) starvation: it lacks fairness request next, to minimize seek time.
- Performance (efficiency): provides better performance.
- Possibility of Example: Track initial position: 11

✓ Track request: 1,36,16,34,9,12

✓ Service order: 12,9,16, 1, 34, 36

SCAN (Elevator) Algorithm

- The disk arm keeps moving in the same direction until there are no more outstanding requests in that direction, and then it switches direction.
- It has the head start at track 0 and move towards the highest numbered track, servicing all requests for a track as it passes the track.
- SCAN algorithm is guaranteed to service every request in one complete pass through the disk.
- It provides better service distribution.

Example. Track initial position: 11

✓ Track request: 1,36,16,34,9,12

✓ Direction bit: 1

✓ Service order: 12, 16, 34, 36, 9, 1

C-SCAN (Modified Elevator) Algorithm (Circular-SCAN)

- A slight modification of elevator algorithm that has smaller variance in response times is always scans in the same direction .
- When the highest numbered cylinder with a pending request has been serviced, the arm goes to the lowest-numbered cylinder with a pending request and then continues moving in an upward direction.
- In effect, the lowest-numbered cylinder is thought of as being just above the highest-numbered cylinder.
- It reduces the maximum delay experienced by new request.

Example: Track initial position: 11

- ✓ Track request: 1,36,16,34,9,12
- ✓ Direction bit: 1
- ✓ Service order: 12, 16, 34, 36, 1, 9

Redundant array independent disk(RAID)

- Redundant array of independent disks may be used to increase disk reliability.
- In a RAID system, a single large file is stored in several separate disk units by breaking the file up into a number of smaller pieces and storing these pieces on different disks.
- When a file is accessed for a read, all disks deliver their data in parallel.
- RAID may be implemented in hardware or in the operating system.