



Marwadi
University

Department of
Computer Engineering

Unit –6
File System Interface

Operating System -
01CE1401

Dr. Krunal Vaghela

FileConcept

3 problems

- All computer applications need to store and retrieve information. While a process is running, it can store a limited amount of information within its own address space.
- A second problem with keeping information within a process' address space is that when the process terminates, the information is lost.
- A third problem is that it is frequently necessary for multiple processes to access (parts of) the information at the same time.

FileConcept

Solution:

- To store information on disks and other external media in units called files.

File:

- A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks.
- In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.
- Files are logical unit of information created by processes.
- Every file has name and its data.
- OS is having various information with file, like data, time, last modified detail etc.
- This information called file attributes.

FileConcept

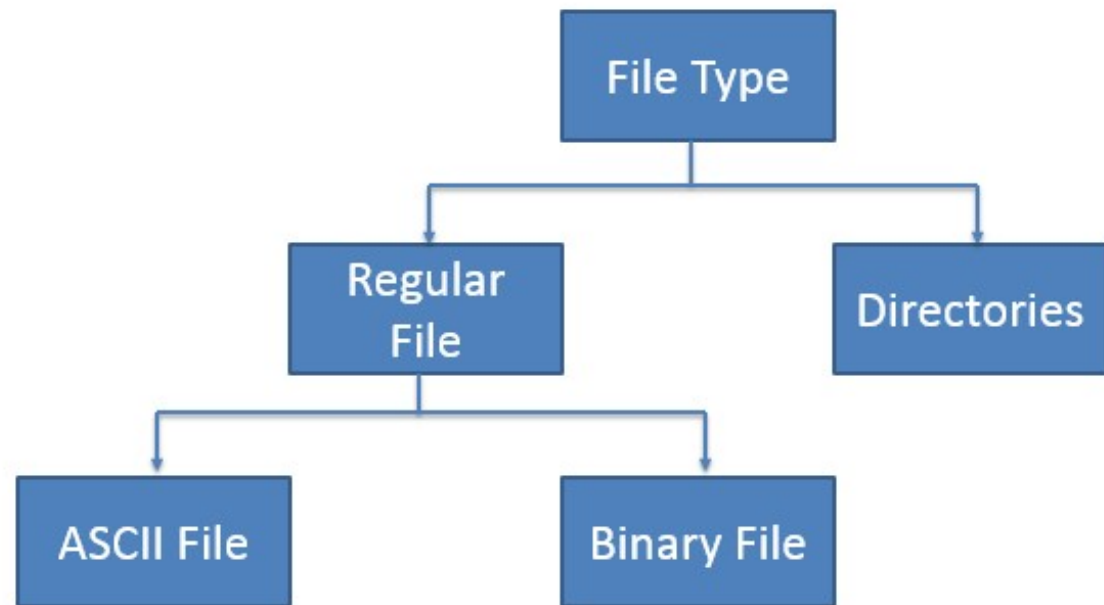
List of file attributes:

- ☐ Protection : who can access
- ☐ Password: needed to access file
- ☐ Creator
- ☐ Owner
- various attributes related to flag:
 - ☐ Read flag
 - ☐ Hidden flag
 - ☐ Random access flag
 - ☐ Lock flag

File Concept: Types of file extension

Extension	Meaning
.bak	Backup file
.c	Cprogram
.gif	Graphical interchange format image
.hlp	Help file
.html	Hyper Text Mark-up Language file
.jpg	Still picture(joint photographic group)
.obj	Object file
.pdf	Portable document file
.txt	General text file
.zip	Compressed file

File Types



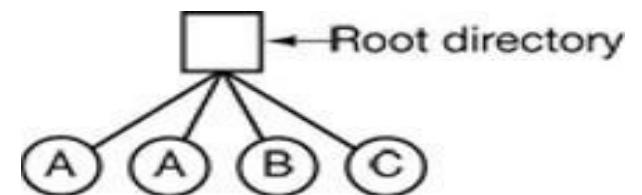
Operation on Directory

- Search for a file
- Create new file
- Delete a file
- List a directory
- Rename a file
- Traverse file system

Directory structure

Single-Level Directory Systems:

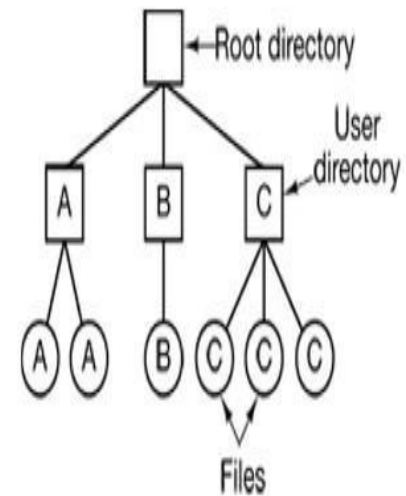
- The simplest form of directory system is having one directory containing all the files. Sometimes it is called the root directory
- The advantages of this scheme are its simplicity and the ability to locate files quickly—there is only one place to look, after all.
- A single-level directory system containing four files, owned by three different people, A , B , and C .
- The problem with having only one directory in a system with multiple users is that different users may accidentally use the same names for their files.



Directory structure

Two-level Directory Systems

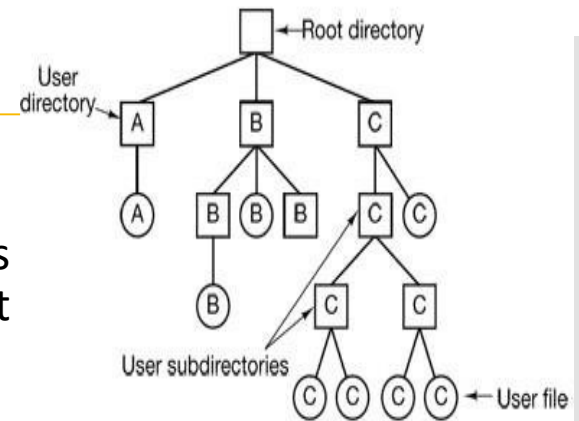
- To avoid conflicts caused by different users choosing the same file name for their own files, the next step up is giving each user a private directory.
- in that way, names chosen by one user do not interfere with names chosen by a different user and there is no problem caused by the same name occurring in two or more directories.



Directory structure

Hierarchical Directory Systems

- The two-level hierarchy eliminates name conflicts among users but is not satisfactory for users with a large number of files.
- Even on a single-user personal computer, it is inconvenient.
- It is quite common for users to want to group their files together in logical ways.
- The ability for users to create an arbitrary number of subdirectories provides a powerful structuring tool for users to organize their work.



FileStructure

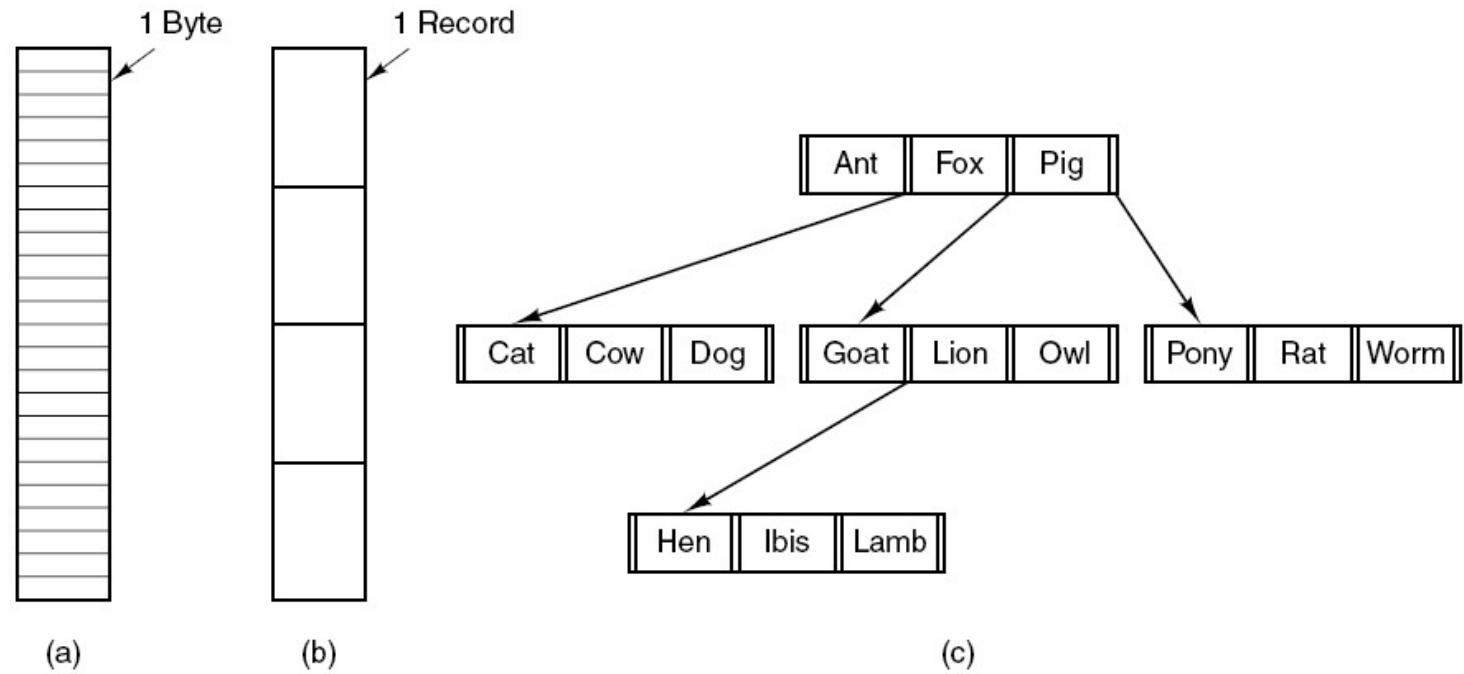


Figure : Three kinds of file structure. (a) Byte sequence. (b) Record sequence. (c) Tree.

FileStructure

Byte Sequence:

- Is an unstructured sequence of bytes.
- the operating system does not know or care what is in the file.
- User programs can put anything they want in their files and name them any way that is convenient.

Record Sequence:

- In this model, a file is a sequence of fixed-length records, each with some internal structure.
- Central to the idea of a file being a sequence of records is the idea that the read operation returns one record and the write operation overwrites or appends one record.

Tree:

- A file consists of a tree of records, not necessarily all the same length.
- each containing a key field in a fixed position in the record.
- The tree is sorted on the key field, to allow rapid searching for a particular key.

FileTypes

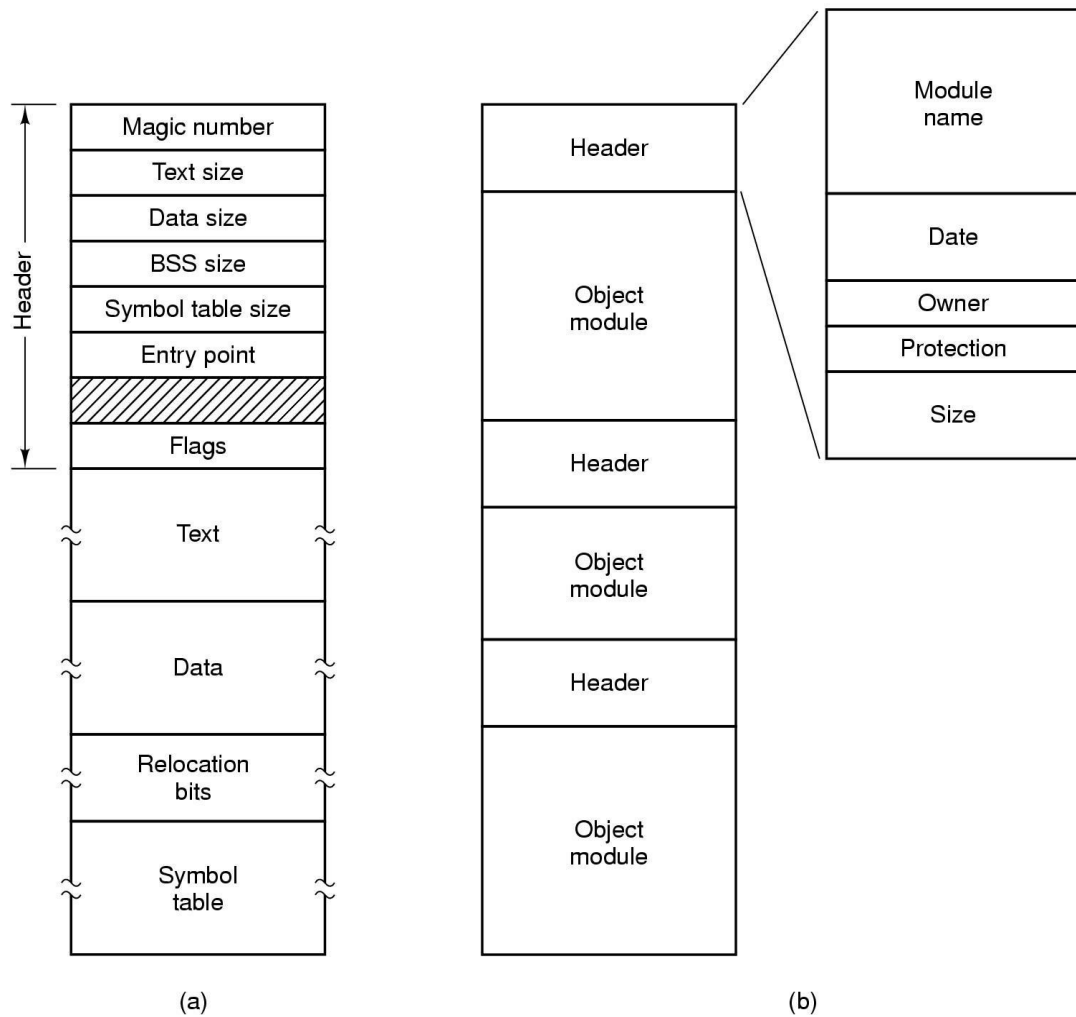


Figure : (a) An executable file. (b) An archive.

FileTypes

Executable file

- it contains header, text, data, relocation bits, and symbol table.
- The header starts with a magic number identifying the file as an executable file
- Then come the sizes of the various pieces of the file, the address at which execution starts, and some flag bits.
- Following the header are the text and data of the program itself.
- These are loaded into memory and relocated using the relocation bits.
- The symbol table is used for debugging.

FileTypes

An Archive file

- It consists of a collection of library procedures (modules) compiled but not linked.
- Each one is prefaced by a header telling its name, creation date, owner, protection code, and size.

FileTypes

Regular files:

- are the ones that contain user information.

Directories:

- are system files for maintaining the structure of the file system.

Character special:

- files are related to input/output and used to model serial I/O devices such as terminals, printers, and networks.

Block special files:

- are used to model disks.

Access Methods

Sequential Access

- a process could read all the bytes or records in a file in order, starting at the beginning, but could not skip around and read them out of order.
- Sequential files were convenient when the storage medium was magnetic tape, rather than disk.

Random Access

- When disks came into use for storing files, it became possible to read the bytes or records of a file out of order, or to access records by key, rather than by position.
- for example, database systems. If an airline customer calls up and wants to reserve a seat on a particular flight, the reservation program must be able to access the record for that flight without having to read the records for thousands of other flights first.

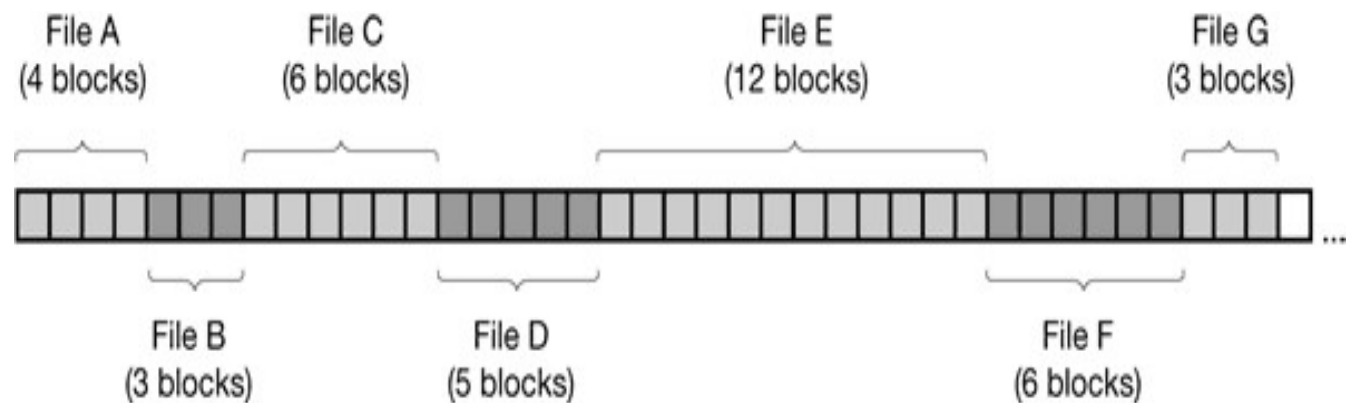
Allocation methods of file

- ☐ Contiguous
- ☐ Linked
- ☐ Indexed

Allocation methods of file

Contiguous allocation method:

- The simplest allocation scheme is to store each file as a contiguous run of disk blocks.
- Thus on a disk with 1-KB blocks, a 50-KB file would be allocated 50 consecutive blocks. With 2- KB blocks, it would be allocated 25 consecutive blocks.



Allocation methods of file

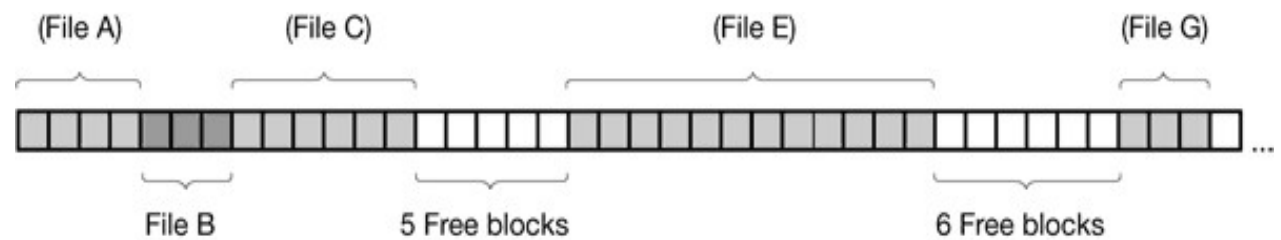
Advantage:

- it is simple to implement because keeping track of where a file's blocks are is reduced to remembering two numbers: the disk address of the first block and the number of blocks in the file.
- Second, the read performance is excellent because the entire file can be read from the disk in a single operation. Only one seek is needed (to the first block).

Allocation methods of file

Disadvantage:

- in time, the disk becomes fragmented.



Allocation methods of file

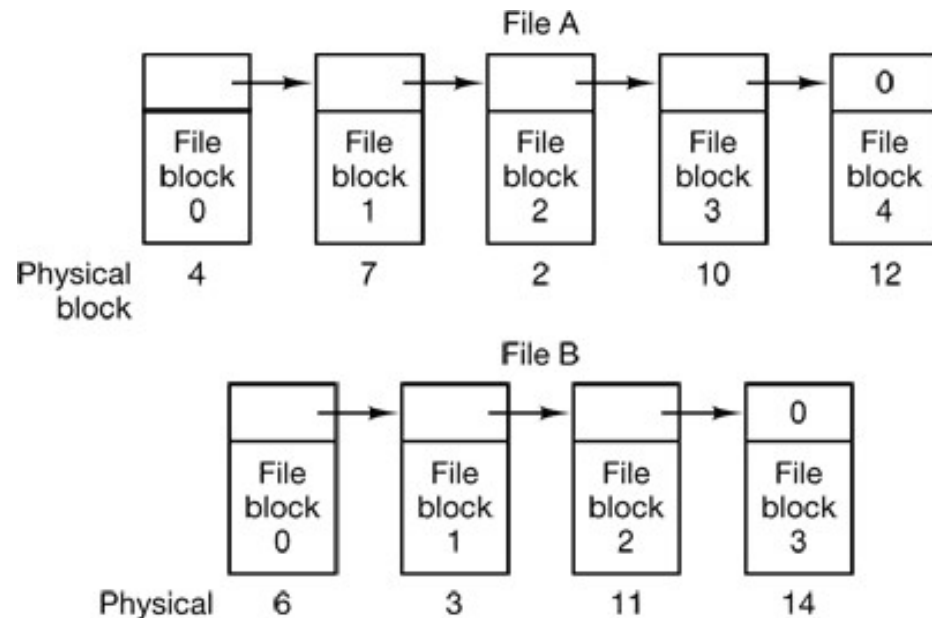
Contiguous allocation method:

- **Disadvantage:**
 - Initially, this fragmentation is not a problem since each new file can be written at the end of disk, following the previous one. However, eventually the disk will fill up and it will become necessary to either compact the disk, which is prohibitively expensive, or to reuse the free space in the holes.
 - Reusing the space requires maintaining a list of holes.
 - However, when a new file is to be created, it is necessary to know its final size in order to choose a hole of the correct size to place it in.

Allocation methods of file

□ Linked allocation:

- The second method for storing files is to keep each one as a linked list of disk blocks. The first word of each block is used as a pointer to the next one. The rest of the block is for data.



Allocation methods of file

Advantage:

- Unlike contiguous allocation, every disk block can be used in this method. No space is lost to disk fragmentation (except for internal fragmentation in the last block).
- Also, it is sufficient for the directory entry to merely store the disk address of the first block. The rest can be found starting there.

Drawback:

- On the other hand, although reading a file sequentially is straightforward, random access is extremely slow.
- Also, the amount of data storage in a block is no longer a power of two because the pointer takes up a few bytes.

Allocation methods of file

Linked List Allocation Using a Table in Memory:

- Both disadvantages of the linked list allocation can be eliminated by taking the pointer word from each disk block and putting it in a table in memory (File Allocation Table).
- **Advantage:**
 - Using this organization, the entire block is available for data. Furthermore, random access is much easier.
 - We only need the starting block number to locate entire file.
- **Disadvantage:**
 - entire table must be in memory all the time to make it work. With a 20-GB disk and a 1-KB block size, the table needs 20 million entries, one for each of the 20 million disk blocks.

Allocation methods: Linked List Allocation Using a Table in Memory

Physical block		
0		
1		
2	10	
3	11	
4	7	← File A starts here
5		
6	3	← File B starts here
7	2	
8		
9		
10	12	
11	14	
12	-1	
13		
14	-1	
15		← Unused block

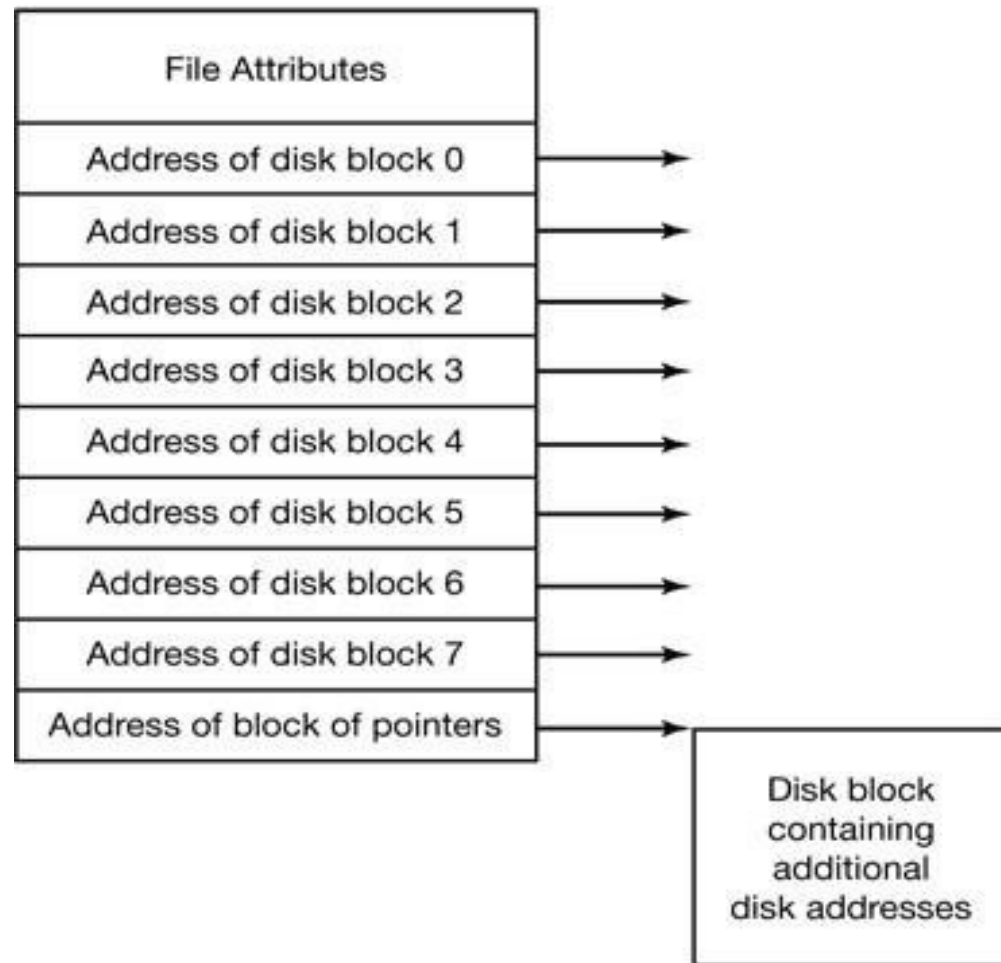
Allocation methods

i-nodes/ indexed:

- Our last method for keeping track of which blocks belong to which file is to associate with each file a data structure called an i-node (index-node), which lists the attributes and disk addresses of the files blocks.
- Given the i-node, it is then possible to find all the blocks of the file.
- The big advantage of this scheme over linked files using an in-memory table is that the i-node need only be in memory when the corresponding file is open.
- This array is usually far smaller than the space occupied by the file table described in the previous section. The reason is simple. The table for holding the linked list of all disk blocks is proportional in size to the disk itself. If the disk has n blocks, the table needs n entries. As disks grow larger, this table grows linearly with them.
- In contrast, the i-node scheme requires an array in memory whose size is proportional to the maximum number of files that may be open at once. It does not matter if the disk is 1 GB or 10 GB or 100 GB.

Allocation methods

I-nodes/ indexed:





☐ FreeSpace Management

- ☐ BitVector

- ☐ Linked list

- ☐ Grouping

Free Space Management

Bit Vector

- Frequently, the free-space list is implemented as a bit map or bit vector.
- Each block is represented by 1 bit.
- If the block is free, the bit is 1; if the block is allocated, the bit is 0.
- The main advantage of this approach is its relative simplicity and its efficiency in finding the first free block or n consecutive free blocks on the disk.

Free Space Management

Linked List

- Another approach to free-space management is to link together all the free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory.
- This first block contains a pointer to the next free disk block, and so on.
- This scheme is not efficient; to traverse the list, we must read each block, which requires substantial I/O time.
- Fortunately, however, traversing the free list is not a frequent action.
- Usually, the operating system simply needs a free block so that it can allocate that block to a file, so the first block in the free list is used.

Free Space Management

Grouping

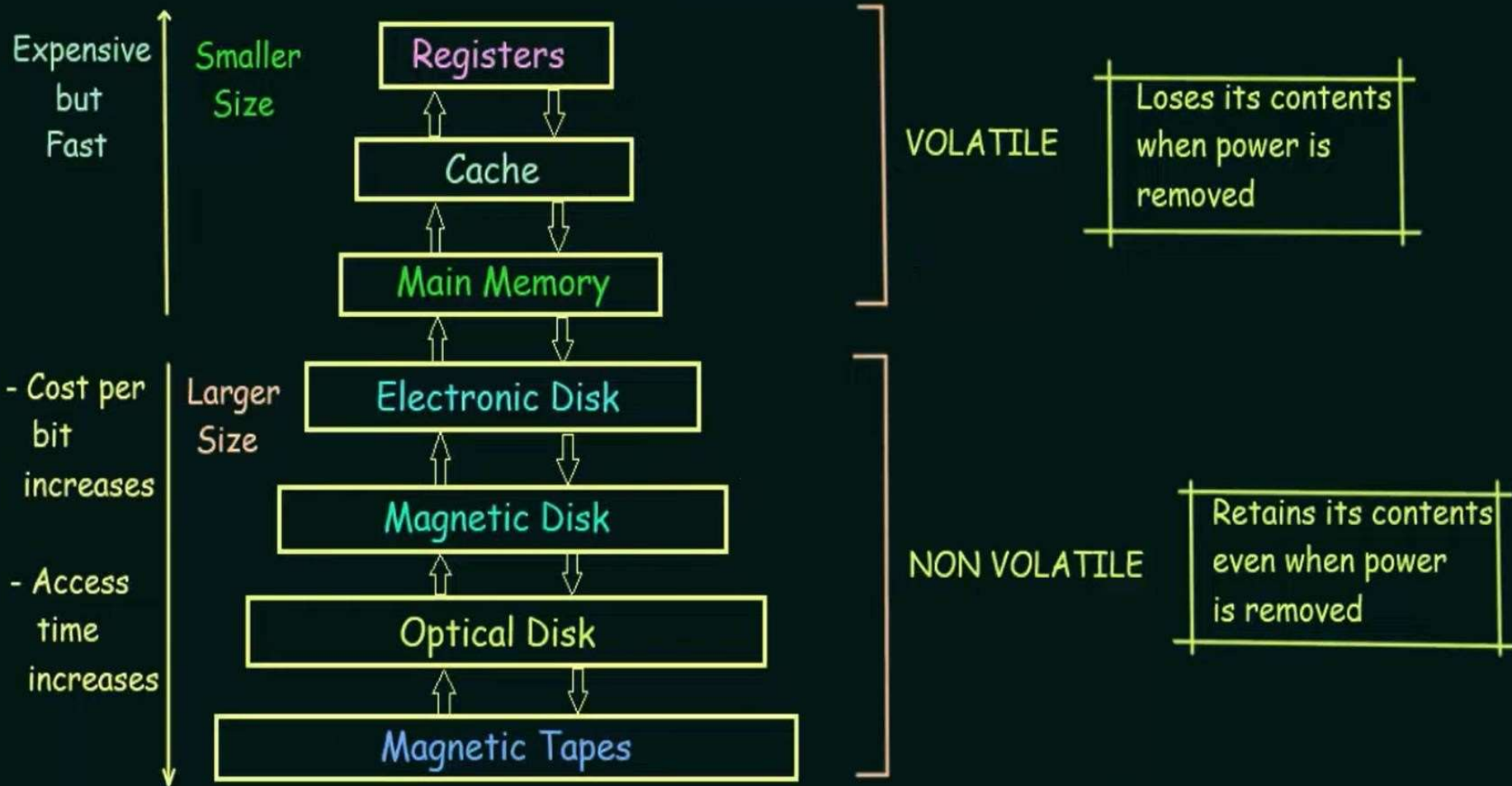
- A modification of the free-list approach stores the addresses of n free blocks in the first free block.
- The first $n-1$ of these blocks are actually free.
- The last block contains the addresses of another n free blocks, and so on.
- The addresses of a large number of free blocks can now be found quickly, unlike the situation when the standard linked-list approach is used.

THANKYOU

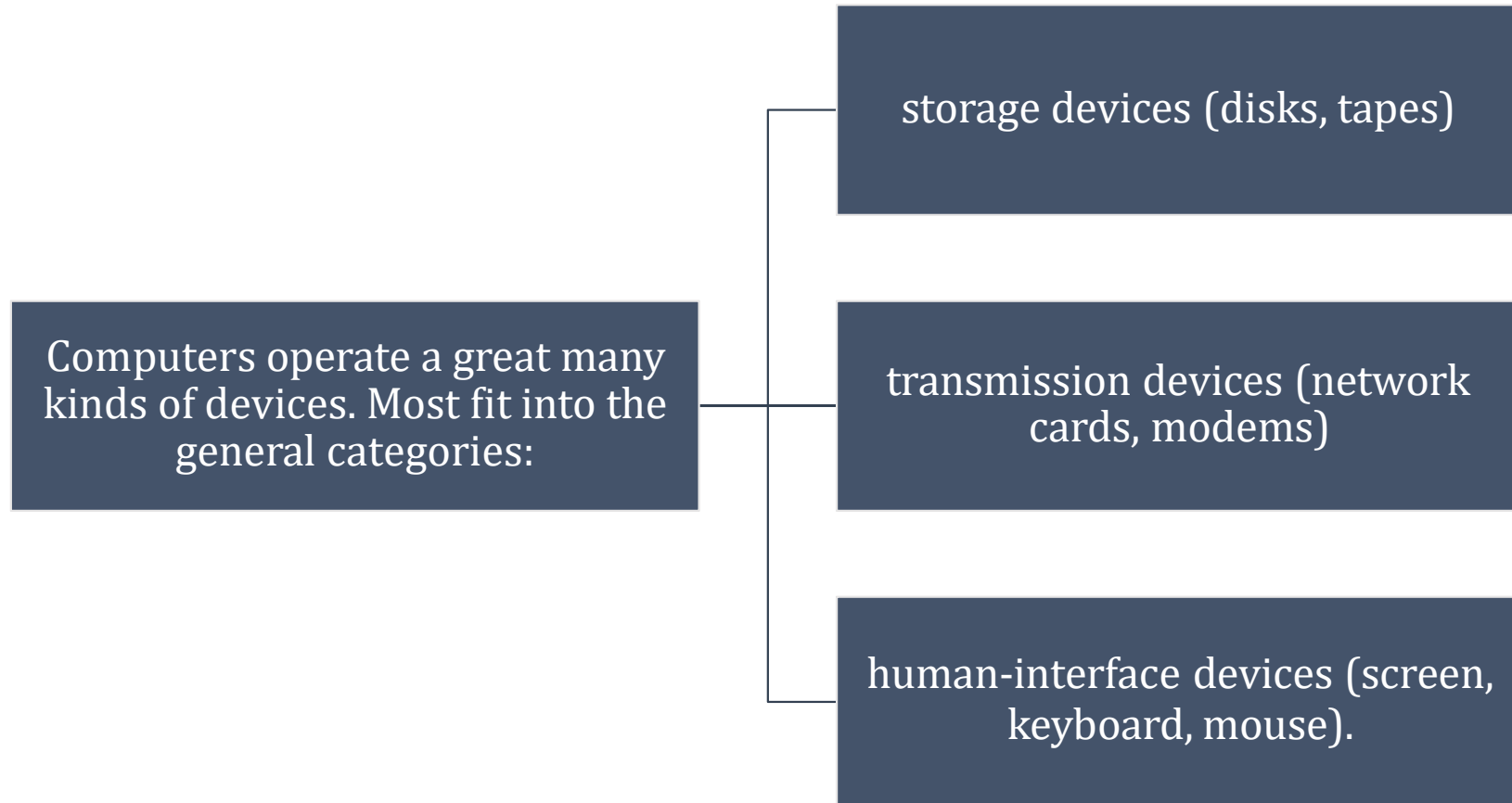
Unit-7

I/O Management

Basics of Operating System (Storage Structure)



Principles of I/O Hardware: I/O device



Principles of I/O Hardware: I/O device

Two types of I/O devices

- block, character

Block device:

- It is one that stores information in fixed-size blocks each one with its own address.
- can read blocks independently of one another
- Hard disks, CD-ROMs, USB sticks
- 512 bytes to 32,768 bytes

Character device:

- accepts characters without regard to block structure
- Printers, mouse, network interfaces

Not everything fits:

- e.g. clocks don't fit

I/O Devices

Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Scanner	400 KB/sec
Digital camcorder	3.5 MB/sec
802.11g Wireless	6.75 MB/sec
52x CD-ROM	7.8 MB/sec
Fast Ethernet	12.5 MB/sec
Compact flash card	40 MB/sec
FireWire (IEEE 1394)	50 MB/sec
USB 2.0	60 MB/sec
SONET OC-12 network	78 MB/sec
SCSI Ultra 2 disk	80 MB/sec
Gigabit Ethernet	125 MB/sec
SATA disk drive	300 MB/sec
Ultrium tape	320 MB/sec
PCI bus	528 MB/sec

Figure : Some typical device, network, and bus data rates.

Device Controllers

I/O unit has two components:

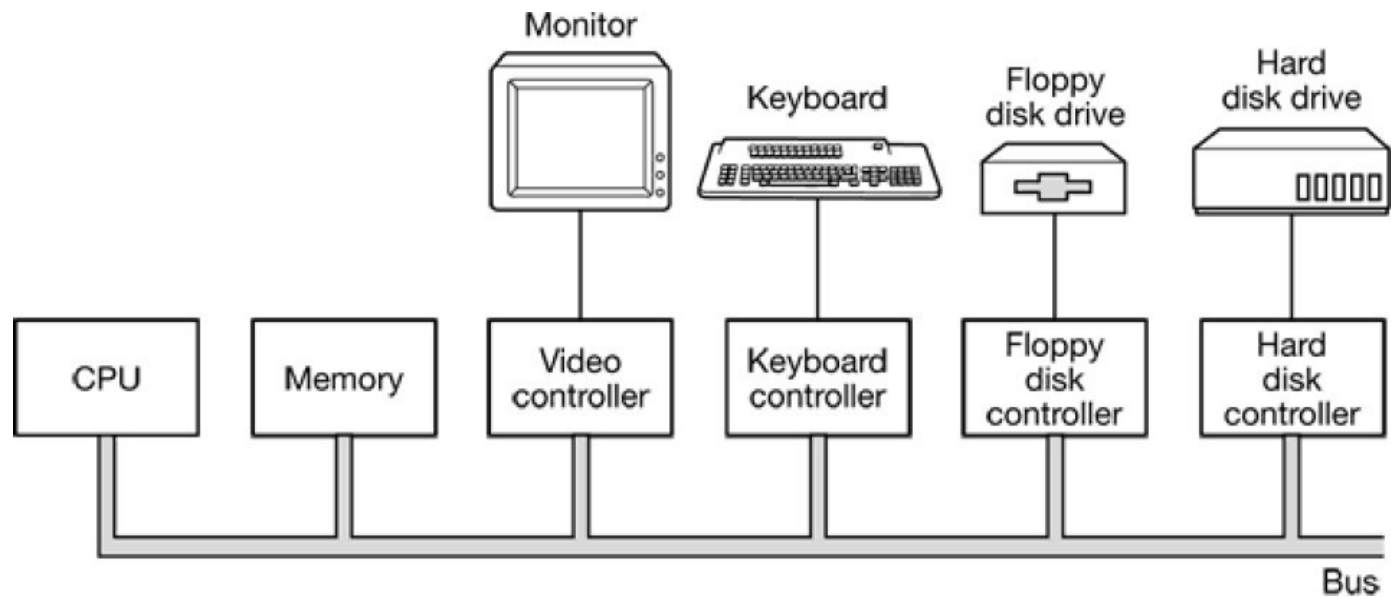
Mechanical components

- Device itself.

Electronic components:

- Known as device controller or adapter .
- The controller card usually has a connector on it, into which a cable leading to the device itself can be plugged.

Device Controller



Device Controllers

Device controller for disk:

- convert the serial bit stream into a block of bytes and perform any error correction:
- The block of bytes is typically first assembled, bit by bit, in a buffer inside the controller. After its checksum has been verified and the block declared to be error free, it can then be copied to main memory or disk.

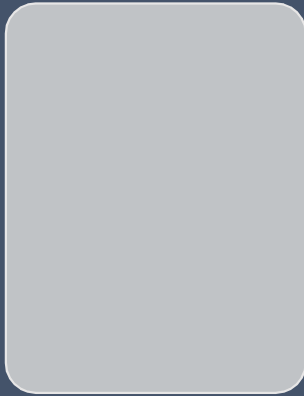
The controller for a monitor:

- It reads bytes containing the characters to be displayed from memory and generates the signals used to modulate the CRT beam to cause it to write on the screen.
- The controller also generates the signals for making the CRT beam do a horizontal retrace after it has finished a scan line, as well as the signals for making it do a vertical retrace after the entire screen has been scanned.
- If it were not for the CRT controller, the operating system programmer would have to explicitly program the analog scanning of the tube.

Direct Memory Access (DMA)



CPU could request data *one byte* at a time from I/O controller, Big waste of time. So other schema can be used DMA.




DMA controller can access system bus independent of CPU. DMA contains several registers that can be written and read by CPU.


- Memory address register
- Byte count register
- Control registers-I/O port, direction of transfer, transfer units (byte/word), number of bytes to transfer in a burst.

How Disk read occurs when DMA is not used??

First the controller reads the block (one or more sectors) from the drive serially, bit by bit, until the entire block is in the controller's internal buffer.



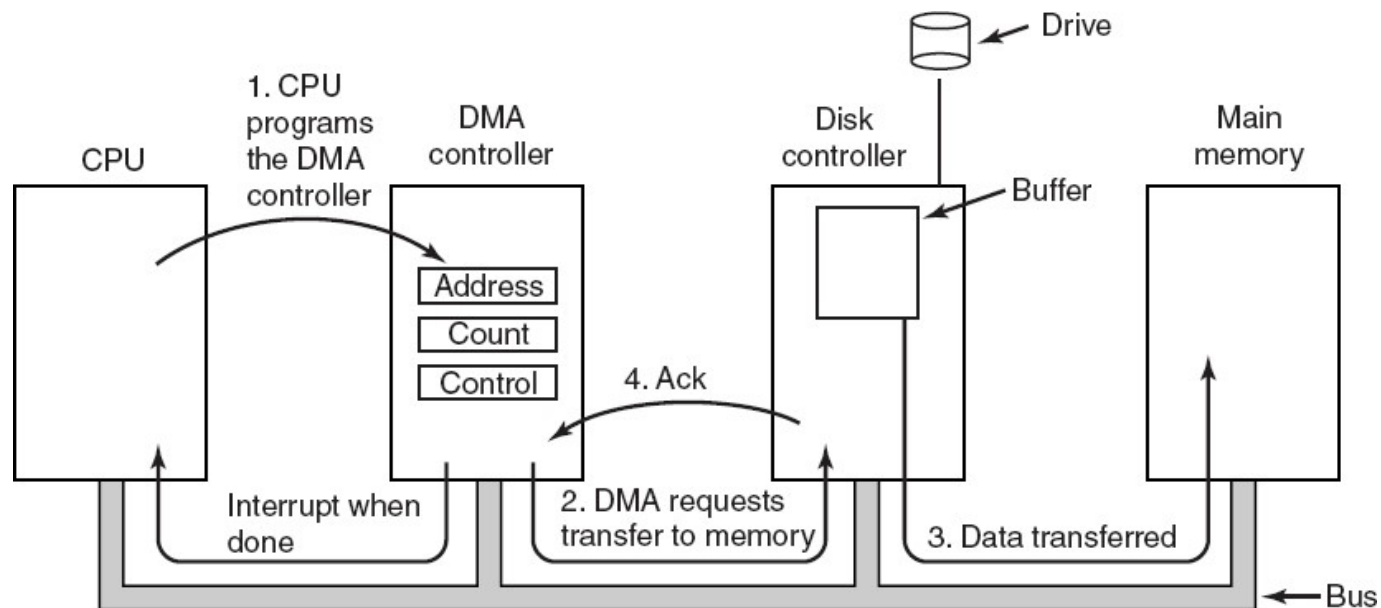
Next, it computes the checksum to verify that no read errors have occurred. Then the controller causes an interrupt.



When the operating system starts running, it can read the disk block from the controller's buffer a byte or a word at a time by executing a loop, with each iteration reading one byte or word from a controller device register and storing it in main memory.


How does DMA work?

- Operation of a DMA transfer.



Direct Memory Access (DMA)

When DMA is used, the procedure is different. First the CPU programs the DMA controller by setting its registers so it knows what to transfer where (step 1).



The DMA controller initiates the transfer by issuing a read request over the bus to the disk controller (step 2).



The write to memory is another standard bus cycle (step 3).



When the write is complete, the disk controller sends an acknowledgement signal to the disk controller, also over the bus (step 4).

DMA controller modes

Cycle stealing mode (word at a time mode)

- the DMA controller requests for the transfer of one word and gets it. If the CPU also wants the bus, it has to wait.
- delaying CPU slightly.

Burst mode (Block mode)

- DMA controller grabs bus and sends a block.
- Here multiple word can be transfer.

Fly by mode

- DMA controller tells device controller to transfer data directly to main memory. Otherwise device controller send the word to DMA controller, which then use bus to write the word.

I/O Software-Goals

- *Shared **devices** (disks) and un-shared devices (tapes) must **be handled***
- *OS needs to make **I/O operations blocking** (e.g. program blocks until data arrives on a read) because it is easy to write blocking operations*

I/O Software Layers

I/O software is typically organized in four layers.

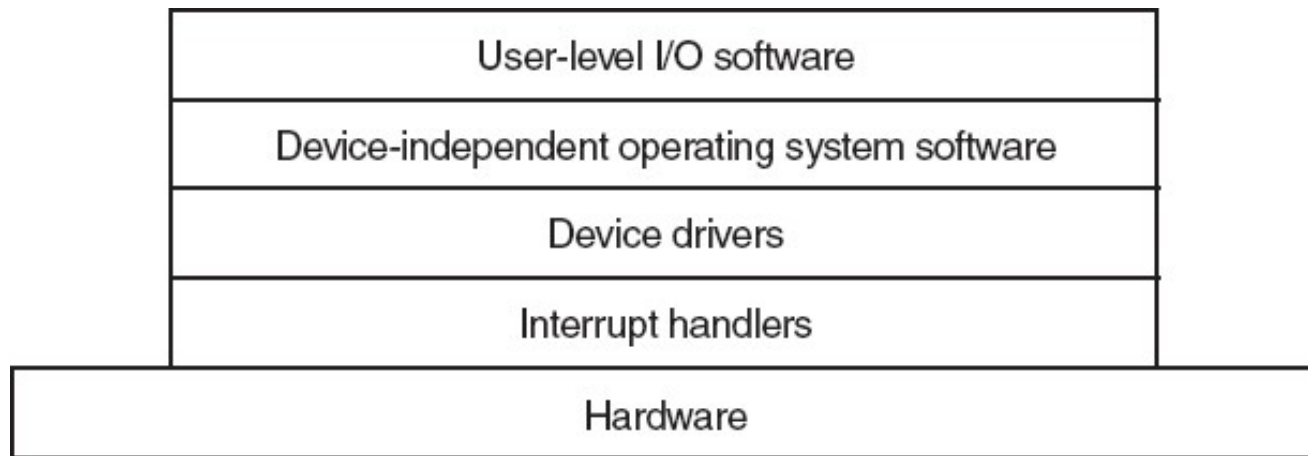


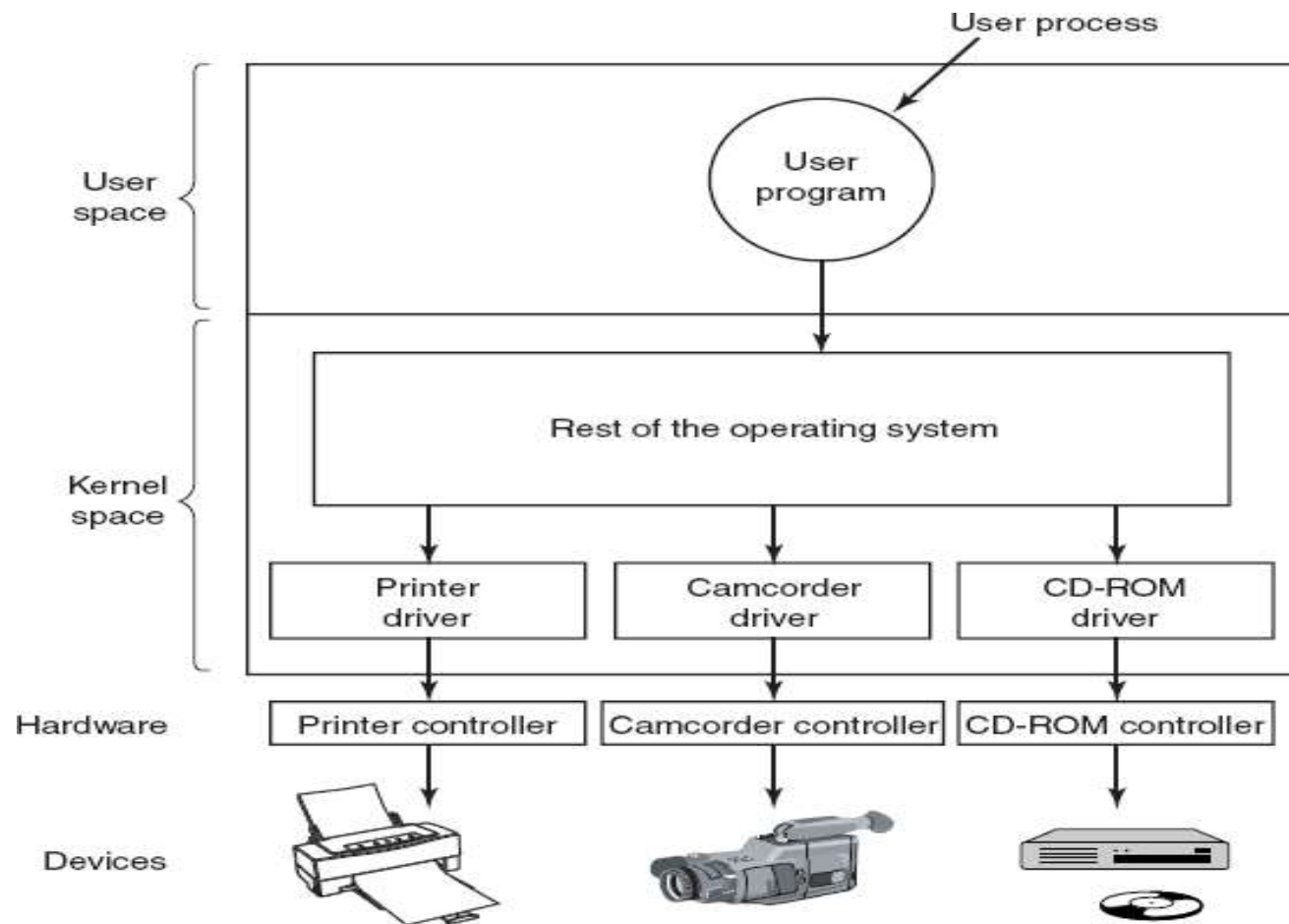
Figure : Layers of the I/O software system.

Device Drivers

Device Drivers are the software through which, the kernel of a computer communicates with different hardware, without having to go into the details of how the hardware works. It also provides a common interface so that the operating system or the Kernel can communicate with the hardware.

Thus, the purpose of device drivers is to allow smooth functioning of the hardware for which it is created and to allow it to be used with different operating systems.

Device Drivers



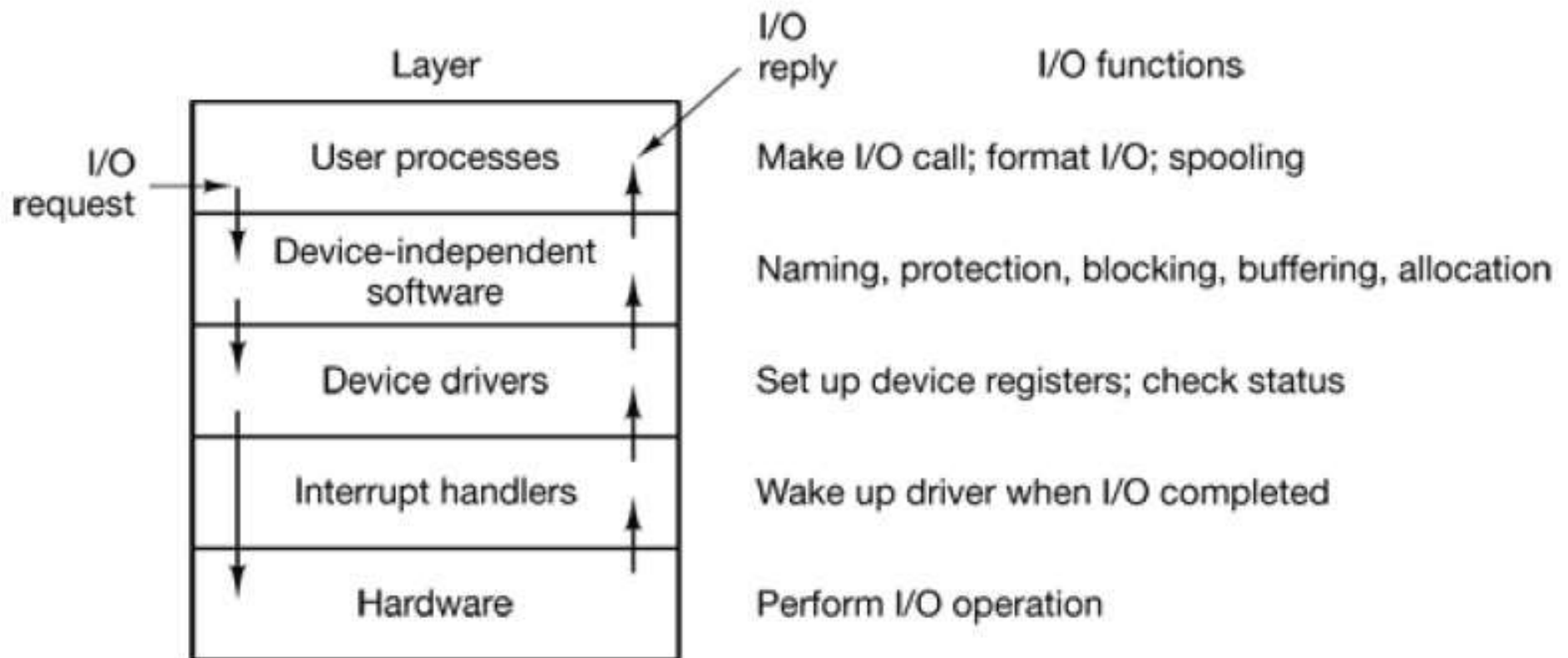
Device-Independent I/O Software

Uniform interfacing for device drivers
Buffering
Error reporting
Allocating and releasing dedicated devices
Providing a device-independent block size

User Space I/O Software

- Library routines are involved with I/O- printf, scanf, write for example. These routines makes system calls
- Spooling-is a way of dealing with dedicated IO devices
- Spooling systems-keep track of device requests made by users.
- Think printing.
 - User generates file, puts it in a spooling directory.
 - Daemon process monitors the directory, printing the user file
- File transfers also use a spooling directory

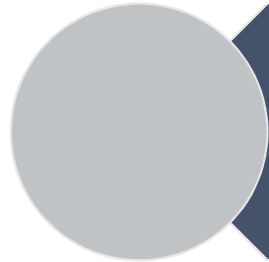
Layers of I/O Software & Function of Each Layer



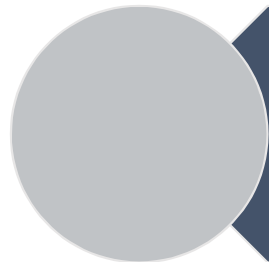
Disks



It is relatively *permanent* and can hold *large quantities* of data. Its *access time is slow* compared with that of main memory.



Disk platter: has a flat circular shape, like a CD. The two surfaces of a platter are covered with a magnetic material.



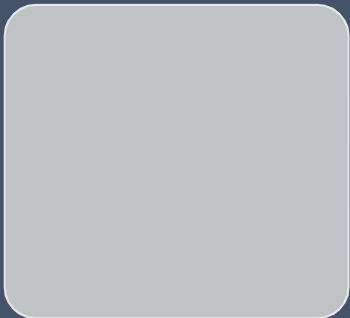
The surface of a platter is logically divided into circular *tracks* which are subdivided into *sectors*. The set of tracks that are at one arm position makes up a *cylinder*.

Disks



Disk speed has two parts.

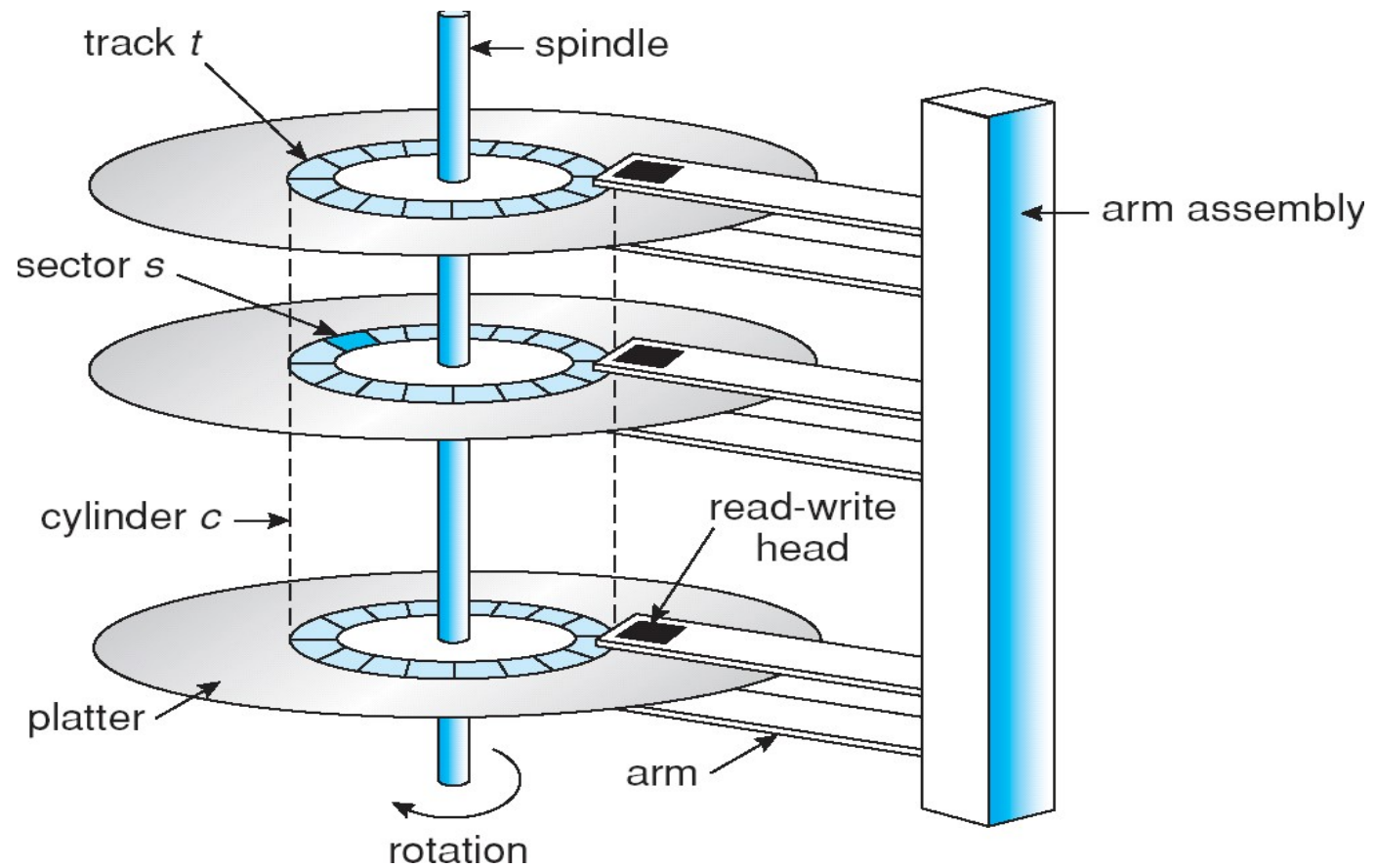
- ***transfer rate***: is the rate at which data flow between the drive and the computer.
- ***positioning time / Seek time***: sometimes called random access time the consists of the time necessary to move the disk arm to the desired cylinder.



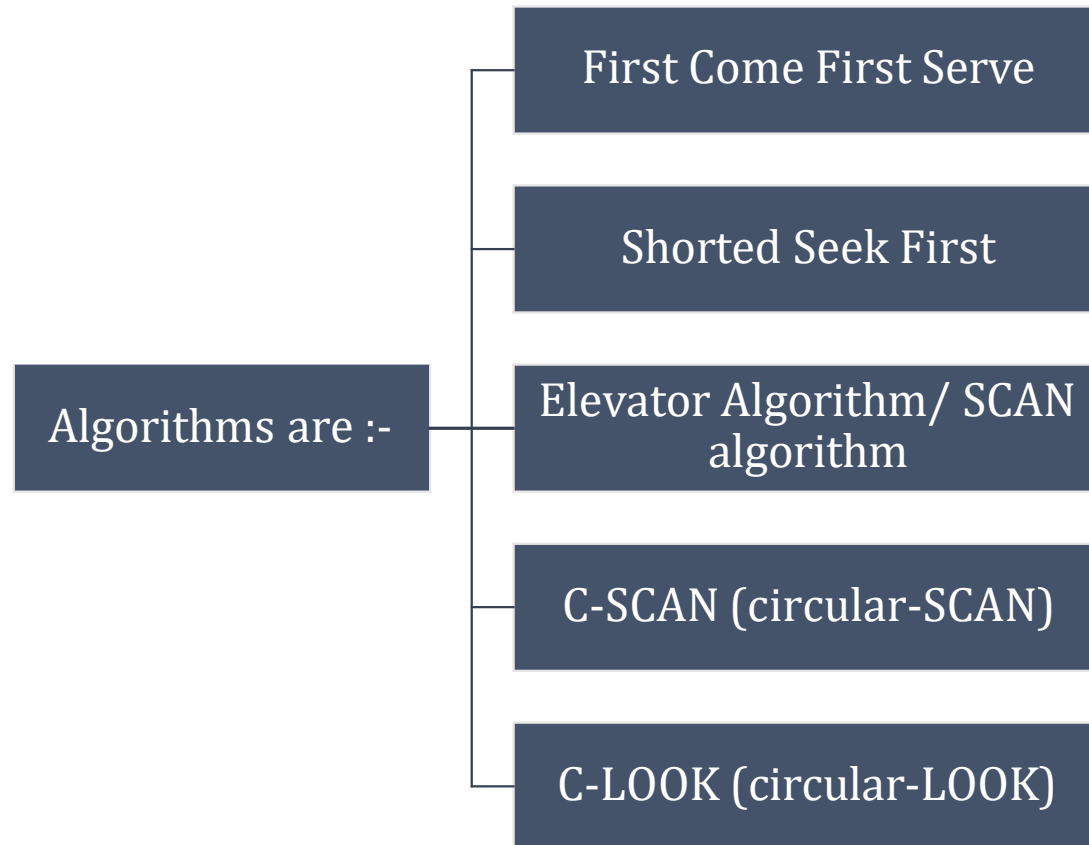
rotation latency:

- the time necessary for the desired sector to rotate to the disk head.
- **Bad sector** in computing refers to a disk sector on a disk storage unit that is permanently damaged. Upon taking damage, all information stored on that sector is lost. When a bad sector is found and marked, the operating system skips it in the future.

Moving-head Disk Mechanism



Disk Arm Scheduling Algorithms



First Come First Serve (FCFS)

- Handle I/O requests sequentially.
- Fair to all processes.

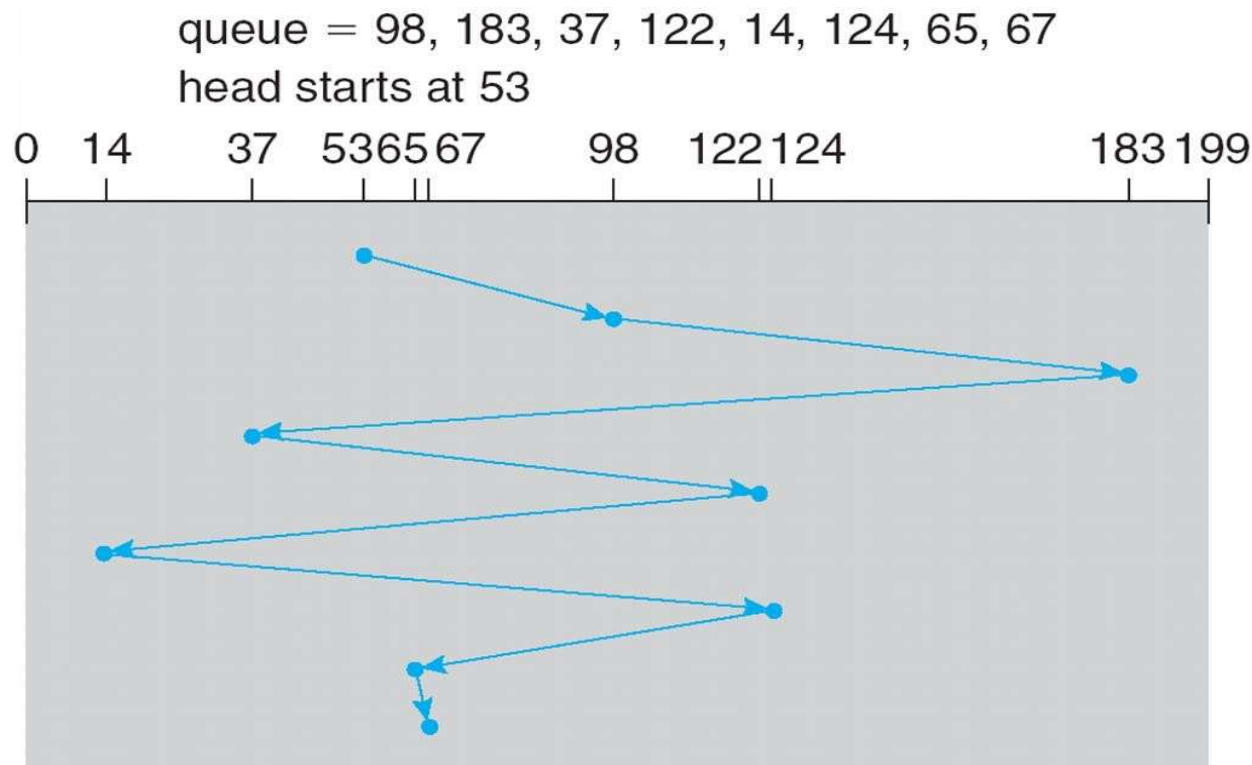
First Come First Serve (FCFS)

- We have disk queue with requests for I/O to block on cylinders(cylinders are between 0-199):

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53

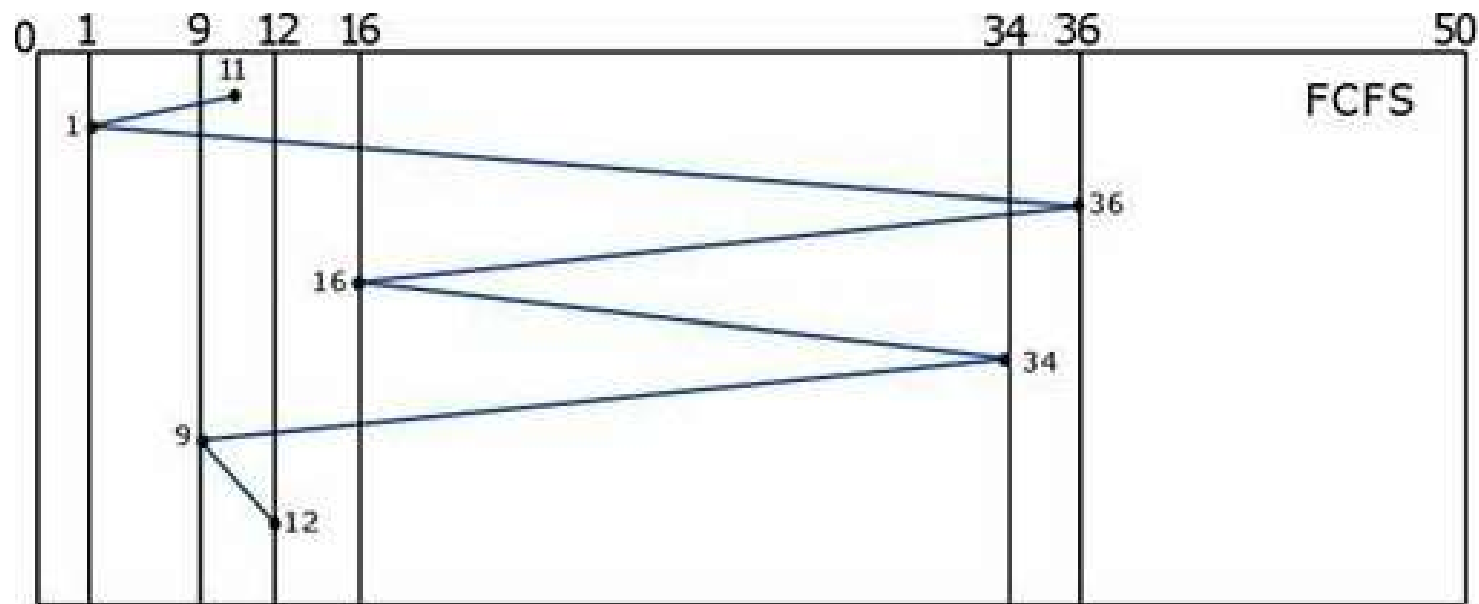
First Come First Serve (FCFS) Example



Total cylinder movement: $|53-98| + |98-183| + |183-37| + |37-122| + |122-14| + |14-124| + |124-65| + |65-67|$

Illustration shows total head movement or seek distance is 640 cylinders.

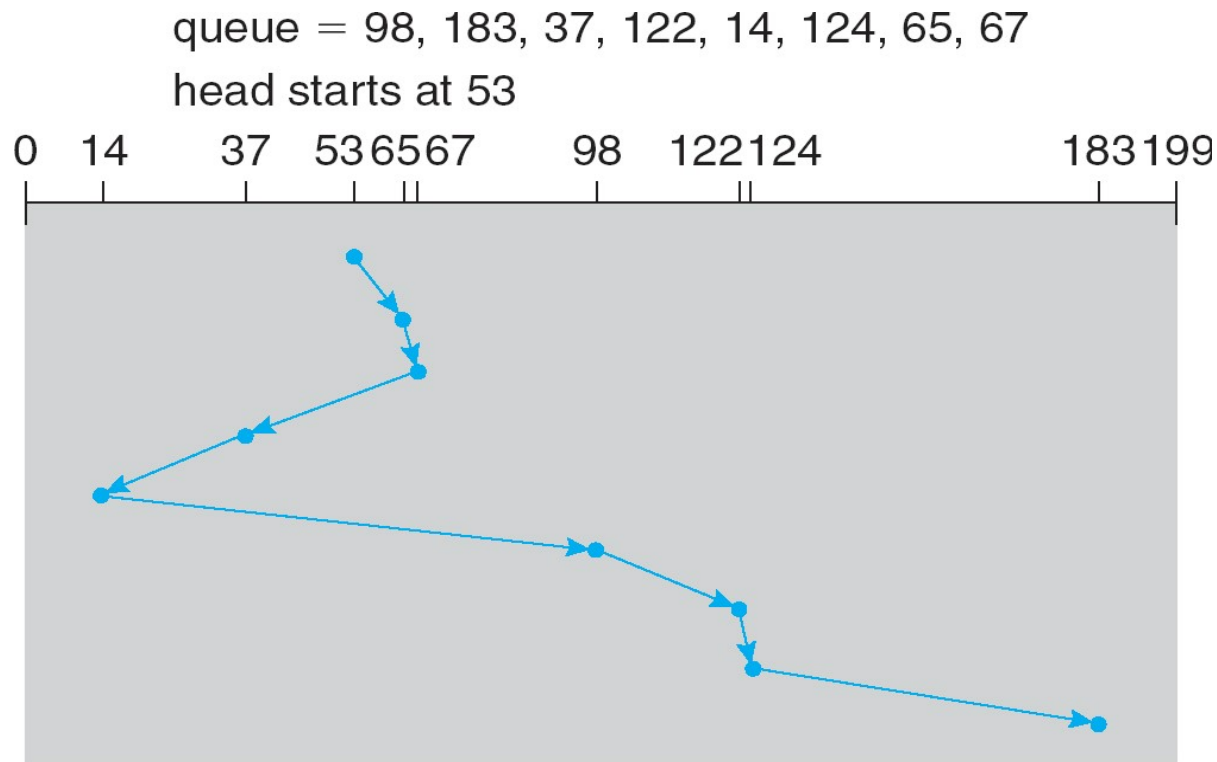
- Example:
New requests come in for cylinders 1, 36, 16, 34, 9, and 12 Starting from the current head position 11, what is the total distance (in cylinders)



Shortest Seek Time First (SSTF)

- Selects the request with the minimum seek time from the current head position.
- Also called Shortest Seek Distance First (SSDF) – It's easier to compute distances.
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests.

SSF (Shortest Seek Time First)

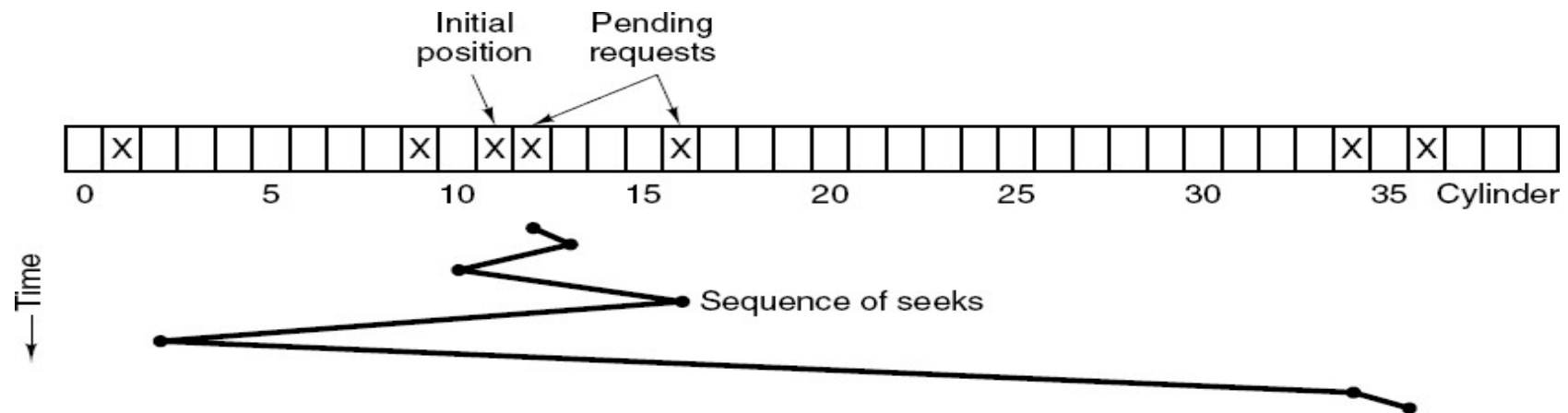


- Illustration shows total head movement of 236 cylinders.

SSF (Shortest Seek Time First)

Example:

New requests come in for cylinders 1, 36, 16, 34, 9, and 12 Starting from the current head position 11, what is the total distance (in cylinders)



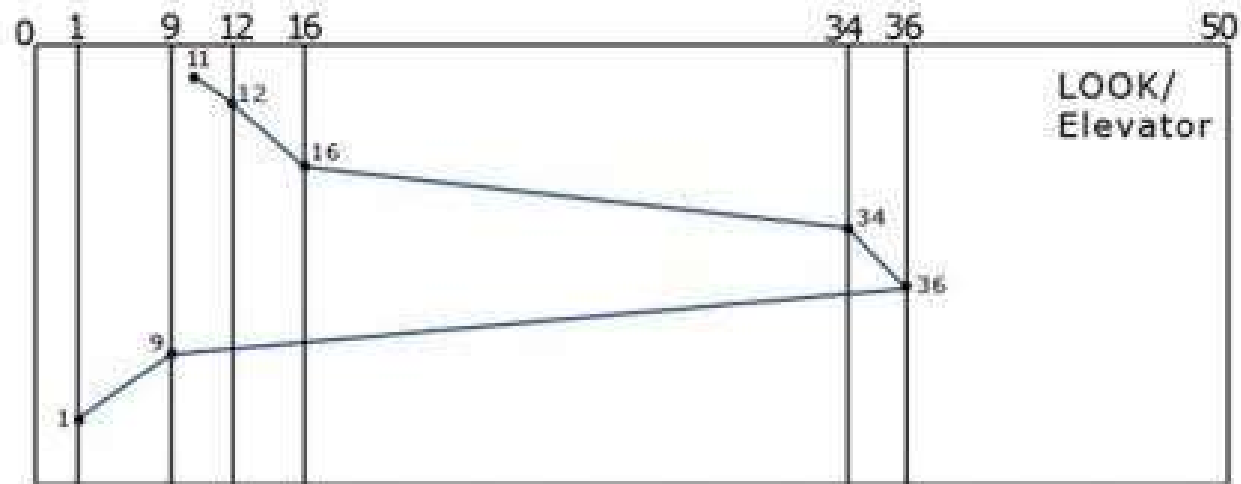
- While head is on cylinder 11, requests for 1,36,16,34,9,12 come in
- FCFS would result in total of 111 cylinders
- SSF would require Total cylinder movement: $(12-11) + (12-9) + (16-9) + (16-1) + (34-1) + (36-34)$
- $1+3+7+15+33+2 = 61$ cylinders

Elevator algorithm

- Keep moving in the same direction until there are no more outstanding requests pending in that direction, then algorithm switches direction.
- After switching the direction the arm will move to handle any requests on the way.

The Look/Elevator

- 11,1,12,36,16,34,9
- Uses 60 cylinders, a bit better



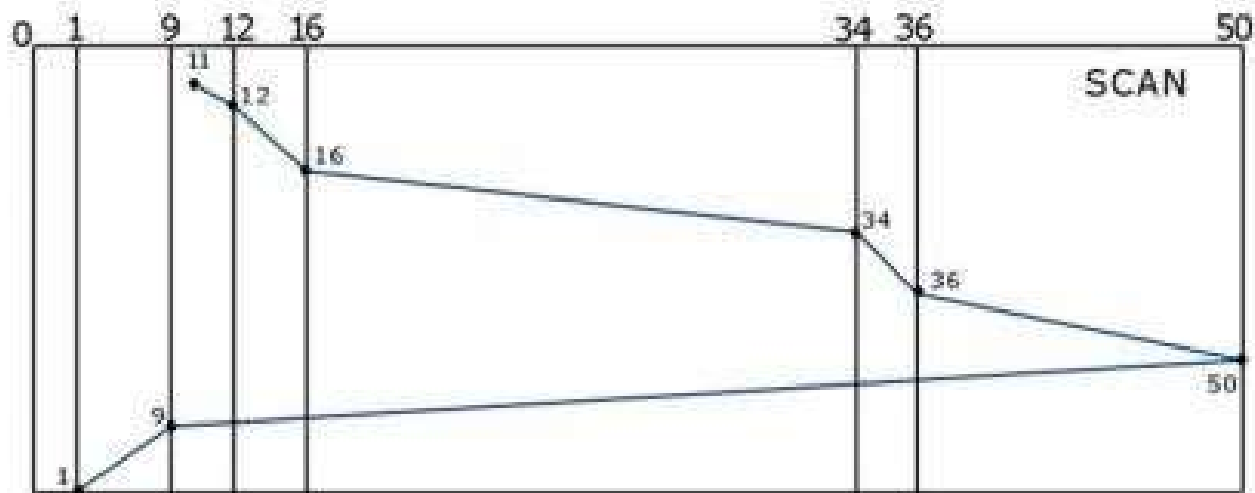
Scan Algorithm

- From the current position disk arm starts in up direction and moves towards the end, serving all the pending requests until end.
- At the end arm direction is reversed(down) and moves towards the other end serving the pending requests on the way.

Scan algorithm

Cylinder are between 0 - 50

11,1,12,36,16,34,9



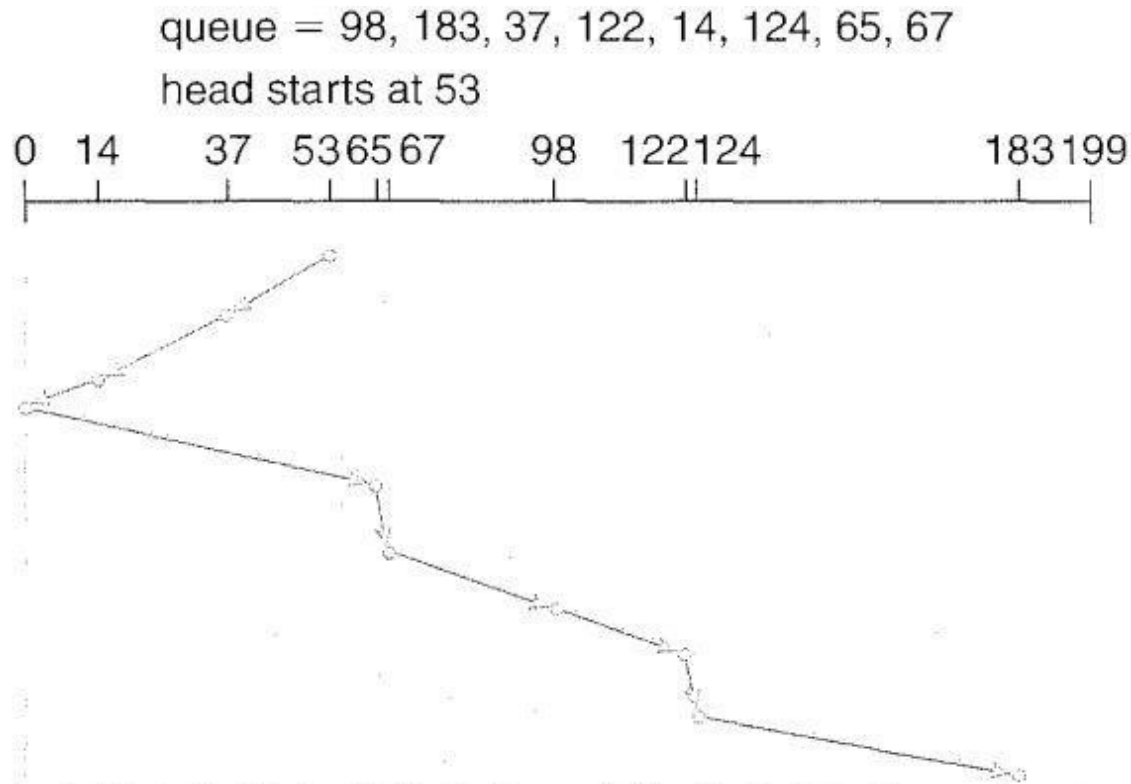
Uses 88 cylinders

Example:

Cylinder are between 0-199

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53

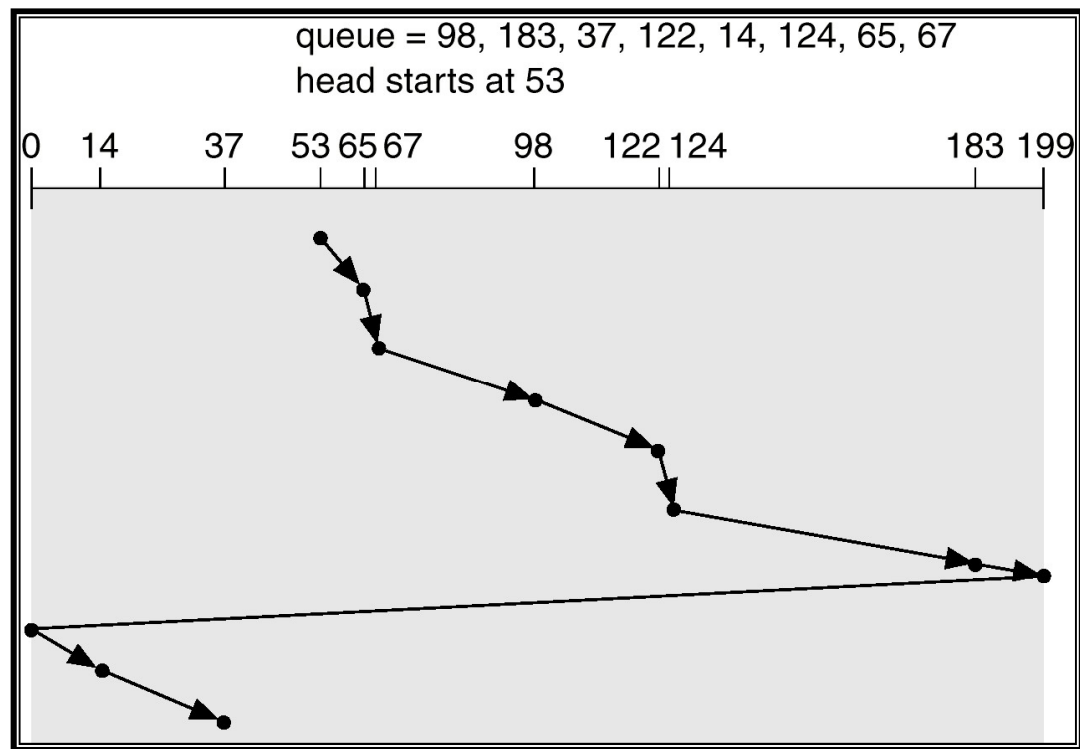


- Answer is 331 cylinders with up direction
- Answer is 236 cylinder with down direction
- Note that head reaches one end and reverse direction

C-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.
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one.

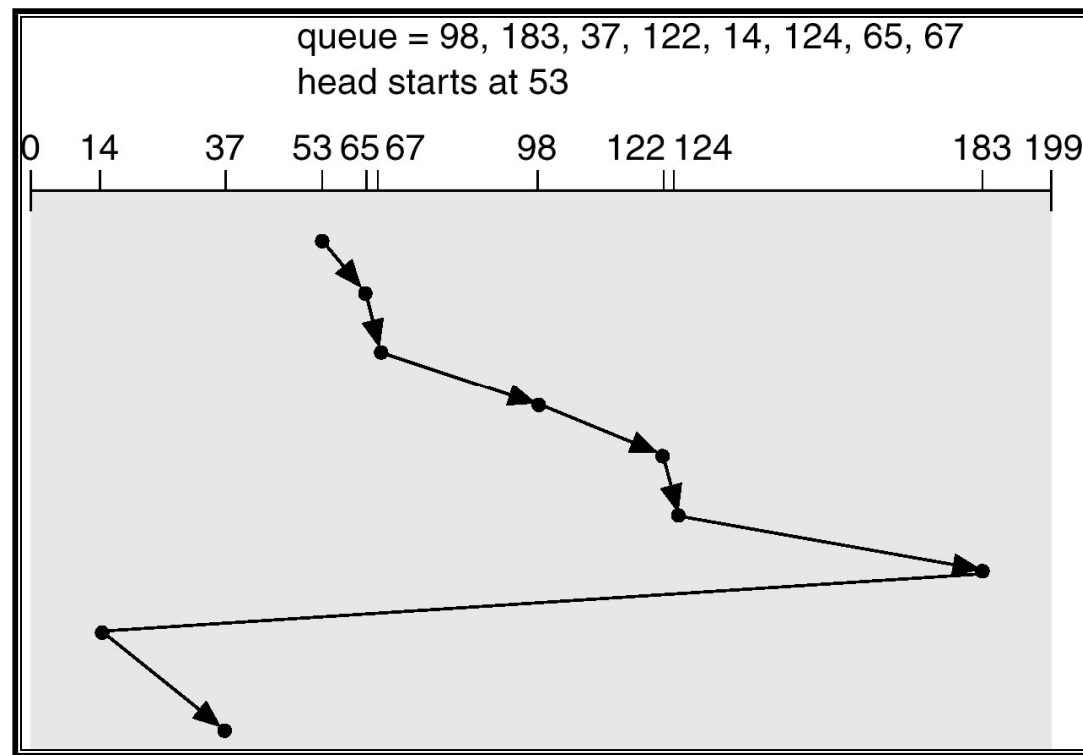
C-SCAN (Cont.)



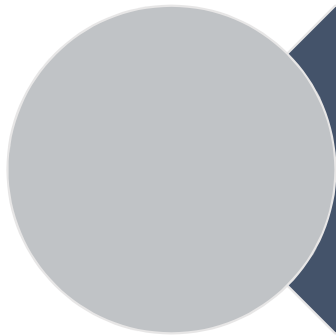
C-LOOK

- Version 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.

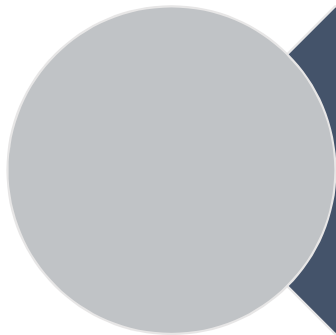
C-LOOK (Cont.)



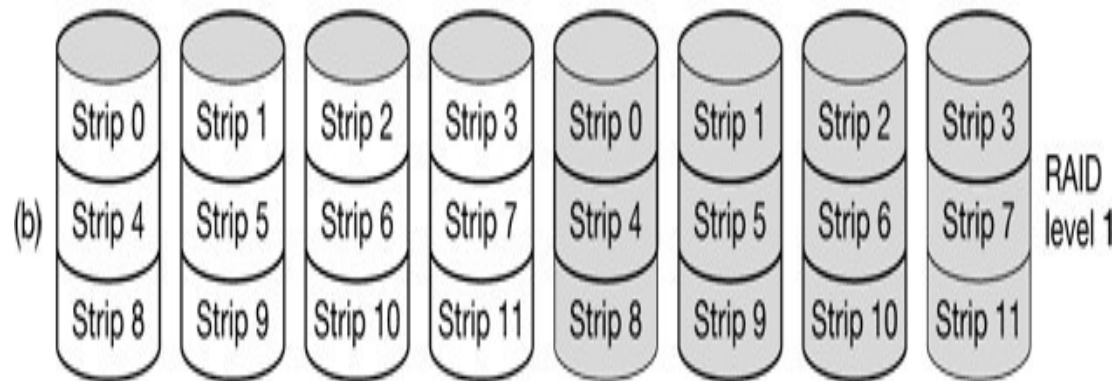
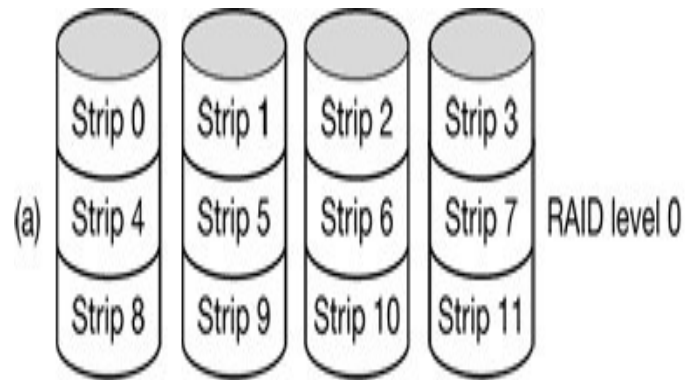
RAID model



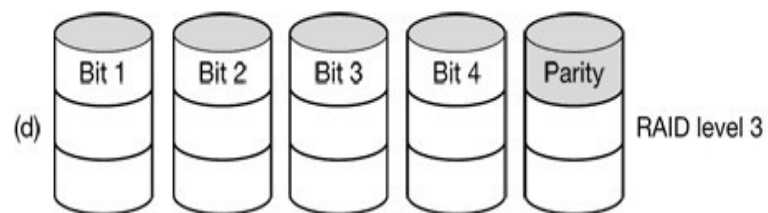
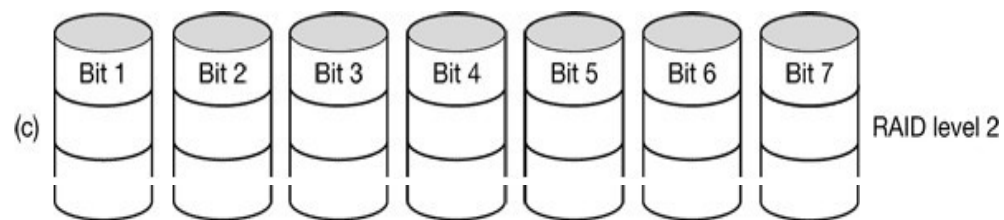
Redundant Array of Independent Disks.



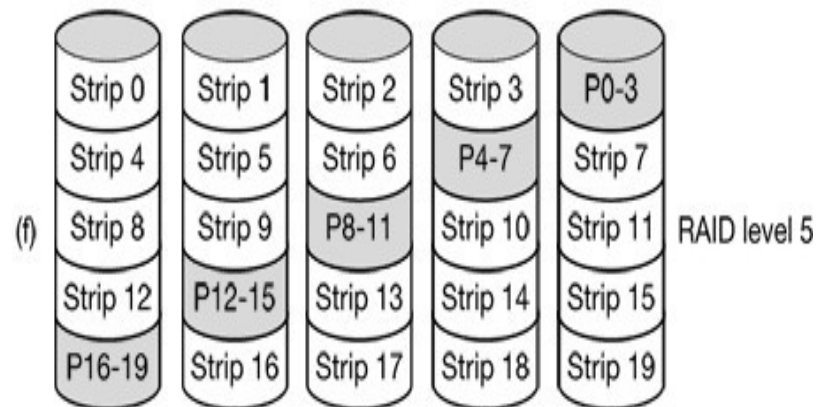
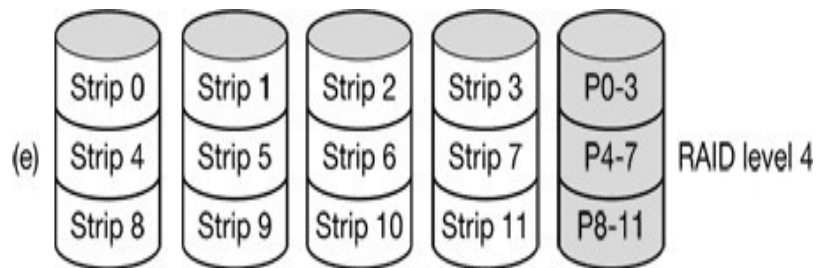
Parallel processing is used more and more
to increasing speed of the CPU.



- RAID 0:
 - stripping
 - Data is split up into blocks.
- RAID 1:
 - Mirroring
 - Data are stored twice.
 - Offers excellent read and write speed.



- RAID 2:
 - Unlike level 0 and 1 level 2 work on word basis.
 - Splitting each byte and then add humming code into this bytes.
 - Loss of the drive only lost 1 byte.
- RAID 3:
 - Here will use single parity bit.
 - Individual data word are spread over multiple drives.



- RAID Level 4:
 - work with strips again,
 - If drive crashes, the lost byte can be recomputed from parity drive.
 - Even if one drive change we have to recomputed parity.
- RAID level 5:
 - Single parity bottleneck is eliminated.
 - By distributing parity bit uniformly over disk.

- **Reference Books:**

1. Operating Systems: Internals & Design Principles, 8th Edition, William Stallings, Pearson Education India
2. Operating System Concepts, 9th edition Peter B. Galvin, Greg Gagne, Abraham Silberschatz, John Wiley & Sons, Inc.
3. Modern Operating Systems-By Andrew S. Tanenbaum (PHI)