File System

* A file is a collection of related information that is recorded on secondary storage.
* File is a collection of logically related entries.
* A text file is a sequence of characters organized into lines.
* A source file is a sequence of functions, each of which is further organized as declarations followed by executable stmts.
* An executable file is a series of code sections that the loader can bring into memory & execute.

File attributes

* Name :- The symbolic file name is the only info kept in human-readable form.
* Identifier :- This unique Tag, usually a number, identifies the file within the file s/m; it is the non-human readable name for the file.
* Type :- This information is needed for s/ms that support diff types of files.
* Location :- This information is a pointer to a device & to the location of the file on that device.

* Size :- The current size of the file (in bytes, words or blocks) & possibly the maximum allowed size are included in this attribute.

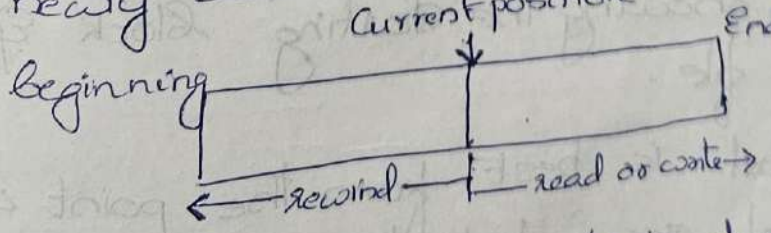
* Protection :- Access-control info^n determines who can do reading, writing, executing and so on.

* Time, date & User identification :- This info^n may be kept for creation, last modification & last use. These data can be useful for protection, security & usage monitoring.

File Access Methods

1. Sequential Access :-

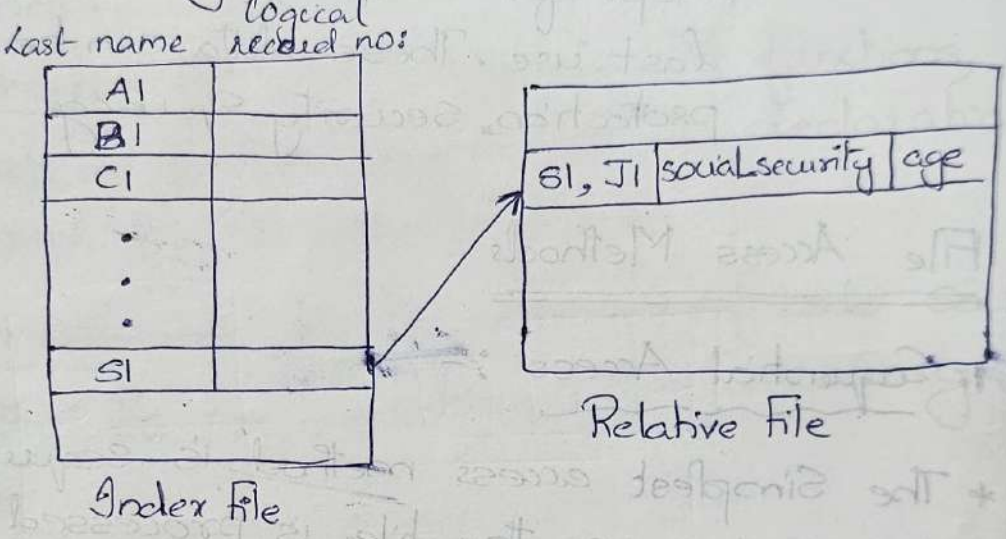
* The simplest access method is sequential Access.
* Information in the file is processed in order, one record after the other.
* Reads & writes make up the bulk of the operation on a file.
* A read oper^n _ read_next() → reads the next portion of the file & automatically advances a file pointer, which tracks the I/O location.
* Similarly, the write oper^n _ write_next() → appends to the end of the file and advances to the end of the newly written material (the new end of file).



↳ is done based on the pointer current position.

2. Index Access :-

* The index, like an index in the back of a book, contains the pointer to the various blocks.
* To find a record in the file, we first search the index & then by the help of pointer we access the file directly.

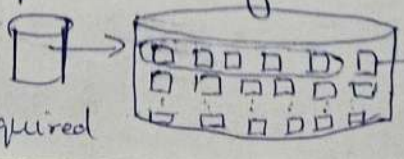


File Allocation Methods

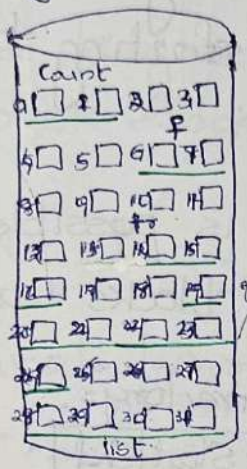
1. Continuous Allocation :-

* A single continuous set of blocks is allowed to a file at the time of file creation.
* Thus, this is a pre-allocation strategy, using variable size portions.
* The file allocation table (FAT) needs just a single entry for each file, showing the starting block & the length of the file.
* This method is best from the point of view of the individual sequential file.

Eg: 50KB file.



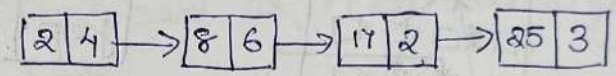
* Multiple blocks can be read in at a time to improve I/O performance for sequential processing.
* It is also easy to retrieve a single block.
* For eg, if a file starts at block b & the ith block of the file is wanted, its location on secondary storage is simply b+i-1.



directory

File	Start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

(FAT)

∴ free List => 

If a file named 'Test' requires 8 blocks, it cannot be allocated since there is no continuous free 8 blocks available.

∴ External & Internal Fragmentation occurs in Continuous Allocation.

Disadvantage (compaction)

If we need a particular 3rd block of mail file.
we can get it by $b + i - 1$
(b: starting position of file mail; i: Required block no.)

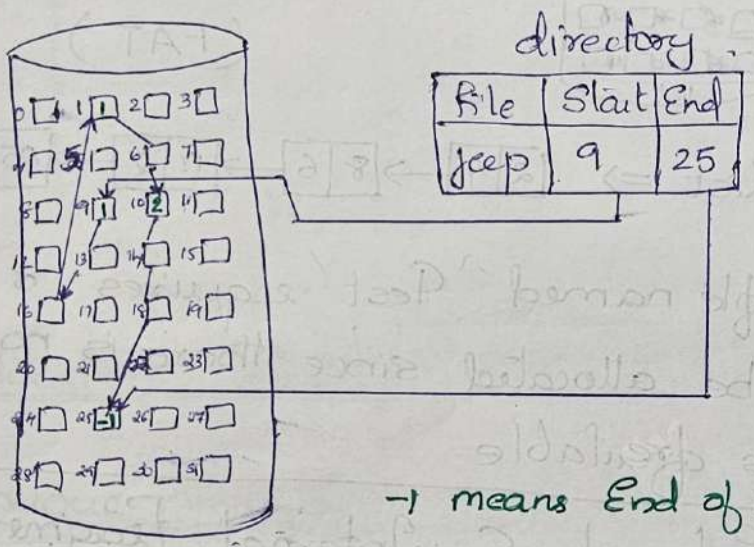
∴ $b = 19$ $i = 3$

∴ $b + i - 1 = 19 + 3 - 1$

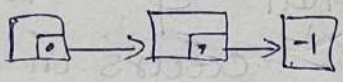
$= \underline{\underline{21}}$

2. Linked Allocation :- (Chained Allocation).

* Allocation is on an individual block basis.
* Each block contains a pointer to the next block in the chain.
* Again the file table needs just a single entry for each file showing the starting block and the length of the file.
* Although pre-allocation is possible, it is more common simply to allocate blocks as needed.



-1 means End of file.



→ Each block requires pointer & if a block miss pointer other blocks cannot be accessed.

No External fragmentation

↳ Since at a time single block of my is allocated.

Internal Fragmentation Occurs

↳ If the last block requires only 5KB, then the remainning 5KB space is unallocated.

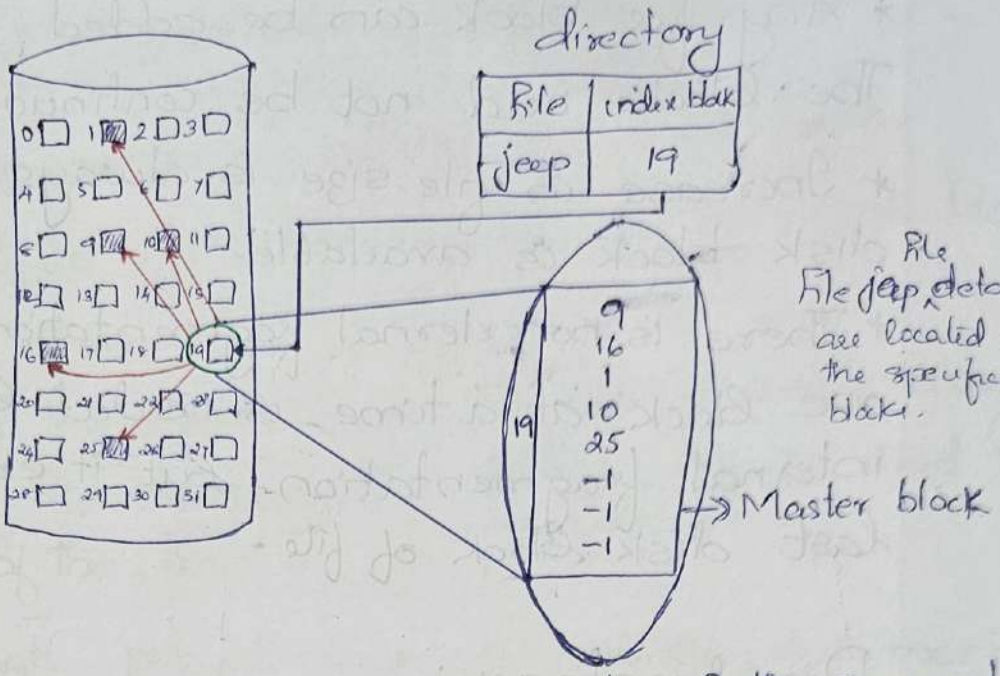
* Any free block can be added to the chain. The blocks need not be continuous.
* Increase in file size is always possible if free disk block is available.
* There is no external fragmentation because only one block at a time is needed but there can be internal fragmentation but it exists only in the last disk block of file.

Disadvantages :-

* Internal fragmentation exists in last disk block of file.
* There is an overhead of maintaining the pointer in every disk block.
* If the pointer of any disk block is lost, the file will be truncated.
* It supports only the sequential access of files.

3. Indexed Allocation :- { Mostly Used }

* It addresses many of the pblms of contiguous and chained allocation.
* In this case, the FAT contains a separate one-level index for each file: The index has one entry for each block allocated to the file.
* Allocation may be on the basis of fixed-size blocks or variable-sized blocks.
* Allocation by blocks eliminates external fragmentation, whereas allocation by variable-size blocks improve the locality.



* This allocation technique supports both sequential & direct access to the file and thus is the most popular form of file allocation.

If b blocks needed.

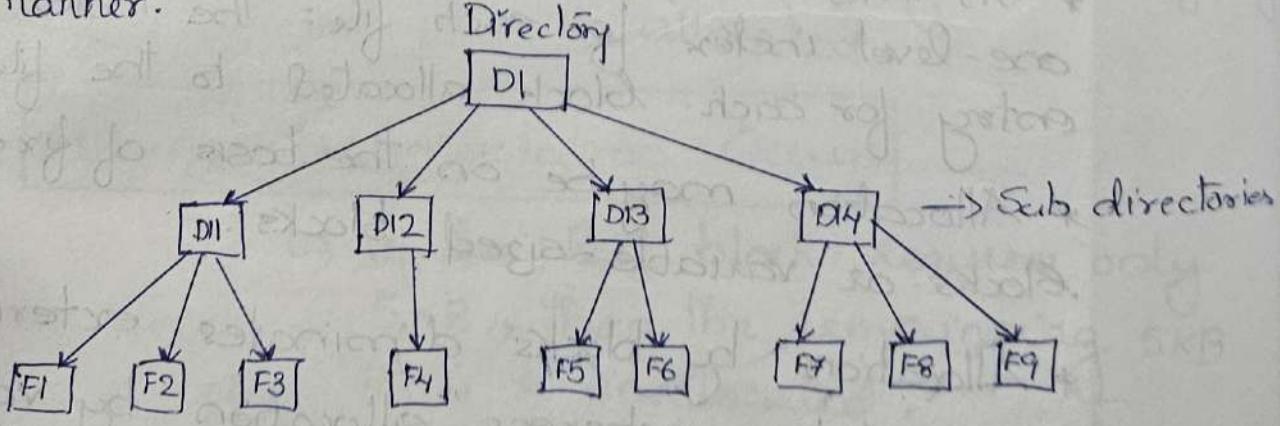
total (b+1) blocks for a file

↳ index block.

Directory Structure

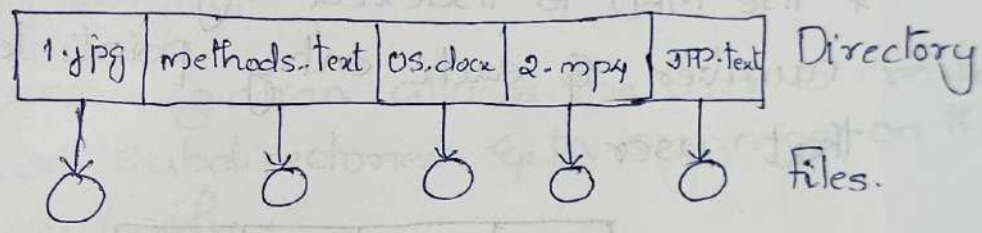
* A directory is a container that is used to contain folders and file.

* It organizes files and folders into hierarchical manner.



1. Single-level directory:

* Simplest directory structure.
* In it all files are contained in same directory which make it easy to support & understand.
* Limitation :- When the no: of files increases or when the s/m has more than one user. Since all the files are in the same directory, they must have the unique name. If 2 users call the dataset, then the unique name rule violated.
* Since all the files are in the same directory, they must have the unique name.



Advantages:

* Since it is a single directory, so its implementation is very easy.
* If files are smaller in size, searching will faster.
* The operations like file creation, searching, deletion, updating are very easy in such a directory structure.

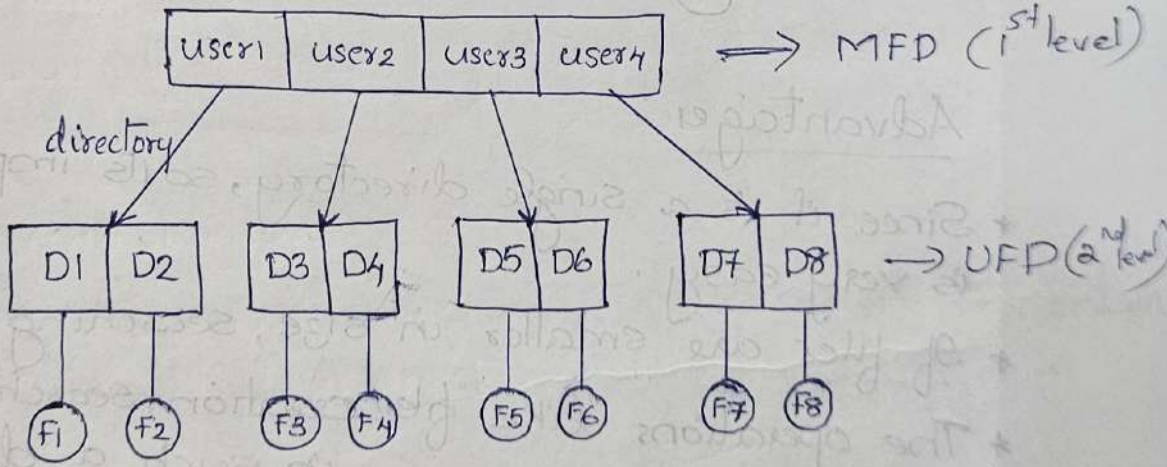
Disadvantages :-

* There may chance of name collission because 2 files cannot have the same name.
* Searching will become time taking if directory will large.

* In this cannot group the same type of files together.

2. Two-level directory: (height 2)

* Each user has their own user files directory (UFD).
* The UFDs has similar structures, but each lists only the files of a single user.
* System's master file directory (MFD) is searched whenever a new user id's logged in.
* The MFD is indexed by username or account number & each entry points to the UFD for that user.



Advantages:-

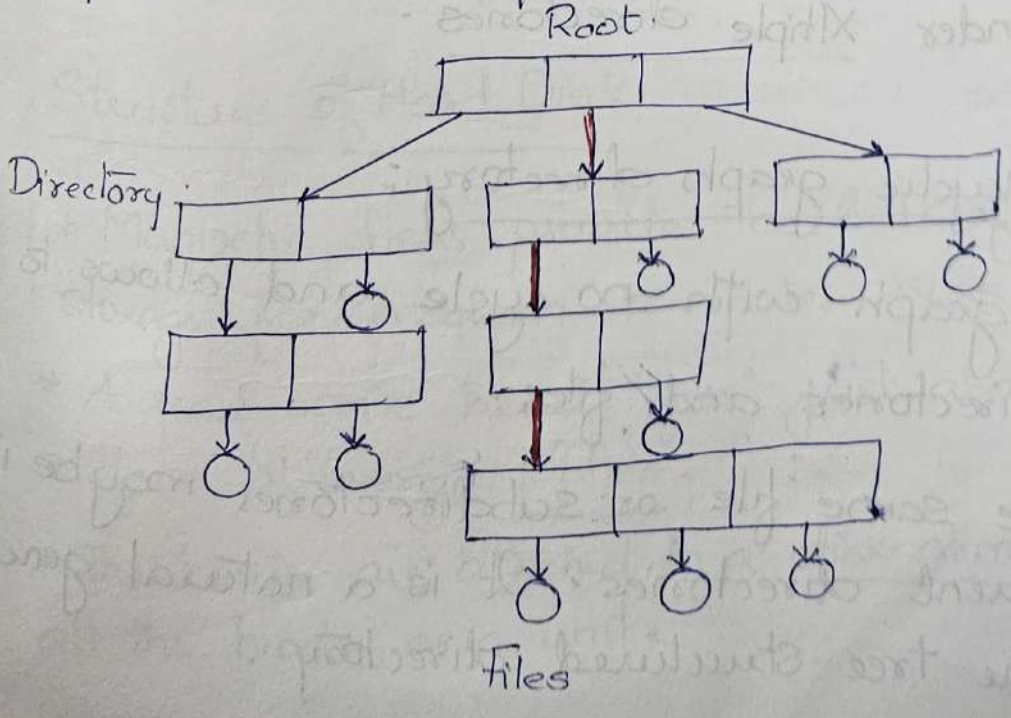
* We can give full path like /username/directoryname/
* Different users can have same directory as well as file name.
* Searching of files become easier due to path name and user-grouping.

Disadvantages :-

* A user is not allowed to share files with other users.
* Still, it is not very scalable, 2 files of the same type cannot be grouped together in the same user.

3. Tree Structure directory : (height ∞)

* Once we have seen a two-level directory as a tree of height 2, the natural generalization is to extend the directory structure to a tree of arbitrary height.
* This generalization allows the user to create their own subdirectories & to organize on their files accordingly.
* A tree structure is the most common directory structure.
* The tree has a root directory & every file in the S/m have a unique path.



Single path to access a file.

Eg: /.../.../../..

Advantages

* Very generalize, since full path name can be given.
* Very scalable, the probability of name collission is less.
* Searching becomes very easy, we can use both absolute path as well as relative.

Eg: c:/Program files / Java / bin / javac.exe
↳ Absolute Path.

$Home : C:/Program files / Java

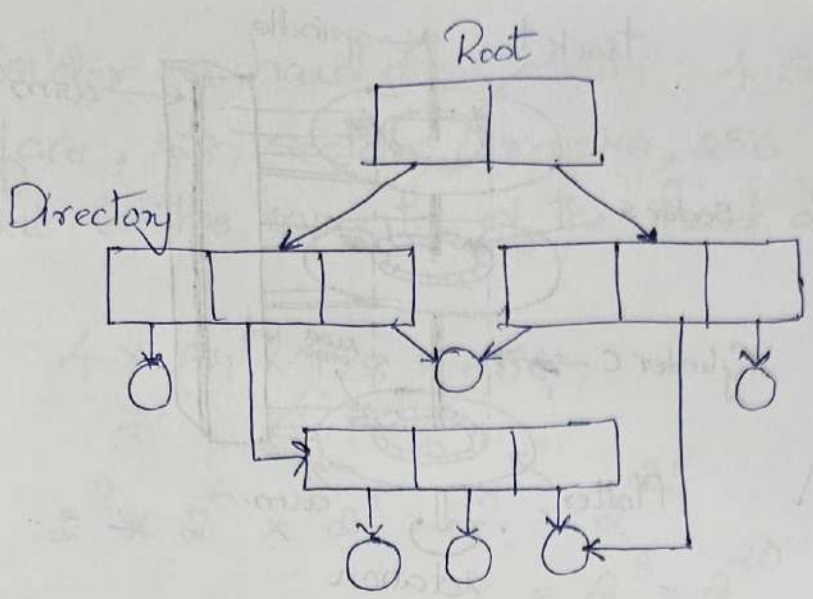
/bin/javac.exe
↳ Relative Path

Disadvantages :-

* Every file doesnot fit into the hierarchical model, files may be saved into Xltiple directories.
* We cannot share files.
* It is inefficient because accessing a file may go under Xltiple directories.

4. Acyclic graph directory :

* A graph with no cycle and allows to share subdirectories and files.
* The same file or subdirectories may be in two different directories. It is a natural generalization of the tree-structured directory.



Advantages

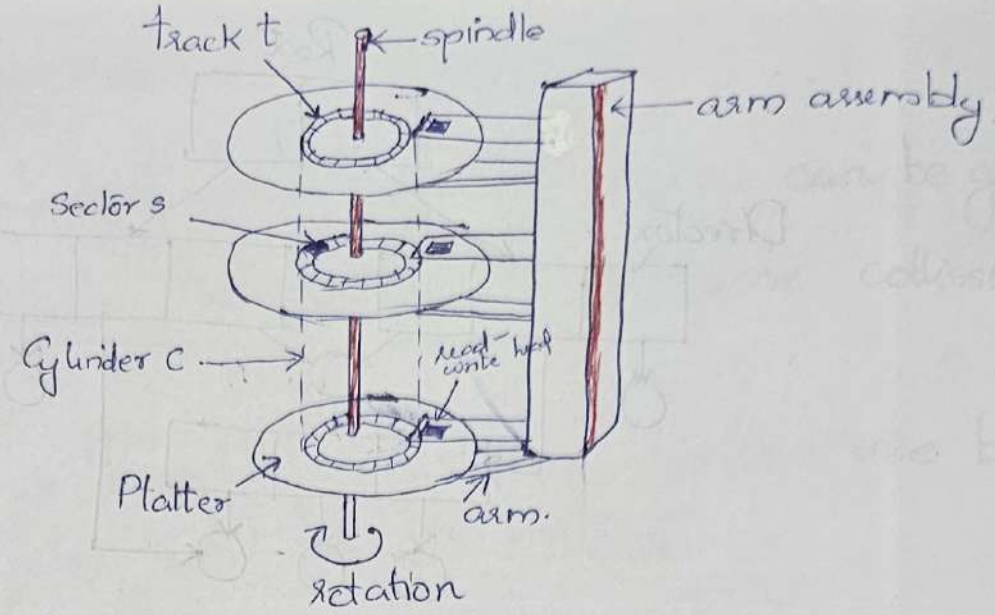
* We can share files.
* Searching is easy due to different-different paths.

Disadvantages

* We share the files via linking, in case of deleting it may create the pblm.
* If the link is softlink then after deleting the file we left with a dangling pointer.
* In case of hardlink, to delete a file we have to delete all the refernce associated with it.

Structure of Hard Disk

* Magnetic disks provide the bulk of secondary storage for modern computer s/ms.
* A read-write head "flies" just above each surface of every platter.
* The heads are attached to a disk arm that moves all the heads as a unit.



* Disk Arm moves as Zig-Zag motion.

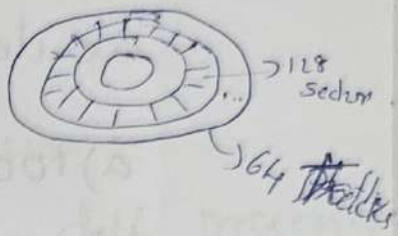
ie Platters
↓ divided
Tracks
↓ divided
Sectors
↳ used to store Bytes.

* Grouping of tracks form Cylinder.
* 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.
* There may be thousands of concentric cylinders in a disk drive & each track may contain hundreds of sectors.
* The storage capacity of common disk drives is measured in gigabytes.

Q. Consider a hard disk with : 4 surfaces, 64 tracks/surface, 128 sectors/tracks, 256 bytes/sector. What is the capacity of the hard disk?

$4 \times 64 \times 128 \times 256$

have (3) have (2) have (1)



$2^2 \times 2^6 \times 2^7 \times 2^8 = 2^{23}$

$= 2^3 \times 2^{20}$

$= 2^3$ MB

$= 8$ MB

Q. Consider a disk pack with the follo: specification: 16 surfaces, 128 tracks per surface, 256 sectors per track & 512 bytes per sector. What is the capacity of disk pack?

a) 200 MB b) 256 MB c) 512 MB d) 500 MB

$16 \times 128 \times 256 \times 512$

$2^4 \times 2^7 \times 2^8 \times 2^9 = 2^{28} = 2^8 \times 2^{20}$

$= 2^8$ MB

$= 256$ MB //

Disk Scheduling (I/O Scheduling)

* Done by OS to schedule I/O requests arriving for the disk.
* Reduces no: of head movements.
* Disk Scheduling is important because :-

→ Multiple I/O requests may arrive by diff: processes & only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be Scheduled.

→ Two or more request may be far from each other so can result in greater disks arm movement

→ Hard drives are one of the slowest parts of the computer s/m & thus need to be accessed in an efficient manner.

Disk Scheduling Algorithms

1. FCFS

* Simplest Algthm.
* The requests are addressed in the order they arrive in the disk queue.

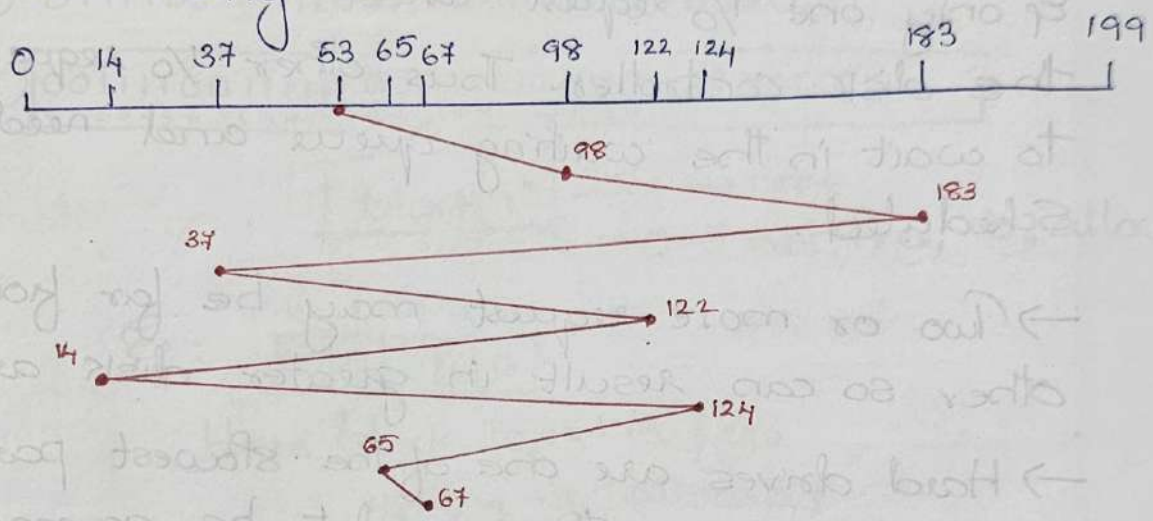
Advantages

* Every request gets a fair chance.
* No indefinite postponement.

Disadvantages

* Doesnot try to optimize seek time.
* May not provide the best possible service.

Q. Queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53.
How many head movements ?



(98-53) + (183-98) + (183-37) + (122-37) + (122-14)
+ (124-14) + (124-65)
(67-65)

= 45 + 85 + 146 + 65 + 108 + 110 + 59 + 2

= 640

2. Shortest Seek Time first (SSTF) {Commonly}.

* In SSTF, requests having shortest seek time are executed first.
* So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their calculated seek time.

* As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of system.

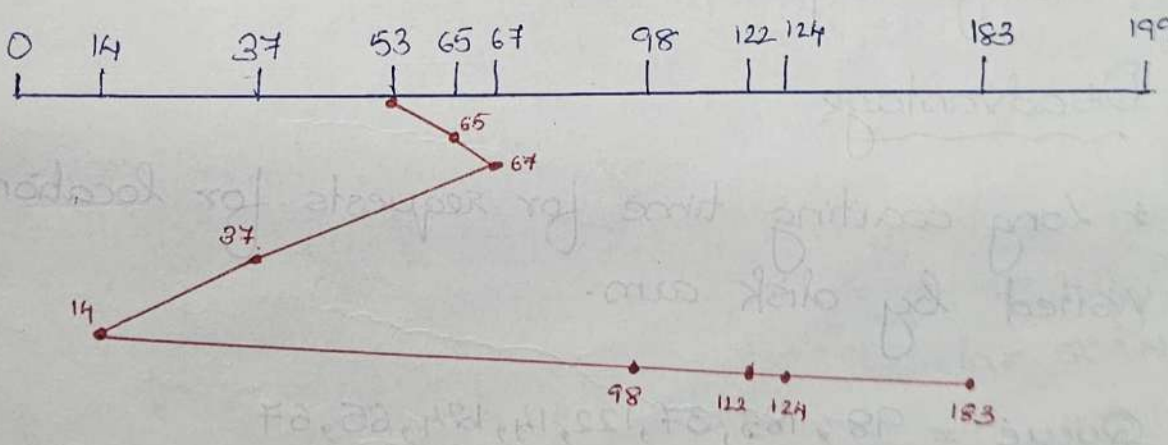
Advantages

* Average Response Time decreases.
* Throughput increases.

Disadvantages

* Overhead to calculate seek time in advance.
* Can cause starvation for a request if it has higher seek time as compared to incoming requests.
* High variance of response time as SSTF favours only some requests.

Q. Queue = 98, 183, 37, 122, 14, 124, 65, 67.
head starts at 53



Shortest distance Selected
53 (37, 65)
65 (37, 67)
67 (37, 98)
37 (14, 98)
When reached left end head moves directly to last value 183. During this movement it access 98, 122, 124.

(65-53) + (67-65) + (67-37) + (37-14) + (98-14) + (122-98) + (124-122) + (183-124).

or

(67-53) + (67-14) + (183-14) = 14 + 53 + 169
= 236.

3. SCAN (Elevator Algorithm)

* The disk arm moves into a particular direction and services the requests coming on its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path.

* So, this algthm works as an elevator and hence also known as elevator algorithm.

* As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

Advantage

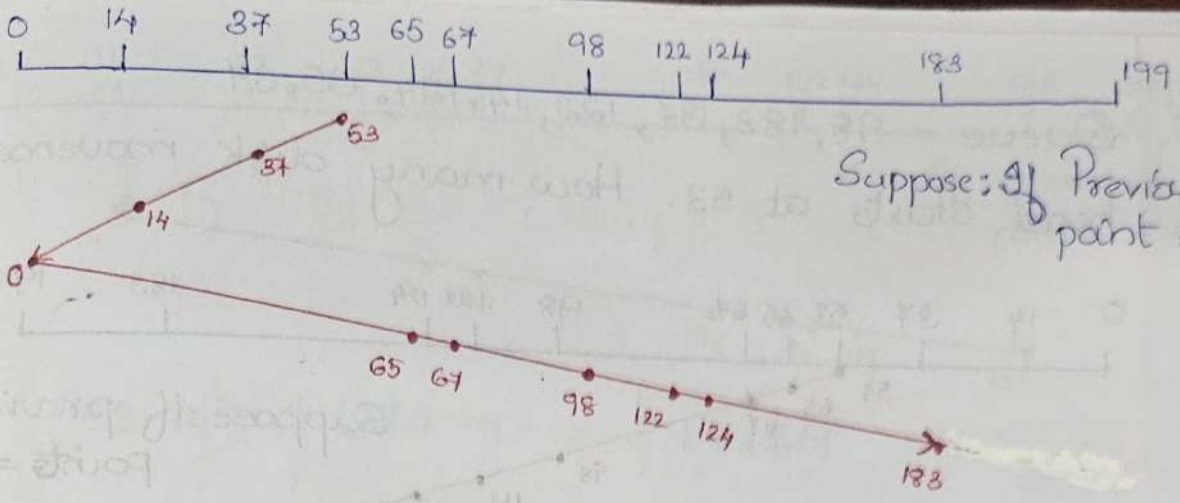
* High throughput
* Low variance of response time.
* Average response time.

Disadvantage

* Long waiting time for requests for locations just visited by disk arm.

Q. Queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53. How many disk movements?

Note :- If Previous head point is given we can understand which direction the head moves.
& if it is not given calculate from both sides & check answers given.



$(53-0) + (183-0) = 53 + 183$

$= \underline{\underline{236}}$.

4. CSCAN (Circular SCAN) {Used in S/ms that place a heavy load on disk}

* In SCAN algthm, the disk arm again scans the path that has been scanned, after reversing its direction.

* So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

* These situations are avoided in CSCAN algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there.

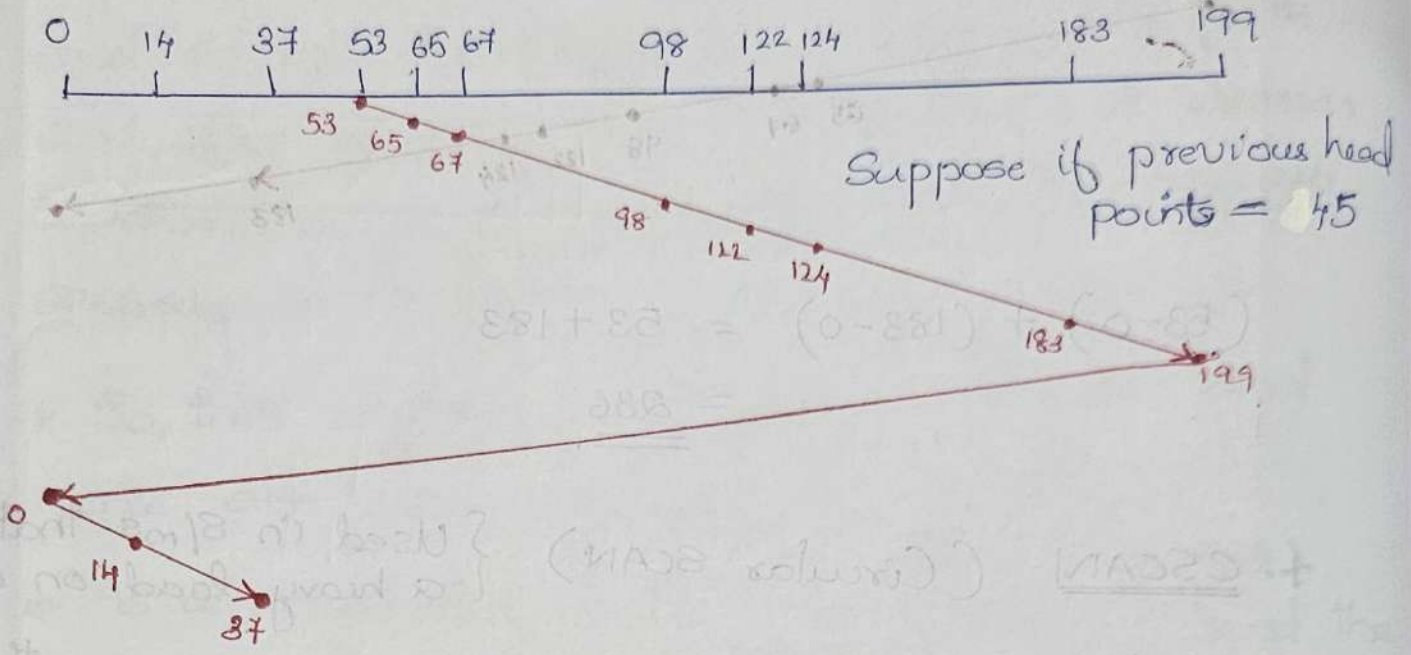
* So, the disk arm moves in a circular fashion & this algorithm is also similar to SCAN algthm & hence it is known as C-SCAN (Circular SCAN).

Advantages

* Provides more uniform wait time compared to SCAN.

* Less starvation pblm.

Q. Queue = 98, 183, 37, 122, 14, 124, 65, 67.
head starts at 53. How many disk movements?



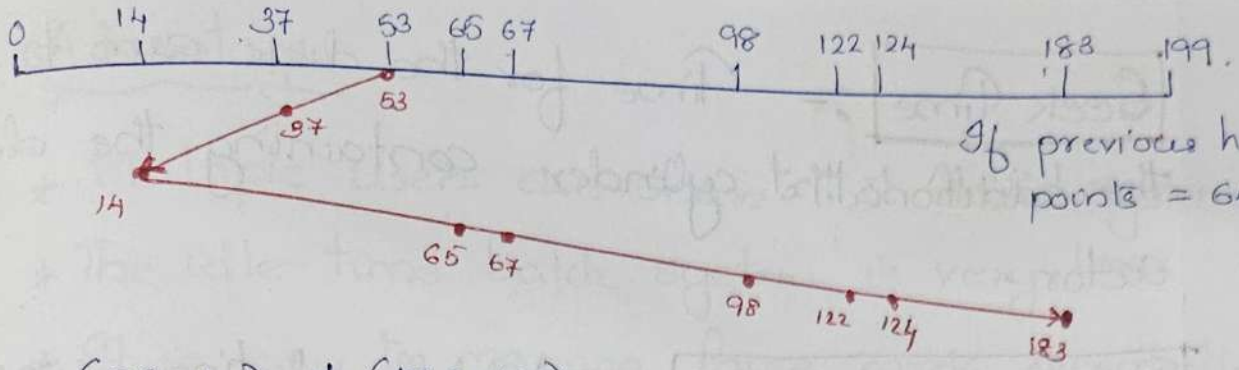
(199-53) + (199-0) + (37-0) = 146 + 199 + 37
= 382

5. LOOK

* It is similar to SCAN disk scheduling algthm except for the difference that the disk arm inspite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only.

* Thus it prevents the extra delay which occured due to unnecessary traversal to the end of the disk.

Q. Queue = 98, 183, 37, 122, 14, 124, 65, 67.
head starts at 53. How many disk movements?



If previous head points = 64.

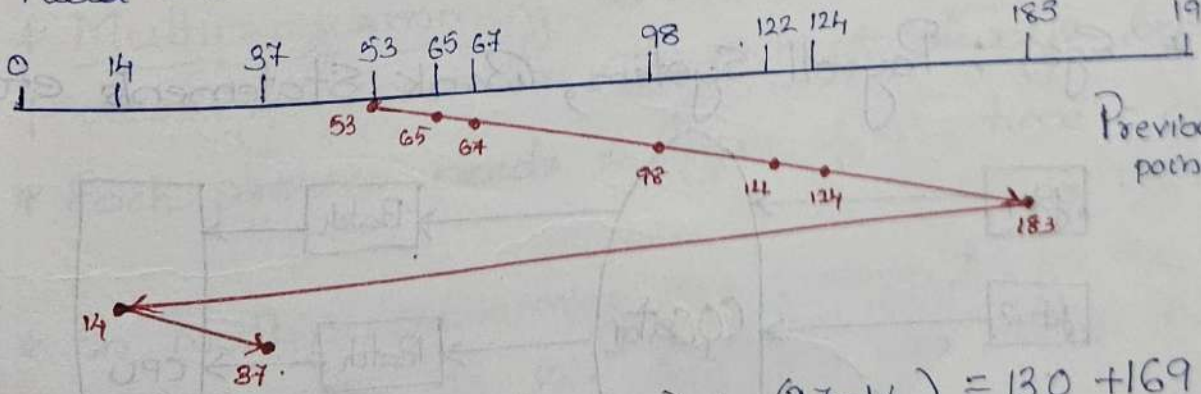
$(53-14) + (183-14) = 39+169$

$= \underline{\underline{208}}$

6. CLOOK

* As LOOK is similar to SCAN algthm, here CLOOK is similar to CSCAN disk scheduling algthm.
* In CLOOK, the disk arm inspite of going to the end goes only to the last request to be serviced in front of the head & then from there goes to the other end's last request.
* Thus, it also prevents the extra delay which occured due to unnecessary traversal to the end of the disk.

Q. Queue = 98, 183, 37, 122, 14, 124, 65, 67.
head starts at 53. How many disk movements?



Previous head points = 45

$(183-53) + (183-14) + (37-14) = 130 + 169 + 23$

$= \underline{\underline{322}}$

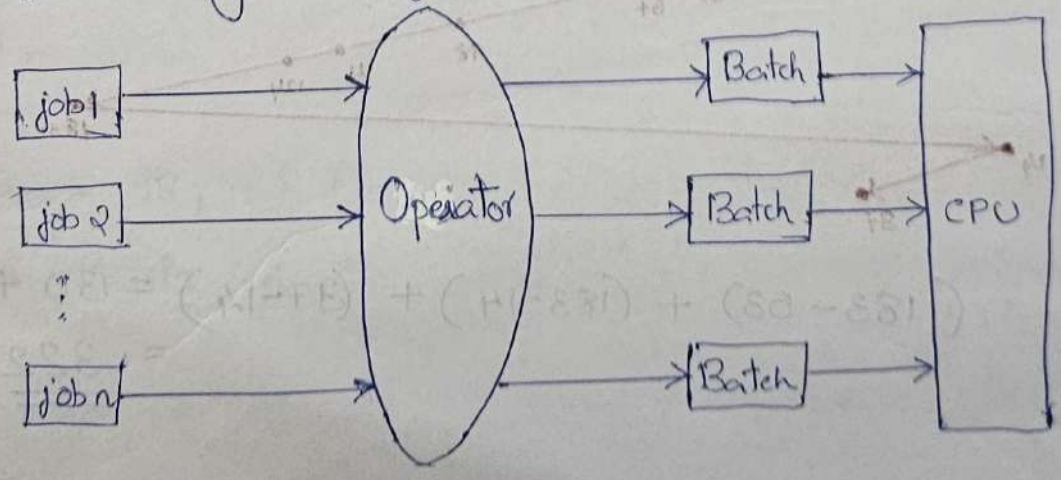
Seek Time :- Time for the disk arm to move the heads to the cylinder containing the desired sector.

Rotational Latency :- Additional time waiting for the disk to rotate the desired sector to the disk head.

Bandwidth :- Total no: of bytes transferred, divided by the total time b/w the first request for service & the completion of the last transfer.

Batch Operating System

* Donot interact with the computer directly.
* There is an operator which takes similar jobs having same requirement (or functionality) & group them into batches.
* It is the responsibility of operator to sort the jobs with similar needs.
* Egs : Payroll System, Bank Statements etc.



Advantages

* Multiple users can share the batch systems.
* The idle time batch system is very less.
* It is easy to manage large work repeatedly in batch systems.

Disadvantages

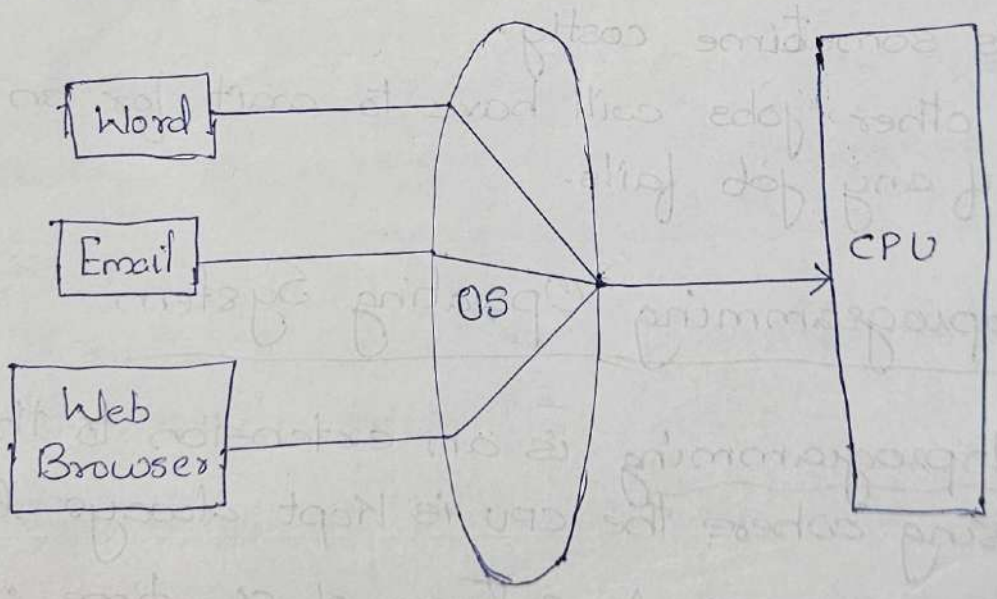
* It is very difficult to guess or know the time required by any job to complete. Processors of the batch systems knows how long the job would be when it is in queue.
* The computer operators should be well known with batch s/ms.
* Batch s/ms are hard to debug.
* It is sometime costly.
* The other jobs will have to wait for an unknown time if any job fails.

Microprogramming Operating System

* Multiprogramming is an extension to the batch processing where the CPU is kept always busy.
* Each process needs 2 types of s/m time : CPU time & I/O time.
* In multiprogramming environment, for the time a process does its I/O. The CPU can start the execution of other processes.
* ∴, Multiprogramming improves the efficiency of the s/m.

Time Sharing Operating System (Multitasking OS)

* Each task has given some time to execute, so that all the tasks work smoothly.
* Each user gets time of CPU as they use single system.
* These systems are also known as Multitasking Systems.
* The task can be from single user or from different users.
* The time that each task gets to execute is called quantum. (Context Switching occurs).
* After this time interval is over OS switches over to next tasks.



Advantages

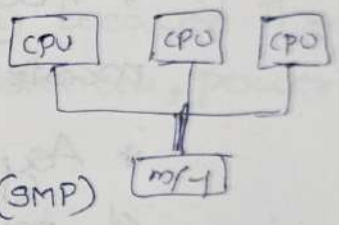
* Each task gets an equal opportunity.
* Less chances of duplication of software.
* CPU idle time can be reduced.

Disadvantages

* Reliability problem.
* One must have to take care of security and integrity of user programs and data.
* Data communication problem.
* Egs :- Multics, Unix etc.

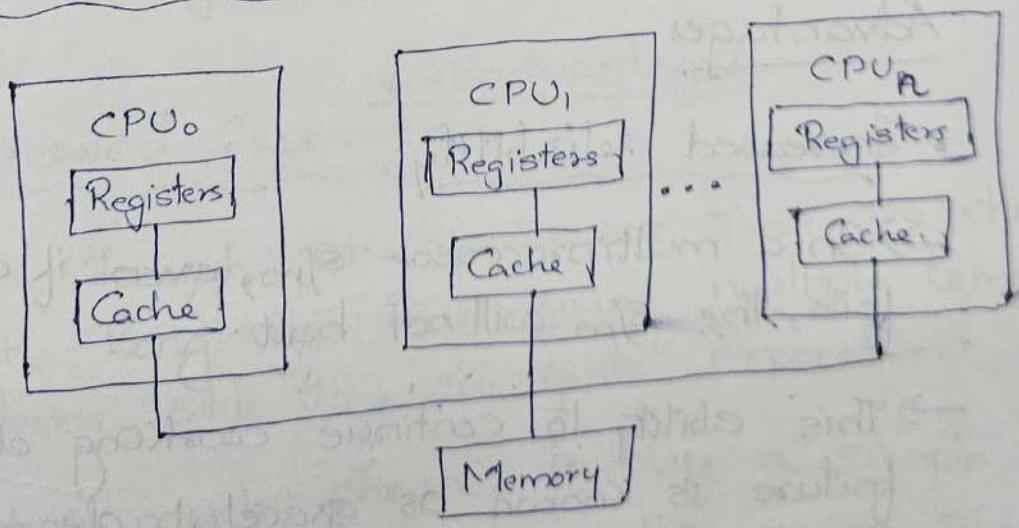
Multiprocessor Systems

* Multiprocessor or parallel or multicore s/ms have multiple processors working in parallel that share the computer clock, memory, bus & peripheral devices.



* Types of Multiprocessors :-
  1. Symmetric multiprocessors (SMP)
  2. Asymmetric multiprocessors.

1. Symmetric Multiprocessors. (Commonly Used).



* Each processor runs an identical copy of the OS & these copies communicate with one another as needed.

* All processors are peers.
* No master-slave relationship exists between processors. Each processor concurrently runs a copy of the OS.

2. Asymmetric Multiprocessors.

* In asymmetric s/ms, each processor is given a predefined task.
* There is a master processor that gives instruction to all the other processors.
* Asymmetric multiprocessor s/m contains a master slave relationship.
* Asymmetric multiprocessor was the only type of multiprocessor available before symmetric multiprocessors were created.
* Now also, this is the cheaper option.

Advantages

- Increased reliability

→ In a multiprocessor s/m, even if one processor fails, the s/m will not halt.

→ This ability to continue working despite h/w failure is known as graceful degradation.

→ For eg:- If there are 5 processors in a Multiprocessor s/m & one of them fails, then also 4 processors are still working.

→ So the s/m only become slower & doesnot ground to a halt.

- Increased Throughput

→ If multiple processors are working in tandem, then the throughput of the s/m increases ie; no: of processes getting executed per unit of time increase.

→ If there are N processors then the throughput increases by an amount just under N.

- Economy of Scale

→ Multiprocessor s/ms are cheaper than single processor systems in the long run because they share the data storage, peripheral devices, power supplies etc.

→ If there are multiple processes that share data, it is better to schedule them on multiprocessor s/ms with shared data than have different computer systems with multiple copies of the data.

Disadvantages

- Increased Expense

→ Even though multiprocessor s/ms are cheaper in the long run than using multiple computer systems, still they are quite expensive.

→ It is much cheaper to buy a simple single processor s/m than a multiprocessor s/m.

- Complicated Operating System Required.

→ There are multiple processors in a multiprocessor system that share peripherals, memory etc.
→ So, it is much more complicated to schedule processes and impart resources to processes than in single processor systems.
→ Hence, a more complex & complicated OS is required in multiprocessor s/ms.

- Large Main Memory Required.

→ All the processors in the multiprocessor s/m share the memory.
→ So a much larger pool of memory is required as compared to single processor s/ms.

Real-Time Operating System { Embedded s/w }

* These types of OSs serves the real-time s/ms.
* The time interval required to process and respond to inputs is very small.
  ↳ is called Response Time

* Real Time s/ms are used when there are time requirements are very strict like missile s/ms, air traffic control systems, robots etc.

* 2 types of Real-Time Operating System which are as follows :- 1. Hard Real-Time Systems
2. Soft Real-Time Systems.

1. Hard - Real Time Systems :-

* These OSs are meant for the applications where time constraints are very strict and even the shortest possible delay is not acceptable.
* These S/ms are built for saving life like automatic parachutes or air bags which are required to be readily available in case of any accident.
* Virtual memory is almost never found in these S/ms.

2. Soft - Real Time Systems :-

* These OSs are for applications where for time-constraint is less strict.

Advantages

* RTOS in embedded S/m : Since size of pgms are small, RTOS can also be used in embeddded S/ms like in transport & others.
* Error free : These types of S/ms are error free.
* Memory Allocation : is best managed in these type of systems.
* Maximum Consumption : Maximum utilization of devices & S/ms, thus more o/p from all the resources.

* Task Shifting : Time assigned for shifting tasks in these s/ms are very less.
For eg :- In older s/ms it takes about 10 ms in shifting one task to another & in latest s/ms it take 3 ms.

* focus on Application : focus on running applications & less importance to applications which are in queue.

Disadvantages

* Limited Tasks : Very few task run at the same time and their concentration is very less on few applications to avoid errors.

* Use heavy system resources : Sometimes the s/ms resources are not so good & they are expensive as well.

* Complex Algorithms : The algorithms are very complex and difficult for the designer to write on.

* Device driver and Interrupt signals :- It needs specific device drivers and interrupt signals to response earliest to interrupts.

when a new resource
mes, where these resource
re to be placed are determined by.

* Thread Priority : It is not good to set thread priority as these s/ms are very less prone to switching tasks.