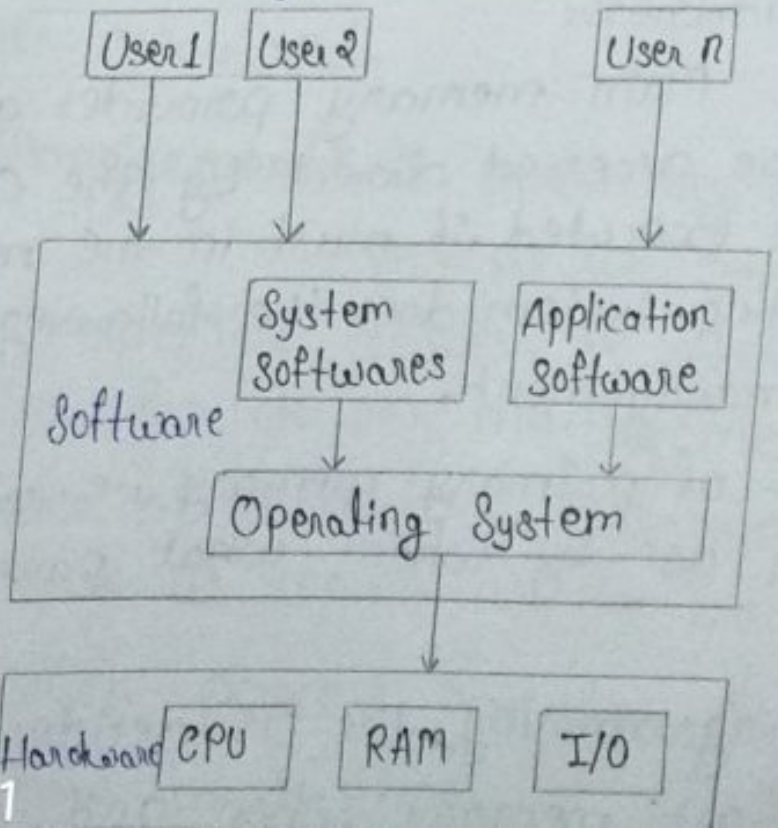
UNIT-I

Introduction

* Operating System: An operating System (OS) is an interface between a Computer user and Computer hardware. An operating System is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

Definition: An operating System is a program that acts as an interface between the user and the Computer hardware and Control the execution of all kinds of programs.



Following are some of important function of an Operating System:

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over System performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

* Memory Management: Memory management refers to management of primary memory or main memory. Main memory is a large array of words or bytes where each word or byte has its own address.

Main memory provides a fast storage that can be accessed direct by the CPU. For a program to be executed it must in the main memory. An operating system does the following activities for memory management.

- Keeps tracks of primary memory, i.e. what part of it are in use by whom, what part are not in use.
- In multiprogramming the OS decides which process will get memory when and how much.

- Allocates the memory when a process request it to do so.
- De-allocates the memory when a process no longer needs it or has been terminated.

* Processor Management: In multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called process scheduling. An operating System does the following activities for processor management.
- Keeps tracks of processor and status of process. The program responsible for this tasks is known as traffic Controller.
- Allocates the processor (CPU) to a process.
- De-allocates processar when a processar is no longer required.

* Device Management: An operating System manages device Communication via their respective drives. It does the following activities for devices management.
- Keeps tracks of all devices. Program responsible for this task is known as the I/O Controller.
- Decides which process gets the device when and for how much time.

- Allocates the devices in the efficient way.
- De-allocates devices.

* File Management: A file System is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.

An operating System does the following activities for file management.

- Keeps track of information, location, uses, Status etc. The Collective facilities are often known as files System.
- Decides who gets the resources.
- Allocates the resources.
- De-allocates the resources.

Other Important Activities-

Following are Some of the important activities that an operating System performs:-

* Security: By means of password and Similar other techniques, it prevents unauthorized access to programs and data.

* Control over System performance: Recording delays between request for a Service and response from the System.

* Job accounting: Keeping track of time and resources used by various jobs and users.

* Error detecting aids: Production of dumps, traces, error messages, and other debugging and error detecting aids.

* Coordination bet^n other softwares and users: Coordination and assignment of Compilers, interperters, assemblers and other software to the various users of the Computer Systems.

Evolution of Operating System

An operating system may process its task serially i.e. sequentially or concorrently. It means that resources of the computer system may be dedicated be a single program until its completion or they may be allocated to several programs in different stages of the execution.

1. Serial Processing:- In early computer system programming in 0's or 1's instructions and data where fed into the computer by means of Console switches and hexadecimal keyboards Program's will started by loading the program registers with address of first instruction of the program. These type of programming caused loco utilization of both user and resource.

With the invent of punch cards, paper taps and languages translators program were web coded using programming language which were changed into object code and then loaded into object the memory using program called loader. After transferring the control to the loaded program the execution of the program will be displayed on printed. The process of development and preparation of the program in such an environment is slow due to serial processing.

In typical serial processing first the editor is called to create a source code using programming language. Translator is called to convert source code into object code and finally loader is called to load the executable program into the main memory. If syntax errors are detected the whole process must be re-started from the beginning. Hence serial processing leads to low utilization of resources.

2. Batch Processing:- The next step in the evaluation of operating system was to automate the sequencing of operations involved in the program execution and into the mechanical aspects of program development. In batch processing jobs with similar requirement which batch together and execute together. In a batch mode

Each user prepares the programs offline and submits it at the computer collects center. The computer operator collects the program, punched on the cards and stacks one program on the top of the other. Then the batch of program which is loaded into the computer where they are executed one after another. Batch processing is also known as serial, sequential, offline or stack job processing. In batch processing utilization of system resources improve but still during the transition from one job to another the CPU sat idle.

To overcome this idle time of small program called resident monitor was created which is always resided in the main memory. It automatically sequenced one job to another job.

In order that the operating system can identify a new job and determine what action should be taken for the job. Some control information is necessary such as making of job beginning and ending, commands for loading and executing the programs etc. These control statements are written in a language known as JCL (Job Control language).

There are two approaches to improve system performance by over lapping input, processing and output. These are called buffering

and Spooling.

Buffering :- Buffering is a method of overlapping input, output and processing of single job. After the data has been read and the CPU is about to start operating on it the input devices is instructed to begin the next instruction input immediately. The CPU and the input device are both busy by the time. The CPU is ready for next data item the input device will have finished reading it and this process can be done for output also. Buffering overlaps input, output and processing of a single job where as spooling allows CPU to overlaps the input of one job with the computation and output of other jobs.

Spooling :- In batch mode of operation the processing speed of a computer system can be further enhanced by the technique known as Spooling i.e simultaneously peripheral, output, onlining.

The dedicated I/O devices such as printers and card readers are considerably slower as compared to the speed of CPU. While a card reader is supplying one character to main memory. The CPU can perform thousands of internal operation during the reading of

information into the main memory. The CPU has to wait because of slow reading. Similar speed mis-match exist for large number of peripheral devices such as printer, tally type writter etc.

Spooling is a tehnique that has been used to reduce the mis-match between CPU and input devices or to reduce the idle time of CPU. In this process the data that comes from input device or goes to an output device is placed on either a magnatic type or drive.

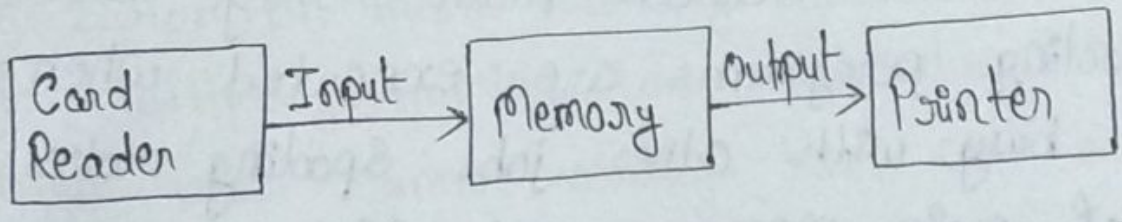


fig. Mode of transfer without Spooling facility

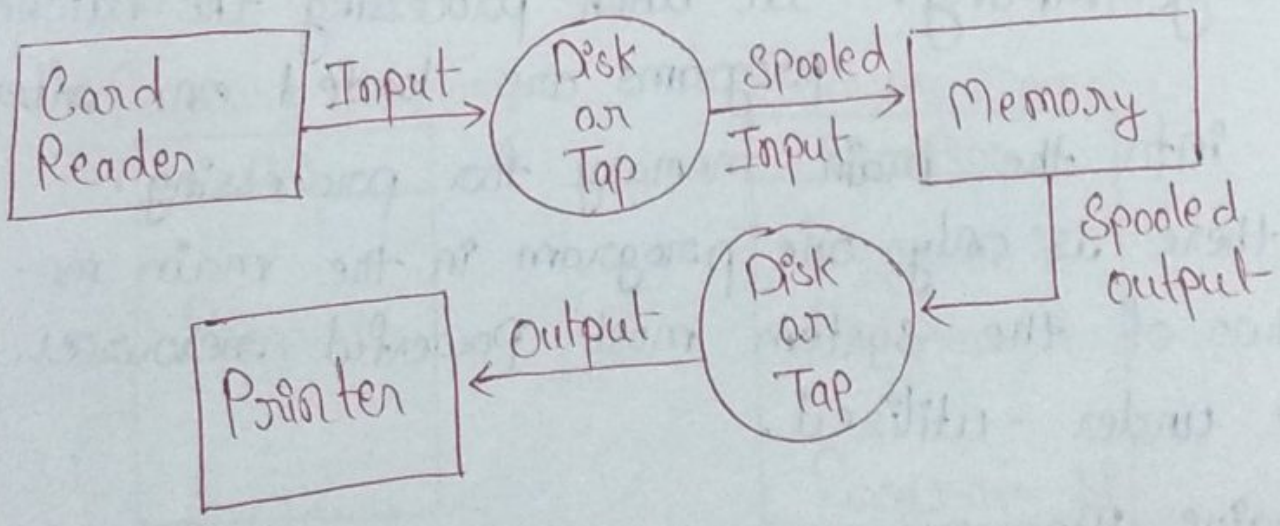


Fig. MOT using Spooling facility

A batch of program when fed to the Card reader is read and temporily Stored in magnetic tape or disk instead of main memary. The program are then fed and processed by the Computer. The result obtained are again Stored in magnetic tape or disk. The Contents of the output tape or disk are later printed through printer. The process of storing input data and output data on tape disk is known as Spooling. The tape or disk device acts as a buffer area between main Storage and I/O devices. Spooling programs are executed when CPU is not busy with other jobs Spooling makes better use of main memary and CPU.

Multiprogramming:- In batch processing the batched programs are loaded one after another into the main memary for processing. When there is only one program in the main memary two of the system most powerful resources. May be under-utilized.

(i) Expensive Memary
(ii) Full Capabilities of CPU

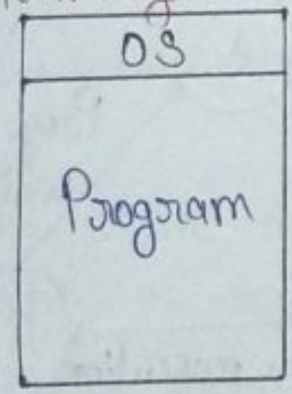
There are 2 types of Programs-

1) I/O bound
2) CPU bound

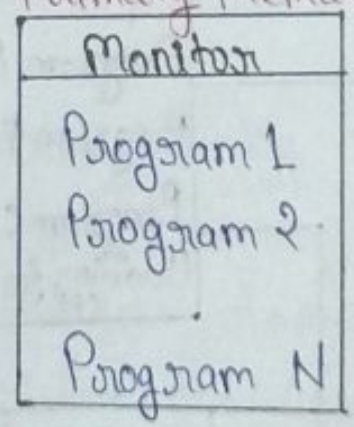
Program used for commerical data processing read in large amount of data, perform little computation and output large amount of information and are known as I/O bound programs.

Program used for scientific and engineering applications need very little I/O but require anomous computation. These are called CPU bound programs.

To overcome the problem of under-utilization of main memory and CPU. The concept of multiprogramming was introduced in OS. A single user cannot always keep CPU or I/O devices busy all the time. Multiprogramming offers a more efficient approach to system performance.

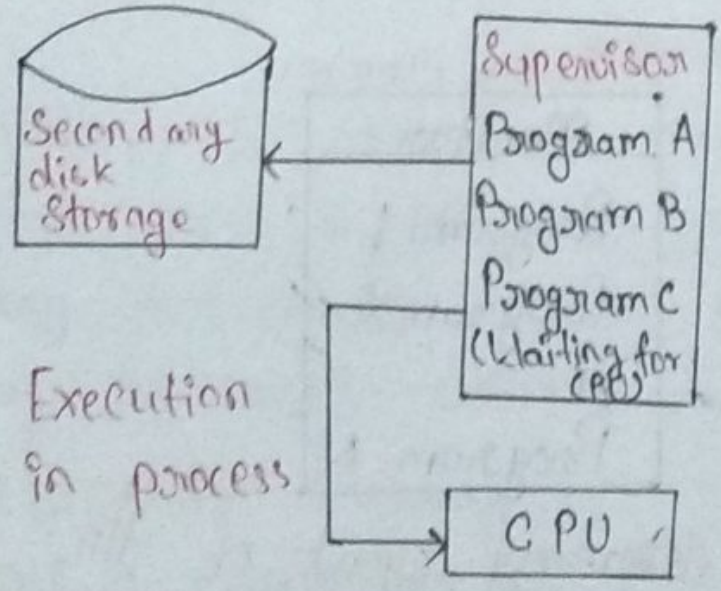


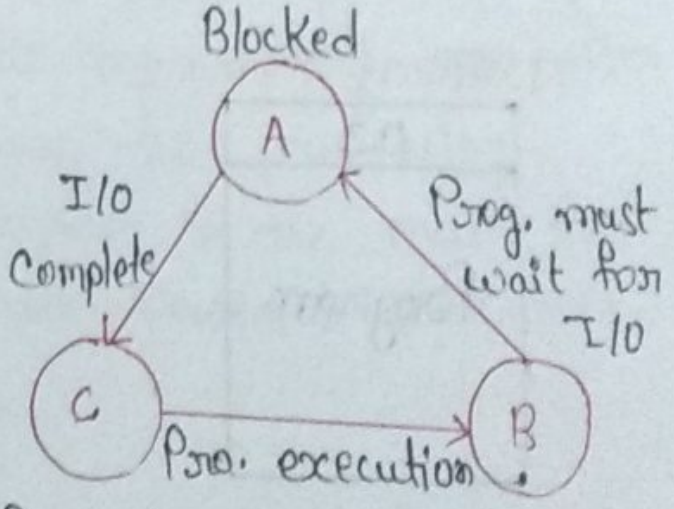
Single User OS



Memory layout of the System in multiprogramming

Multiprogramming is the process of execution of 2 or more different program by the same computer. Two or more programs are placed to the main memory and executed con-corrently. In sequential execution when one program need I/O operation CPU would sit idle. In multipromming system when one program is waiting for I/O transfer there is another program ready to be excecuted by the CPU. In some multiprogramming system only a fixed number of jobs can be processed con-corrently i.e multiprogramming with fixed number of tasks may vary i.e. multiprogramming with variable number of tasks.





Multiprogramming has 2 main advantages-

(1) Increased throughput

(2) Lowered Response time.

In multiprogramming all the programs residing in the main memory will be in one of the three states-

(1) Running i.e CPU is being used.

(2) Blocked i.e I/O operations is being performed.

(3) Ready state i.e waiting for CPU.

Program A is busy writing output data into the disk. Program B is being currently executed by CPU and Program C is waiting for CPU to become free. Program A, B and C are in blocked, running and ready state respectively.

The area occupied by each program in the main memory is called memory partition. As soon as any one of the jobs already occupying the main memory is completed the corresponding memory partition becomes free. In multiprogramming system CPU is always busy.

Time Sharing :- In time Sharing System many users are allowed to simultaneously share the computer resources. Time Sharing System is a logical extension of multiprogram. Multiprogrammed Batched System refers to the allocation of Computer resources. In a time dependent fashion. The principle notion of time Sharing System is to provide a large number of user direct access to the computer. In multiprogramming System where programs are executed in a priority basis where as in time Sharing System the CPU time is divided among different users on a scheduled basis. A time Sharing System uses Scheduling and multiprogramming to provide the user with Small portion of time Shared Computer i.e CPU time. Each user program is allocated a very small portion of CPU time for its execution and this time period is called as time slice, time quantum allows the CPU or time slot. Time Sharing System allows the CPU to Switch from one user Station to another and perform a part of each job in allocated time - slice. Until the job is completed. In time Sharing like a multiprogramming System only one

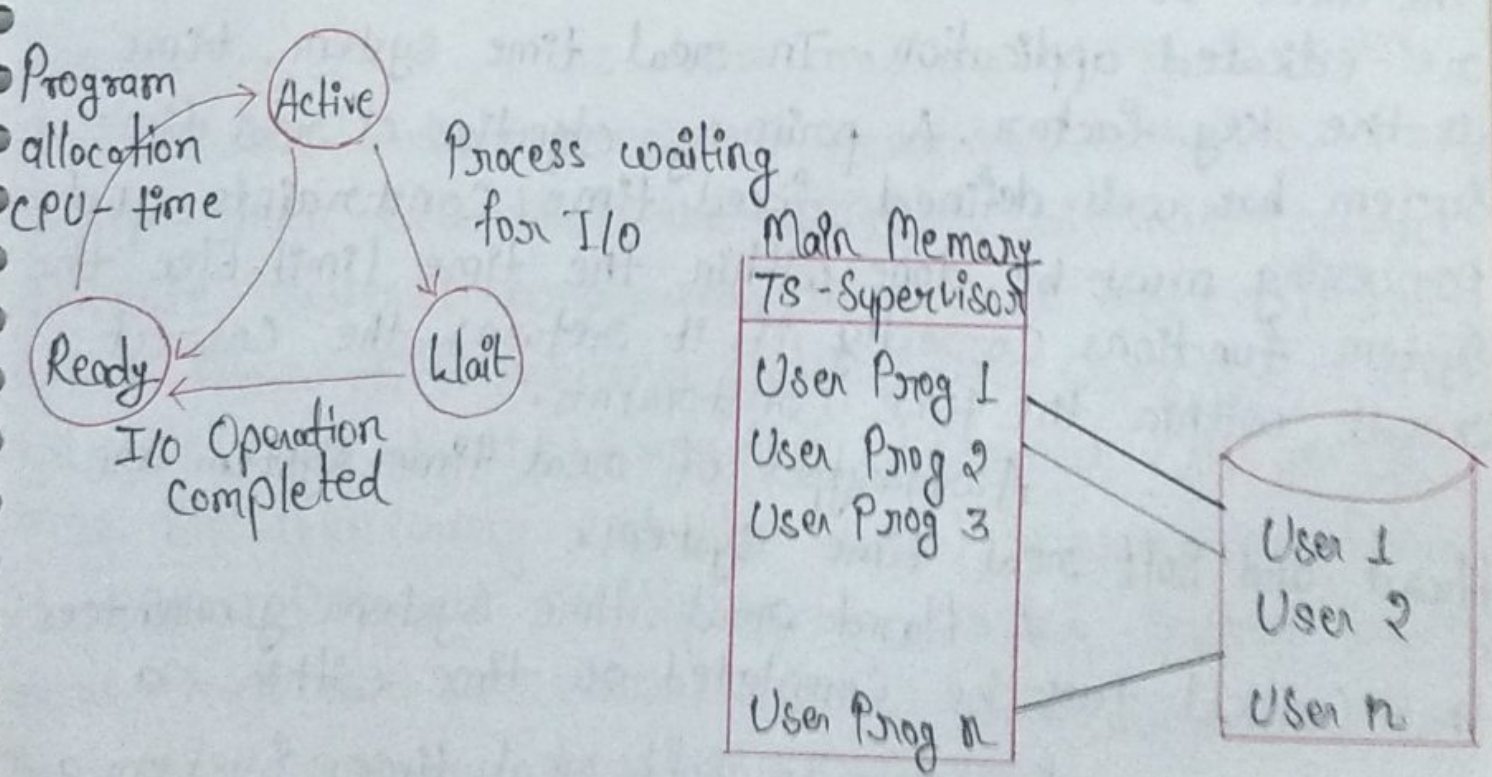
Program can be in Control of CPU at any given time.

The users of time - Sharing System will fall in one of the 3 State-

(1) Active State :- The user program Currently has Control of the CPU and only one user program will be active at a particular time.

(2) Ready :- The user program is ready to Continue but is waiting for its turn to get the get the CPU. More than one user can be in ready state at a particular time.

(3) Wait :- The users program is waiting for some I/O operations. More than one can be in waiting State at a particular time or in a given time period.



In time sharing system to obtain a reasonable response time jobs have to be swapped in and out of the main memory into the disk which serves as a backing store for the main memory. This operation of transferring program from main memory to the backing store and back is known as swapping or roll-in, roll-out process.

Advantages:-

1. Reduces the CPU idle time i.e. CPU utilization and throughput increases.
2. Turn around time and response time decrease.
3. Improve the user efficiency.

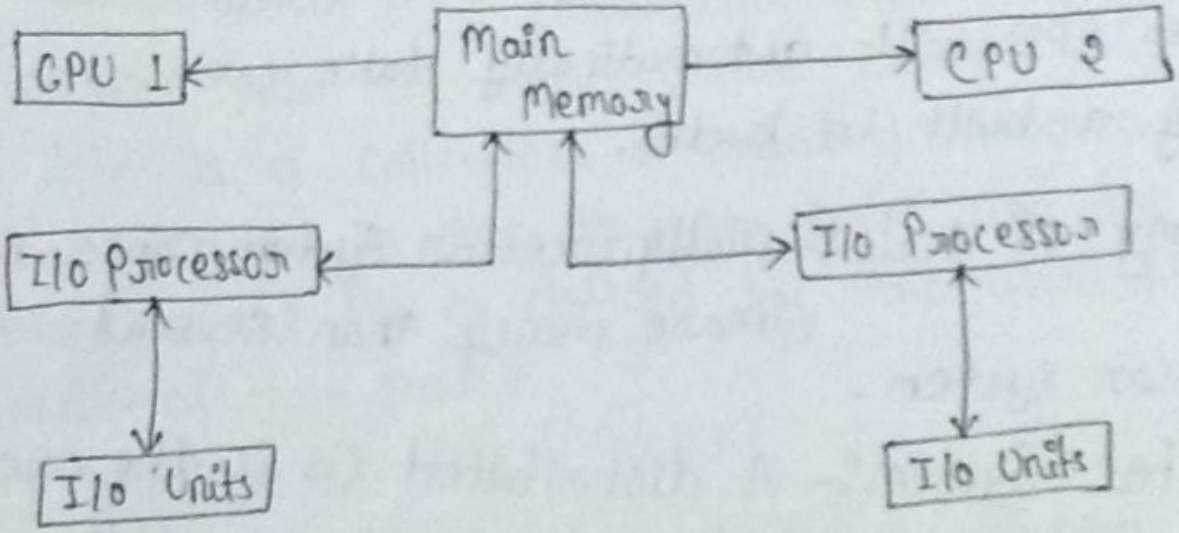
Real Time:- A real time system is used when rigid time constraints have been placed on the operation of a processor or flow of the data. It is often used as a control device in re-edicated application. In real time system, time is the key factor. A primary objective of real time system has well defined fixed time constraints and processing must be done within the time limit. Else the system functions correctly if it returns the correct result within the time constraints.

Two types of real time system are- Hard and Soft real time system.

A Hard real time system guarentees that critical task be completed on time within a certain range of time. In Soft real time system a

Critical real time task gets the priority over other task but don not support deadline scheduling.

Multiprocessing System:- The idea of using of I/O processor is to improve the performance of the computer system was farther enhanced by designing system that make use of more than one CPU. Such system are called multiprocessing system.



The term multiprocessing is used to describe inter-connected computer with 2 or more independent CPU. that the ability to simultaneously execute several programs. In such a system instruction from different and independent program can be processed at the same time different CPU or CPU. may simultaneously execute several instructions from the same program. Multiprogram-cessing system have more than one processor in close communication sharing the computer bus and also memory and peri-

pheral devices. Multiprocessor System has 3 main advantages -

(1) Increase throughput :- Increase throughput by increasing the no. of processor more work is done in less time.

(2) Increase Reliability :- If functions can be distributed properly amongly several processor then failure of one processor will not halt the entire system. If one CPU breaks down, the other CPU will automatically take over, thus providing a built in backer.

(3) Economy of scale :- Multiprocessor System can save more money than several uniprocessor system.

Distributed System :- A distributed Os is the one that looks to its user like a Ordinary centralized Os but runs on multiple independent CPU's. The distributed Os consist of multiprocessor but provide an illusion to its user that it is an uniprocessor system. In this system a user interact with the system for running the program or accessing the file but does not know how were the progs. executed all the processing are automatically and efficiently handled by the Os distributed Os allow program to run on several process at the same time which requires more

conflict processor scheduling algorithm to achieve maximum utilization of CPU. Distributed system and distributed Os depend on networking concept for their functionality. It is capable of sharing different computing task and communicate with each other through communication lines or networks. Distributed Os is a collection of processors that do not store memory, clock or peripheral devices.

FILE SYSTEM:- The file system is the most important visible part of any Os and also the most visible services of an OS. A file is a collection of related information that is recorded in the storage device and the information in the file is defined by the user. File System consits of two parts-

(1) Collection of file

(2) Directory Structure which organizes and provides info. about all the files in the System.

A file has a certain defined structure according to its type. A text file is a sequence of characters organised into lines. A source file is a sequence of sub-routines and functions consisting of declarative and executable statement. An object file is a sequence of bytes organised into blocks. An executable file is a series of code sections which is loaded into the memory

by loader and executed.

A file has certain attributes-

(1) Name:- Name is a unique identifier which identifies the file within the file system.

(2) Type:- This information is needed for these system that support different types.

(3) Location:- This info. is a pointer to the device and to the location of the file on that device.

(4) Size of the file.

(5) Protection:- Acess control info. about the file determine which users can do the reading, writing or executing the file.

(6) Date, time and User identification:- Creation, last modification or last use of the file.

These data can be used for protection, security & usage monitoring.

All this info. about all the files is kept in the directory structure.

FILE OPERATIONS :- Os provide System calls to Create, write, read, reposition, delete, rename & truncket the files.

(1) Creating a file :- 2 steps for creating a file -

(i) Space in file System must be found for that file.

(ii) An entry for the new file in the directory must be made i.e. name of the file, location in the file System and other information.

(2) Writing into the file :- To write a file a System call specifying both the name of the file and info. to be written into the file is made.

The Os searchs the directory to find the location of the file and keep a write pointer to the location where the write is to take place which is updated whenever write operation occurs.

(3) Reading a file :- To read from the file System call is made that specifies the name of the file and where the read operation should start. The Os search the directory Structure for the entry and keep a read pointer to the location in the file where next read is to take place and updated when read operation is performed.

(4) Re-position :- The directory is searched for appropriate entry and current file position is set to the given value.

(5) Deleting a file :- To delete the file the directory is searched and then all the file space is released and the directory entry is also erased.

(6) Truncating a file :- This function allows all the attributes of the file to remain unchanged except that all the contents of file are erased and size is set to zero and also its file space is released.

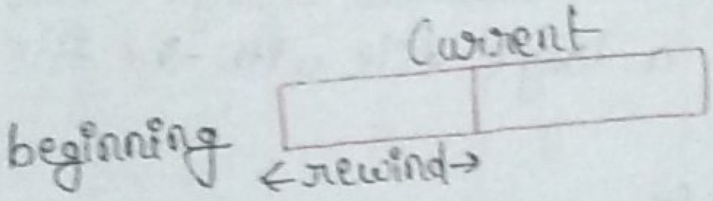
Other operations on the file are renaming a file, copying a file, closing a file etc.

Access Methods in file Systems :-

File store info. which must be accessed and read into the computer memory. The info. in the file can be accessed in several ways-

(1) Sequential Access :- Sequential files are those files which are read sequentially one record after the other. In an order starting at the begining to the end of file. A file is made up of fixed length logical records that allow the program to read and write the program to read and write the records rapidly.

The bulk of operation on the file are read and write and is based on pointer. A read operation reads next portion of the file and automatically advances the files pointer which tracks a I/O location. Similarly a write operation approaches to the end of the file and updates the new info. to the end of the file. Such a file can be reset to the beginning or in some systems a program may be able to skip forward or backward n records. This scheme is known as sequential access.



(2) Direct / Random / Relative Access:- Direct access is based on a disk model of the file. A file is viewed as numbered sequence of blocks or records. A direct access file allows arbitrary blocks to be read / written i.e. without any order. Direct access files are important for immidiate access to large among of info. The block containing answer to the query can be read directly to provide the desired info. the file operations must be modified to include the block no. as the parameter i.e the operation can be written as -

Read n	Read next
Write n	Write next

The block no. provided by the user to the OS is a relative block no. A relative block no. is an index relative to the beginning of file.

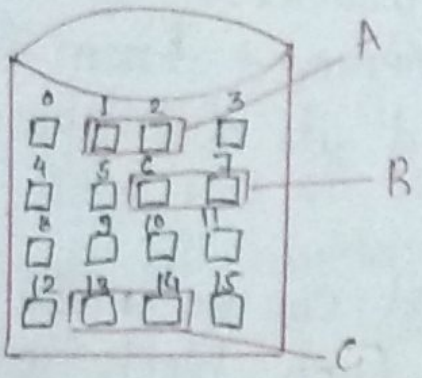
(3) Index Sequential Access Method:- In this method an index for the file is constructed which contains pointer to various blocks. To find an entry in the file first search the index and then use the pointer directly access the file and find the desired entry. This method allows to search a large file with the file information.

Allocation Method:- The file can be created, opened, written, rewound, read, closed and deleted. The implementation of the file is another important function for OS. The direct access nature of the disk allows more flexibility in the implementation of the files. The main problem for OS is how to allocate space to these files so that disk space is effectively utilized and files can be quickly accessed.

There are three method of allocation of disk space are-

1) Contiguous
2) Linked
3) Indexed

1) Contiguous :- Contiguous allocation method require each file to occupy a set of contiguous addresses. On the disk. This disk addresses define a linear ordering on the disk. Contiguous allocation of the file is defined by disk address of first-block and its length. A file is n block long and the starting location is b then the file occupies the locations.



File	Start	Length
A	1	2
B	6	3
C	13	2

The directory entry for each file indicates the address of the starting block and the length of the memory area allocated for this file. For sequential access the file system records the disk address of the last block and then reads the next block. For direct access the block i of a file which starts at block b then the block b+i can be immediatly accessed.

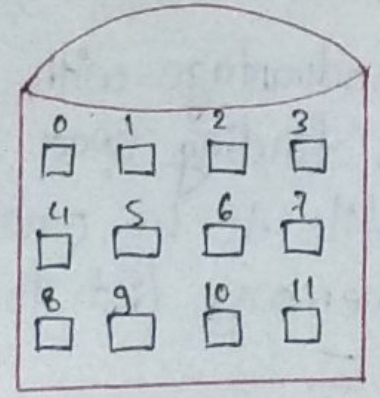
Disadvantage :- The disadvantage with contiguous allocation is finding space for the files. If the file to be created is n blocks long search the free memory list for n free contiguous blocks.

Free Space Management:- Files are Created and deleted during operation therefore the Space from the deleted file can be re-used for new files. To keep the track of free disk Spaces the file system maintain a free disk Spaces the file System maintain a free Space list. The free Space List records all the blocks which are free.

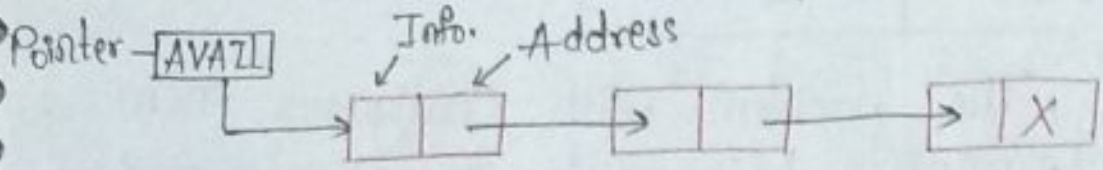
To Create a file the free Space List for required amount of Space list for required amount of Space is Searched and allocated to the new file. And this free Space is removed from free Space list. When the file is deleted its disk space is added to the free Space list.

The free Space list can be maintained by a Bit map / Bit vector. Each block is represented by one-bit. If the block is free then it is represented by 0 and if it is allocated is represented by 1.

For example: If the blocks 2, 3, 4, 5, 8, 9 are free and rest are allocated then the Space list map is denoted by 110000110011

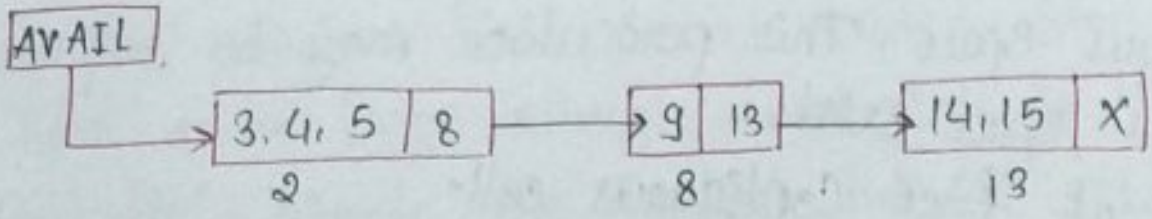


Another approach is to link all the free cells or disk blocks together keeping a pointer to the first free block. This block will contain a pointer to the next free block and so on. which is called as linked free space scheme.



This scheme require more I/O time as to traverse the list each block must be read.

A modicational approach is to store the address of n free blocks in first free block. The first n-1 of these are free and the last is the address of another block containing the addresses of another n free blocks. The advantage is that the addresses of large no. of free blocks can be found quickly.



Another method is to keep the address of first free block and the no. n of free contiguous blocks which follow it. Each entry in the free space list consists of a disk address and it count.

	Address	Count
Free Space	2	3
	8	1
	13	2

The problem with Contiguous memory allocation is determining how much space is needed for a file. If little space is allocated to a file it cannot be extended.

The two solutions are -

(1) The user programs are terminated with an appropriate error message. The user is allocated more space and the program is executed again. The repeated execution increases the cost.

(2) Another solution is to find a large hole, copy the contents of the file to the new hole and release the previous space. This proceduce may be repeated as long as space exist.

Hole: Set of free contiguous cell.

Dynamic Storage Allocation:- The problem with the Contiguous allocation method can be removed by dynamic Storage allocation. Disk space is a large array of disk blocks. Disk space is a collection of free and used segments.

A Segment is a contiguous set of disk blocks. An unallocated segment is called a hole. The dynamic allocation storage problem is to specify a size n from a list of free holes.

The set of holes is searched to determine which hole is best suited to allocate to a file or user program.

(1) First fit
(2) Best fit
(3) Worst fit are the Stratigies used to select a free hole from the set of available holes.

1) First fit :- Allocate the first hole that is large enough or big enough searching can start either at the beginning of the set of holes or where the previous first search ended. Stop searching as soon as a large enough free cell is found.

2) Best fit :- Allocate the smallest hole i.e. big enough search the entire list until the list is kept ordered by size. This produces the smallest left over hole.

3) Worst fit :- Allocate the largest hole search the entire list until the list is stored by size. This stratigy produces the largest left over hole.

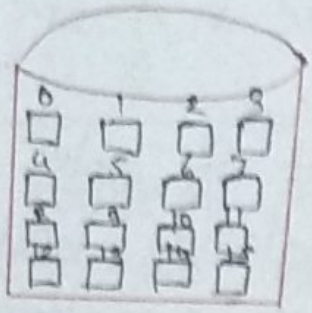
Example:- File → 3

	1	2	3
Free Space	2 | 3	6, 5, 7, 8	13, 14, 15

First fit - 2

Best fit - 3

Worst fit - 2



The algorithm first fit, Best fit, and worst fit Suffer from external fragmentation. External fragmentation exist when enough disk space exist to satisfy a request but is not contiguous. Storage is fragmented into a no. of small holes. The Soln to external fragmentation is a technique called compaction.

Compaction:- The prevent lossing significant amount of disk space to external fragmentation a re-paking routine is run which copies the entire file System onto another floppy disk or tape. The original floppy is completely freed executing one large contiguous hole. The file then can be copied back by allocating contiguous space from this one large hole. This Scheme efficiently Compacts all the free spaces into one hole Solving fragmentation.

2) Linked - Allocation Method :- In linked allocation each file a linked list of disk blocks. Disk blocks may be scattered any where on the disk. The directory contains a pointer to the first and last blocks of the file. Each block contains a pointer to the next block of the file. With linked allocation each directory entry for the file has a pointer to the first block of the file. And this is initialized to NULL to signify an empty file the directory entry for the file contains the address or pointer to the last block of the file.

File	Start	End
X	8	25

The advantages of the linked allocation is that there is no external fragmentation problem as any free block from the free space list can be used to satisfy a request since all blocks are linked together.

Disadvantages:-

(i) The linked allocation can only be used effectively for sequential access files. To find block i searching must start at the beginning of that file and follow the pointer till block i is reached.

(ii) Another disadvantage is the space required for pointer.

(iii) Reliability is less since the files are linked together by pointer scattered all over the disk and if the pointer is loossed or damaged. error will be there in the final result.

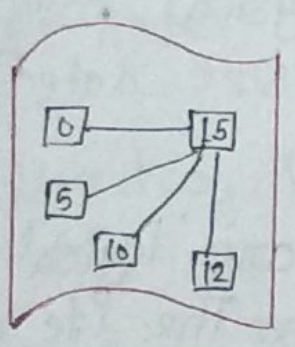
3) Indexed Allocation:- Linked allocation do not support direct access as disk blocks or scattered throughout the disk. Also the pointer to the blocks are scattered. Indexed allocation solve the problem by bringing all the pointer together into one location i.e. the index block. Each file has its own index block containing which is an array of disk block addresses.

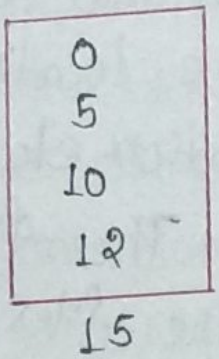
The ith entry in the index block points to the block of the file. The directory

entry contains the address of the index block. To read ith block pointer in the ith index block entry is used to find and relief the desired block.

When file is created all the pointer in the index block are set to NULL. When ith block are set is first written a block is removed from the free space list and its address is put in the ith block in the index block. Index block allocation supports direct access without suffering from external fragmentation.

File Name	Index block
X	15





It suffers from wasted space with index allocation method. An entire index block must be allocated even if one or more pointers will be NULL. Every file must have an index block for large files several index files may be

linked together to access a block. The OS uses a first level index and then can get the desired data block.

Directory System:- The files are represented by the entries in the device directory or the volume table of contents. A directory structure provides a mechanism for organizing many files in the file system.

There are two types of directory structure:-

1. Device directory
2. File directory

The device directory stored on each physical device. The device directory entry describes all the files on that device and physical properties of each files i.e. location, address, size, date and time of creation etc.

The file directory are logical organization of the files on all devices. The file directory entry describes logical properties of each file. i.e. name, file type, owner of the file, protection access method / code etc. The operation to be performed on the directories are -

1. Search :- To search a directory structure to find entry for a particular file.

2. Create directory :- The directory is created and the files are added to the directory.

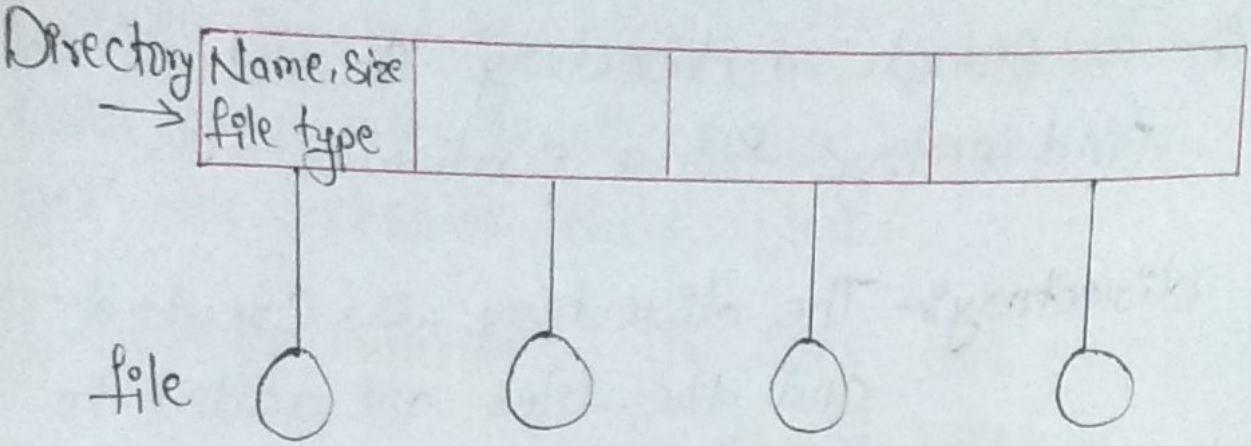
3. Deleting :- When files are no longer needed, delete all the files and remove the directory

4. List the directory.

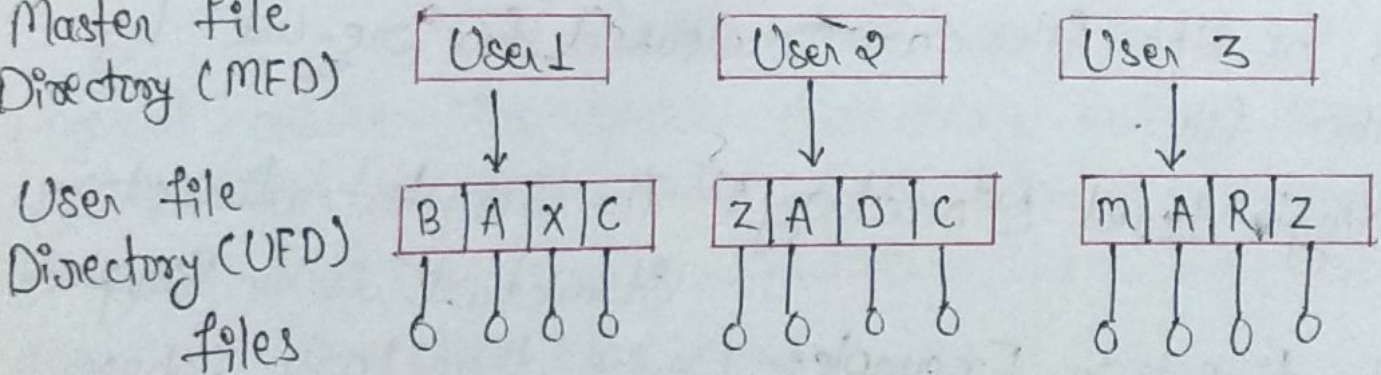
5. Backup :- Copying all the files from the disk to another disk or drive and the disk space of the files are released for re-use by another file.

I. Single Level Directory :- The simplest directory structure is a single level directory. Example :- Device directory where all the files are contained in the same directory. Therefore each file must have a unique name.

The limitation is that the no. of files increases or there is more than one user. The problem arises as no. of files increases to create unique name of the file.



II. Two level Directory Structure :- The major disadvantages is the file names betⁿ different user's in the case of single level directory structure. The standard solⁿ is to create separate directory for each user. The user directory is logical.



In two level directory structure each user has its own user file directory (UFD). that list the files directory (UF for single user's. The master file directory (MFD) is indexed by user name or access no. and each entry points to the user directory for that user.

When a user refers to a particular file their own user directory is searched. For that particular user file. Different user may have files with same name but file name should be unique for a user to create a file for a user or searches only that user file directory for any duplicate name.

The advantage with two level directory structure is that it is effectively isolated one user from another user.

The two level directory structure is termed as tree or height two whose root is MFD and its direct descendents are UFD's whose discendents are files. Files are the leafs of the trees.

To store system file in two level directory structure two methods are-

i) To copy system files into each user file directory. This leads to space wastage.

ii) Special file directory is defined to contain the system files OS will search for a system file within the local user directory first if the file is found it is used. If the file is not found the system automatically searches the

Special user directory which contains the System files. The sequences of directories Searches when a file is named is called Search path.

III. Tree Structure directory:- The directory Structure can be extended to an arbitrary tree which allows the users to Create their own sub-directories and organize their files accordingly. The tree has a root directory and every file in the system has unique path name. A path name is the path from root through all the sub-directories to a specified file.

To Search a file, firstly the Current directory of the user is Searched and if the file is not found in the Current directory then the user must Specify a path name or Change the Current directory. A path name is of two types path name and relative pathname. A Complete pathname begins at the root and follows path to the Specified file giving the directory names on the path. The relatives pathname defines a path from the Current directory.

IV Acyclic graph directory Structure:- A common sub-directory which is used by more than 1 user should be shared. A shared directory or file will exist in the file system in two or more places. A shared file is not same as two copies of the file.

In Copies of the file if one user changes the file, changes will not appears in other copy.

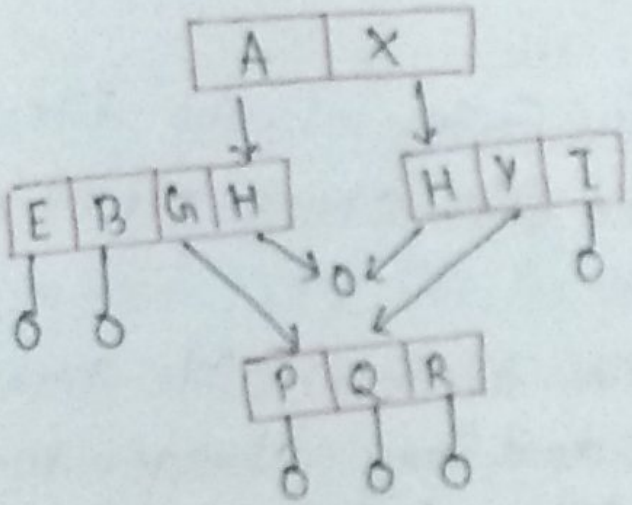
With a shared file there is only one actual file and any changes made by any user would be visible to others. For shared sub-directories a new file created by one user will automatically appear in all of the shared sub-directories.

Acyclic graph can be implemented for shared files and sub-directories in two ways -

(i) One way is to create a new entry directory called link. A link is a pointer to another file, of sub-directory. For example - A link may be implemented as a complete pathname when a reference to a file is made search the directory. The directory entry is mark as a link and

the name of the real file is given. The link is resolved by using the pathname to locate the real file.

(ii) Another approach is to duplicate all the info. related to file in both the sharing directories.



An acyclic graph directory structure is more flexible than simple tree structure. A file may have multiple complete pathname. Distinct file names may refer to the same file.

File Protaction:- The info. in the computer system have to be protected from physical damage and unauthorized access. Reliability is provided by making duplicate copies of the file. Protaction mechanism provide controlled access i.e. read or write access or execute access.

File protaction can be provided by password, access list or by special file protaction adhoc mechanism. The simplest approach is to use a password with each file. If passwords are randomly chosen and changed frequently than the method is effective in limiting the file access. If one password is associated with each file then to provide protaction on detailed level multiple passwords are needed.

If separate password is associated with each file the no. of passwords increases. If one password is used for all the files all the files are accessible using one password and protaction is on and all or nothing bases.

Another approach is to make access dependent on the identity of the user and access list can be associated with each file and directory specifying user names and types of access. allot for each user can be provided. When a user requests the file, the OS checks the access list if the user is listed for the requested access the access is allowed else a protaction volution access occurs and users request is terminated.

An System recongnises three types of users -

Owner, group and other. Each file has an owner i.e. the user who Created the file. The owner may belong to a group of users for sharing the file and need similar access.

Finally there is every one else in the system who fall in other type.

UNIT-II

* Processor Management (CPU Scheduling) *

Scheduling :- Operating System is a collection of program to manage the system resources i.e. processor, memory, I/O devices and file system. It is a job of an OS to see that resources are used in very efficient & co-operative manner. The Operating System must keep track of the status of each resource. Allocate resources to different processes based on certain policy decide how long these processes will be utilizing these resources and finally deallocate them.

Scheduling is a fundamental Operating System function. All comp. resources are scheduled before use. Since CPU is the primary comp. resource its scheduling is central to the operating system.

A Scheduling is an Operating System program that selects the next job to be admitted for execution.

CPU Scheduling is the basic of OS which supports multiprogramming concepts. This mechanism improves the overall efficiency of comp. System by getting more work done in less time.

Process State:- A process is a program in execution as the program executes the process changes state. The state of a process is defined by its current activity. Process execution is an alternative sequence of CPU and I/O execution. Each process may be in one of the following states.

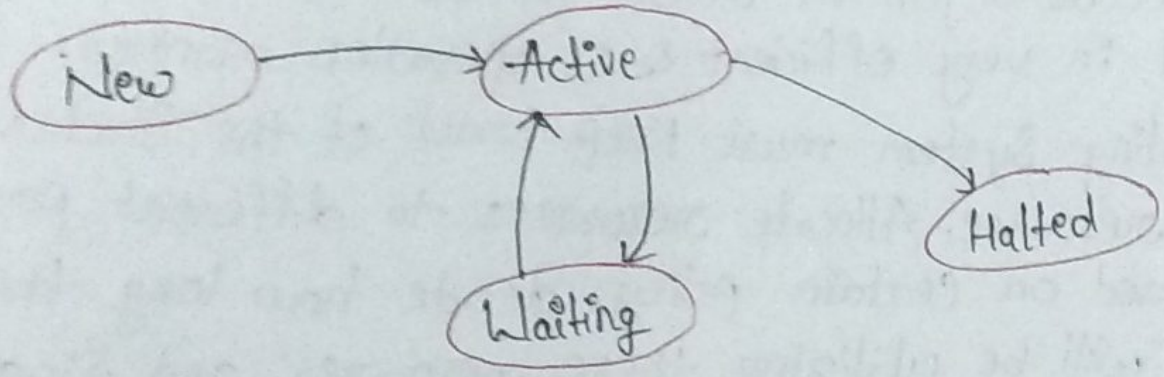


fig:- Process State Diagram

A process is in new state when it is just created. An active process is a process in execution. A process which is waiting for the CPU is also called ready state. A process which has been allocated to CPU is said to be running state.

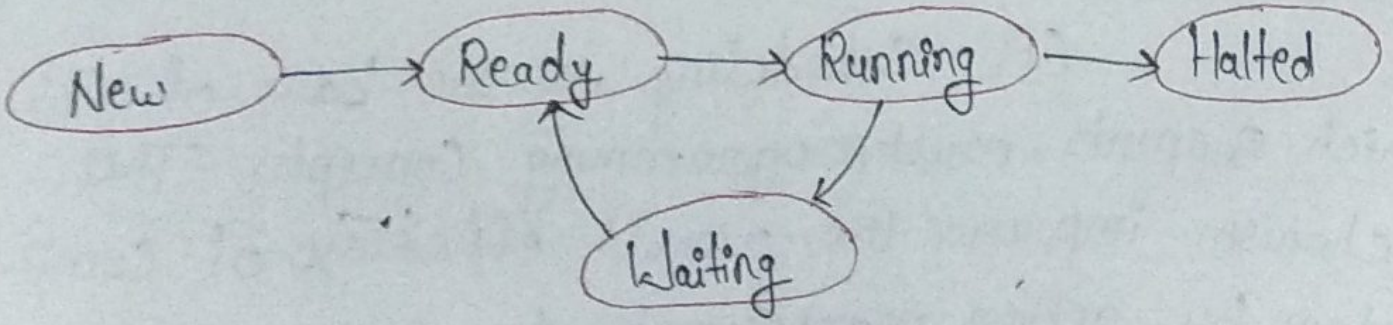


fig:- Refined Process State diagram

Process Control Block (PCB):- Each process is represented in the Operating System by its own process control block also called task control block or job control block. A process control block is a data block containing infor. associated with a specific process. Such as-

1) Process state i.e. new, ready, running, waiting or Halted.

2) Program Counter indicates the address of next instruction to be executed by the CPU or I/O device for the process.

3) The Registers which include accumulator, general purpose PC etc. All the register info. is there in the block.

4) Any memory mgt info. such as base and bound register, pch cable etc.

5) Accounting info. including the amount of CPU & device time used, time limits, job or process no.s etc.

6) I/O status information, I/O devices, I/O requested, List of open files etc.

7) CPU Scheduling information including process priority, pointer to scheduling queues etc.

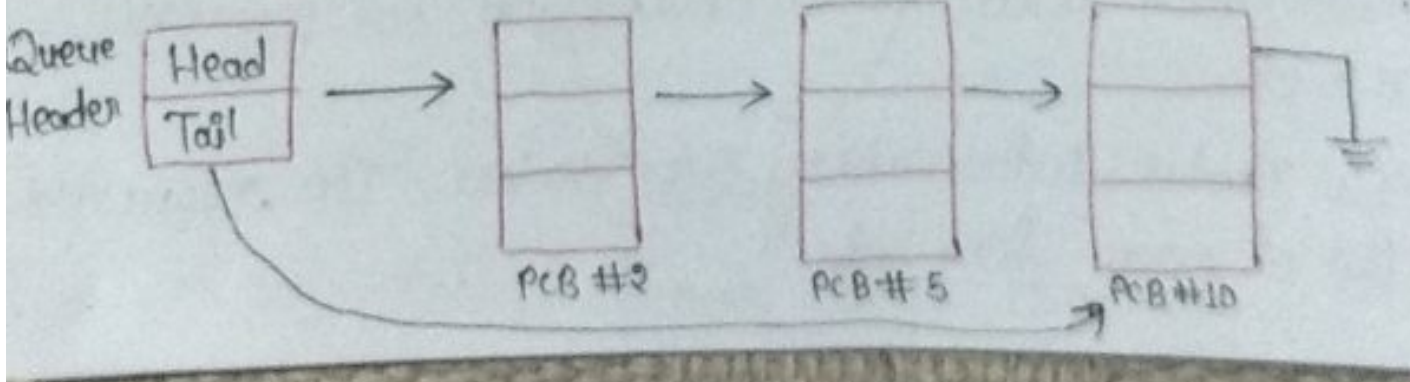
PCB

Process id
State
pointer
priority
Program Counter
CPU Register
I/O info.
Accounting

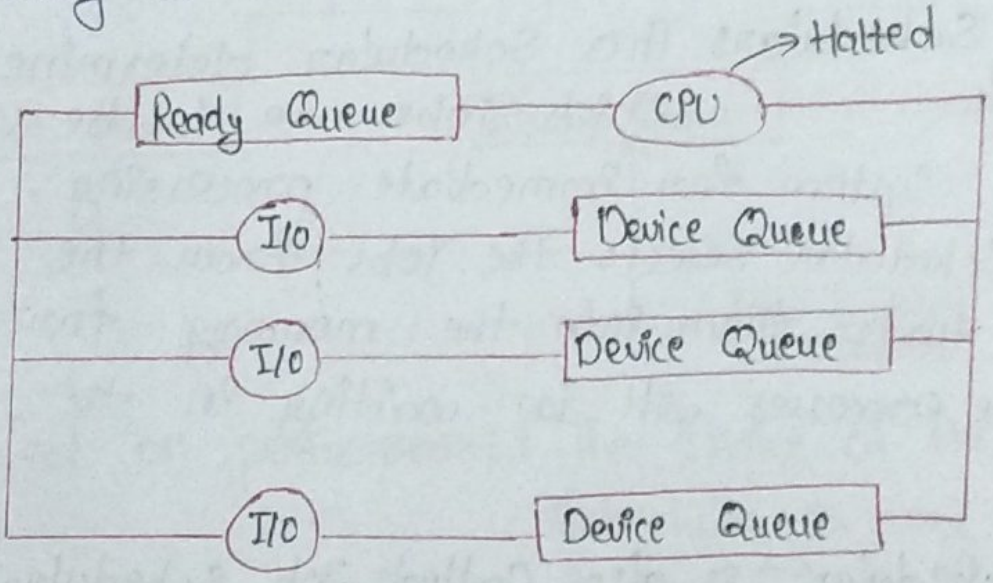
Process State
Program Counter
Registers
Memory mgt
Process no.

No. of open files
privileges

Scheduling Queues:- In uniprocessor system there will be only one running process always. The objective of multiprogramming is to have a process running at all the times in order to maximize CPU utilization. The rest of the programs will have to wait until the CPU is free and be rescheduled. The processes which are ready and kept on a waiting to be executed are kept on a list called the ready queue which is a type of linked list. A ready queue header will contain pointers to first and last process control blocks or PCB's of the queue.



Each PCB has a pointer field which points to the next process in the ready queue. A ready queue may be implemented as FIFO queue, priority queue, tree, stack, or an unordered linked list. There are also other queues in the system when a process is allocated the CPU it executes for a while then it either quits or waits for an I/O request. Since there are many processes in the system the disk may be busy in the I/O request of other processes & the process may have to wait for the disk. The list of processes waiting for the I/O device is called as a device queue.



Each device has its own device queue, if the device is a dedicated device. The device queue will never have more than one process in it. If the device is sharable several processes may be in the device queue. Each rectangle box represents a queue. The circle represents the resources which serve. The queues and arrows indicate the flow of the process in the system. A process enters the system and its

put in the ready queue. It waits in the ready queue till it is selected for CPU. After the execution it waits for I/O operation in I/O queue or device queue. It is served to the ready queue. A process continuous those CPU. I/O cycle until it finishes and then it leaves from the System.

An operating System has many Schedular-

1. Long term Schedular
2. Medium term Schedular
3. Short term Schedular

1) Long term Schedular: This Schedular determines which jobs are to be admitted to the System for immediate processing. A long term Schedular selects the jobs from the job pool and loads them into the memory for execution. These processes will be waiting in the ready Queue.

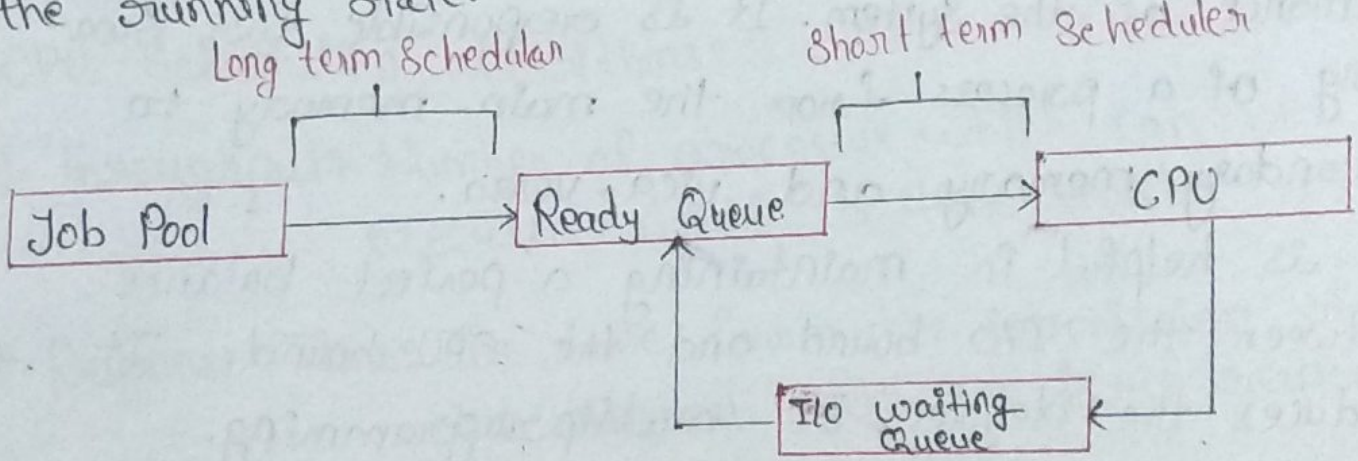
OR Job Schedular

Long-Term Scheduler is also called Job Scheduler and is responsible for controlling the Degree of multiprogramming i.e. the total number of processes that are present in the ready State.

2) Short term Scheduler: Short-term Scheduler are those Schedulers whose decision will have a short-term effect on the performance of the System. The duty of the short

term Scheduler is to Scheduler the process from the ready State to the running state. This is the place where all the scheduling algorithms are used i.e. It can be FCFS or Round-Robin or SJF or any other scheduling algorithm.

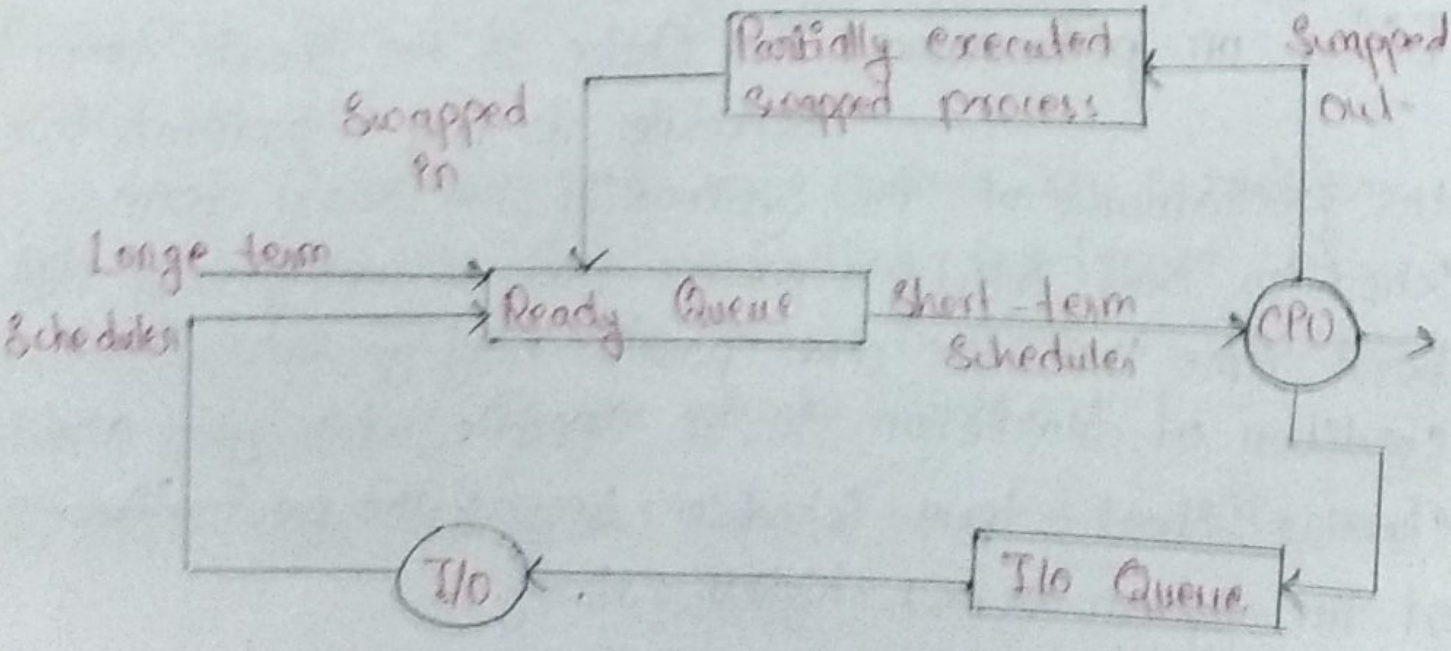
Short-Term Scheduler is also known as CPU Scheduler and is responsible for selecting one process from the ready State for scheduling it on the running state.



Effect on performance: The choice of the Short-term Scheduler is very important for the performance of the System. If the short term scheduler only selects a process that is having very high burst time then the other process may go into the condition of Starvation. So, be specific when you are choosing Short-term Scheduler because the performance of the System is our highest priority.

3) Medium term Scheduler: Sometimes, you need to send the running process to the ready state or to the wait/block state. For example, in the round-robin process, after a fixed time quantum, the process is again sent to the ready state from the running state. So, these things are done with the help of medium-term scheduler.

Medium-term scheduler are those scheduler whose decision will have a mid-term effect on the performance of the system. It is responsible for swapping of a process from the main memory to secondary memory and vica-versa.
It is helpful in maintaining a perfect balance between the I/O bound and the CPU bound. It reduces the degree of multiprogramming.



Scheduling Algorithms: Scheduling algorithm support preemtive or non-preemtive deciplines of Scheduling. A Scheduling decipline is non-preemtive if once the process has been given the CPU, the CPU cannot be taken away from that process. A Scheduling algorithm is preemtive, if the CPU can be snatched away from the process.

Scheduling & performance Criteria for Comparing CPU Scheduling algorithms:

1. Throughput: Number of processes which can be executed within a time limit.

2. Response Time: The time from the submission of the request until the first response is produced is called response time.

3. Turn around time: The time interval from time of submission to the time of Completion is called turn around time. Turn around time is the sum of time periods, spent, waiting in the queue, waiting to get into the memory, execuding on the CPU and completing I/O operation.

4. Waiting Time: The algorithm only affects by the amount of the time that the job spends waiting in the ready queue.

5. CPU utilization: The algorithm is chosen for increasing the utilization of the CPU by taking a good mixture of I/O bound and CPU bound jobs.

The Criteria of choosing the scheduling algorithm is to maximize CPU utilization and throughput and to minimize turn around time waiting time & response time.

FCFS (First Come first Serve):- FCFS is the simplest (Non-preemtive) Scheduling algorithm i.e. the process that requests the CPU first will be allocated the CPU first and the FCFS is maintained by FIFO queue. When a process enters the ready queue its PCB is linked to the tail of the queue. When CPU is free it is allocated to the process which is at the head of the queue. The running process is then removed from the ready queue. FCFS Scheduling is non-preemtive which results in poor performance.

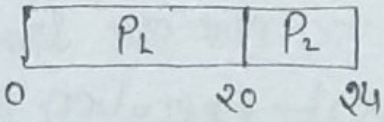
Short jobs suffer considerable turn around delays and waiting time when CPU has been allocated to longer jobs.

Consider two processes P_1 and P_2 arriving in the order P_1, P_2 with their corresponding time 20 and 4.

Process	Execution Time / Burst time (MS)
P_1	20
P_2	4

$P_1 - P_2$

Gantt Chart



Average turn around time = $\frac{(20-0)+(24-0)}{2}$

= 22 ms

Average waiting time = $\frac{0+20}{2}$ = 10ms

If P_1 and P_2 arrive in the order P_1-P_2, the turn around times for P_1 and P_2 are 20 and 24 respectively. Giving an average of 22ms. P_2 must wait for P_1 to complete and the corresponding waiting times for P_1 and P_2 are 0 and 20 and average waiting time is 10ms.

When the same processes arrived in the order P_2-P_1, the turn around times are 4 and 24 respectively. Giving an average of 14ms. The average waiting time is 2ms. There is a Substantial reduction in average waiting turn around

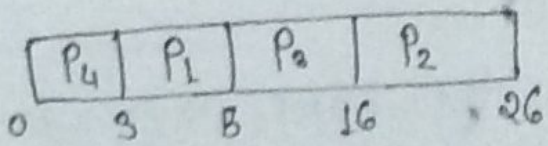
time and average waiting time. FCFS algorithm may vary substencially and there is a convey effect as all the processes wait for one big process to get the CPU. This results in lower CPU and device utilization than if shorter jobs are given the CPU first.

SJF (Shortest Job first):- SJF Scheduling of a job or process is done on the basic of its having shortest execution time. If two processes have the same CPU time FCFS is used. (Non-preemtive)

SJF associates with each job the length of its next CPU execution time. Consider the following set of processes -

Process	Execution Time/ Burst Time
P_1	5
P_2	10
P_3	8
P_4	3

Giantt Chart:



Average turn around time = $\frac{(3-0)+(8-0)+(16-0)+(26-0)}{4}$

$= \frac{53}{4}$

$= 13.25$ ms

Average waiting time = $\frac{0+3+8+16}{4}$

$= \frac{27}{4}$

$= 6.75$ ms

SJF Scheduling is an optimal scheduling algorithm with respect to minimum average time for a given set of processes. SJF proves that moving a short job before a longer job results in decreased waiting time of the shorter job SJF works optimally only when the exact future execution time of the jobs are known at the time of scheduling.

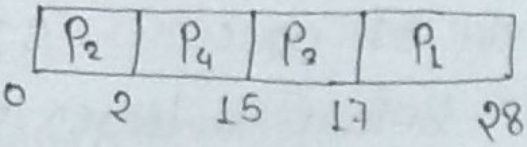
Priority Scheduling Algorithm: SJF is a special case (Non-preemtive) of priority scheduling algorithm. Each job is assigned a priority and CPU is allocated to a job with highest priority. FCFS is used to schedule equal priority jobs. SJF is a priority scheduling algorithm where priority is inverse of CPU execution time. The larger the CPU execution time lower the priority in SJF.

Consider the set of processes in the order, P_1, P_2, P_3, P_4 with CPU execution time

11, 2, 2, 13 and the priority 4, 1, 3 2.

Process	CPU execution Time	Priority
P_1	11	4
P_2	2	1
P_3	2	3
P_4	13	2

Gantt Chart:



Average turn around time = $\frac{(2-0)+(15-0)+(17-0)+(28-0)}{4}$

$= 62/4$

$= 15.5$

Average waiting time = $\frac{0+2+15+17}{4}$

$= 34/4$

$= 8.5$

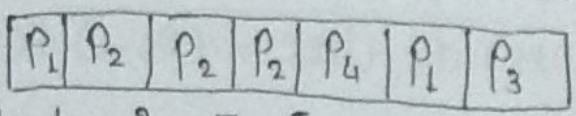
A major problem with the priority Scheduling algorithm is indefinite blocking or starvation. A process which is ready to run but lacking the CPU can be considered blocked waiting for the CPU. A soln to indefinite blockage of low priority job is Aging. Aging is a technique which

gradually increases the priority of the jobs that wait in the system or ready queue for a long time.

SJF, FCFS and priority scheduling algorithms are non-preemtive scheduling algorithm i.e. once the CPU is allocated to a process, it will be released only when the process once. Completed. SJF may be either preemtive or non-preemtive. The question arrises when a new job arrives in ready queue when the previous job is executing. The new job may have shorter next CPU execution time than what is left of currently executing job. A preemtive SJF algorithm will preempt the currently-executing job. Preemtive SJF is called SRTF i.e. shortest remaining time first

Process	CPU execution Time	Arrival time
P_1	8	0
P_2	4	1
P_3	9	2
P_4	5	3

Grantt chart:



Average turn around time = (17-0) + (5-1) + (26-2) + (10-3)
TAT = Compt^n time + AT

$= 52/4$

$= 13$

Average waiting time = $\frac{(10-1)+0+(17-2)+(5-3)}{4}$

$= 6.5$

NOTE - Criteria - Burst Time, Mode - Preemptive
↓
STRF

TAT = CT - AT

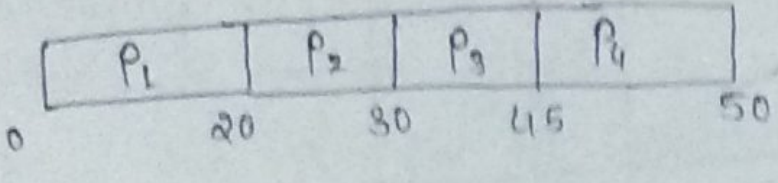
LIT = TAT - BT (Burst Time)

RT = (CPU first time - AT)

Q.

Process	CPU execution Time	Arrival Time
P_1	20	0
P_2	10	4
P_3	15	6
P_4	5	10

(I) Grantt chart - FCFS



Average turn around time = (20-0) + (30-0) + (45-0) + (50-0)

4

$= \frac{145}{4}$

$= 36.25$

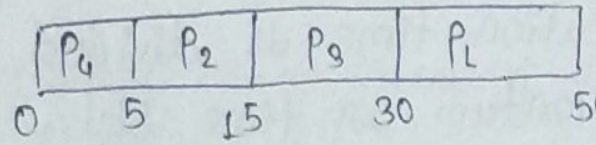
Average waiting time = Rest - AT

$= \frac{0 + 20 + 30 + 45}{4}$

$= \frac{95}{4}$

$= 23.75$

② SJF - Gantt Chart:



Average turn around time = $\frac{(50-0)+(15-0)+(30-0)+(5-0)}{4}$

$= 100/4$

$= 25$

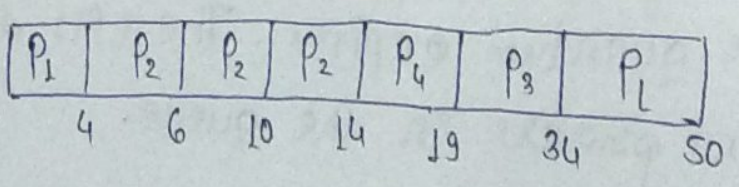
Average waiting time = $\frac{30 + 5 + 15 + 0}{4}$

$= \frac{60}{4}$

$= 15$

③ Priority Scheduling - (SRTF)

① Gantt Chart:



Average turn around time = $\frac{(50-0)+(14-4)+(34-6)+(19-10)}{4}$

$= 24.2$

Average waiting time = $\frac{(34-4)+0+(19-6)+(14-10)}{4}$

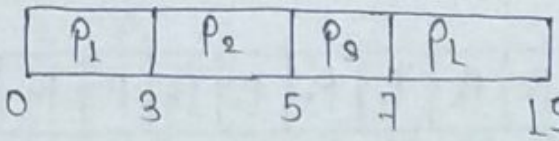
$= 11.75$

Round - Robin Algorithm: The RR algorithm is used in time sharing & multi-user system. The CPU execution time is divided into time slices or time quantum or time slot. The ready queue is treated as a circular queue, the CPU Scheduler goes round the circular queue allocating the CPU to each process for a time interval of upto 1 quantum of length. If the CPU execution time of currently running process is longer than 1 time quantum, the timer will go off and will cause and interrupt to OS and the process will be put at the tail of the ready queue. The CPU schedular will then select the next process in the ready queue. If a process needs less time than 1 time quantum expires. The CPU is given to the next process in the queue.

Process	CPU execution Time (ms)
P_1	15
P_2	2
P_3	2

Time Slice = 3ms

Gantt chart:



Average turn around time = $\frac{(19-0)+(5-0)+(7-0)}{3}$

$= \frac{31}{3}$

$= 10.33$

Average waiting time = $\frac{(7-3)+(3-0)+(5-0)}{3}$

$= \frac{4+3+5}{3}$

$= \frac{12}{3}$

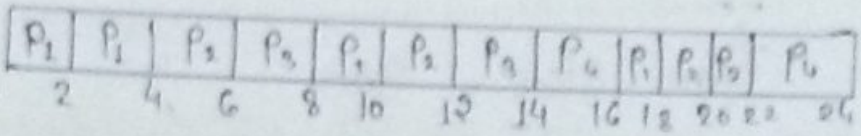
$= 4$

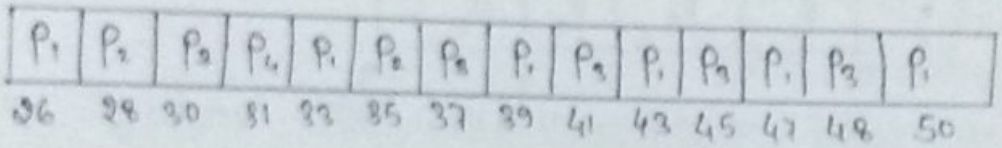
Q

Process	CPU exe. Time	Arrival Time	Priority
P_1	20	0	2
P_2	10	4	1
P_3	15	6	1
P_4	5	10	2

Time Slice - 2ms

1. Gantt chart:





Average turn around time = $\frac{(50-0)+(35-4)+(48-6)+(31-10)}{4}$

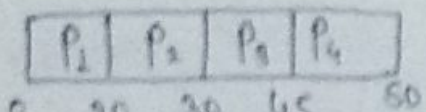
= 36 ms

Average waiting time = $[(8-4)+(16-10)+(24-18)+(7-26)$
$+(37-33)+(41-39)+(45-43)+(48-47)+[(10-6)+(18-12)$
$+(26-20)+(33-28)+(12-8)+(20-14)+(28-22)+(35-30)+$
$(39-37)+(43-41)+(47-45)+(14-10)+(30-24)]$

$\frac{}{21}$

= 4.093

2. Priority / FCFS:



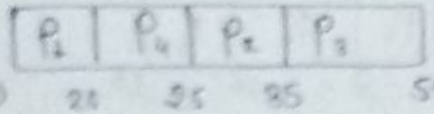
Average TAT = $\frac{(20-0)+(30-4)+(35-6)+(50-10)}{4}$

= 26.25

Average waiting time = $\frac{(0-0)+(20-4)+(30-6)+(43-10)}{4}$

= 18.25.

Q. SJF:



Average TAT = $\frac{(20-0)+(35-4)+(50-6)+(35-10)}{4}$

= 27.5

Average waiting time = $\frac{(0-0)+(35-4)+(35-6)+(20-10)}{4}$

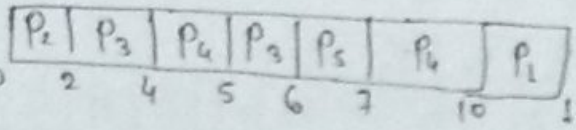
= 15

Q. Using preemtive SJF (SRTF), FCFS and Round Robin T-S = (2ms). Compute Average turn around time and Average waiting time which of these is the best.

Process	CPU Burst Time (ms)	Arrival Time (ms)
P_1	8	0
P_2	2	0
P_3	4	2
P_4	3	4
P_5	1	5

1. Preemtive SJF (SRTF):

Gantt chart:



Average turn around Time = $\frac{(18-0)+(2-0)+(6-2)+(10-4)+(7-5)}{5}$

$= 31/5$

$= 6.2$ ms

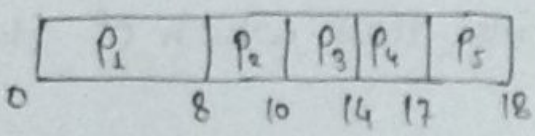
Average waiting time = $\frac{(10-0)+(0-0)+(2-2)+(7-4)+(6-5)}{5}$

$= 14/5$

$= 2.8$ ms.

2. FCFS:

Gantt chart:



Average turn around time = $\frac{(8-0)+(10-0)+(14-2)+(17-4)+(18-5)}{5}$

$= 56/5$. 11.5 ms

Average waiting time = $\frac{(0-0)+(8-0)+(10-2)+(17-4)+(18-5)}{5}$

$= 42/5$

$= 8.4$ ms

3. Round Robin:

Gantt Chart:

P_1	P_3	P_4	P_5	P_2	P_4	P_1

0 2 4 5 6 8 10 18

Av. TAT = $\frac{(18-0)+(2-0)+(8-2)+(10-5)+(6-0)}{5}$

$= \frac{35}{5} = 7$

Av. waiting time = $\frac{(10-0)+(0-0)+(2-2)+(6-4)+(8-4)+(5-0)}{6}$

$= \frac{21}{6}$

$= 3.5$ ms

Q

Process	CPU Burst time (ms)	Arrival Time (ms)
P_1	8	0.0
P_2	4	0.4
P_3	1	1.0

(1) FCFS: Gantt Chart:

P_1	P_2	P_3

0 8 12 13

Average TAT = $\frac{(8-0)+(12-0.4)+(13-1)}{3}$

$= \frac{31.6}{3}$

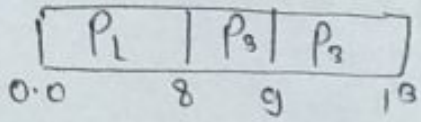
$= 10.53$ ms

Av. waiting time = $\frac{0+(8-0.4)+(12-1)}{3}$

$= 6.2$ ms

② SJF:

Gantt chart:



Average TAT = $\frac{(8-0)+(13-0.4)+(9-1)}{3}$

$= \frac{28.6}{3}$

$= 9.53$ ms

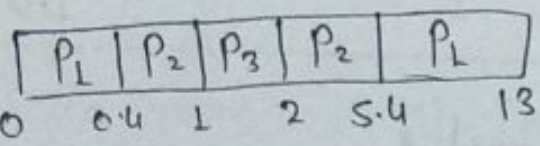
Average waiting time = $\frac{(5-0)+(1-0.4)+(0-1)}{3}$

$= \frac{4.6}{3}$

$= 1.53$ ms

③ SRTF:

Gantt chard:



Average TAT = $\frac{(13-0)+(5.4-0.4)+(2.0-1.0)}{3}$

$= \frac{19}{3}$

$= 6.33$ ms

Average waiting time $= \frac{(5.14 - 0.43) + (8 - 1) + (1 - 1)}{3}$

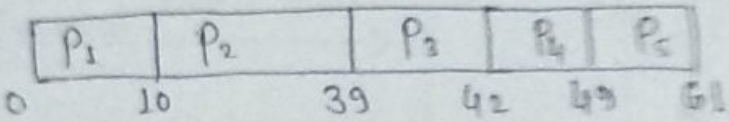
$= 9$ ms

Q

Process	CPU Burst Time (ms)
P_1	10
P_2	29
P_3	3
P_4	7
P_5	12

(I) FCFS:

Gantt Chart:



Average TAT $= \frac{(10-0) + (39-0) + (42-0) + (49-0) + (61-0)}{5}$

$= \frac{201}{5}$

$= 40.2$ ms

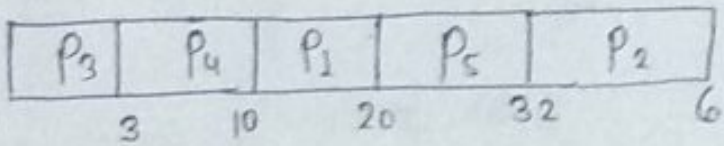
Average WT $= \frac{(0-0) + (10-0) + (39-0) + (42-0) + (49-0)}{5}$

$= \frac{140}{5}$

$= 28$ ms

② SJF:

Gantt Chart:



$$\text{Av. TAT} = \frac{(20-0)+(61-0)+(3-0)+(10-0)+(32-0)}{5}$$

$$= \frac{126}{5}$$

$$= 25.5 \text{ ms}$$

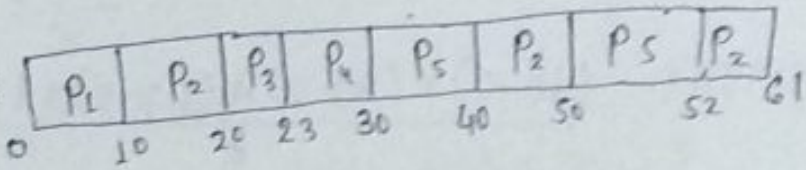
$$\text{Av. WT} = \frac{(10-0)+(32-0)+(0-0)+(3-0)+(20-0)}{5}$$

$$= \frac{65}{5}$$

$$= 13 \text{ ms}$$

③ RR (10ms):

Gantt chart:



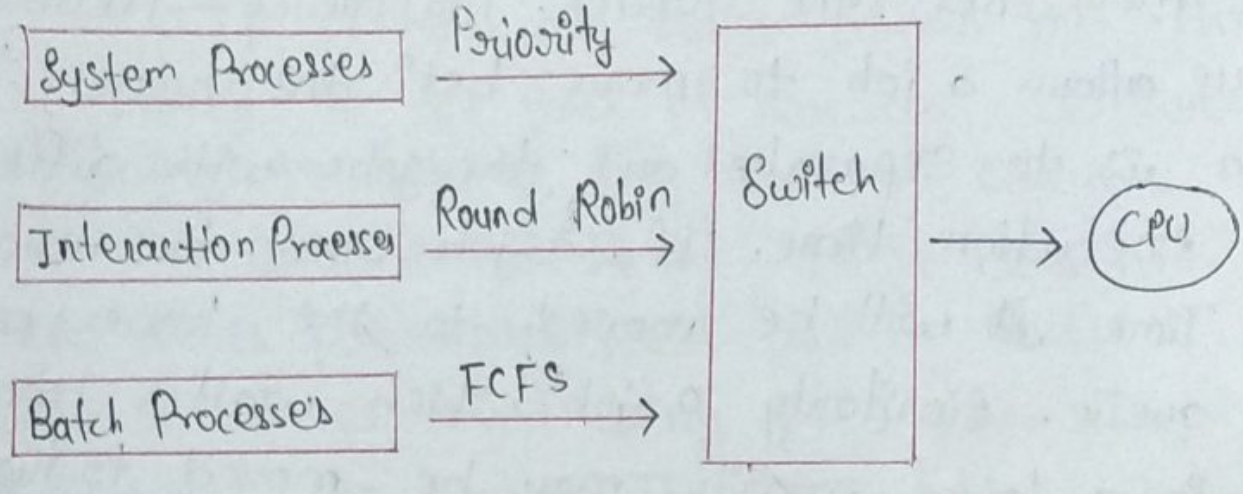
$$\text{Av. TAT} = \frac{10+61+23+30+52}{5}$$

$$= \frac{176}{5} = 35.2 \text{ ms}$$

$$\text{Av. WT} = \frac{(0-0)+[10+(40-20)+(52-50)]+(20-0)+(23-0)+[(30-0)+(50-40)]}{5}$$

$$= 23$$

Multi-level Queues: Jobs are classified into 2 different groups called back-ground jobs and fore-ground jobs. A multi level queue scheduling algorithm partitions the ready queue into separate queues. According to the different types of jobs. Memory size needed by the jobs etc. and each queue has its own scheduling algorithm. Scheduling processes are classified into different groups.



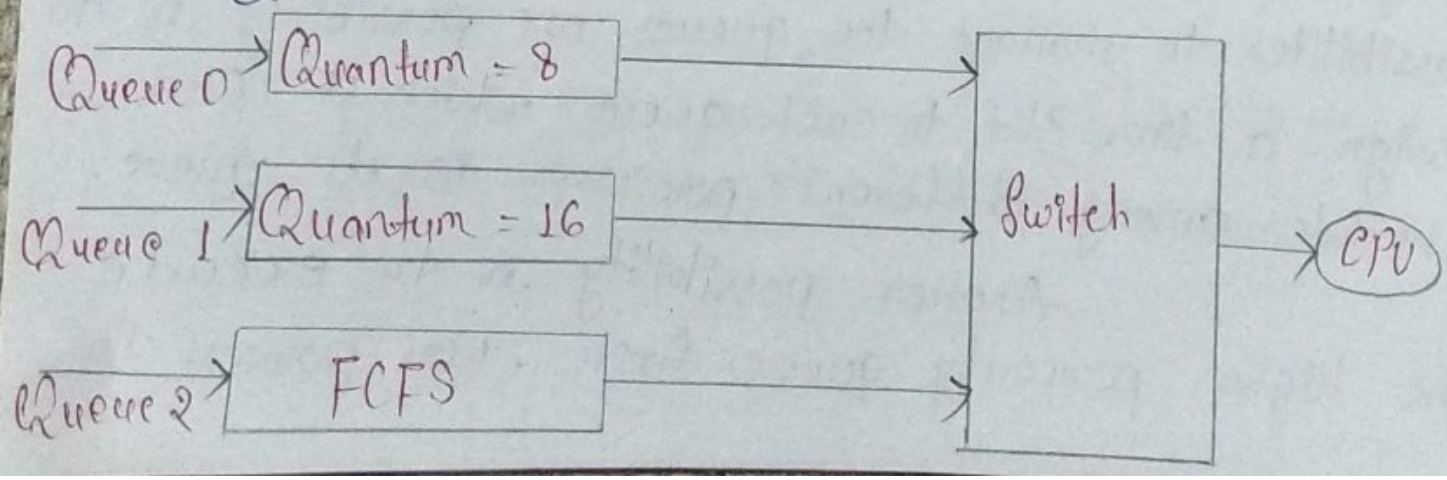
Each queue is seived by some scheduling decision best suited to the type of the processes stored in the queue. Each queue has priority over the lower priority queues. There are many possibilites to manage the queues one possibility is to assign a time slice to each queue which it can schedule among different processes in its queue.

Another possibility is to execute the higher priority queue first. No process in

the batch queue could run unless the queue for system processes and interactive processes are empty. If an interactive entered the ready queue while a batch process is running the batch process would be preemted.

Multilevel feedback Queues: In a multilevel or multiqueue scheduling algorithm jobs are permanently assigned to a queue upon entry to the system. And the jobs do not move betn the queues. Multilevel feedback queue allow a job to move betn the queues. The idea is to separate out the jobs with different CPU execution time. If a job uses too much CPU Time it will be moved to the lower priority queue. Similarly a job which waits too long in a lower priority may be moved to higher priority queues.

Consider a multilevel feedback queue Schedular with 3 queues Queue 0, Queue 1, and Queue 2.



The Schedular first executes all the jobs in Queue 0. the jobs in Queue 1 will be executed only when Queue 0 is empty. Similarly jobs in Queue 2 will be executed only if Queues 0 and 1 are empty. A job which arrives for Queue 1 will preempt a job in Queue 2. A job in Queue 1 will be preempt by a job arriving for Queue 0. A job entering the ready queue is put in Queue 0 and is given a time quantum of 8 ms.

If it does not finish within this time, it is moved to the tail of Queue 1. If Queue 0 is empty the job at the head of Queue 1 is given a time quantum of 16 ms. If it does not complete it is preempted and put in Queue 2. Jobs in Queue 2 are executed using FCFS when Queue 0 and Queue 1 are empty. This Scheduling algorithm gives highest priority to any job with CPU execution time of 8ms or less. Long jobs automatically sink to Queue 2 and are served on FCFS basis. A multilevel feedback queue Schedular is defined by the following parameters:-

1) Number of queues.
2) Scheduling algorithm for each queue.

3) A method to determined when to upgrade a job to higher priority queue.

4) A method to determined when to denote a job to lower priority queue.

5) A method to determind which queue a job will enter when it needs CPU Service.

Multi Processor Queue Scheduling: In a system if multiple CPU's are available the scheduling problem is more complex. One major factor is the type of processors involved. Processors may be identical i.e a homogenous system or may be different i.e. a heterogeneous.

In heterogeneous system each processor has its own queue and its own scheduling algorithm. In homogeneous system each processor is ideal with an empty queue while another processor is very busy. To prevent this situation a common ready queue is used. All the jobs are put in the ready queue & are scheduled onto any available processor. One approach is for each processor to be self scheduling. Each processor scans the ready queue and selects a process to execute. Each processor must be programmed carefully to ensure that two processors do not choose the same process and that processors are not loosed from the queue.

Another approach solves this problem by approach appointing one processor as scheduler for other processors i.e a master Slave Structure approach is Considered.

Disk and Drum Scheduling: Disk provides the primary online Storage of info. both programs and data. Each disk has a flat circular shape with 2 Surfaces Covered Shape with 2 Surfaces covered with a magnetic material. A Read /write head is positioned above the Surface of the disk. The disk surface is divided into tracks where the info. is stored. A fixed head disk has a separate head for each track. A moving head disk has only one head which moves in and out to access different tracks. A fixed head disk is same as a drum Same/Shape as a Silender & recording is done on its side, drums are used as backing Store also called as swapping drum or pagging drum. Within a track info. Stored in blocks of fixed Size Called Sectors. Disk speed is Composed of 3 parts.

To access a block on the disk the System must move the head to appropriate track or Cilender. This head moment is Called Seem and

the time to complete is called seek time.

Once the head is at the right tracks it must wait until the desired block rotates under the read/write head. This delay is called latency time.

Finally the actual transfer of the data betn the disk and the main memory takes place. This is called transfer time.

The total time to service a disk request is the sum of seek time, latency time and transfer time. A disk service requires that the head be moved to the desired track then wait for the latency and finally transfer the data.

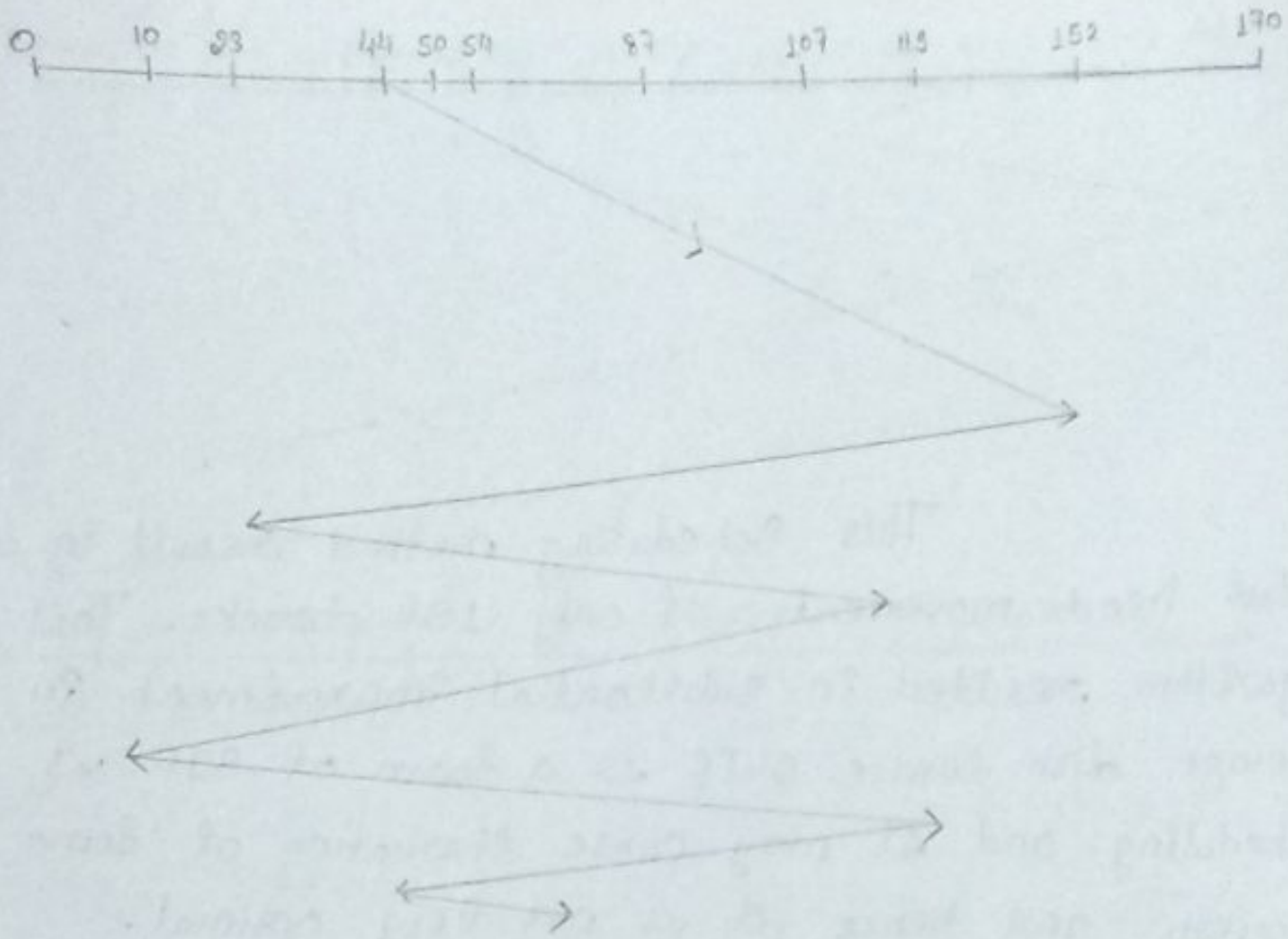
1) FCFS:- The simplest form of disk scheduling is FCFS.

Consider an ordered disk queue with request involving the track numbers 87, 152, 93, 107 10, 119, 50, 54. If the read/write head is initially at a track 44. Then the head will first move from 44.

Total Head Movement -

$(87-44)+(152-87)+(152-93)+(107-93)+(107-10)+$
$(119-10)+(119-50)+(54-50)$

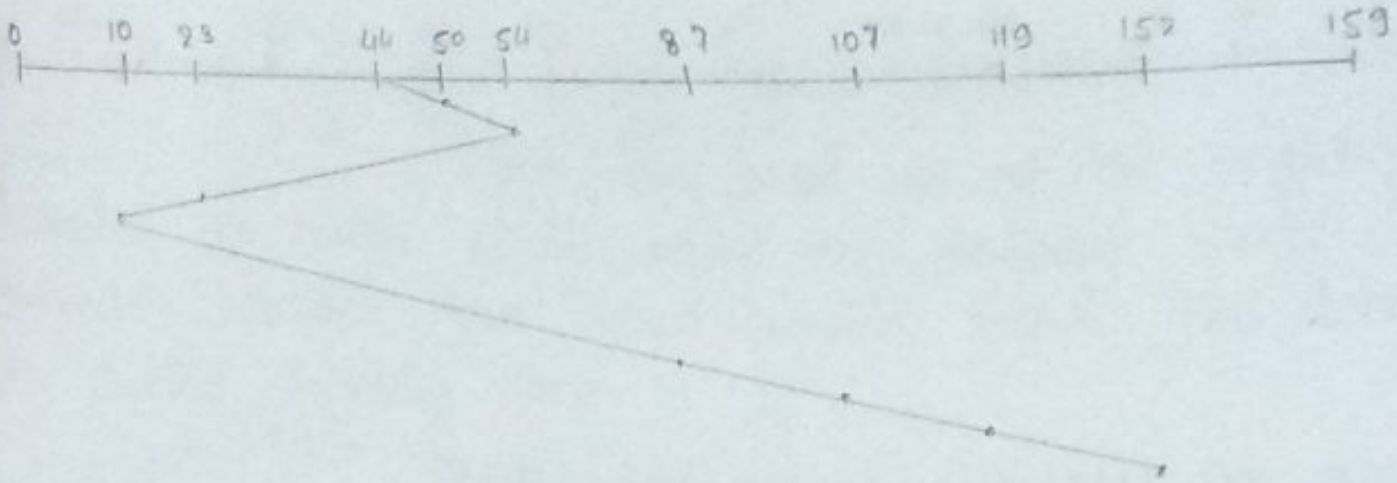
$= 600$ tracks



2) SSTF (Shortest Seek time first): The main approach is to service all the requests close to current head position. SSTF selects the request with minimum seek time from current head position. Since seek time is generally proportional to the track difference between the request SSTF is implemented by moving the head to the closes track in the request queue.

Total Head Movement = (50-44) + (54-50) + (54-23) + (23-10) + (87-10) + (107-87) + (119-107) + (152-119)

= 196 tracks.

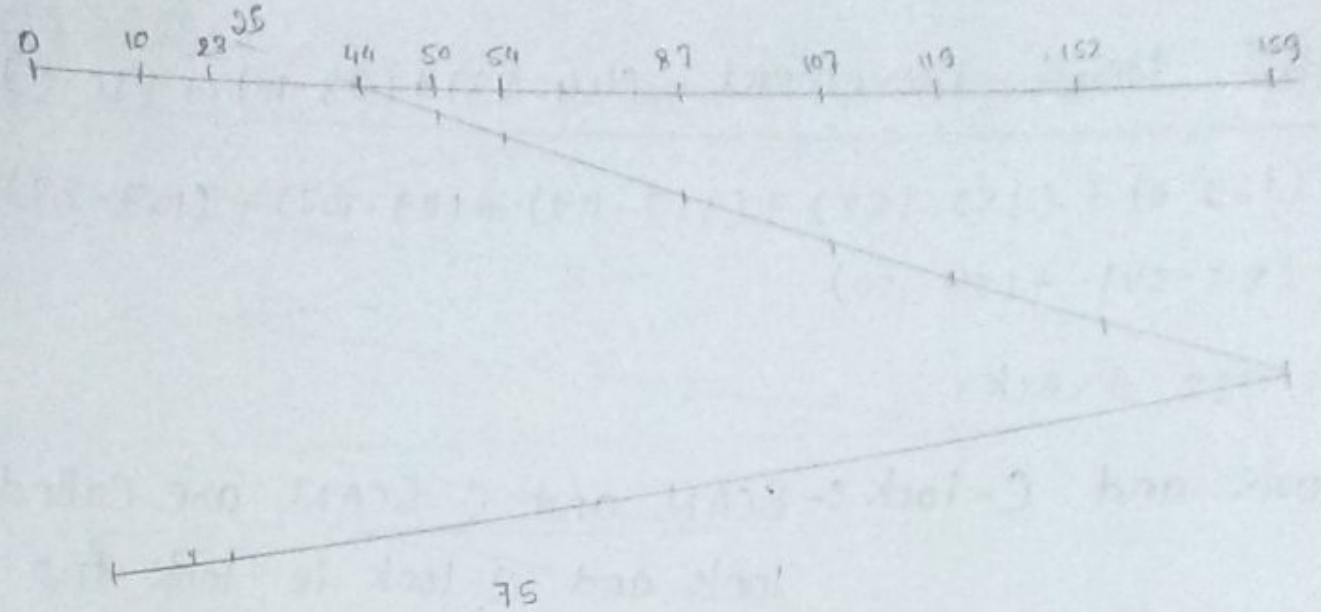


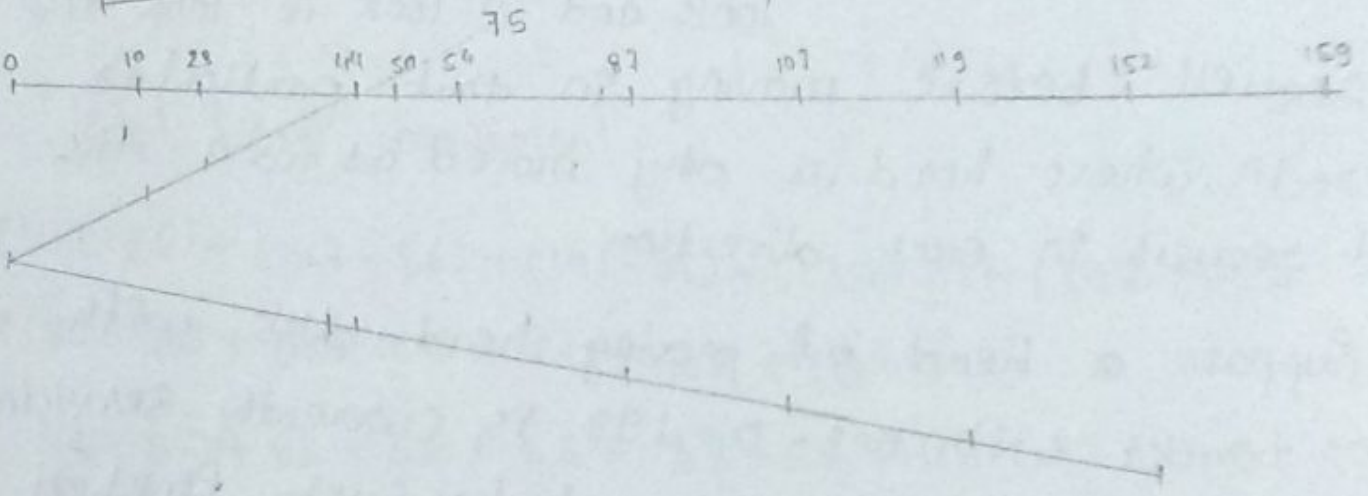
This scheduling method result in a total head movements of only 196 tracks. This algorithm resulted in substantial improvement in average disk service. SSTF is a form of SJF is scheduling and it may cause starvation of some requests and hence it is not very optimal.

3) SCAN OR ELEVATOR: The dynamic nature of the request queue leads to the SCAN algorithm. The read / write head starts at one end of disk and move towards other end servicing all the request as it reaches each track. Until it gets to another end of the disk. At the other end the head movement is reversed and servicing continue. The head continuosly scans the disk from end to end.

Before applying the SCAN the direction of the head movement and its last position should be known.

87, 152, 23, 107, 10, 119, 50, 54





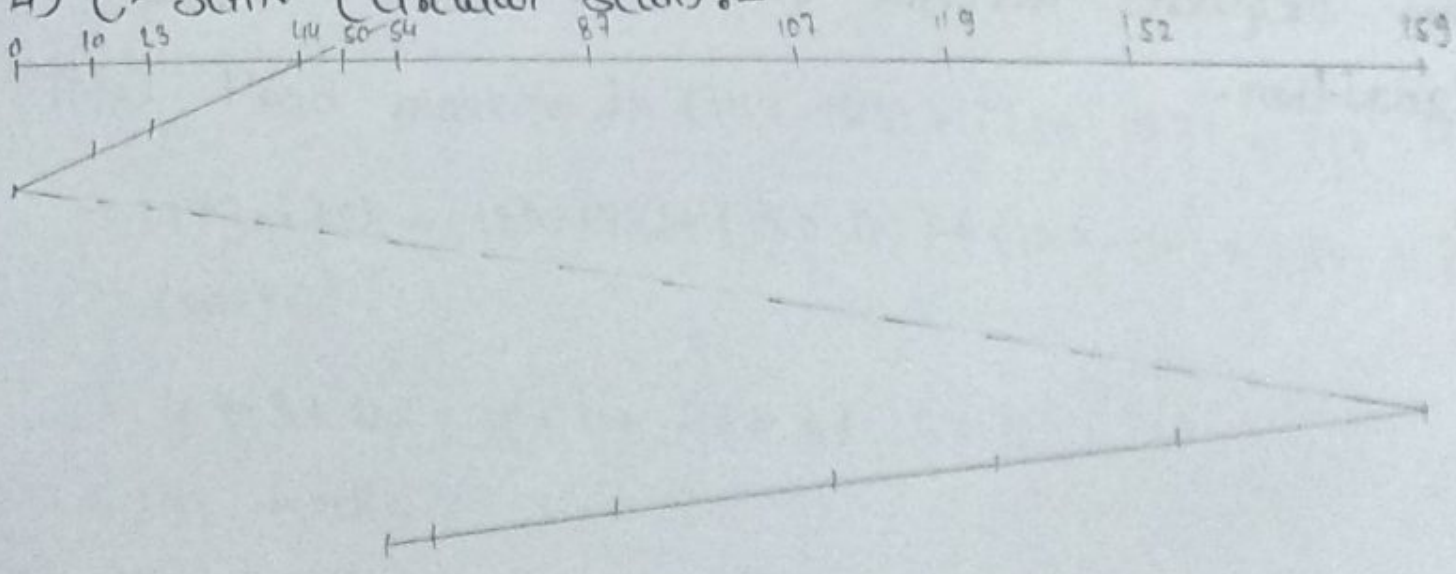
Total Head movement = (44-23) + (23-10) + (10-0) + (50-0)

+ (54-50) + (87-54) + (107-87) + (119-107) + (152-119)

= 21 + 13 + 10 + 50 + 4 + 33 + 20 + 12 + 33

= 196 tracks.

4) C-SCAN (Circular Scan):-



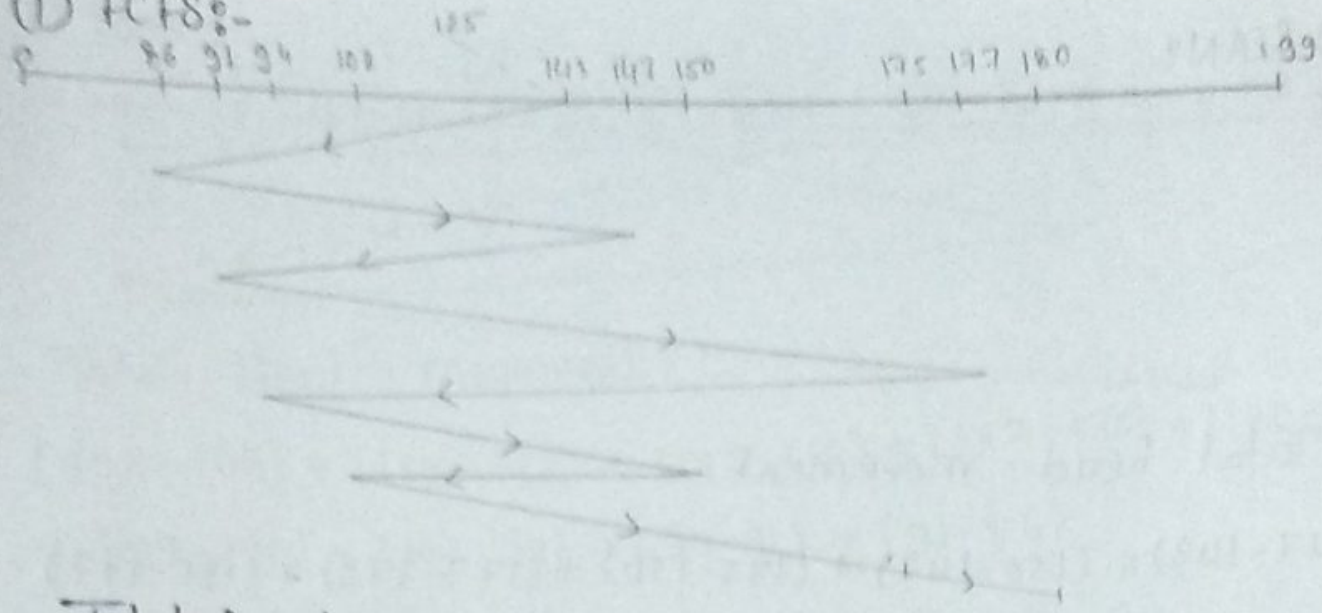
Total Head movement = (44-23)+(23-10)+(10-0)
+(159-0)+(159-152)+(152-119)+(119-107)+(107-87)
+(87-54)+(54-50)

= 312 tracks

5) Look and C-look :- SCAN and C-SCAN are called look and C-look ie. look for a request before moving in that particular direct^n where head is only moved as for a the last request in each direction.

① Suppose a head of moving head disk with 200 tracks with no. 0-199 is currently servicing a request with track 143 and has just finished a request at track 125. If the queue of the requests is kept in FIFO order.
86, 147, 91, 177, 94, 150, 102, 175, 180.
What is the total head movement to satisfy this requests for the following disk scheduling algorithm -

(i) FCFS:-



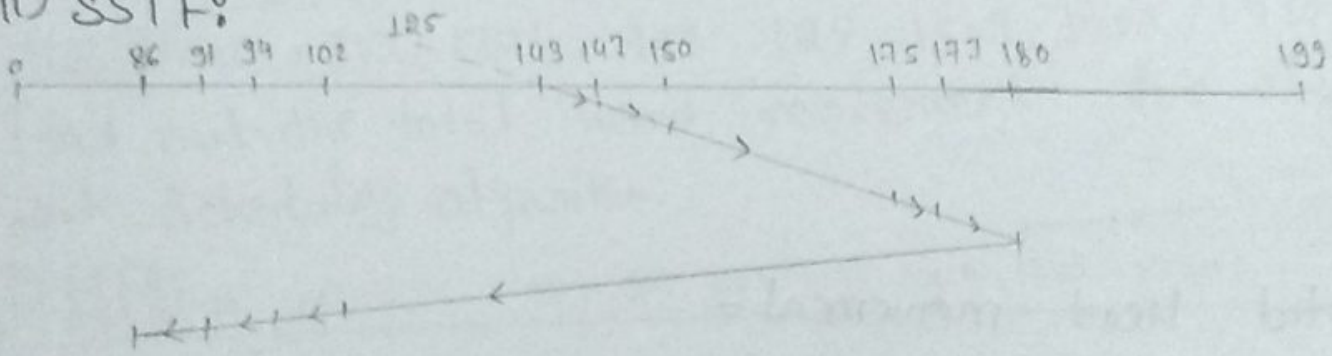
Total head movement.

(143-86) + (147-86) + (145-91) + (177-91) + (177-94) + (150-94) + (150-102) + (175-102) + (180-175)

= 57 + 61 + 56 + 86 + 83 + 56 + 48 + 73 + 5

= 525 tracks.

(ii) SSTF:

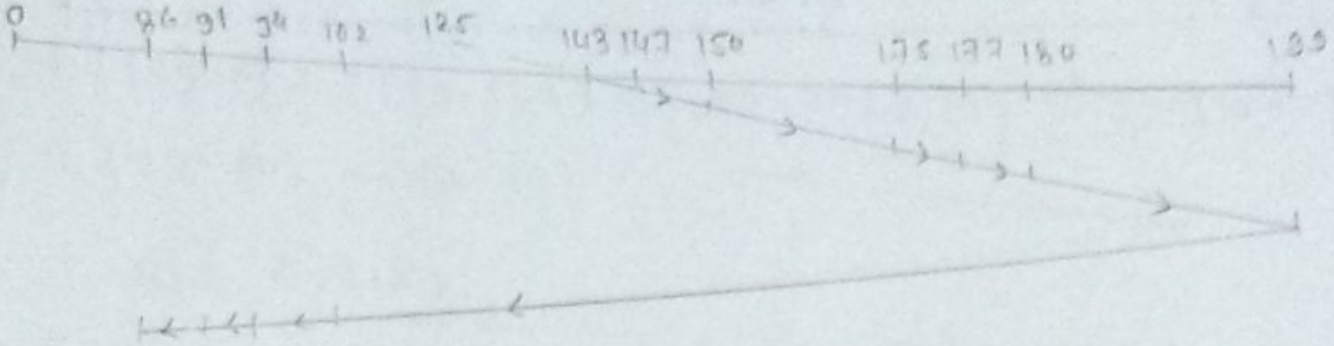


Total Head movement = (147-143) + (150-147) + (175-150) + (177-175) + (180-177) + (180-102) + (102-94) + (94-91) + (91-86)

= 4 + 3 + 25 + 2 + 3 + 78 + 8 + 3 + 5

= 131 tracks.

iii) SCAN:-



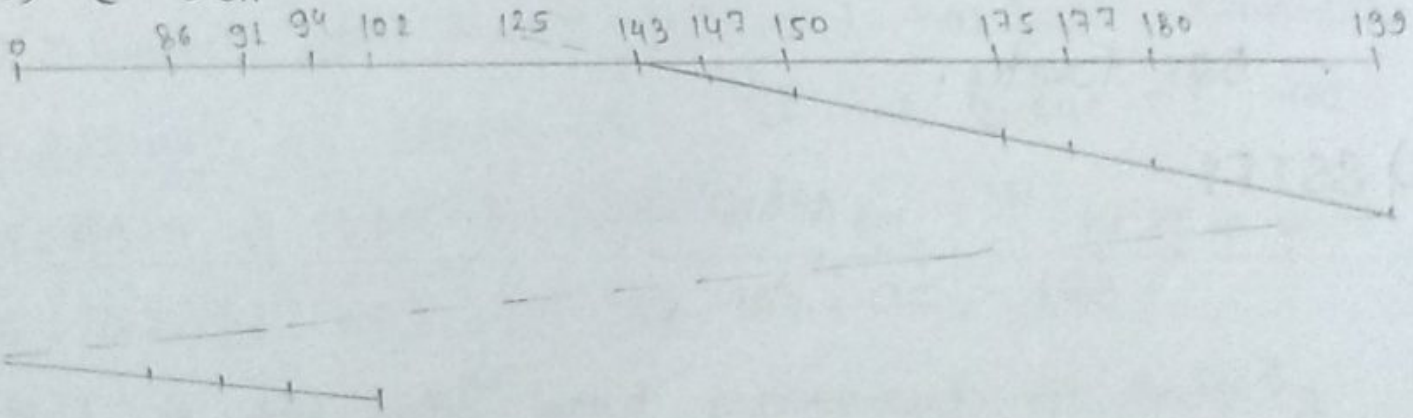
Total head movement =

(147-143) + (150-147) + (175-150) + (177-175) + (180-177)
+ (199-180) + (199-102) + (102-94) + (94-91) + (91-86)

= 4 + 3 + 25 + 2 + 3 + 19 + 97 + 8 + 3 + 5

= 169 track.

iv) C-SCAN:-



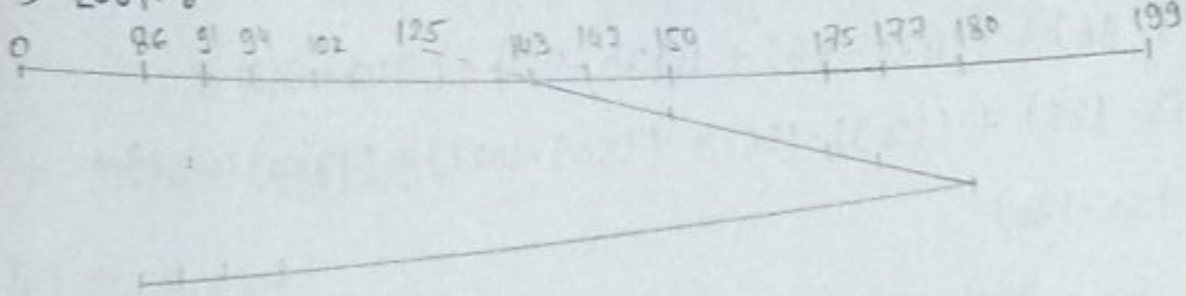
Total Head movement =

(147-143) + (150-147) + (175-150) + (177-175) + (180-177).
+ (199-180) + (199-0) + (86-0) + (91-86) + (94-91) +
(102-94)

= 4 + 3 + 25 + 2 + 3 + 19 + 199 + 86 + 5 + 3 + 8

= 357

v) Look:

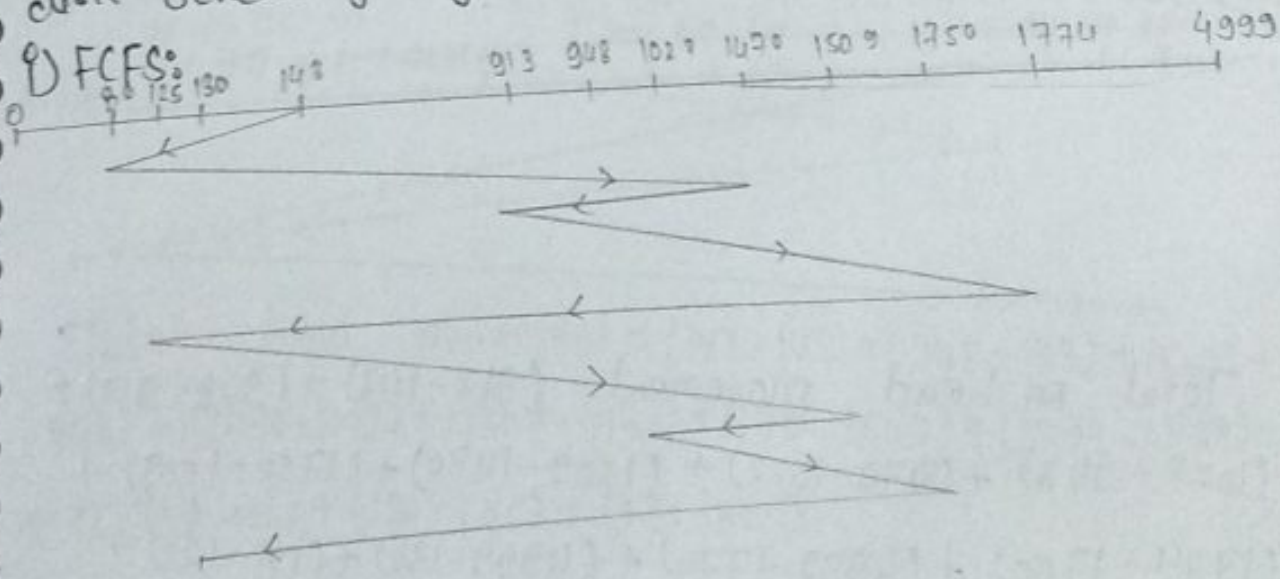


Total Head movement =

$(147-143)+(150-147)+(175-150)+(177-175)+(180-177)$
$+(199-180)+(102-94)+(94-91)+(91-86)$
$= 4+3+25+2+3+19+8+3+5$
$= 72$

② Consider a disk drive having 5000 cilenders with no. 0-4999 currently servicing a request at the cilender 143 and previous request at 125. The queue of pending request in FIFO order. 86, 1470, 913, 1774, 948, 125, 1509, 1022, 1750, 130 Find out the total head movement for all disk scheduling algorithm.

i) FCFS:



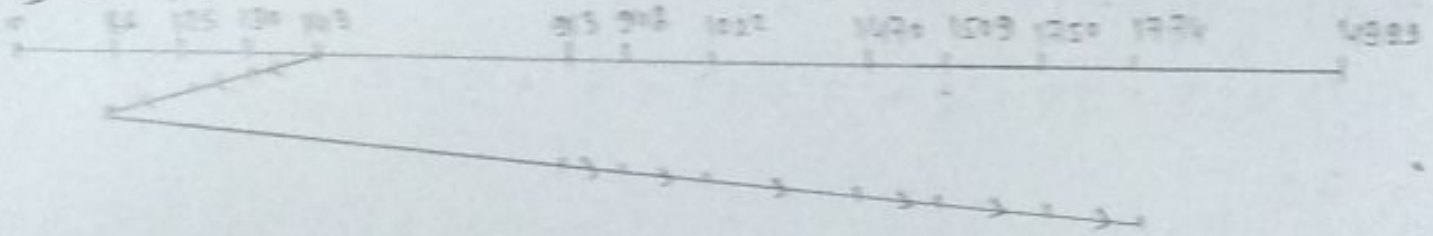
Total Head Movement =

(143 - 86) + (1470 - 86) + (1470 - 913) + (948 - 913) +
(913 - 125) + (1509 - 125) + (1509 - 1022) + (1750 - 1022) +
(1750 - 130)

= 57 + 1384 + 557 + 35 + 788 + 1384 + 487 + 728

= 5420 tracks

iii) SSTF:



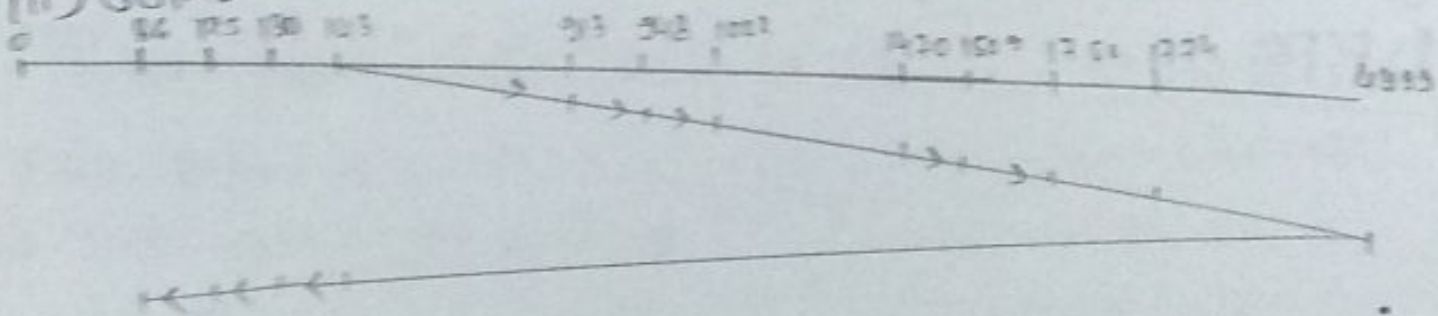
Total Head Movement =

(143 - 130) + (130 - 125) + (125 - 86) + (913 - 86) + (948 - 913)
+ (1022 - 948) + (1470 - 1022) + (1509 - 1470) + (1750 - 1509)
+ (1774 - 1750)

= 13 + 5 + 39 + 827 + 35 + 74 + 448 + 39 + 241 + 24

= 1745 tracks.

iii) SCAN:

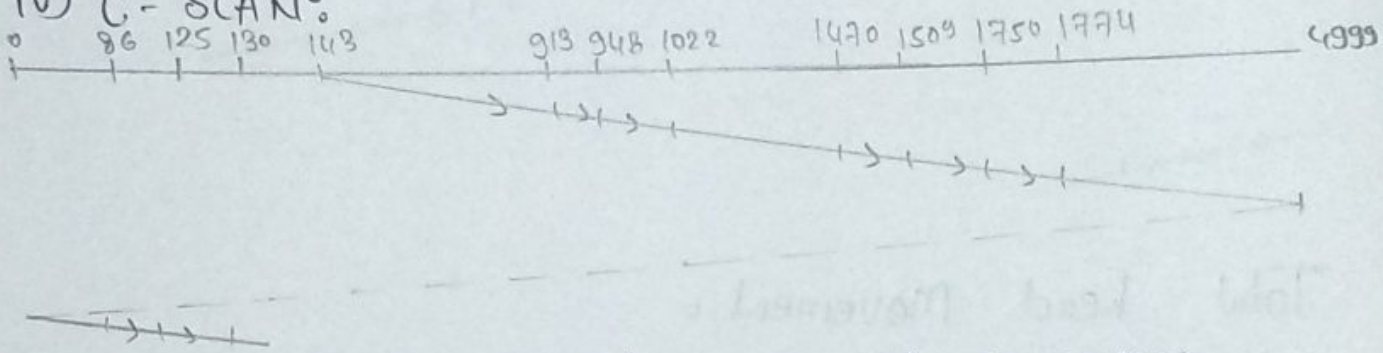


Total head movement = (913 - 143) + (948 - 913) +
(1022 - 948) + (1470 - 1022) + (1509 - 1470) + (1750 - 1509) +
(1774 - 1750) + (4999 - 1774) + (4999 - 130) + (130 - 125) +
(125 - 86)

$= 770 + 35 + 74 + 448 + 39 + 241 + 24 + 3225 + 4869 + 5 + 39$

$= 9769$ tracks.

iv) C-SCAN:

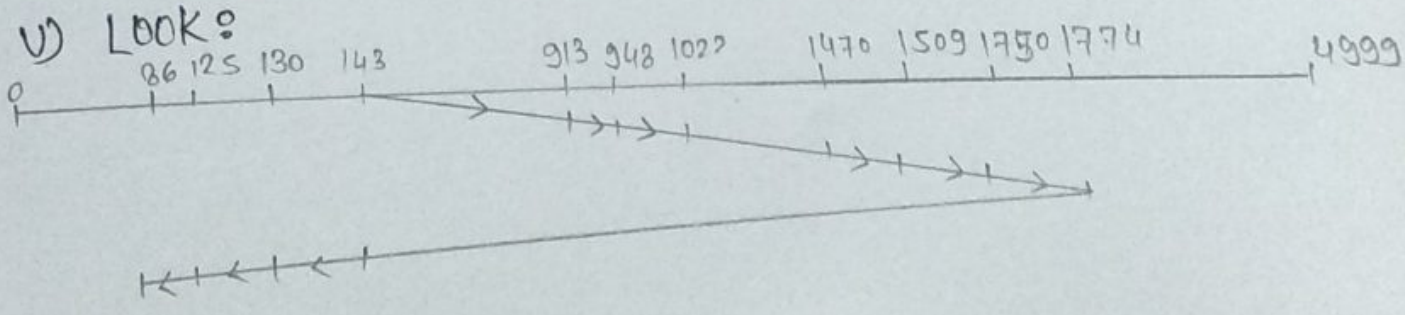


Total Head Movement = (913-143) + (948-913) + (1022-948) + (1470-1022) + (1509-1470) + (1750-1509) + (1774-1750) + (4999-1774) + (4999-0) + (86-0) + (125-86) + (130-125)

$= 770 + 35 + 74 + 448 + 39 + 24 + 24 + 3225 + 4999 + 86 + 39 + 5$

$= 9985$ tracks.

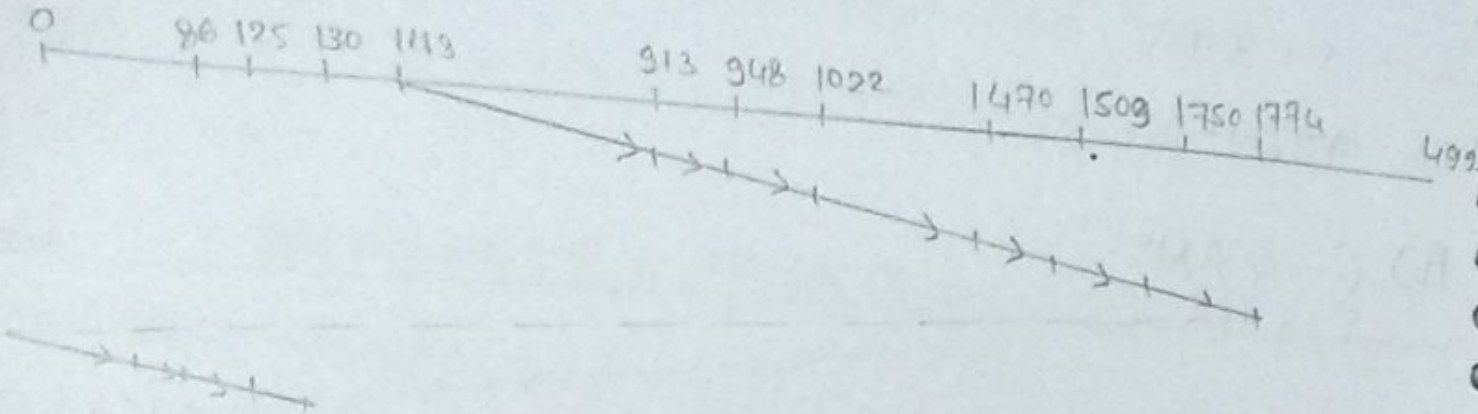
v) LOOK:



Total head movement = (913-143) + (948-913) + (1022-948) + (1470-1022) + (1509-1470) + (1750-1509) + (1774-1750) + (1774-130) + (130-125) + (125-86)

$= 1745$ tracks.

vi) C-look:



Total head movement =

(913 - 143) + (948 - 913) + (1022 - 948) + (1470 - 1022) + (1509 - 1470) + (1750 - 1509) + (1774 - 1750) + (1774 - 0) + (86 - 0) + (125 - 86) + (130 - 125)

= 770 + 35 + 74 + 448 + 39 + 241 + 24 + 1774 + 86 + 39 + 5

= 3535 tracks.