4.1 DEVICE MANAGEMENT FUNCTION :

There are following Device management functions :

Job Management : (Set of processes)

Job management controls the order and time in which programs are run and is more sophisticated in the mainframe environment where scheduling the daily work has always been routine. IBM's job control language (JCL) was developed decades ago. In a desktop environment, batch files can be written to perform a sequence of operations that can be scheduled to start at a given time.

Task Management : (unit of work done by process)

Multitasking, which is the ability to simultaneously execute multiple programs, is available in all operating systems today. Critical in the mainframe and server environment, applications can be prioritized to run faster or slower depending on their purpose. In the desktop world, multitasking is necessary for keeping several applications open at the same time so you can bounce back and forth among them.

Data Management :

Data management keeps track of the data on disk, tape and optical storage devices. The application program deals with data by file name and a particular location within the file. The operating system's file system knows where that data are physically stored (which sectors on disk) and interaction between the application and operating system is through the programming interface. Whenever an application needs to read or write data, it makes a call to the operating system

Device Management :

Device management controls peripheral devices by sending them commands in their own proprietary language. The software routine that knows how to deal with each device is called a "driver," and the OS requires drivers for the peripherals attached to the computer. When a new peripheral is added, that device's driver is installed into the operating system.

User Interface : CUI , GUI

All graphics based today, the user interface includes the windows, menus and method of interaction between you and the computer. Prior to graphical user interfaces (GUIs), all operation of the computer was performed by typing in commands. Not at all extinct, command-line interfaces are alive and well and provide an alternate way of running programs on all major operating systems.

Operating systems may support optional interfaces, both graphical and command line. Although the overwhelming majority of people work with the default interfaces, different "shells" offer variations of appearance and functionality.

Security :

Operating systems provide password protection to keep unauthorized users out of the system. Some operating systems also maintain activity logs and accounting of the user's time for billing purposes. They also provide backup and recovery routines for starting over in the event of a system failure.

4.2 DEVICE CHARACTERISTICS :

Operating systems are normally unique to their manufacturers and the hardware in which they are run. Generally, when a new computer system is installed, operational software suitable to that hardware is purchased. Users want reliable operational software that can effectively support their processing activities.

Though operational software varies between manufacturers, it has similar characteristics. Modern hardware, because of its sophistication, requires that operating systems meet certain specific

standards. For example, considering the present state of the field, an operating system must support some form of online processing. Functions normally associated with operational software are :

1. Job management
2. Resource management
3. Control of I/O operations
4. Error recovery
5. Memory management

Job Management :

A very important responsibility of any operational software is the scheduling of jobs to be handled by a computer system. This is one of the main tasks of the job management function. The operating system sets up the order in which programs are processed, and defines the sequence in which particular jobs are executed. The term job queue is often used to describe the series of jobs awaiting execution. The operating system weighs a variety of factors in creating the job queue. These include which jobs are currently being processed, the system's resources being used, which resources will be needed to handle upcoming programs, the priority of the job compared to other tasks, and any special processing requirements to which the system must respond.

The operational software must be able to assess these factors and control the order in which jobs are processed.

Resource Management :

The management of resources in a computer system is another major concern of the operating system. Obviously, a program cannot use a device if that hardware is unavailable. As we have seen, the operational software oversees the execution of all programs. It also monitors the devices being used. To accomplish this, it establishes a table in which programs are matched against the devices they are using or will use. The operating system checks this table to approve or deny use of a specific device.

Control of I/O Operations :

Allocation of a system's resources is closely tied to the opera-

tional software's control of I/O operations. As access is often necessary to a particular device before I/O operations may begin, the operating system must coordinate I/O operations and the devices on which they are performed. In effect, it sets up a directory of programs undergoing execution and the devices they must use in completing I/O operations. Using control statements, jobs may call for specific devices. This lets users read data from specific sites or print information at selected offices. Taking advantage of this facility, data read from one location may be distributed throughout computerized system.

To facilitate execution of I/O operations, most operating systems have a standard set of control instructions to handle the processing of all input and output instructions. These standard instructions, referred to as the input/output control system (IOCS), are an integral part of most operating systems. They simplify the means by which all programs being processed may undertake I/O operations.

In effect, the program undergoing execution signals the operating system that an I/O operation is desired, using a specific I/O device. The controlling software calls on the IOCS software to actually complete the I/O operation. Considering the level of I/O activity in most programs, the IOCS instructions are extremely vital.

4.3 DISK SPACE MANAGEMENT:

Disk Space Management :

Disk Space Management tools provide data that system administrators need to track disk space availability for designated systems, typically as part of Service Level Agreement (SLA) measures. A disk space management tool can establish baseline measures against which administrators can track ongoing capacity. A disk space management tool makes it possible to collect metrics and monitor disk space availability around the clock.

- Low-level formatting, or physical formatting - Dividing a disk into sectors that the disk controller can read and write.
- To use a disk to hold files, the operating system still needs to record its own data structures on the disk.

- * Partition the disk into one or more groups of cylinders.
- * Logical formatting or "making a file system".

➠ Boot block initializes system.

- * The bootstrap is stored in ROM.
- * Bootstrap loader program.

➠ Methods such as sector sparing used to handle bad blocks.

Swap-Space Management Space Management :

➠ Swap-space : Virtual memory uses disk space as an extension of main memory.

➠ Since disk access is much slower than the memory access, using swap space significantly reduces the system performance so the main goal for design and implementation of swap space is to provide the best throughput for the virtual memory system.

➠ Swap-space can reside in two places :

1. Can be carved out of the normal file system, or,
2. More commonly, it can be in a separate disk partition.

➠ If swap space is simply a large file within the file system, the normal file system routines can be used to create it, name it and allocate its space.

➠ Swap space can be created in a separate disk portioning, no file system or directory structure is kept is placed on this space. Rather a separate swap-space storage manager is used to allocate and deallocate the blocks

Swap-space management

➠ 4.3BSD allocates swap space when process starts; holds text segment(the program) and data segment.

➠ Kernel uses swap maps to track swap-space use.

➠ Solaris 2 allocates swap space only when a page is forced out of physical memory, not when the virtual memory page is first created.

➠ Data Structures for Swapping on Linux Systems is shown below

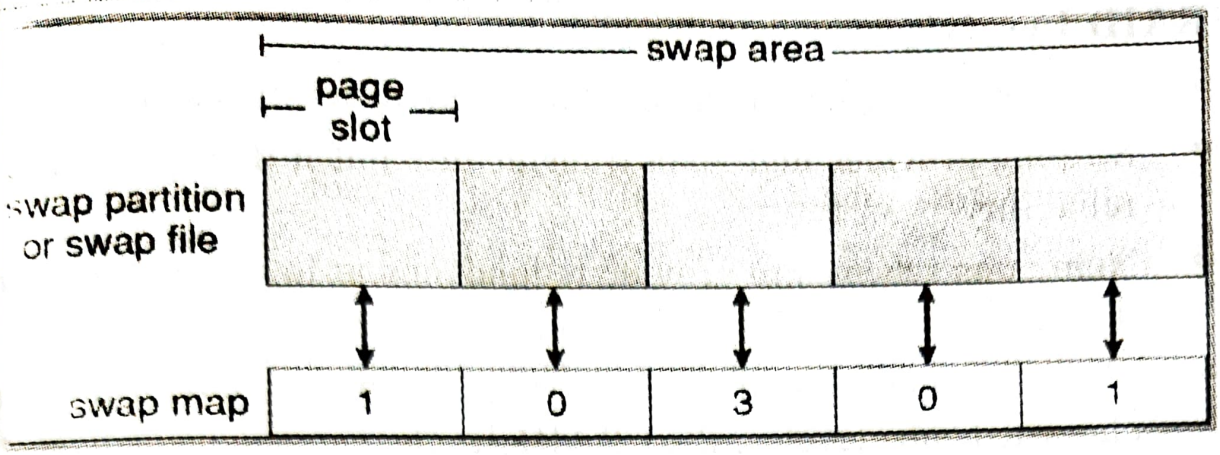


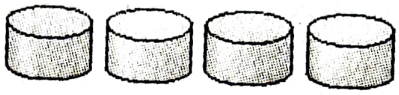
Figure 4.1 Data Structures for Swapping on Linux Systems

RAID (Redundant Array of Inexpensive or Independent Disks)

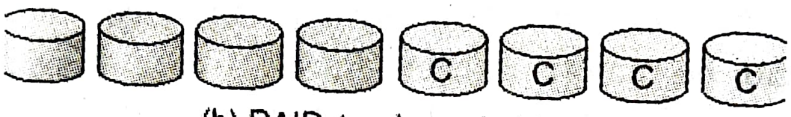
- Having large no of disks in a system improves the rate at which the data can be read or written. Further more this setup offers the potential for improving the reliability of data storage b/c redundant data can be stored on multiple disks.
- RAID is the term used to describe a storage systems' resilience to disk failure through the use of multiple disks and by the use of data distribution and correction techniques. RAID stands for: Redundant Array of Inexpensive or Independent Disks
- RAID multiple disk drives provide reliability via redundancy. so variety of disk organization technique is collectively called as Redundant Array of Independent Disks (RAID)
- RAID is arranged into six different levels.
- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively.
- Disk striping uses a group of disks as one storage unit.
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data.
- The simplest approach to introducing redundancy is to duplicate every disk this technique I known as Mirroring or shadowing
- A logical disk then consists of two physical disk and every write is carried out on both disks. If one of the disks fails, the data can be read from the other.

RAID Levels

- Mirroring provides high reliability but it is expensive
- Striping provides high data transfer rate but it does not improve reliability
- Numerous schemes to provide redundancy at lower cost by using the idea of disk striping combined with parity bits
- These schemes have different cost-performance tradeoff and are classified into levels called RAID levels



(a) RAID 0: non-redundant striping.

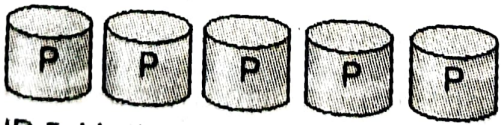


(b) RAID 1: mirrored disks.

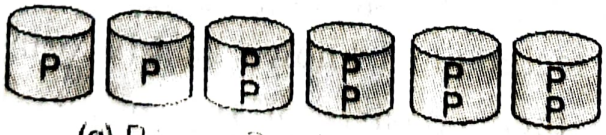
(c) RAID 2: memory-style error-correcting codes.

(d) RAID 3: bit-interleaved parity.

(e) RAID 4: block-interleaved parity.



(f) RAID 5: block-interleaved distributed parity.



(g) R[illegible] + Q redundancy.

4.2 RAID LEVELS

- P indicates error correcting bits
- C indicates second copy of data

Level 0

- RAID level 0 does not provide fault tolerance. This level is also known as disk striping because it uses a disk file system called a stripe set. Data is divided into blocks and is spread in a fixed order among all the disks in the array. RAID level0 improves read and write performance by spreading operations across multiple disks, so that operations can be performed independently.

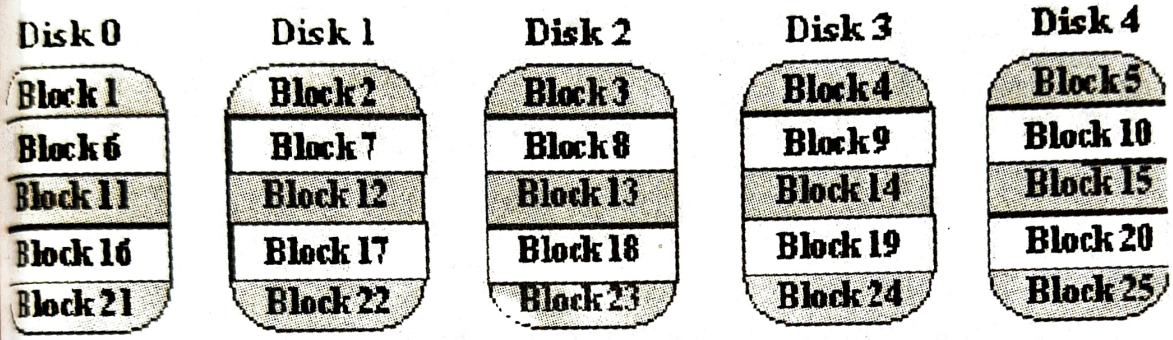


Figure 4.3

Level 1

- RAID level 1 provides fault tolerance. This level is also known as disk mirroring because it uses a disk file system called a mirror set. Disk mirroring provides a redundant, identical copy of a selected disk. All data written to the primary disk is written to the mirror disk. It also generally improves read performance (but may degrade write performance).

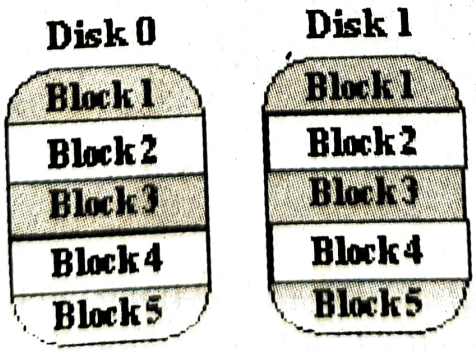


Figure 4.4

Level 2

➡ RAID level 2 uses error correcting algorithm that employs disk-striping strategy that breaks a file into bytes and spreads it across multiple disks. The error-correction method requires several disks. RAID level 2 is more advanced than Level 0, because it provides fault tolerance, but is not as efficient as other RAID levels and is not generally used.

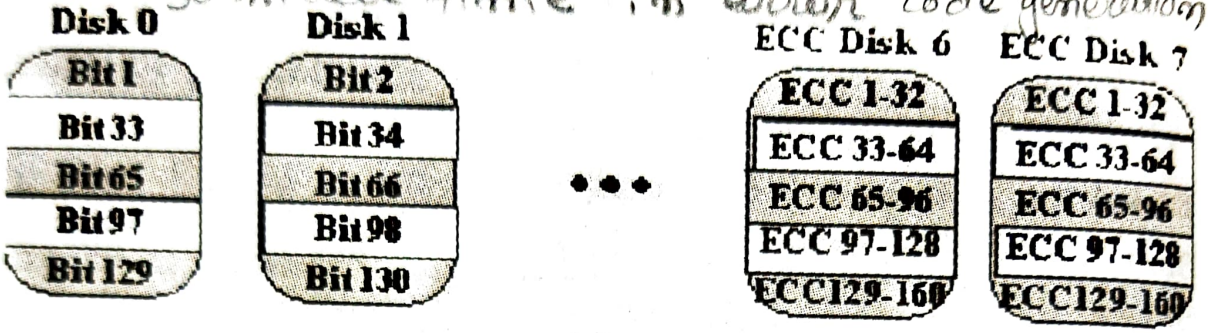


Figure 4.5

Level 3

➡ RAID level 3 is similar to RAID level2, because it uses the same striping method as level 2, but it requires only one disk for parity data. RAID 3suffers from a write bottleneck, because all parity data is written to a single drive but provides some read and write performance improvement.

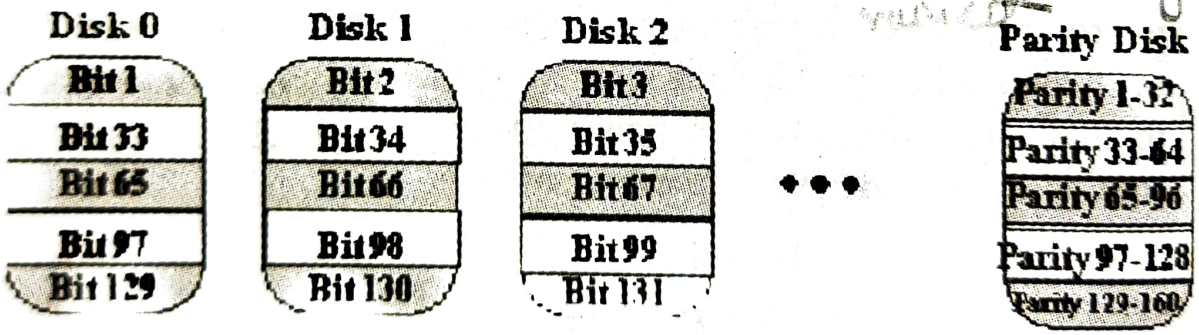


Figure 4.6

Level 4

➡ RAID level 4 is similar to RAID level3, because it uses the similar striping method as level 3 and requires only one disk for parity data, but it employs striped data in much larger blocks or segments. RAID level 4 is not as efficient as RAID level 5,

because (as in RAID level 3) all parity data is written to a single drive, so RAID level 4 suffers from a write bottleneck and is not generally used.

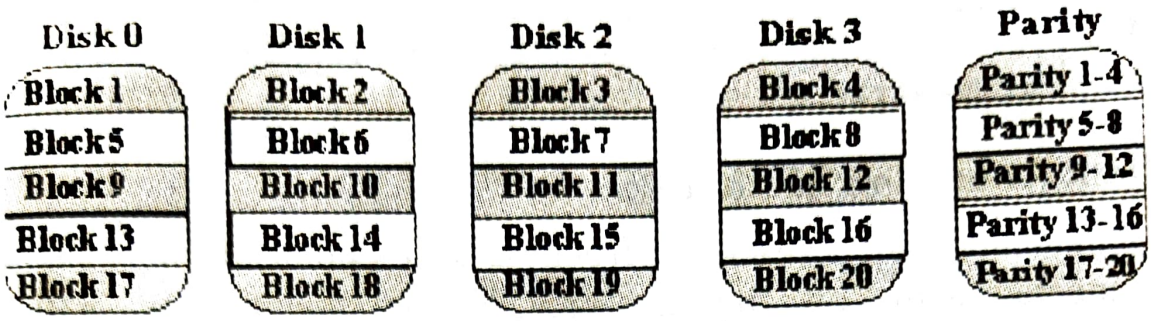


Figure 4.7

Level 5

- RAID level 5 is known as striping with parity. This is the most popular RAID level. It is similar to level 4 in that it stripes the data in large blocks across all the disks in the array. It differs in that it writes the parity across all the disks. The data redundancy is provided by the parity information. The data and parity information are arranged on the disk array so that the two are always on different disks. RAID level 5 has better performance than RAID level 1 and provides fault tolerance.

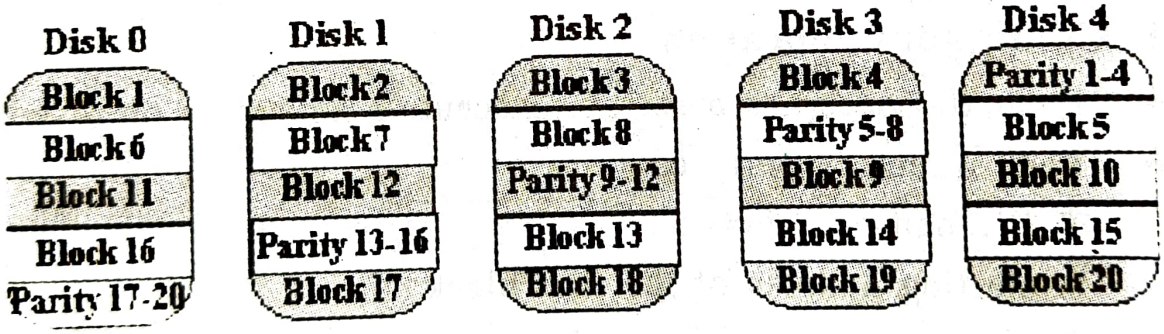


Figure 4.8

Level 6

- It is called P+Q redundancy scheme which store extra redundant information to guard against multiple disk failure
- It uses reed-Solomon code
- System can tolerate 2disk failure

copy of Parity Disk as well as Hard Disk.

create copy of parity

RAID 0 + 1 Parity Disk.

- It is combination of level 0 and level 1
- Provides better performance and reliability

4.4 ALLOCATION AND DISK SCHEDULING METHODS :

Disk Allocation Methods :

File system is described from the user's point of view. Users are connected with now files are named, what the directory structure looks like and so on. Users need convenience using the file system.

There are two main goals.

1. **Disk space should be utilized effectively.**
2. **Files should be accessed quickly.**

There is a need to allocate storage space to a file, one or more free disk blocks are found and allocated to file.

There are mainly three different allocation methods.

1. Contiguous Allocation
2. Linked Allocation
3. Indexed Allocation

1. Contiguous allocation

- Each file occupies a set of consecutive addresses on disk
- Each directory entry contains :
  - * file name
  - * starting address of the first block
  - * block address = sector id (e.g., block = 4K)
  - * length in blocks
- Usual dynamic storage allocation problem
  - * use first fit, best fit, or worst fit algorithms to manage storage
- if the file can increase in size, either
  - * leave no extra space, and copy the file elsewhere if it expands
  - * leave extra space

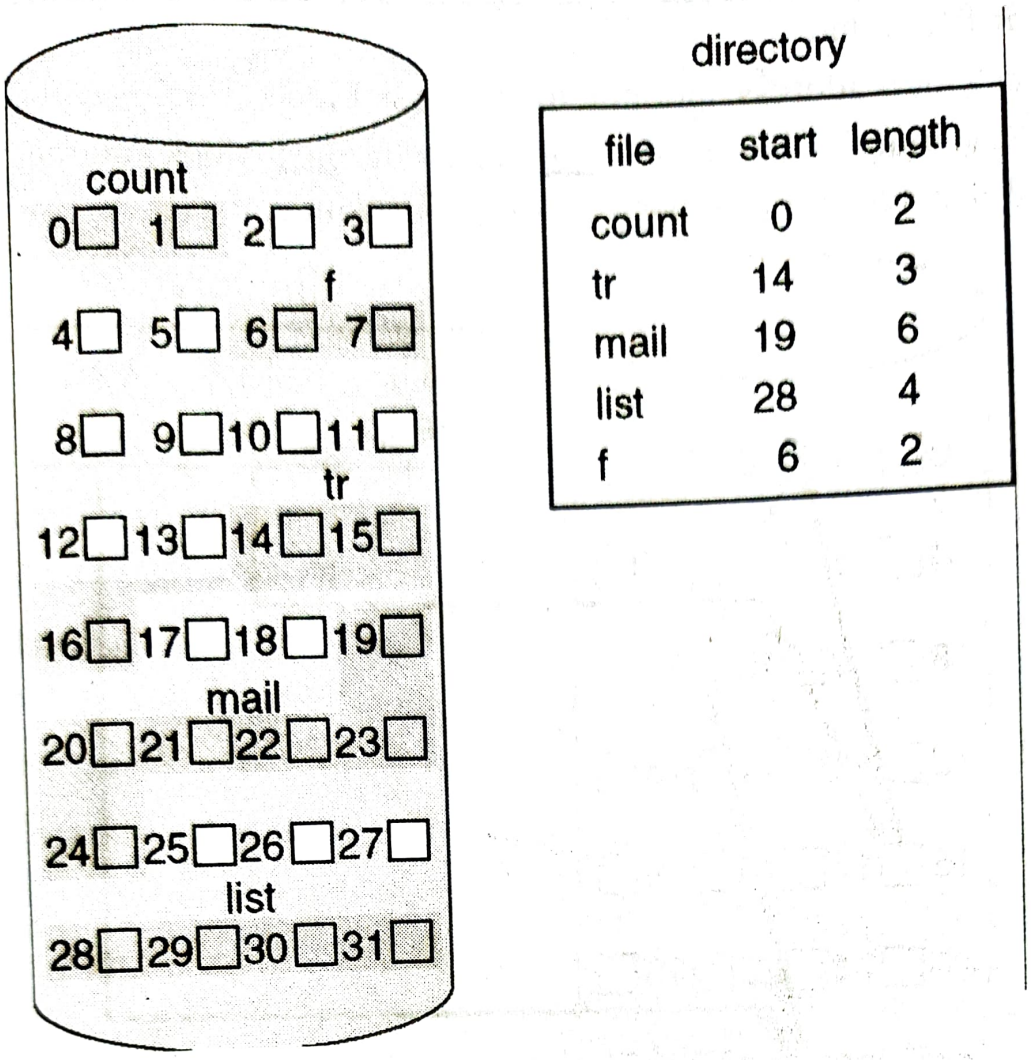


Figure 4.9 Contiguous allocation

Advantages :

The advantage of contiguous memory allocation is

1. It supports fast sequential and direct access
2. It provides a good performance
3. The number of disk seek required is minimal

Disadvantages :

The disadvantage of contiguous memory allocation is fragmentation.

2. Linked allocation

- Each data block contains the block address of the next block in the file
- Each directory entry contains :

* file name
* block address: pointer to the first block

Sometimes, also have a pointer to the last block (adding to the end of the file is much faster using this pointer)

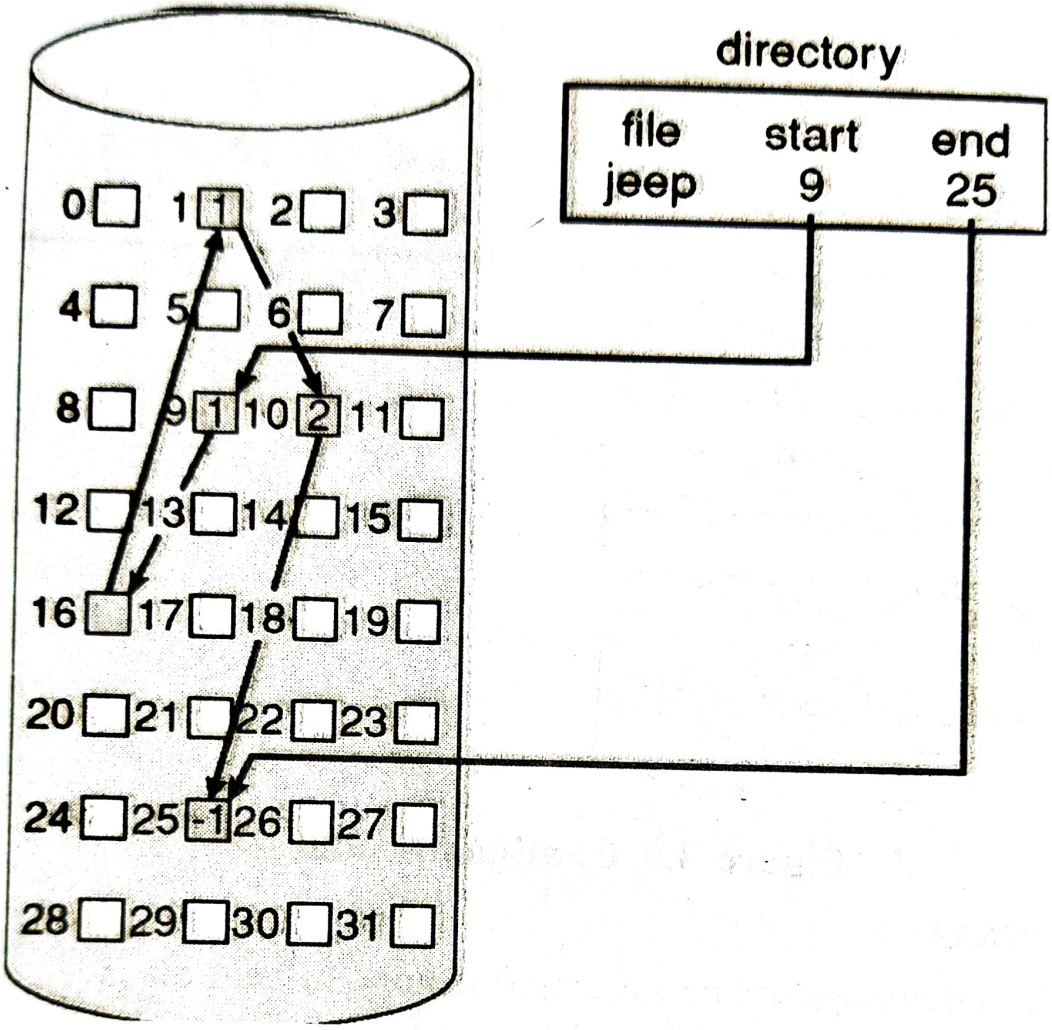


Figure 4.10 Linked allocation

➡ A view of the linked list

Advantages :

1. The advantage is that while accessing a block that is stored at the middle of a file.
2. Its location can be determined by chasing the pointers stored in the FAT as opposed to accessing all of the individual blocks of the file in a sequential manner to find the pointer to the target block.

3. Typically, most of the FAT can be cached in memory and therefore the pointers can be determined with just memory accesses instead of having to access the disk blocks.

Disadvantages :

1. Used only for sequential access of files.
2. Direct access is not supported
3. Memory space required for the pointers.
4. Reliability is compromised if the pointers are lost or damaged

3. Indexed allocation

- Store all pointers together in an index table
  - * The index table is stored in several index blocks
  - * Assume index table has been loaded into main memory

All files in one index

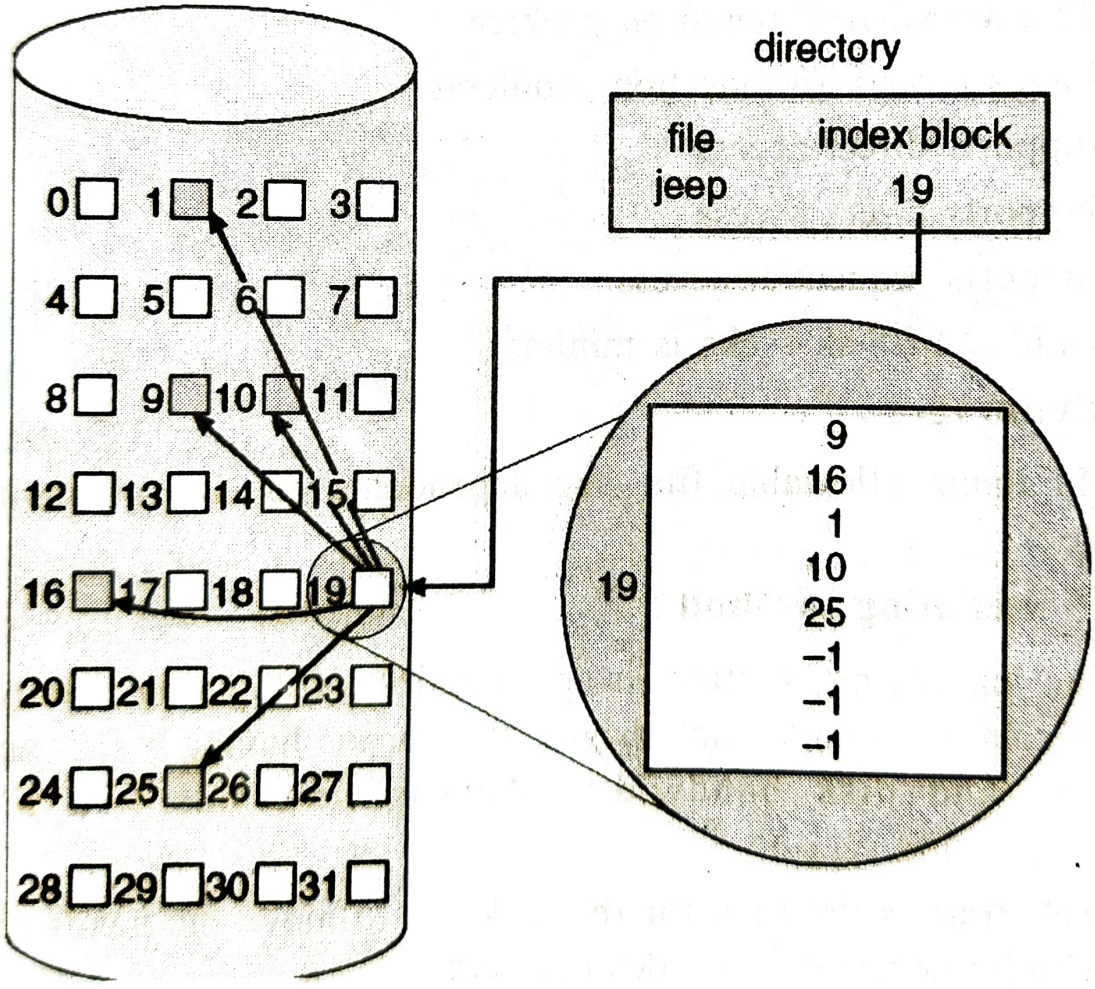


Figure 4.11 Indexed allocation

The index has one entry for each block on disk.

- better than linked allocation if we want to seek a particular offset of a file because many links are stored together instead of each one in a separate block
- SGG call this organization a "linked" scheme, but I call it an "indexed" scheme because an index is kept in main memory.
- problem: index is too large to fit in main memory for large disks
  - * FAT may get really large and we may need to store FAT on disk, which will increase access time
  - * e.g., 500 Mb disk with 1 Kb blocks = 4 bytes * 500 K
    = 2Mb entries

Advantages :

The advantages are

1. No external-fragmentation problem
2. Solves the size-declaration problems.
3. Supports direct access
4. Supports direct access
5. Supports sequential access
6. Number of disk seeks is minimal.

Disadvantages :

1. Maximum allowable file size dependents on size of an index block.

Disk Scheduling Method :

- The operating system is responsible for using hardware efficiently - for the disk drives, this means having a fast access time and disk bandwidth. Access time has two major components :

  Seek time is the time for the disk are to move the heads to the cylinder containing the desired sector.

  Rotational latency is the additional time waiting for the disk to rotate the desired sector to the disk head.

- Minimize seek time
- Seek time seek distance
- Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.
- Both bandwidth and access time can be improved by processing requests in a good order.
- Disk requests include the disk address, memory address, number of sectors to transfer, and whether the request is for reading or writing.
- Several algorithms exist to schedule the servicing of disk I/O requests.

TYPES OF DISK SCHEDULING ALGORITHMS :

Although there are other algorithms that reduce the seek time of ll requests, I will only concentrate on the following disk scheduling lgorithms :

1. First-Come, First-Serve (FCFS)
2. Shortest Seek Time First (SSTF)
3. Elevator (SCAN)
4. Circular SCAN (C-SCAN)
5. LOOK
6. C-LOOK

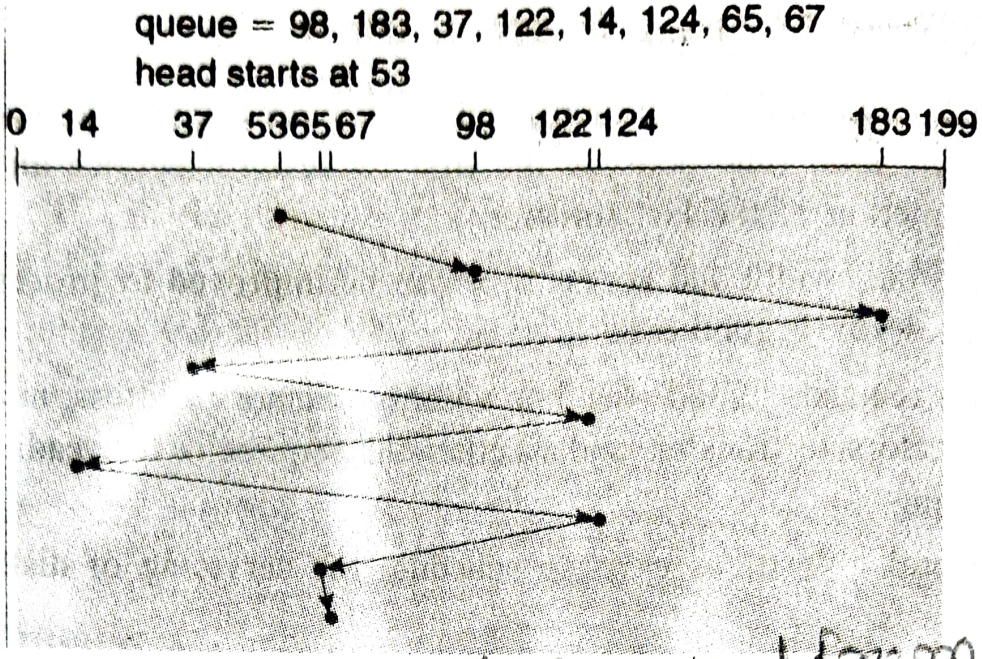
We illustrate them with a request queue (0-199).

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

Head pointer 53

1 FCFS (First Come First Serve) Scheduling :

First-Come First-Serve is simple and intrinsically fair, but it enerally does not provide the fastest service.



To set Head from desire place

Figure 4.12 FCFS DISK SCHEDULING

First-come First-served Scheduling follow first in first out method. As each process becomes ready, it joins the ready queue. When the current running process ceases to execute, the oldest process in the Ready queue is selected for running. That is first entered process among the available processes in the ready queue. The average waiting time for FCFS is often quite long. It is non-preemptive.

In the above example, the disk head is initially at cylinder 53, it will first move from 53 to 98, then to 183,37,122,14,124,65, and finally to 67, for a total head movement of 640 cylinders.

Advantages :

- Better for long processes
- Simple method (i.e., minimum overhead on processor)
- No starvation

Disadvantages :

- Convoy effect occurs. Even very small process should wait for its turn to come to utilize the CPU. Short process behind long process results in lower CPU utilization.
- Throughput is not emphasized.

2 SSTF Scheduling :

- In Shortest Seek Time First scheduling Selects the request with the minimum seek time from the current head position.
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests.
- SSTF reduces the total head movement to 236 cylinders, down from 640 required for the same set of requests under FCFS. Note, however that the distance could be reduced still further to 208 by starting with 37 and then 14 first before processing the rest of the requests.

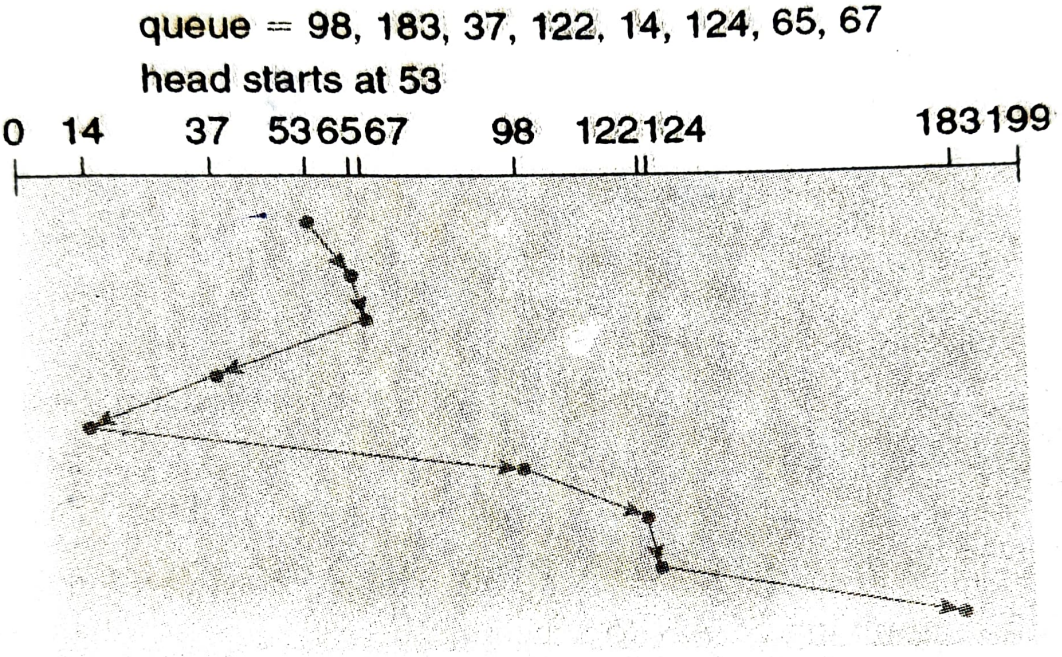


Figure 4.13 SSTF DISK SCHEDULING

Advantages :

- After a request, go to the closest request in the work queue, regardless of direction reduces total seek time compared to FCFS

Disadvantages:

- Starvation is possible; stay in one area of the disk if very busy switching directions slows things down

3. SCAN Scheduling :

- The **SCAN** algorithm, the **elevator** algorithm moves back and forth from one end of the disk to the other, similarly to an elevator processing requests in a tall building.

- The disk arm starts at one end of the disk. and moves toward the other end. servicing requests until it gets to the other end of the disk. where the head movement is reversed and servicing continues. The head continuously scans back and forth across the disk.

always check previous & then Move

head = 60 50 60 to 53
if = 06 50 0 to 53

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

head starts at 53

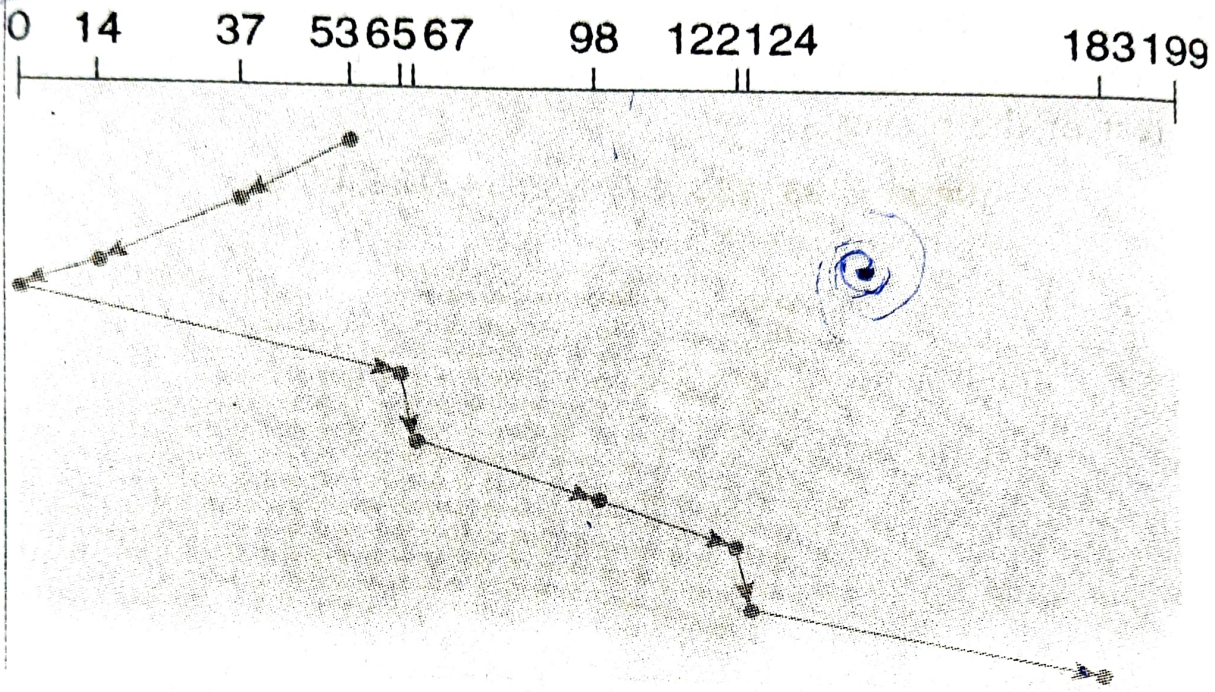


Figure 4.14 SCAN DISK SCHEDULING

- Before applying SCAN to schedule the request on cylinders, 98, 183, 37, 122, 14, 124, 65, and 67, we need to know the direction of head movement, in addition to the head's current position (53). If the disk arm moving toward 0, the head will service 37 and then 14. At cylinder 0, the arm will reverse and will move toward the other end of the disk, servicing the requests at 65, 67, 98, 122, 124, and 183 (Figure 4.14).
- Illustration shows total head movement of 208 cylinders.

Advantages :

- Also there is no hunger for any request.
- More efficient than FCFS.

Disadvantages :

- Not so fair; cylinder which are just behind the head will wait longer.
- Requires extra head movement between two extreme points.
- For example, after servicing 5th cylinder, there is no need to visit 0th cylinder. But, though, this algorithm visits the points.

4. C-SCAN (Circular SCAN) Scheduling :

- The Circular-SCAN algorithm improves upon SCAN by treating all requests in a circular queue fashion - Once the head reaches the end of the disk, it returns to the other end without processing any requests, and then starts again from the beginning of the disk.
- Provides a more uniform wait time than SCAN.
- The head moves from one end of the disk to the other. servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one.

→ It work on increasing order only.

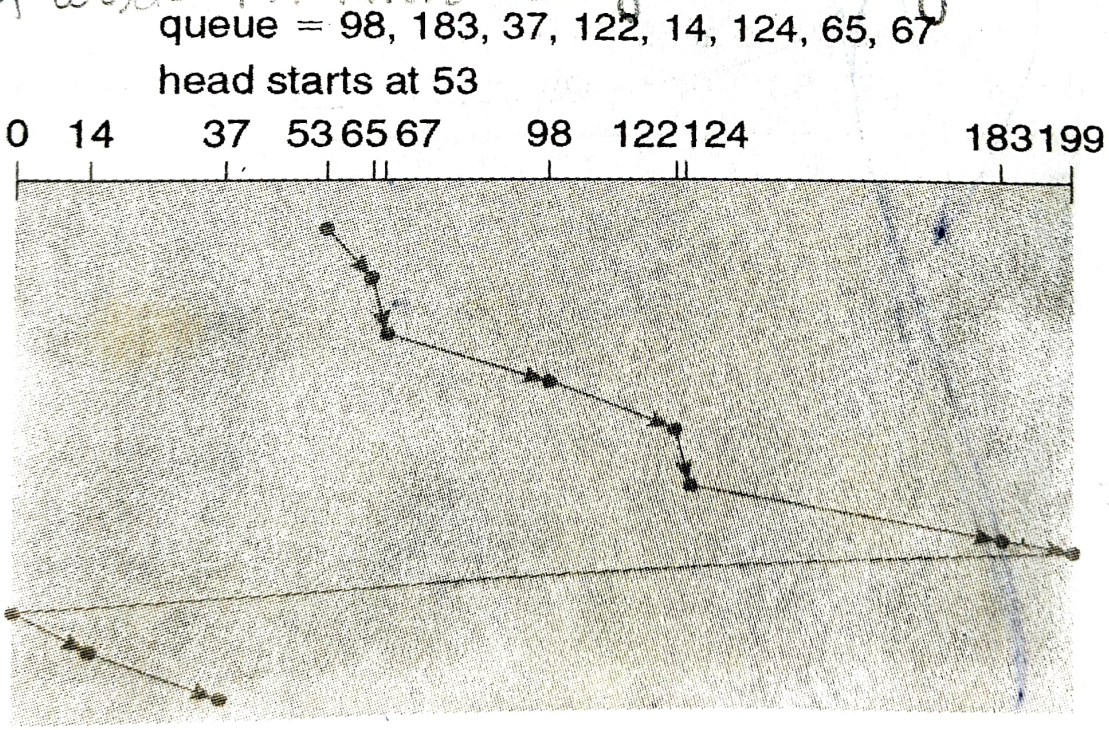


Figure 4.15 C-SCAN DISK SCHEDULING

5. LOOK Scheduling :

➠ LOOK scheduling improves upon SCAN by looking ahead at the queue of pending requests, and not moving the heads any farther towards the end of the disk than is necessary.

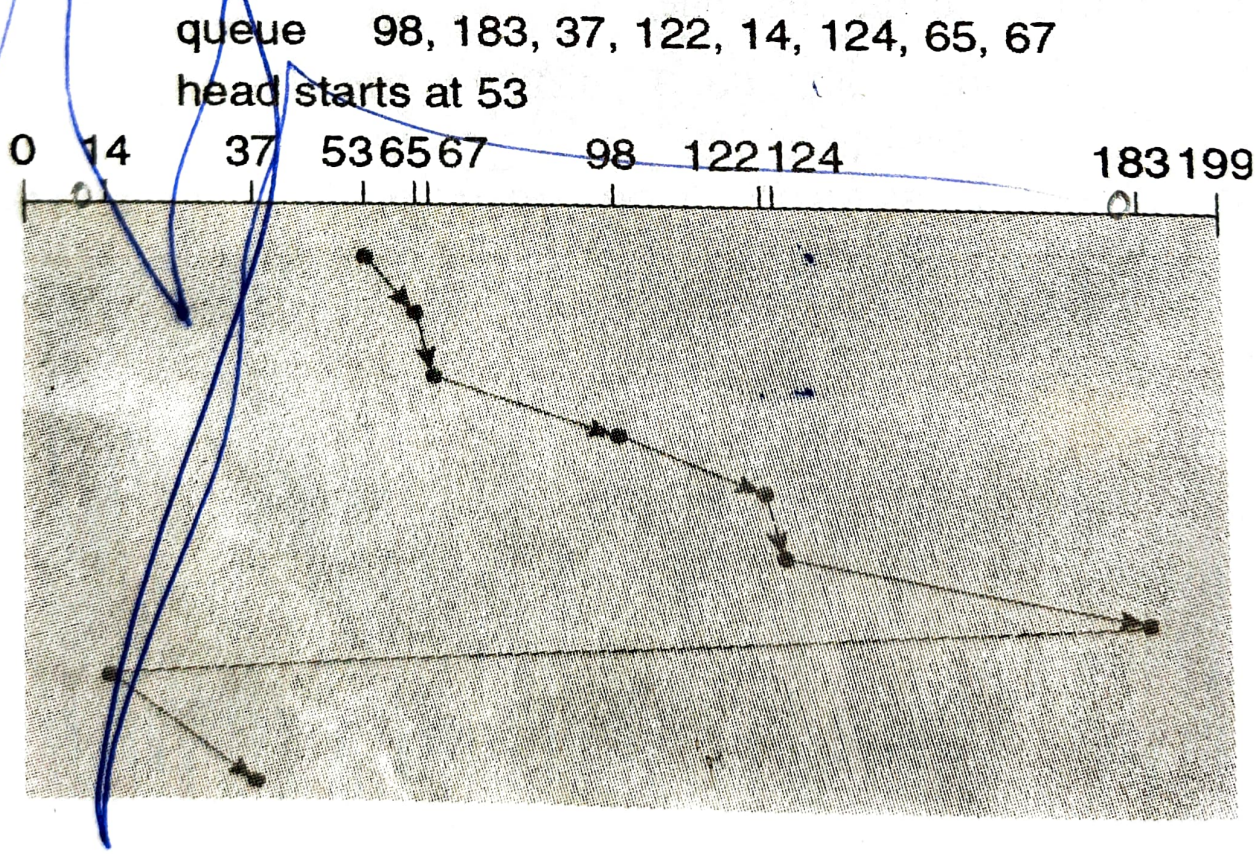
6. C-LOOK (Circular - LOOK) Scheduling :

This is just an enhanced version of C-SCAN.

Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.

In this the scanning doesn't go past the last request in the direction that it is moving. It too jumps to the other end but not all the way to the end. Just to the furthest request. C-SCAN had a total movement of 187 but this scan (C-LOOK) reduced it down to 157 tracks.

From this you were able to see a scan change from 644 total head movements to just 157. You should now have an understanding as to why your operating system truly relies on the type of algorithm it needs when it is dealing with multiple processes.



4.16 C- LOOK SCHEDULING