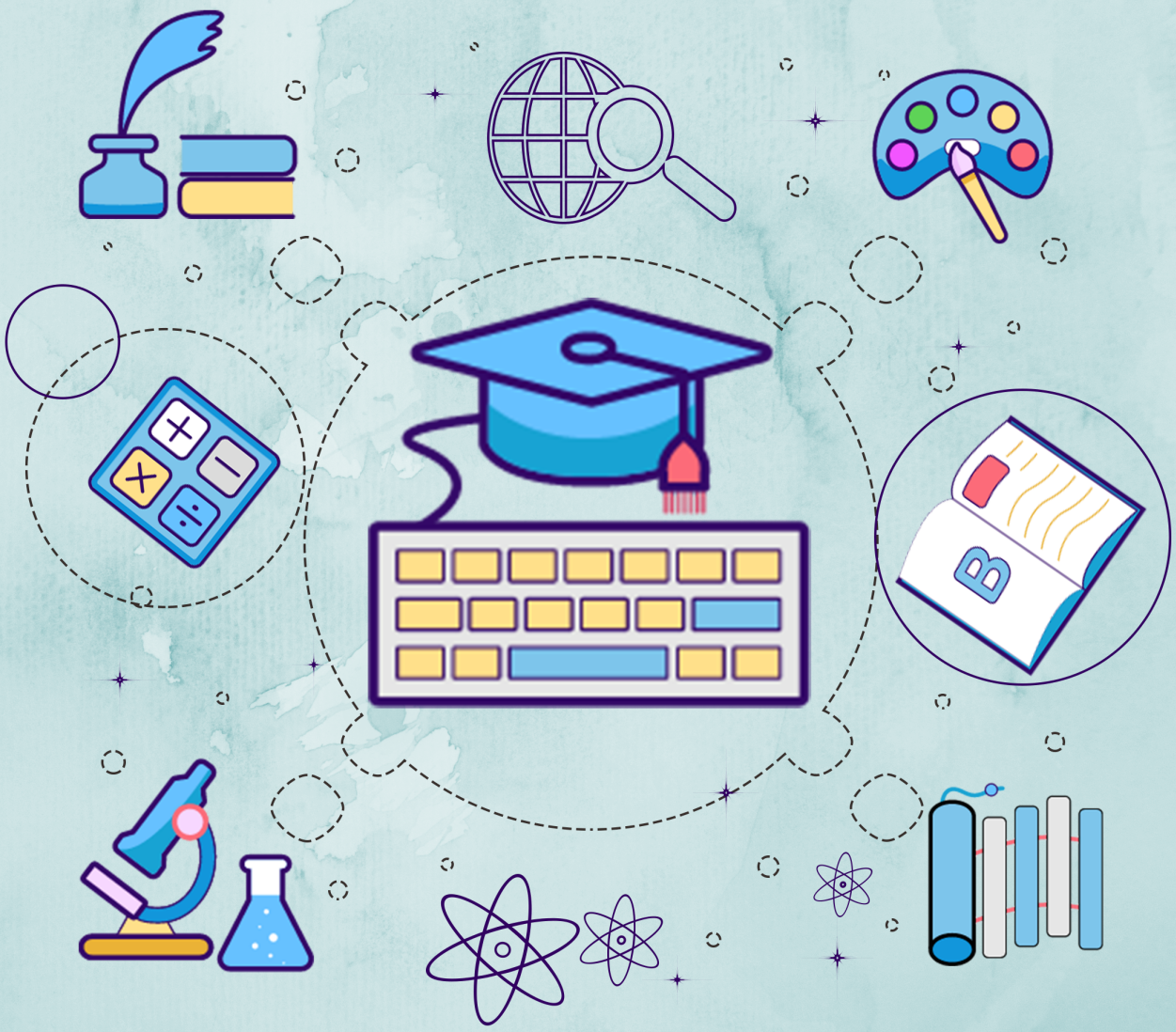
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

KeralaNotes



SYLLABUS | STUDY MATERIALS | TEXTBOOK PDF | SOLVED QUESTION PAPERS

www.keralanotes.com

KeralaNotes

KTU S4 CSE PDF NOTES

OPERATING SYSTEM (CST200)

Module 5

Related Link :

- KTU S4 CSE NOTES 2019 SCHEME
- KTU S4 SYLLABUS CSE COMPUTER SCIENCE
- KTU PREVIOUS QUESTION BANK S4 CSE SOLVED
- KTU CSE TEXTBOOKS S4 B.TECH PDF DOWNLOAD
- KTU S4 CSE NOTES | SYLLABUS | QBANK | TEXTBOOKS DOWNLOAD

www.keralanotes.com

MODULE-5

5.1 Storage Management

This session gives an overview of the physical structure of secondary and tertiary storage devices.

Magnetic Disks

- Magnetic disks provide the bulk of secondary storage for modern computer systems. Conceptually, disks are relatively simple each disk platter has a flat circular shape, like a CD. Common platter diameters range from1.8 to 3.5 inches. The two surfaces of a platter are covered with a magnetic material. We store information by recording it magnetically on the platters.
- A read–write head "flies" just above each surface of every platter. The heads are attached to a disk arm that moves all the heads as a unit. The surface of a platter is logically divided into circular tracks, which are subdivided into sectors. The set of tracks that are at one arm position makes up a cylinder. There may be thousands of concentric cylinders in a disk drive, and each track may contain hundreds of sectors. The storage capacity of common disk drives is measured in gigabytes.
- When the disk is in use, a drive motor spins it at high speed. Most drives rotate 60 to 250 times per second, specified in terms of rotations per minute (RPM). Disk speed has two parts. The transfer rate is the rate at which data flow between the drive and the computer. The positioning time, or random-access time, consists of two parts: the time necessary to move the disk arm to the desired cylinder, called the seek time, and the time necessary for the desired sector to rotate to the disk head, called the rotational latency.

keralanotes.com

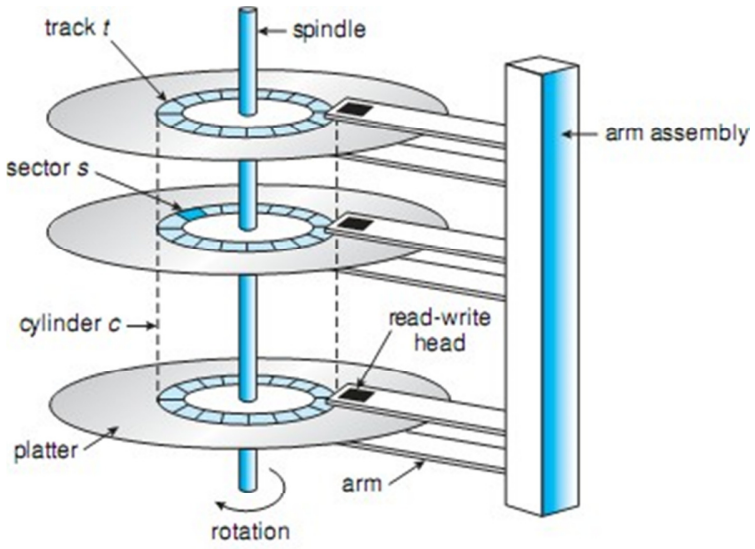


Figure 10.1 Moving-head disk mechanism.

- A disk drive is attached to a computer by a set of wires called an I/O bus. Several kinds of buses are available, including advanced technology attachment (ATA), serial ATA (SATA), e-SATA, universal serial bus (USB), and fiber channel (FC). The data transfers on a bus are carried out by special electronic processors called controllers.

Disk Structure

- Modern magnetic disk drives are addressed as large one-dimensional arrays of logical blocks, where the logical block is the smallest unit of transfer. The size of a logical block is usually 512 bytes, although some disks can be low-level formatted to have a different logical block size, such as 1,024 bytes.

- The one-dimensional array of logical blocks is mapped onto the sectors of the disk sequentially. Sector 0 is the first sector of the first track on the outermost cylinder. The mapping proceeds in order through that track, then through the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost. By using this mapping, we an—at least in theory—convert a logical block number into an old-style disk address that consists of a cylinder number, a track number within that cylinder, and a sector number within that track.
- In practice it is difficult to perform this translation because of the following reasons
    1. most disks have some defective sectors, but the mapping hides this by substituting spare sectors from elsewhere on the disk.

For More Study Materials : https://www.keralanotes.com/

2. Second, the number of sectors per track is not a constant on some drives.

- Arranging tracks in different manner , which includes

1. **Constant Linear Velocity (CLV):** The density of bits per track is uniform. The farther a track is from the center of the disk, the greater its length, so the more sectors it can hold. As we move from outer zones to inner zones, the number of sectors per track decreases.
2. **Constant Angular Velocity (CAV):** Tracks in the outermost zone typically hold 40 percent more sectors than tracks in the innermost zone. The drive increases its rotation speed as the head moves from the outer to the inner tracks to keep the same rate of data moving under the head. This method is used in CD-ROM and DVD-ROM drives. Alternatively, the disk rotation speed can stay constant; in this case, the density of bits decreases from inner tracks to outer tracks to keep the data rate constant. This method is used in hard disks and is known as constant angular velocity (CAV).

- The number of sectors per track has been increasing as disk technology improves, and the outer zone of a disk usually has several hundred sectors per track. Similarly, the number of cylinders per disk has been increasing; large disks have tens of thousands of cylinders.

Disk Scheduling

- The **seek time** is the time for the disk arm to move the heads to the cylinder containing the desired sector.
- The **rotational latency** is the additional time for the disk to rotate the desired sector to the disk head.
- The **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.
- We can improve both the access time and the bandwidth by managing the order in which disk I/O requests are serviced.
- For a multiprogramming system with many processes, the disk queue may often have several pending requests. Thus, when one request is completed, the operating system chooses which pending request to service next. The operating system makes this choice by any one of several disk-scheduling algorithms.

FCFS Scheduling

- It is the simplest of the scheduling algorithms. Consider, for example, a disk queue with requests for I/O to blocks on cylinder in that order.

For More Study Materials : https://www.keralanotes.com/

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

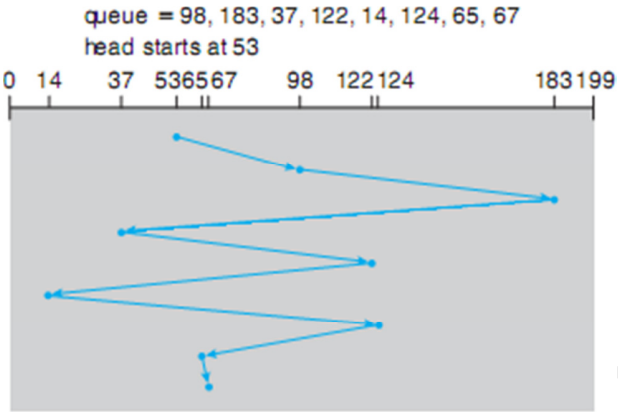


Figure 10.4 FCFS disk scheduling.

- If 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.

- The wild swing from 122 to 14 and then back to 124 illustrates the problem with this schedule. If the requests for cylinders 37 and 14 could be serviced together, before or after the requests for 122 and 124, the total head movement could be decreased substantially, and performance could be thereby improved

SSTF Scheduling

- It seems reasonable to service all the requests close to the current head position before moving the head far away to service other requests. This assumption is the basis for the shortest-seek-time-first (SSTF) algorithm. The SSTF algorithm selects the request with the least seek time from the current head position. In other words, SSTF chooses the pending request closest to the current head position.

For More Study Materials : https://www.keralanotes.com/

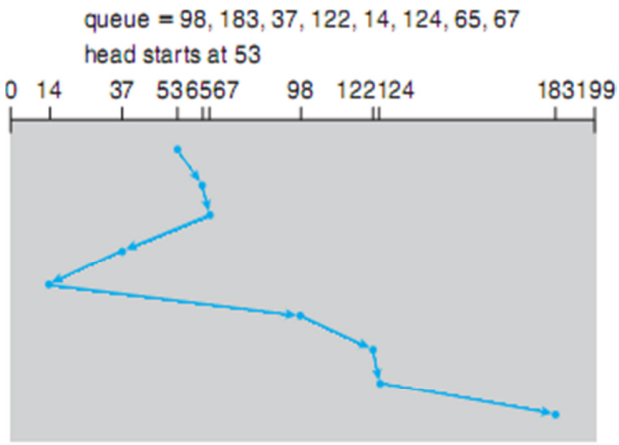


Figure 10.5 SSTF disk scheduling.

- For our example request queue, the closest request to the initial head position (53) is at cylinder 65. Once we are at cylinder 65, the next closest request is at cylinder 67. From there, the request at cylinder 37 is closer than the one at 98, so 37 is served next. Continuing, we service the request at cylinder 14, thn 98, 122, 124, and finally 183 (Figure 10.5). This scheduling method results in a total head movement of only 236 cylinders—little more than one-third of the distance needed for FCFS scheduling of this request queue. Clearly, this algorithm gives a substantial improvement in performance
- This scheduling algorithm has the disadvantages of starvation.
- Although it is not optimal ie.. In the example, we can do better by moving the head from 53 to 37, even though the latter is not closest, and then to 14, before turning around to service 65, 67, 98, 122, 124, and 183. This strategy reduces the total head movement to 208 cylinders.

SCAN Scheduling

- In the SCAN algorithm, the disk arm starts at one end of the disk and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk.
- At the other end, the direction of head movement is reversed, and servicing continues. The head continuously scans back and forth across the disk.
- The SCAN algorithm is sometimes called the elevator algorithm, since the disk arm behaves just like an elevator in a building, first servicing all the requests going up and then reversing to service requests the other way.

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

For More Study Materials : https://www.keralanotes.com/

- Before applying SCAN to schedule the requests 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.
- Assuming that the disk arm is moving toward 0 and that the initial head position is again 53, the head will next service 37 and then 14.

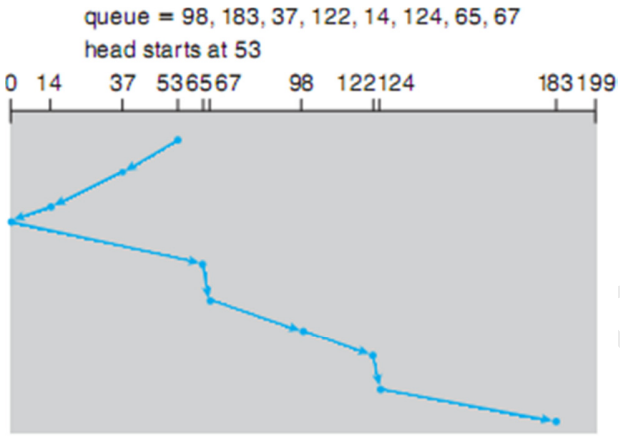


Figure 10.6 SCAN disk scheduling.

- 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.
- If a request arrives in the queue just in front of the head, it will be serviced almost immediately; a request arriving just behind the head will have to wait until the arm moves to the end of the disk, reverses direction, and comes back.

C-SCAN Scheduling

- Circular SCAN (C-SCAN) scheduling is a variant of SCAN designed to provide a more uniform wait time. Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk without servicing any requests on the return trip. The C-SCAN scheduling algorithm essentially treats the cylinders as a circular list that wraps around from the final cylinder to the first one

For More Study Materials : https://www.keralanotes.com/

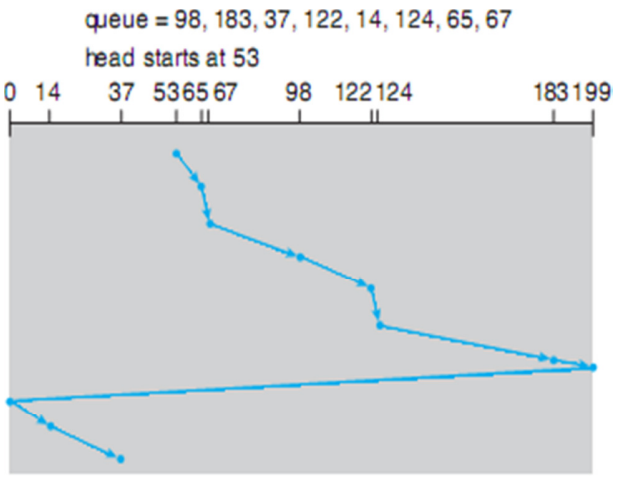


Figure 10.7 C-SCAN disk scheduling.

LOOK and C-LOOK Scheduling

- As we described them, both SCAN and C-SCAN move the disk arm across the full width of the disk. In practice, neither algorithm is often implemented this way.
- More commonly, the arm goes only as far as the final request in each direction. Then, it reverses direction immediately, without going all the way to the end of the disk.

- Versions of SCAN and C-SCAN that follow this pattern are called LOOK and C-LOOK scheduling, because they look for a request before continuing to move in a given direction

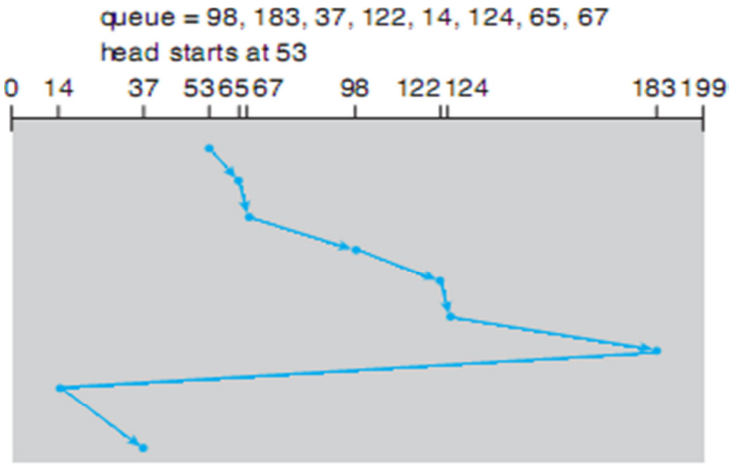


Figure 10.8 C-LOOK disk scheduling.

For More Study Materials : https://www.keralanotes.com/

Different Types of Disk Scheduling Alg.

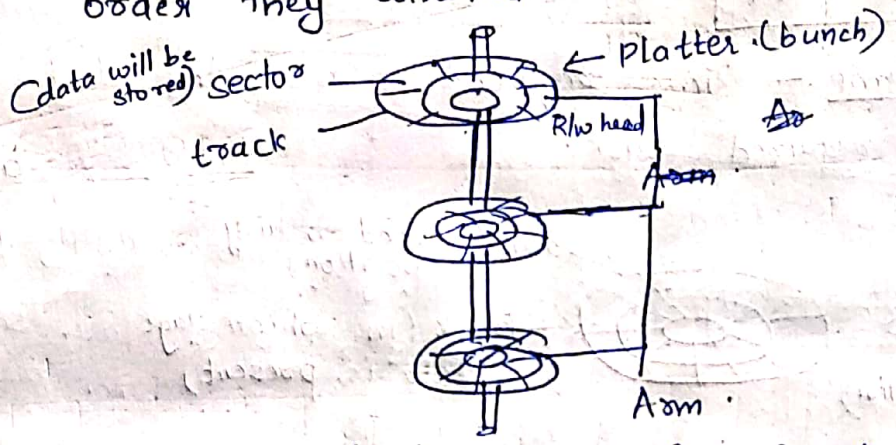
KeralaNotes

Examples

- FCFS.
- SSTF
- SCAN
- CSCAN
- LOOK
- CLOOK

① FCFS Alg. (First Come First Served)

- Simplest of all disk scheduling alg.
- In FCFS, ~~all~~ the requests are addressed in the order they arrived in the disk queue.



* Arm ~~is placed in~~ for read/write R/w head ~~is~~ placed in platter. From platter → surface → Track → Sector

Advantage (FCFS)

- Every request gets a fair chance.
- No indefinite postponment.

Disadvantage (FCFS)

- Does not try to optimize seek time.
- May not provide the best possible service.

Eg:- A disk contains 200 tracks (0-199)

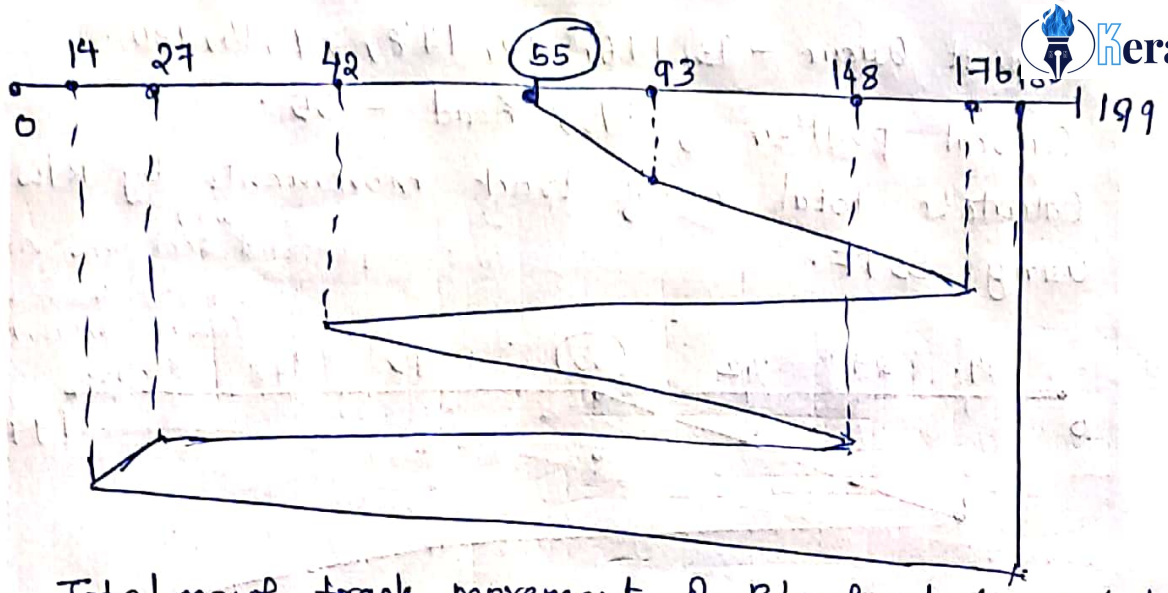
- Request queue contains track no: 93, 176, 42, 148, 27, 14, 180 ~~82, 170, 43, 140, 24, 16, 190~~.
- Current position of R/w head = 55

Calculate total no. of track movement of R/w head using FCFS.

For More Study Materials : https://www.keralanotes.com/

Scanned with CamScanner

93, 176, 42, 148, 27, 14, 180



Total no. of track movement of R/W head (Forward & backward)

$= (176-55) + (176-42) + (148-42) + (148-14) + (180-14) = \underline{\underline{661}}$

② SSTF (Shortest Seek Time First)

* In SSTF, request having shortest seek time are executed first.
* So the seek time of every request is calculated in advance in queue & schedule according to their seek time.
* As a result, request near the disk arm will get executed first.

Advantages (SSTF)

* Avg response time will be decreased.
* Throughput increased.

Disadvantage (SSTF)

* Overhead to calculate the seek time in advance.
* Causes of starvation (waiting for long time) for a request if it has higher seek time.

For More Study Materials : https://www.keralanotes.com/

Scanned with CamScanner

Eg: Disk contains 200 tracks (0-199)

Request Queue – 93, 176, 42, 148, 27, 14, 180

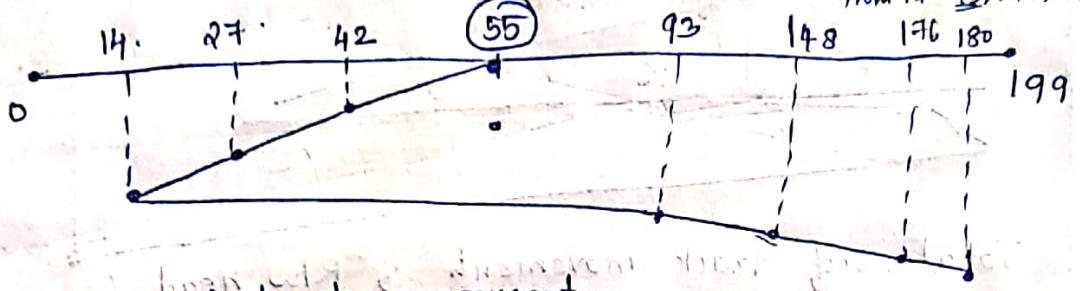
Current position of R/W head – 55.

Calculate total no. of track movements by R/W head using SSTF.

[Shortest (nearest) seek time visited first)

[42, 93 – 42 is nearest]

from 14 – 93, 176, 148, 180



Total no. of track movement

$= (55 - 14) + (180 - 14) = \underline{\underline{207}}$.

③ SCAN.

* The arm moves in one direction only & satisfying all outstanding requests, until there is no more request in that direction.
* Service direction is then reversed.
* It works like an elevator.

Advantages

* High Throughput
* Low variance of response time
* Avg response time

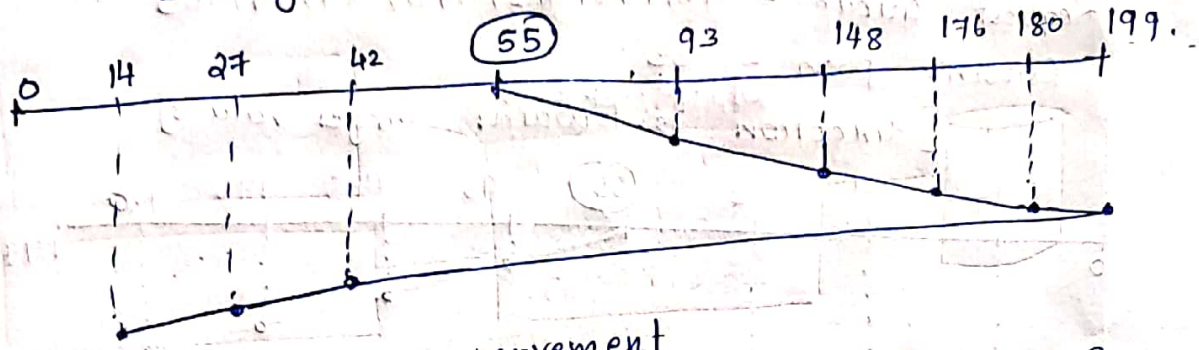
Disadvantage.

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

For More Study Materials : https://www.keralanotes.com/

Scanned with CamScanner

Eg: Disk contains 200 tracks (0-199)
Request queue - 93, 176, 42, 148, 27, 14,
Current position of R/w head - 55
[Moving towards larger values].



Total no. of track movement
$= (199-55) + (199-14) = \underline{329}$

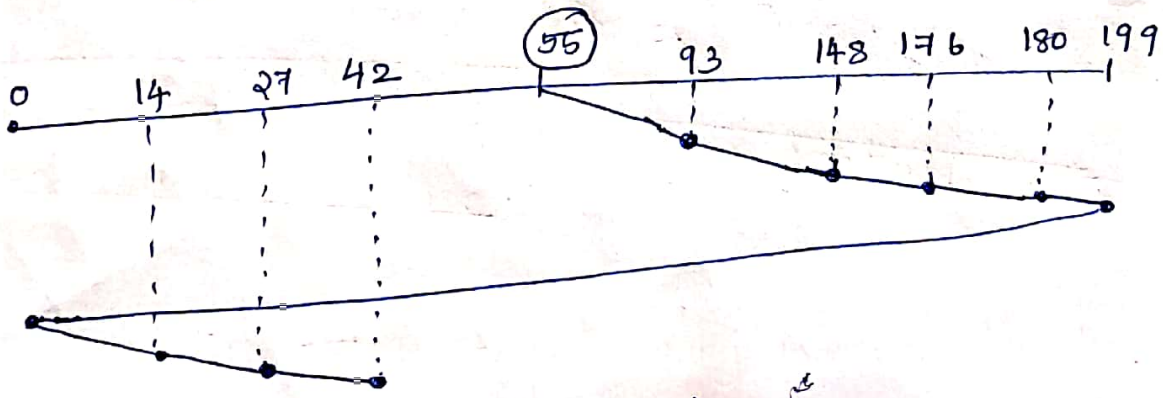
(4) CSCAN. (Circular SCAN)

* It restrict scanning to one direction only.
* When the last track has been visited in one direction, the arm is returned to opposite end of disk & scan begins again.

Advantages.

* Provides more uniform wait time compared to SCAN.

Eg: Disk contains 200 tracks (0-199)
Request queue - 93, 176, 42, 148, 27, 14, 180.
Current position of R/w head - 55.
[direction is towards large value]



Total no. of track movement
$= (199-55) + (199-0) + (42-0) = \underline{385}$

For More Study Materials : https://www.keralanotes.com/
Scanned with CamScanner

⑤ LOOK

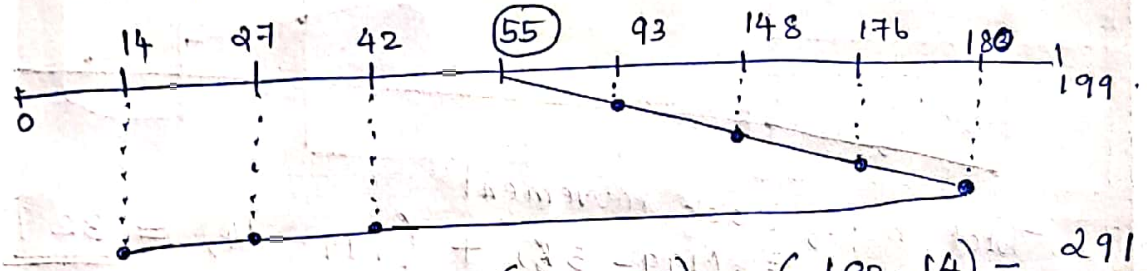
* Same as SCAN algorithm but arm goes only the final request.

Eg: Disk contains – 0 - 199 tracks.

Request queue – 93, 176, 42, 148, 27, 14, 180.

Current posn – 55.

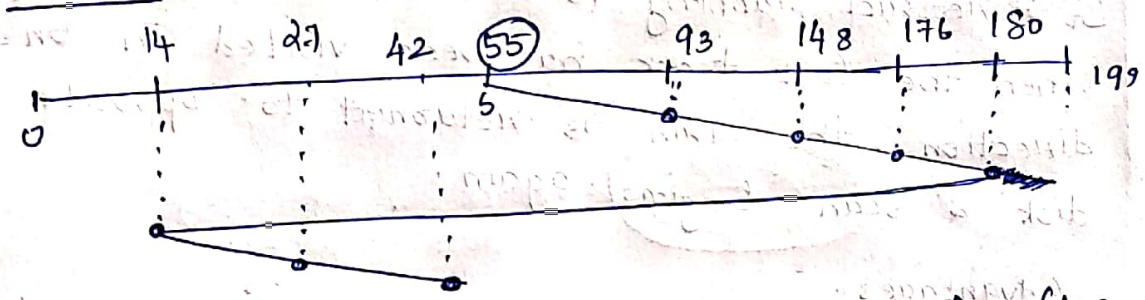
[direction is towards larger values].



Total track mov. = (180 – 55) + (180 – 14) = 291

⑥ CLOOK

Eg:



Total track mov = (180 – 55) + (180 – 14) + (42 – 14)

= 319

For More Study Materials : https://www.keralanotes.com/

Scanned with CamScanner

Disk Formatting

- A new magnetic disk is a blank slate: it is just a platter of a magnetic recording material. Before a disk can store data, it must be divided into sectors that the disk controller can read and write. This process is called low-level formatting,or physical formatting.
- Low-level formatting fills the disk with a special data structure for each sector. The data structure for a sector typically consists of a header, a data area (usually 512 bytes in size), and a trailer.
- The header and trailer contain information used by the disk controller, such as a sector number and an error-correcting code (ECC).
- When the controller writes a sector of data during normal I/O, the ECC is updated with a value calculated from all the bytes in the data area. When the sector is read, the ECC is recalculated and compared with the stored value. If the stored and calculated numbers are different, the data area of the sector has become corrupted and that the disk sector may be bad.
- The ECC is an error-correcting code. If only a few bits of data have been corrupted, ECC enable the controller to identify which bits have been changed and calculate what their correct values should be.
- It then reports a recoverable soft error. The controller automatically does the ECC processing whenever a sector is read or written.
- For many hard disks, when the disk controller is instructed to low-level-format the disk, it can also be told how many bytes of data space to leave between the header and trailer of all sectors.
- It is usually possible to choose among a few sizes, such as 256, 512, and 1,024 bytes. Formatting a disk with a larger sector size means that fewer sectors can fit on each track; but it also means that fewer headers and trailers are written on each track and more space is available for user data.
- Some operating systems can handle only a sector size of 512 bytes.
- The first step is to partition the disk into one or more groups of cylinders. The operating system can treat each partition as though it were a separate disk.
- The second step is logical formatting, or creation of a file system. In this step, the operating system stores the initial file-system data structures onto the disk.
- To increase efficiency, most file systems group blocks together into larger chunks, frequently called clusters.
- Some operating systems give special programs the ability to use a disk partition as a large sequential array of logical blocks, without any file-system data structures.

keralanotes.com

5.2 File Concepts

- A file is a named collection of related information that is recorded on secondary storage.
- files represent programs and data.
- The information in a file is defined by its creator.
- A file has a certain defined structure which depends on its type.
  - ***text file, source file, executable file***

5.2.1 File Attributes

- A file is referred to by its name.
- A file's attributes vary from one operating system to another but typically consist of these:

- **Name:** The symbolic file name is the only information kept in human readable form.
- **Identifier:** This unique tag, usually a number, identifies the file within the file system; it is the non-human-readable name for the file.
- **Type:** This information is needed for systems that support different types of files.
- **Location:** This information is a pointer to a device and to the location of the file on that device.
- **Size:** The current size of the file (in bytes, words, or blocks) and possibly the maximum allowed size are included in this attribute.
- **Protection:** Access-control information determines who can do reading, writing, executing, and so on.
- **Time, date, and user identification:** This information may be kept for creation, last modification, and last use. These data can be useful for protection, security, and usage monitoring.

5.2.2 File Operations

- A file is an abstract data type.
- To define a file properly, it needs to consider the operations that can be performed on files.

  Creating a file: Two steps are necessary to create a file. First, space in the file system must be found for the file.

  Writing a file: To write a file, we make a system call specifying both the Name of the file and the information to be written to the file. The system must keep a write pointer to the location in the file where the next write is to take place.
- **Reading a file:** To read from a file, we use a system call that specifies the name of

keralanotes.com

the file and where (in memory) the next block of the file should be put.

- **Repositioning within a file.** The directory is searched for the appropriate entry, and the current-file-position pointer is repositioned to a given value (seek).
- **Deleting a file.** To delete a file, we search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase the directory entry.
- **Truncating a file.** The user may want to erase the contents of a file but keep its attributes.

➢ Other common operations include
  - Appending, renaming and copy
  - Several pieces of information are associated with an open file.
  - File pointer.
  - File-open count.
  - Disk location of the file.
  - Access rights

5.2.3 File Types

➢ The operating system should recognize and support file types. If an operating system recognizes the type of a file, it can then operate on the file in reasonable ways.

➢ The system uses the extension to indicate the type of the file and the type of operations that can be done on that file.

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

keralanotes.com

5.2.4 File Structure

- File types can be used to indicate the internal structure of the file.
- The operating requires that an executable file have a specific structure so that it can system termine where in memory to load the file and what the location of

- None - sequence of words, bytes
- Simple record structure
  - Lines
  - Fixed length
  - Variable length
- Complex Structures
  - Formatted document
  - Relocatable load file
- Can simulate last two with first method by inserting appropriate control characters

Internal File Structure

- Internally, locating an offset within a file can be complicated for the operating system.
- Disk systems typically have a well-defined block size determined by the size of a sector.
- All disk I/0 is performed in units of one block (physical record), and all blocks are the same size.
- It is unlikely that the physical record size will exactly match the length of the desired logical record. Logical records may even vary in length. Packing a number of logical records into physical blocks is a common solution to this problem.

keralanotes.com

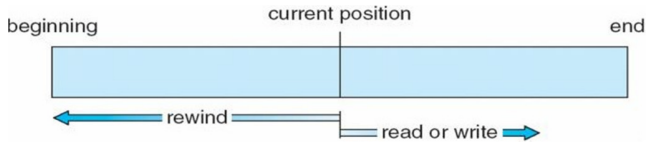
5.3 Access Methods

- Files store information.
- When it is used, this information must be accessed and read into computer memory.
- The information in the file can be accessed in several ways.
- Some systems provide only one access method for files while others provide many access methods.
- A major design problem is choosing one among them.

1. Sequential Access

- Information in the file is processed in order, one record after the other.

- Sequential Access file operations
  - **read_next()** - Read next portion of file and automatically advances a file pointer.
  - **write_next()** – Appends to the end of the file and advances to the end of the newly written material.
  - **Reset** – Back to the beginning of file
  - no read after last write



2. Direct Access (Relative Accesses)

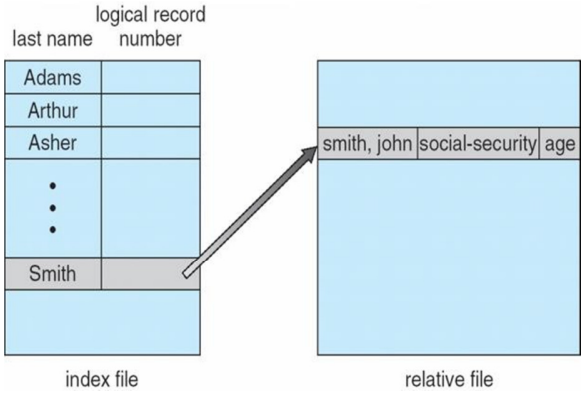
- A file is made up of fixed length logical records, that allow programs to read and write records rapidly in no particular order.
- For direct access disk is viewed as numbered sequence of blocks or record.

- Direct access file operations
  - The file operations were modified to include the block number as a parameter.
  - Read(*n*)
  - Write(*n*)
  - Position_file(n)

    n = relative block number

3. Other Access Methods

- Can be built on top of base methods
- General involve creation of an index for the file
- Keep index in memory for fast determination of location of data to be operated on (consider UPC code plus record of data about that item)
- If too large, index (in memory) of the index (on disk)
- To find a record, we first search the index and then use pointer to access the file directly and to find desired record.



Example of index and relative files.

For More Study Materials : https://www.keralanotes.com/

5.4 Protection

- When information is stored in a computer system, it must be keep safe from physical damage (the issue of reliability) and improper access (the issue of protection).
- Reliability is generally provided by duplicate copies of files.
- Protection can be provided in many ways.
  - Types of Access
  - Access Control
- Other Protection Approaches

5.4.1 Types of Access

- Access is permitted or denied depending on several factors, one of which is the type of access requested.
- Several different types of operations may be controlled:
  - **Read:** Read from the file.
  - **Write:** Write or rewrite the file.
  - **Execute:** Load the file into memory and execute it.
  - **Append:** Write new information at the end of the file.
  - **Delete:** Delete the file and free its space for possible reuse.
  - **List**: List the name and attributes of the file.

5.4.2 Access Control

- The most common approach to the protection problem is to make access dependent on the identity of the user.
- Different users may need different types of access to a file or directory.
- To implement dependent access is to associate with each file and directory an access control list (ACL)specifying user names and the types of access allowed for each user.
- Mode of access: read, write, execute
- Many systems recognize three classifications of users in connection with each file:
  - **Owner:** The user who created the file is the owner.
  - **Group**: A set of users who are sharing the file and need similar access is a group, or work group.
  - **Universe:** All other users in the system constitute the universe.

5.5 File-System Implementation

- Several on-disk and in-memory structures are used to implement a file system.
- The file system may contain information about how to boot an operating system

keralanotes.com

stored there, the total number of blocks, the number and location of free blocks, the directory structure, and individual files.

- Boot control block contains info needed by system to boot OS from that volume
  - Needed if volume contains OS, usually first block of volume
- Volume control block (superblock, master file table) contains volume details

Total number of blocks, number of free blocks, blocks size, free blocks pointers or array

- Directory structure organizes the files
  - Names and inode numbers, master file table
- Per-file File Control Block (FCB) contains many details about the file
  - inode number, permissions, size, dates
  - NFTS stores into in master file table using relational DB structures

file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

A typical file-control block.

- The in-memory information is used for both file-system management and performance improvement via caching.
- The data are loaded at mount time, updated during file-system operations, and discarded at dismount.
- Several types of structures may be included.
  - An in-memory mount table contains information about each mounted volume.
  - An in-memory directory-structure cache holds the directory information of recently accessed directories.
  - The system wide open file table contains a copy of the FCB of each open file, as well as other information.
  - The per process open file table contains a pointer to the appropriate entry in the system-wide open-file table, as well as other information.
- Buffers hold file-system blocks when they are being read from disk or written to disk.

Partitions and Mounting

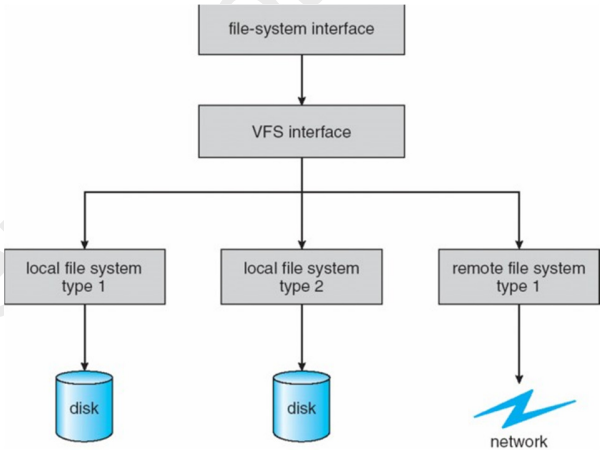
- Partition can be a volume containing a file system ("cooked") or raw – just a sequence of blocks with no file system
- Boot block can point to boot volume or boot loader set of blocks that contain enough code to know how to load the kernel from the file system
  - Or a boot management program for multi-os booting

For More Study Materials : https://www.keralanotes.com/

- Root partition contains the OS, other partitions can hold other Oses, other file systems, or be raw
  - Mounted at boot time
  - Other partitions can mount automatically or manually
- At mount time, file system consistency checked
  - Is all metadata correct?
    - If not, fix it, try again
    - If yes, add to mount table, allow access

Virtual File Systems

- Virtual File Systems (VFS) on Unix provide an object-oriented way of implementing file systems
- VFS allows the same system call interface (the API) to be used for different types of file systems
  - Separates file-system generic operations from implementation details
  - Implementation can be one of many file systems types, or network file system
    - Implements **vnodes** which hold inodes or network file details
  - Then dispatches operation to appropriate file system implementation routines



Schematic view of a virtual file system.

5.6 Directory Implementation

- The selection of directory-allocation and directory-management algorithms significantly affects the efficiency, performance, and reliability of the file system

Linear List

- The simplest method of implementing a directory is to use a linear list of file names with pointers to the data blocks.
- This method is simple to program but time-consuming to execute.
- The real disadvantage of a linear list of directory entries is that finding a file requires a linear search.

Hash Table

- With this method, a linear list stores the directory entries, but a hash data structure is also used. The hash table takes a value computed from the file name and returns a pointer to the file name in the linear list.
- Therefore, it can greatly decrease the directory search time.
- Insertion and deletion are also fairly straightforward, although some provision must be made for collisions-situations in which two file names hash to the same location.
- The major difficulties with a hash table are its generally fixed size and the dependence of the hash function on that size.
- a chained-overflow hash table can be used.
- Each hash entry can be a linked list instead of an individual value, and we can resolve collisions by adding the new entry to the linked list.

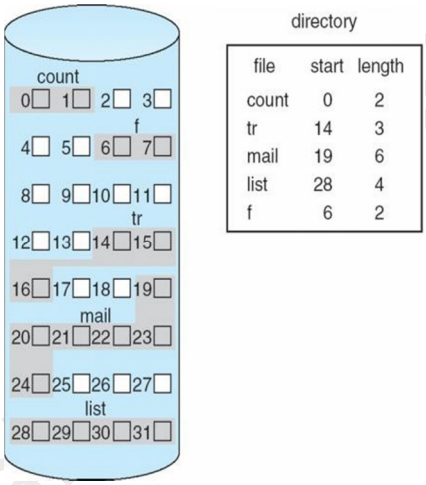
keralanotes.com

5.7 Allocation Methods

- An allocation method refers to how disk blocks are allocated for files.

5.7.1 Contiguous Allocation

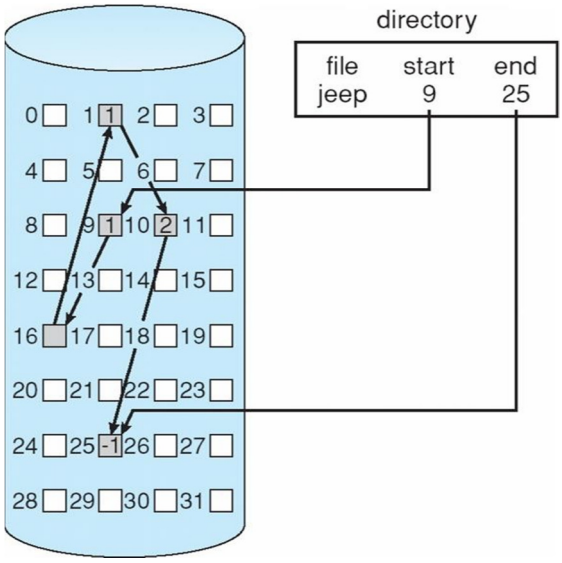
- requires that each file occupy a set of contiguous blocks on disk.
- Disk addresses define a linear ordering on the disk.
- Best performance in most cases
- Simple – only starting location (block number) and length (number of blocks) are required
- Problems include finding space for file, knowing file size, external fragmentation, need for **compaction off-line (downtime) or on-line**



5.7.2 Linked Allocation

- With linked allocation, each file is a linked list of disk blocks; the disk blocks may be scattered anywhere on the disk.
- The directory points to the first and last blocks of the file.
- File ends at nil pointer
- No external fragmentation
- Each block contains pointer to next block
- No compaction, external fragmentation
- Free space management system called when new block needed
- Improve efficiency by clustering blocks into groups but increases internal fragmentation
- Reliability can be a problem
- Locating a block can take many I/Os and disk seeks

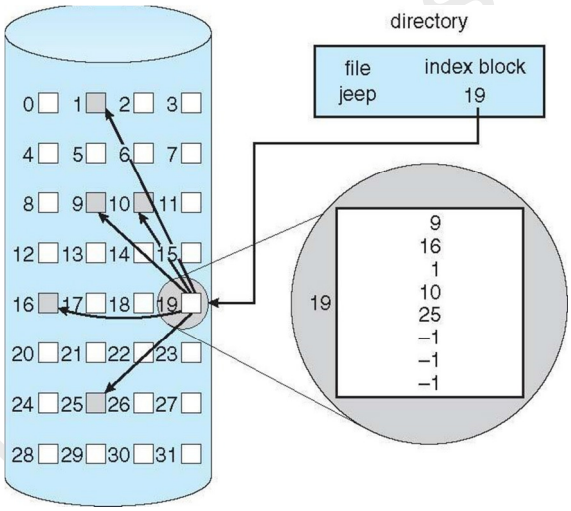
For More Study Materials : https://www.keralanotes.com/



Linked allocation

5.7.3 Indexed allocation

- Linked allocation solves the external-fragmentation and size-declaration problems of contiguous allocation.
- Indexed allocation solves this problem by bringing all the pointers together into one location: the index block.



- Each file has its own index block, which is an array of disk-block addresses. The ith entry in the index block points to the ith block of the file.
- The directory contains the address of the index block. To find and read the ith block, we use the pointer in the ith index-block entry.
  - When the file is created, all pointers in the index block are set to null. When the ith block is first written, a block is obtained from the free-space manager, and its address is put in the ith index-block entry.
  - Indexed allocation supports direct access, without suffering from external fragmentation, because any free block on the disk can satisfy a request for more space. Indexed allocation does suffer from wasted space, however.

For More Study Materials : https://www.keralanotes.com/

SYSTEM PROTECTION

GOALS OF PROTECTION

- Protection mechanisms control access to a system by limiting the types of file access permitted to users. In addition, protection must ensure that only processes that have gained proper authorization from the operating system can operate on memory segments, the CPU, and other resources.
- Protection is provided by a mechanism that controls the access of programs, processes, or users to the resources defined by a computer system. This mechanism must provide a means for specifying the controls to be imposed, together with a means of enforcing them.
- Security ensures the authentication of system users to protect the integrity of the information stored in the system (both data and code), as well as the physical resources of the computer system. The security system prevents unauthorized access, malicious destruction or alteration of data, and accidental introduction of inconsistency.
- Untrustworthy users might safely share a common logical name space, such as a directory of files, or share a common physical name space, such as memory.
- Protection is necessary to
    1. Prevent the mischievous, intentional violation of an access restriction by user.
    2. To ensure that each program component active in a system uses system resources only in ways consistent with stated policies.
    3. A protection-oriented system provides means to distinguish between authorized and unauthorized usage.
    4. The role of protection in a computer system is to provide a mechanism for the enforcement of the policies governing resource use. A protection system must have the flexibility to enforce a variety of policies.
- Note that mechanisms are distinct from policies. Mechanisms determine how something will be done; policies decide what will be done. The separation of policy and mechanism is important for flexibility. Policies are likely to change from place to place or time to time. In the worst case, every change in policy would require a change in the underlying mechanism. Using general mechanisms enables us to avoid such a situation.

For More Study Materials : https://www.keralanotes.com/

6.2 Principles of Protection

- A key, time-tested guiding principle for protection is the principle of least privilege. It dictates that programs, users, and even systems be given just enough privileges to perform their tasks.
- Consider the analogy of a security guard with a passkey. If this key allows the guard into just the public areas that she guards, then misuse of the key will result in minimal damage. If, however, the passkey allows access to all areas, then damage from its being lost, stolen, misused, copied, or otherwise compromised will be much greater.
- The overflow of a buffer in a system daemon might cause the daemon process to fail but should not allow the execution of code from the daemon process's stack that would enable a remote user to gain maximum privileges and access to the entire system
- The creation of audit trails for all privileged function access. The audit trail allows the programmer, system administrator, or law-enforcement officer to trace all protection and security activities on thesystem.
- Managing users with the principle of least privilege entails creating a separate account for each user, with just the privileges that the user needs. An operator who needs to mount tapes and back up files on the system has access to just those commands and files needed to accomplish the job. Some systems

  implement role-based access control (RBAC) to provide this functionality.

6.3 Domain of Protection

- A computer system is a collection of processes and objects. By objects, we mean both hardware objects (such as the CPU, memory segments, printers, disks, and tape drives) and software objects (such as files, programs, and semaphores). Each object has a unique name that differentiates it from all other objects in the system, and each can be accessed only through well-defined and meaningful operations. Objects are essentially abstract data types.
- The operations that are possible may depend on the object.
- A process should be allowed to access only those resources for which it has authorization.
- At any time, a process should be able to access only those resources that it currently requires to complete its task. This second requirement, commonly referred to as the need-to-know principle, is useful in limiting the amount of damage a faulty process can cause in thesystem.

For More Study Materials : https://www.keralanotes.com/

1. Domain Structure

- A protection domain, which specifies the resources that the process may access. Each domain defines a set of objects and the types of operations that may be invoked on each object.
- The ability to execute an operation on an object is an accessright. A domain is a collection of access rights, each of which is an ordered pair <object-name, rights- set>.
- For example, if domain D has the access right <file F, {read,write}>, then a process executing in domain D can both read and write file F. It cannot, however, perform any other operation on that object.
- Domains may share access rights. Ex. we have three domains: D1, D2,and D3. The access right <O4, {print}> is shared by D2 and D3, implying that a process executing in either of these two domains can print object O4.
- The association between a process and a domain may be either static, if the set of resources available to the process is fixed throughout the process's lifetime, or dynamic.
- If the association between processes and domains is fixed, and we want to adhere to the need-to-know principle, then a mechanism must be available to change the content of a domain.

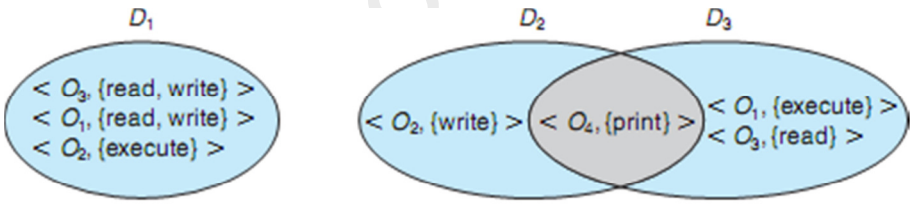


Figure 14.1 System with three protection domains.

- If the association is dynamic, a mechanism is available to allow domain switching, enabling the process to switch from one domain to another. We may also want to allow the content of a domain to be changed. If we cannot change the content of a domain, we can provide the same effect by creating a new domain with the changed content and switching to that new domain when we want to change the domain content.
- A domain can be realized in a variety of ways:
    1. Each user may be a domain.
    2. Each process may be a domain.
    3. Each procedure may be a domain.

For More Study Materials : https://www.keralanotes.com/

6.4 Access Matrix

- Our general model of protection can be viewed abstractly as a matrix, called an access matrix.
- The rows of the access matrix represent domains, and the columns represent objects.
- Each entry in the matrix consists of a set of access rights. Because the column defines objects explicitly, we can omit the object name from the access right. The entry access(i,j) defines the set of operations that a process executing in domain Di can invoke on object Oj.
- Consider a sample Access Matrix

domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Figure 14.3 Access matrix.

- There are four domains and four objects—three files (F1, F2, F3) and one laser printer. A process executing in domain D1 can read files F1 and F3. A process executing in domain D4 has the same privileges as one executing in domain D1; but in addition, it can also write onto files F1 and F3.The laser printer can be accessed only by a process executing in domain D2.
- The access-matrix scheme provides us with the mechanism for specifying a variety of policies. The mechanism consists of implementing the access matrix and ensuring that the semantic properties we have outlined hold. More specifically, we must ensure that a process executing in domain Di can access only those objects specified in row i, and then only as allowed by the access- matrix entries.
- The access matrix can implement policy decisions concerning protection.
- The access matrix provides an appropriate mechanism for defining and implementing strict control for both static and dynamic association

For More Study Materials : https://www.keralanotes.com/

between processes and domains.

- Processes should be able to switch from one domain to another. Switching from domain Di to domain Dj is allowed if and only if the access right switch ∈ access(i, j).
- In the following figure , a process executing in domain D2 can switch to domain D3 or to domain D4. A process in domain D4 can switch to D1,and one in domain D1 can switch to D2.

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

Figure 14.4 Access matrix of Figure 14.3 with domains as objects.

keralanotes