6 File System and Disk Storage

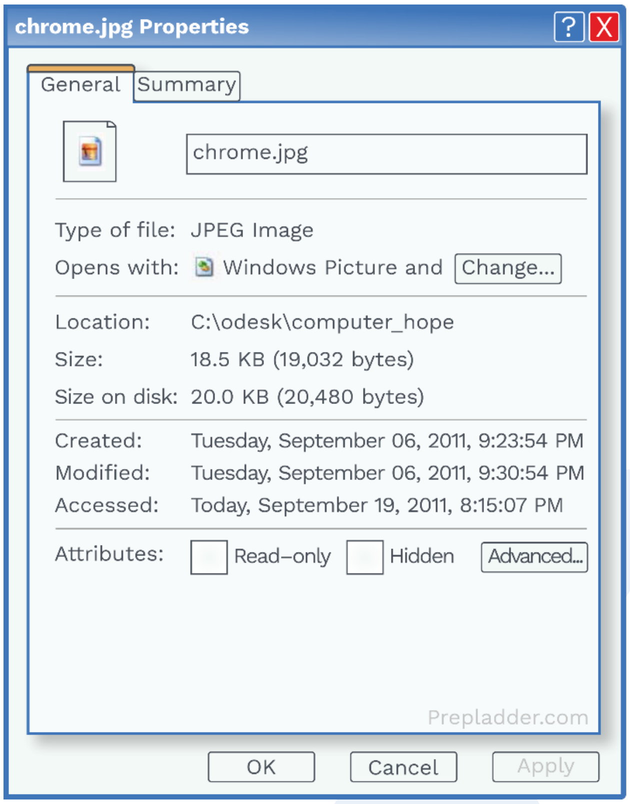
6.1 BASICS OF FILE

- The file system provides an abstract data structure of the file present in the hard disk (secondary storage).
- File system provides the online storage and access mechanism for both data and programs of the operating system.
- File system is a combination or collection of two main objects like directories and files.
- For stored information, the uniform logical view is provided by OS to define the file, i.e., logical storage unit, it (OS) abstracts the physical properties of its storage devices.
- File is a collection of related information, and it is last stage/smallest allotment which could be stored on the secondary storage. Any data could be stored only in the form of files onto secondary storage.
- Files have a defined structure, like for instance, a text file .txt which has a sequence of characters organized into lines, and likewise an (.exe) executable file which has code/programs.

File attributes:

A file is named to get it identified uniquely by its end users. A file has various attributes attached to it, which vary from one operating system to another. It typically consists of:

1) **Name:** The symbolic name of the file is the only information kept in readable form.
2) **Identifier:** A file is allocated in the system with a distinctive tag.
3) **Type:** It represents what type of file it is.
4) **Location:** The location of a file is the actual position on the disk where the file is stored.
5) **Size:** This gives info about the present file size, possibly maximum file size extension.
6) **Protection:** Access–control information determines who can read, write and execute.
7) **Time and date:** This attribute says about creation, last modified, last used times of a file.



chrome.jpg Properties

Type of file: JPEG Image
Opens with: Windows Picture and
Location: C:\odesk\computer_hope
Size: 18.5 KB (19,032 bytes)
Size on disk: 20.0 KB (20,480 bytes)
Created: Tuesday, September 06, 2011, 9:23:54 PM
Modified: Tuesday, September 06, 2011, 9:30:54 PM
Accessed: Today, September 19, 2011, 8:15:07 PM
Attributes: □ Read–only □ Hidden

Fig.6.1 File Attributes

Note:

In a system with many files, the size of the directory itself may be in megabytes because files are non_volatile.

File operations:

A file is an abstract data type. Various types of operations can be performed on the file by the resident operating system using system calls like create, read, write, delete and truncate.

- **Creating a file:** For a file creation, the first thing to have is storage space and secondly need to enter the unmatched or unique file name in the directory, these are prerequisites for file creation.
- **Writing a file:** Writing on to the file is done using a system call with specifying both names of the file and the data to be written to the file. A writer pointer is kept to the location of the file from where or the position

which needs to be start writing and pointer need to be updated accordingly as per writes in the file.

- **Reading a file:** To read a file, a read pointer is provided by the OS.
- **Repositioning within a file:** For an appropriate entry, the directory is searched. The pointer is repositioned in the file to the given entry from the current position of the file. There is no I/O involvement for the repositioning of the pointer within a file. This operation is called file seek.
- **Deleting a file:** A file (content of file) is deleted along with its attributes, and space is released. The space is again reusable.
- **Truncating a file:** A file truncation includes the content removal from the file but not the complete deletion of the file. In file truncation, attributes of the file remains but only data/content from the file is removed.

Note:

Other common operations include appending new information to the end of an existing file and renaming an existing file.

Grey Matter Alert!

Open() system call is used by most of the OS in order to avoid constant searching. Information associated with an open() file system call.

1) Access rights

2) File's location on the disk

There are also operating systems which provide the file lockers. So, a file could be safe when it is a sharable resource. If a file is locked by a process, then it would not allow other processes to gain access.

SOLVED EXAMPLES

Q1 **Which of the following statements is/are TRUE?**

a) The file systems manage only secondary storage data

b) Native and mounted file systems must be similar in type

c) A file system creates a hard link to a file in a mounted file system

d) Tape storage is most appropriately managed by a sequential file organization

Sol: **Option: d)**

a) (FALSE) The file system manages files on secondary as well as main storage

b) (FALSE) One of the primary advantages of mounting a file system is to enable multiple heterogeneous file systems

c) (FALSE) Soft links are path name, and hard links are directory entries

d) Tape storage is accessed sequentially

File types:

A file is recognized by the operating system on its type and operates accordingly. The types could be like .doc, .txt, etc.

The extension type of a file indicates the type of operation we can perform on files.

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	doc, txt	textual data, documents

word processor	tex, rtf, doc	various word–processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	gif, 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, mp3, mp4, avi	binary file containing audio or A/V information

Fig. 6.2 Common File Types

Directory:

- In a computer system, millions of files are stored on random-access storage devices, including the hard disks, optical disks, and memory-based disks.
- To organize these files properly, a directory structure is used by the file system.
- Directory is a collection of correlated files which keeps entries of all files.

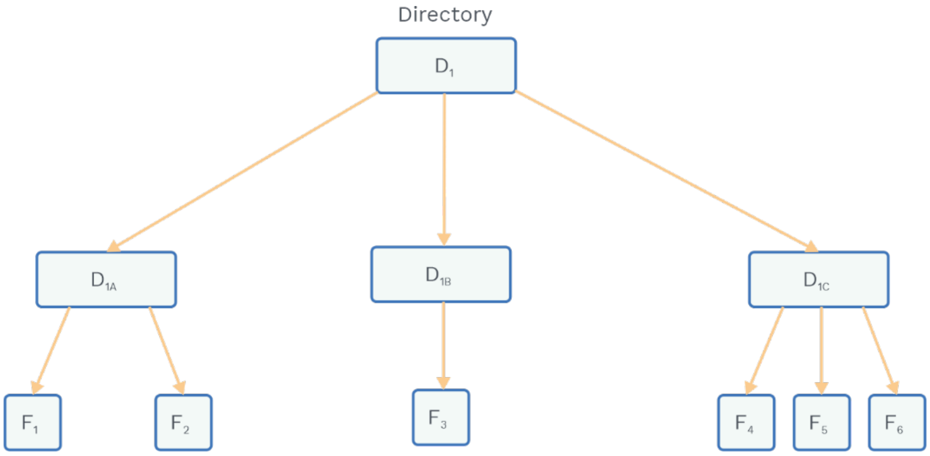
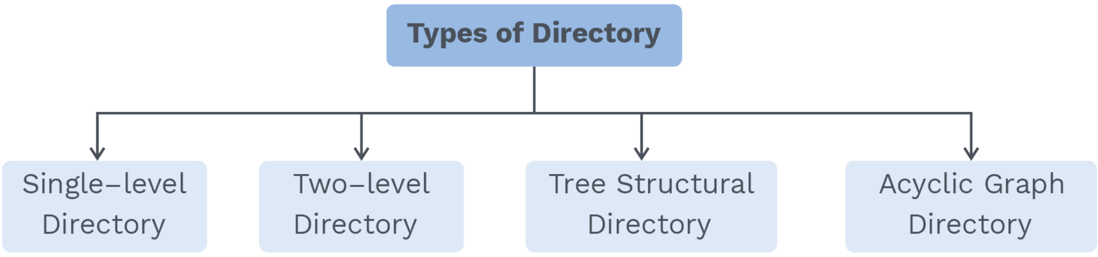


Fig. 6.3 Directory Structure

Types of directory structure:



Single–level directory:

- The simplest directory structure is the single level directory. Which consists of one directory with all the files in the same directory.

Advantages:

1) Since we have only one level structure, implementation becomes easy.
2) It's faster to search a file in this structure if the number of files is low.
3) File operations are easy.

Disadvantages:

1) It consumes more time if files are high in number.
2) Users cannot be separated.

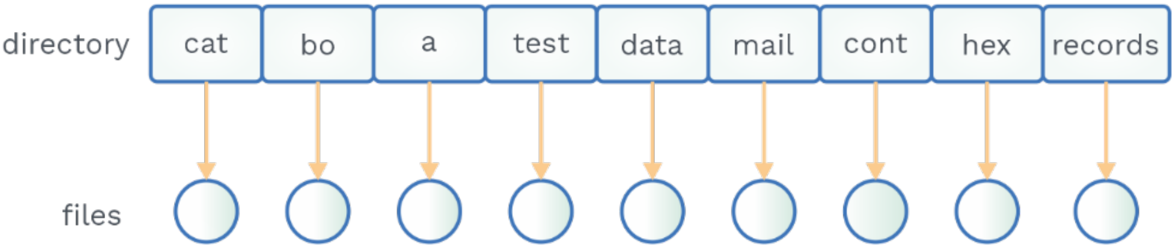


Fig. 6.4 Single-Level Directory

Two–level directory:

- Among multiple users, this (single level directory) structure gives confusion in file names.
- So, to overcome ambiguity, creating a separate directory for an individual user is good.
- Every user has UFD (user file directory) of their own.
- The structure of UFDs is similar, but each directory lists files of single user.
- For unique file naming purpose, user needs to know the path name of file desired.

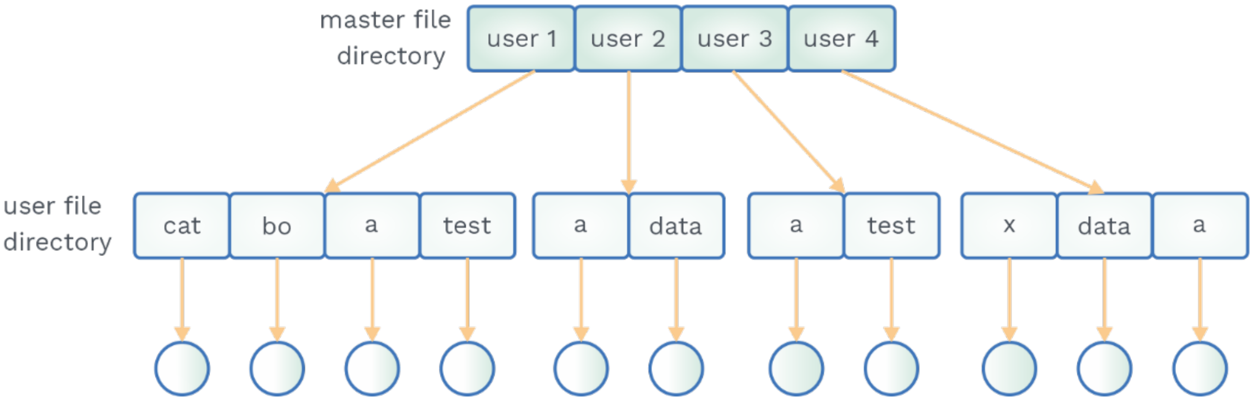


Fig. 6.5 Two-Level Directory

Advantages:

1) Isolation of users from one to another is effectively done in this structure. So, the same directory or same file name could exist for multiple users, because paths are different.
2) Searching of files become easier due to path name and user grouping.

Disadvantages:

1) The implementation of a directory structure is difficult.
2) Files are not permitted to share with users. Thus, if the same file exists in two different directories, then updates made in any one of the files are not reflected in the other file. This leads to the problem of inconsistency.

Tree-structured directory:

- Tree-structured directory is the extension of a two-level directory tree structure to a tree of arbitrary height.
- In this, users can create their own subdirectories and can keep their files in a more organized manner.
- Tree structure is the most frequently used directory structure.
- Unique file route/path for each file and root directory could be found in the tree structure.

Advantages:

1) The probability of name collision is very less as each file has different paths.
2) Full path name of the file is given, which makes searching very easy.
3) By using a relative search or absolute path search, we could find the file.

Disadvantages:

1) Implementation of this directory structure is more complicated.
2) It is not efficient since accessing a file may lead to accessing of multiple directories.
3) Sharing of directories and files is prohibited in this structure.

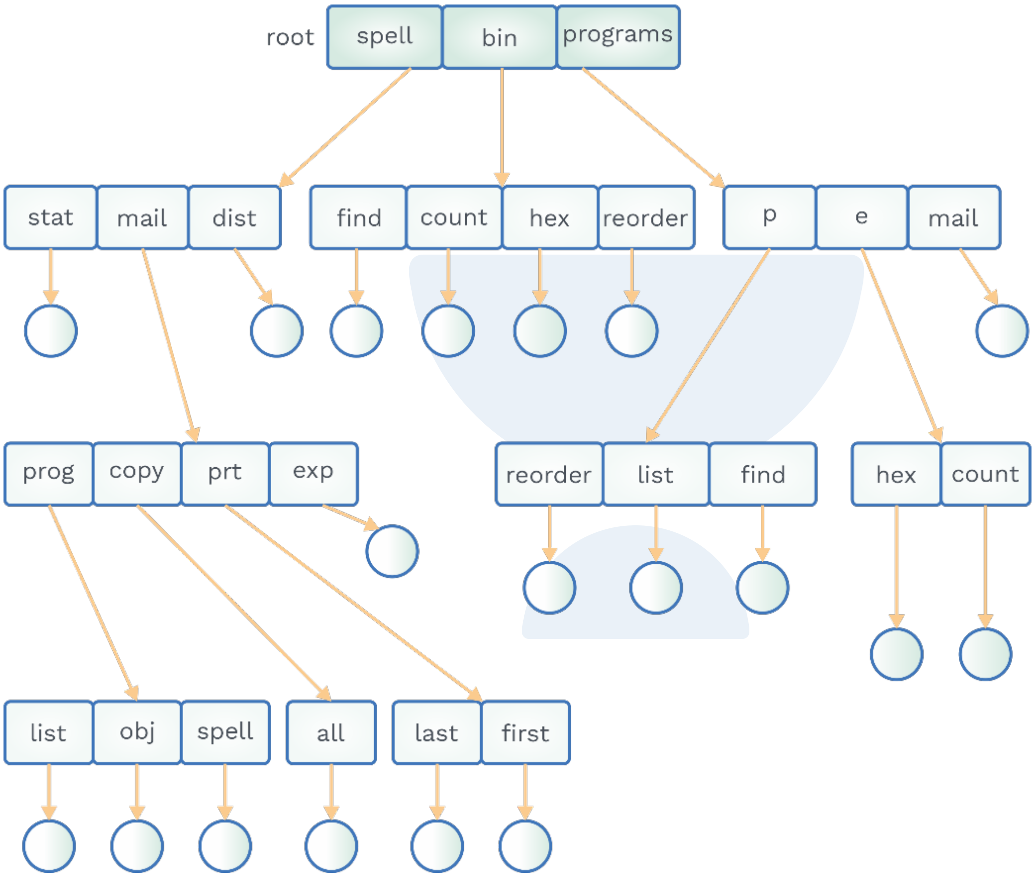


Fig. 6.6 Tree Level Directory

Acyclic-graph directory:

- An acyclic graph directory allows different users to share subdirectories and files.
- In two different directories, there may exist the same files or subdirectories. So, among various users or directories, common subdirectories could be shared.

Note:

Sharing subdirectories (files) is different from copying subdirectories (files) to another place; if a user alters shared subdirectories (files), the changes will get reflected everywhere.

Advantages:

1) There is no problem of inconsistency because files can be shared, and any changes made to one copy of the file are automatically reflected everywhere.
2) Searching a file is easy because it has a unique or individual path.
3) Since there is no cycle, simple graph algorithms can be used to traverse the graph.

Disadvantages:

1) Acyclic-graph directory structure is more complex than a simple tree structure.
2) Deleting files may create a problem because files are shared via links, and it may lead to a dangling pointer.

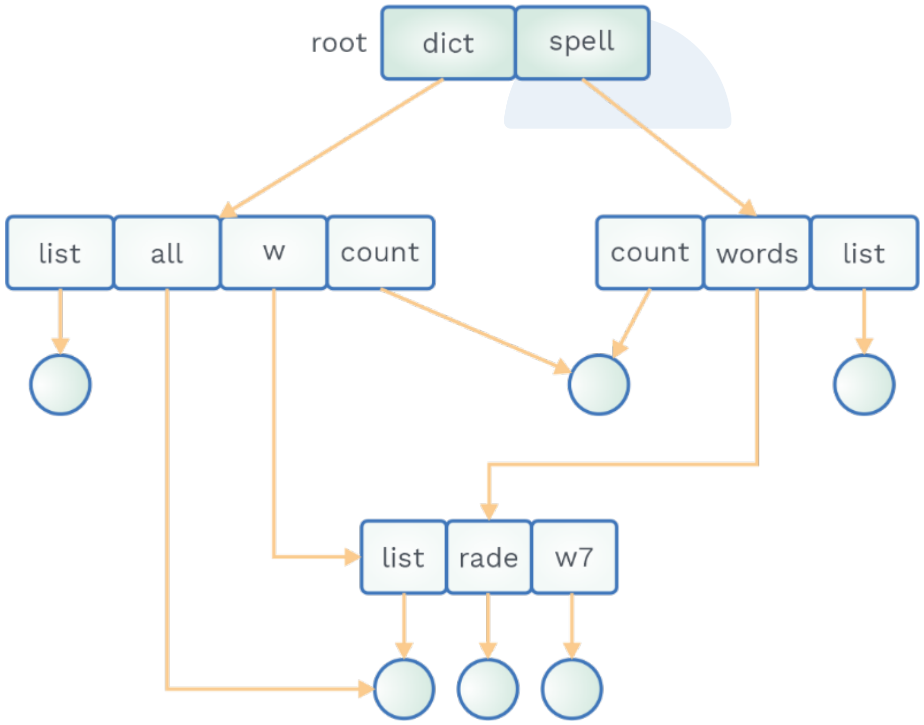


Fig. 6.7 Acyclic–Graph Directory Structure

General graph directory structure:

- Generally, in a graph directory structure, cycles can be present within a directory structure.
- If there is a cycle in the structure, then there are more than one path possible to reach to the file.

Advantages:

1) It allows cycle.
2) It is a flexible structure, since a file can be searched through many paths.

Disadvantages:

It is costly because we might need to design complex algorithms in order to avoid infinite loops while searching through the cycle.

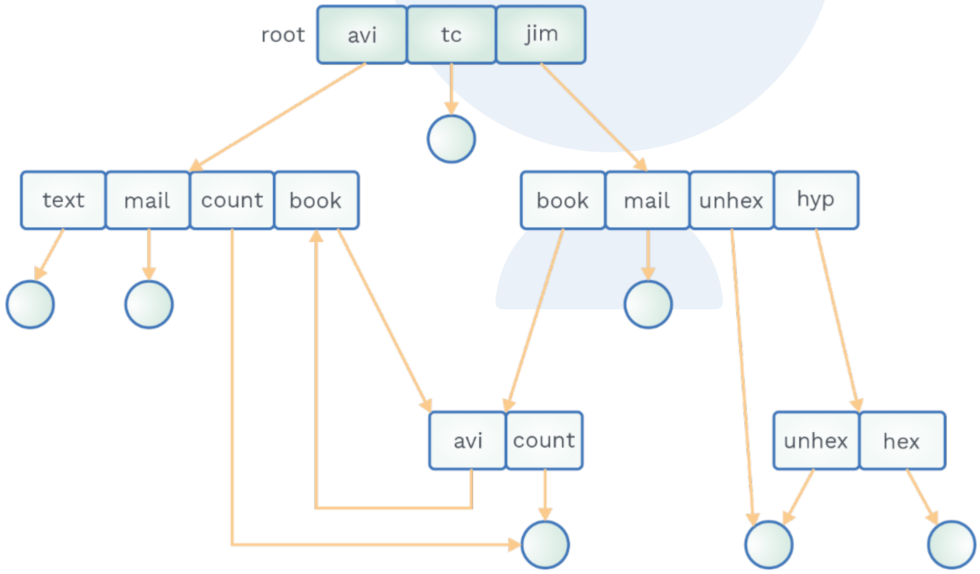


Fig. 6.8 General Graph Directory

SOLVED EXAMPLES

Q2 **Which of the following is/are FALSE?**

a) A hierarchical file system, must have a root directory irrespective of the operating system.

b) In a hierarchical file system a soft link is also known as a relative path.

c) In a hierarchical file system, for a file, multiple soft links but only one hard link can exist.

d) In a hierarchical file system, for a file, only one soft link but multiple hard links can exist.

Sol: **Options: b), d)**

a) (TRUE) A hierarchical file system must have a root directory irrespective of the operating system.

b) (FALSE) In a hierarchical file system, a soft link is a directory entry containing the pathname for another file. The relative path of a file is just a short path of the file in the current working directory.

c) (TRUE) In a hierarchical file system, for a file, multiple soft links but only one hard link can exist.

d) (FALSE) A hard link is a directory entry that specifies the location of the file on the storage device.

Access methods:

There are different methods that can be used to access the information within the file.

a) Sequential access:

Information in the file is accessed and processed orderly (sequentially), i.e., one record after other in this method.

Example: Compiler and editor access a file using sequential access.

Key points are:

1) Orderly, data is accessed one record after another record.
2) A read operation "read next", reads next portion of the file and advances the file pointer.
3) Most of the OS provide this method to access a file.

b) Direct Access:

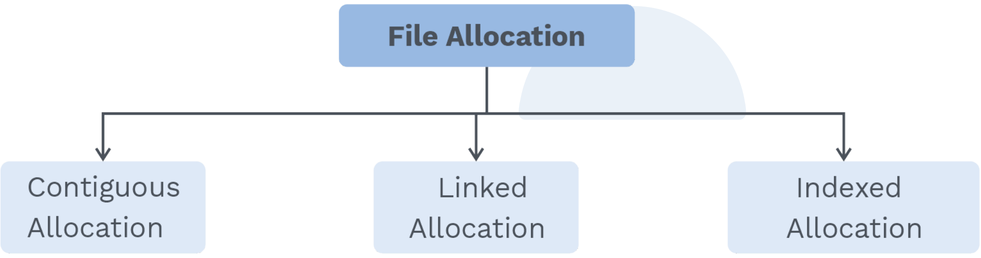
- It is also known as the relative access method.
- In this method, a file can be accessed randomly without any proper sequence(order).
- Random read/write is possible like read on block 10, then 56 and write on block 45 can be performed.

c) Index sequential method:

- Accessing of the file is done by using another file called index file, which is built on sequential method in this method.
- It generally involves index construction for the file, whereas the file contains a pointer to the various blocks of the file.
- For finding a record in the file, first search the index and later with the help of a pointer, directly access the file to find desired the record.

File allocation methods:

There are multiple ways to store a file in the disk blocks; three of them are discussed here:

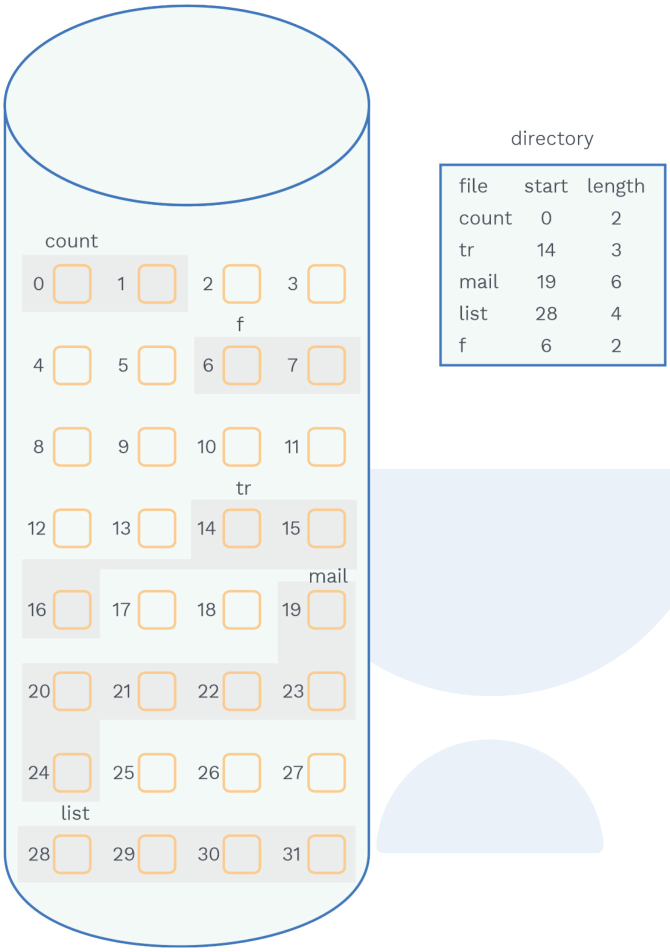


Motive of file allocation:

Utilization of disk space in an effective way and accessing file blocks in fast manner.

a) Contiguous allocation:

- In this, whenever a file is created, it occupies a contiguous set of blocks on the disk. Let us consider a file that requires ‘x’ blocks, if a file is assigned to block ‘p’ then the file will occupy blocks p, p + 1, p + 2, p + 3, ... p + x - 1.
- If starting disk block address and file Length (in terms of blocks) are given, then we can easily find the number of occupied blocks by the file .,
- The directory entry for a file with contiguous allocation contains the first block address and file size.



directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Fig. 6.9 Contiguous Allocation

Advantages:

1) Both sequential and random accesses can be done. For random access, the address of the Xth block of the file, which starts at block ‘p’ can easily be obtained as (p + X).
2) Due to sequential allocation, the number of seeks is minimal, which makes it extremely fast.

Disadvantages:

1) It suffers from external fragmentation because even if the disk block is free, it cannot be allocated, as it may not be a contiguous block.
2) Increasing the file size is difficult because the next contiguous block may or may not be empty.
3) Internal fragmentation may exist in the last disk block allocated to the file.

b) Linked list allocation:

- It is a non-contiguous allocation. In this type of allocation, each file acts as a linked list of disk blocks. Where blocks need not be in a contiguous manner.
- Here, a file could be allocated in disk blocks that are located anywhere on disk. In this allocation, directory contains starting and ending blocks pointers of the file.

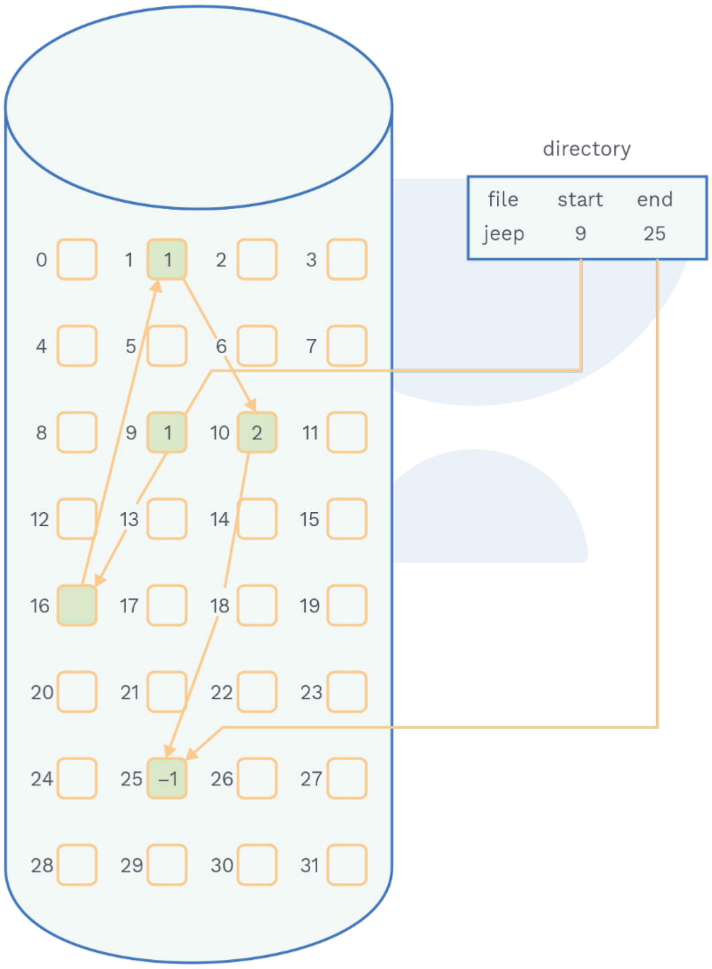


Fig. 6.10 Linked-List Allocation

Advantages:

1) This is flexible allocation because whenever a free disk block is present, a file block can be allocated in that block.
2) External fragmentation is not present. So, no space is lost due to disk fragmentation.

Disadvantages:

1) Need a large number of seeks to access the required disk block, since a file is scattered randomly on the disk.
2) A file can be accessed only in a sequential manner, as random access is not possible.
3) Maintaining the pointer for each disk block is considered as an overhead.

SOLVED EXAMPLES

Q3 **Consider a file which is stored in a disk and occupying disk blocks from 1 to 50. If currently R/W head is on 15th disk block, and a particular record is to be accessed, which is stored on disk block number 42. It takes 2ns to access a disk block.**

Let the time required to access the disk block containing the desired record using the contiguous file allocation method and using the linked file allocation method are x ns and y ns, then find the absolute difference between x and y.

Sol: **Range: 52 – 52**

a) **Contiguous file allocation:**
In this method, random access is possible.
So, from 15th disk block, we can directly access 42nd disk block.
Hence, only single disk block access is required.
Access time = 2ns

b) **Linked file allocation:**
In this method, random access is not possible.
So, from 15th block user needs to access all the blocks to reach 42nd block.
Total block access = 27
Access time = 27 * 2 = 54 ns
The absolute difference between x and y is 54 - 2 = 52 ns.

c) **Indexed allocation**

- In this, every file will have a special block which consists of the pointers to all blocks of that file. This special block is an index block.
- The ith entry of the index block of a file have the adisk address of the ith block of that file. Here each file will have index block.

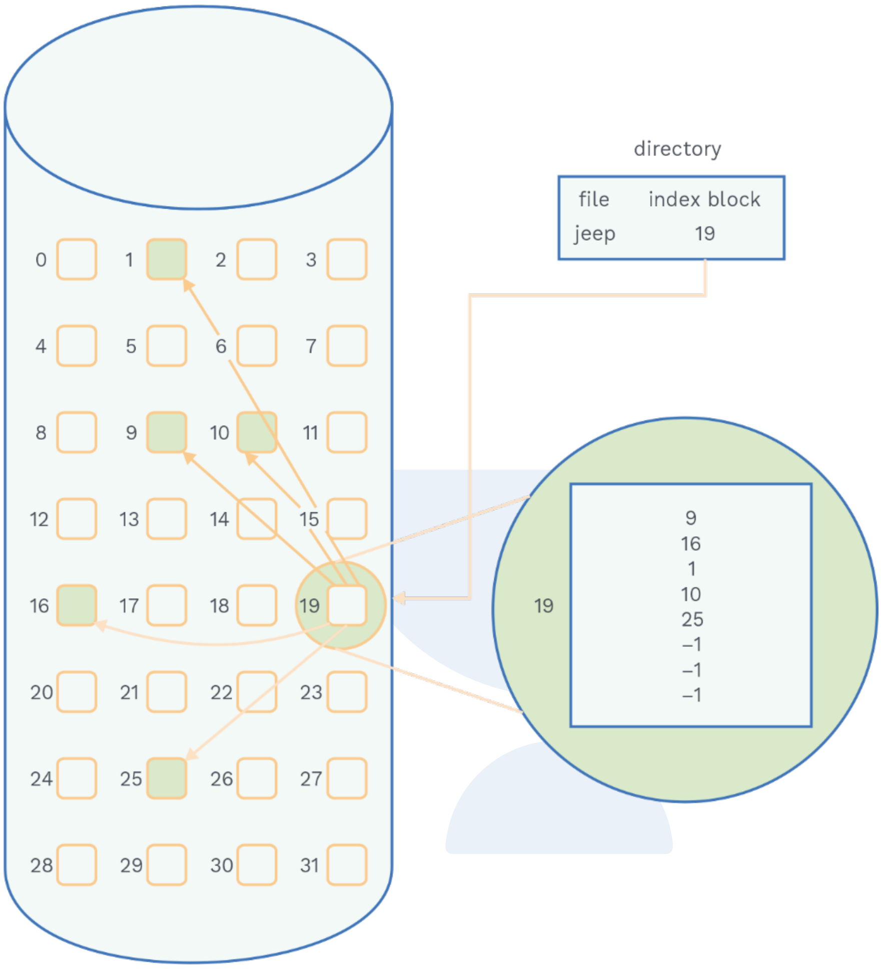


Fig. 6.11 Indexed Allocation of Disk Space

Advantages:

1) It provides random access to the blocks of files, which makes retrieval faster.
2) External Fragmentation is not present.

Disadvantages:

1) Pointer overhead in index allocation is more as compared to linked allocation.
2) A complete block holds the pointers only, which leads to inefficient utilization of space.

Sometimes, a single index block is not sufficient to hold all the pointers for files that are very large.

Thus, there are few mechanisms that is used for this.

- **Linked scheme:**
  As the name specifies, in this scheme,atleast two index blocks are grouped together to hold the pointers. Every index block will have a pointer or address to other index blocks.
- **Multilevel index:**
  There are multiple levels of an index. In case of two levels of an index, the first level index block has pointers to the second level index blocks and the second level block has pointers to the disk blocks of the files stored on the disk.
- **I-node:**
  1) A file attributes like a name, size, permissions, etc are stored in a special block called as I-node (information node).
  2) In an I-node, space which remains empty after storing the meta data of a file is used for storing DBA (disk block address) which has the actual content of a file.
  3) The first few of these pointers in I-node point to the direct blocks. Direct block is the block which has actual data.
  4) A single indirect block exists, whose pointers are pointing to the actual data blocks where the data of a file is stored. This indirect block is never holding the data. But it holds the addresses of the blocks (which holds the actual data).
  5) Similarly, the next few pointers are pointing to double indirect blocks which do not contain the file data but the address of the blocks that contain the address of the blocks containing the file data.

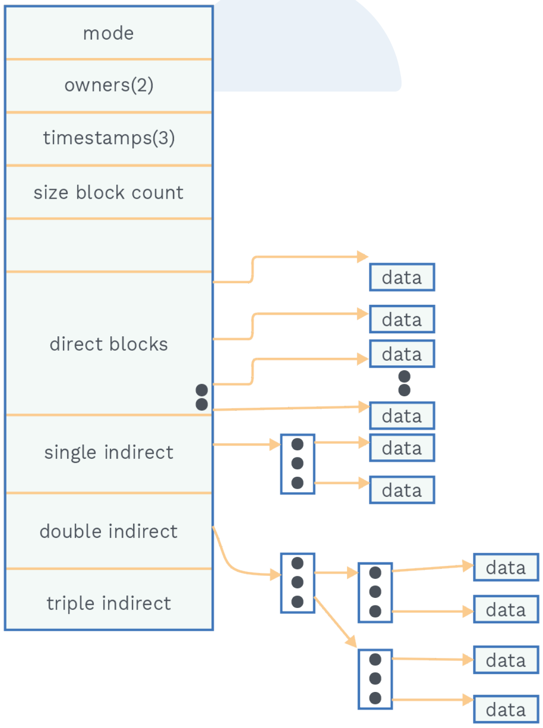


Fig. 6.12 UNIX I-Node

SOLVED EXAMPLES

Q4 **Consider a UNIX-I-NODE structure, which maintains eight direct disk block addresses, two single indirect, one double indirect, one triple indirect disk block addresses. The size of the disk block is 256 bytes, and disk block address requires 32 bit.**

The maximum possible file size and total size of the file system are:

a) 64 MB and 66,594 KB

b) 64 MB and 66,494 KB

c) 32 MB and 66,594 KB

d) 32 MB and 66,494 KB

Sol: **Option: a)**

Disk block size (DB size) = 256 bytes.

Disk block address size (DBA) = 32 bit = $\frac{32}{8}$ = 4 bytes.

a) Maximum file size will be for triple indirect disk block addresses:

$$1*\left(\frac{\text{DB size}}{\text{DBA}}\right)^3 * \text{DB size}$$

$$=\left(\frac{2^8}{4}\right)^3 * 2^8$$

$$= 2^{18} * 2^8 = 2^{26}\text{ B}$$

$$= 64\text{ MB}$$

b) Total size of the file system

$$=\left[8+2*\left(\frac{\text{DB size}}{\text{DBA}}\right)+1*\left(\frac{\text{DB size}}{\text{DBA}}\right)^2+1*\left(\frac{\text{DB size}}{\text{DBA}}\right)^3\right]*\text{DB size}$$

$$=\left[8+2*\frac{2^8}{2^2}+\left(\frac{2^8}{2^2}\right)^2+\left(\frac{2^8}{2^2}\right)^3\right]*2^8\text{B}$$

$$=\left[8+2^7+2^{12}+2^{18}\right]*2^8\text{ B}$$

$$= \left[2 + 2^5 + 2^{10} + 2^{16}\right] 2^{10} \text{ B}$$

$$= \left[2 + 32 + 1024 + 65536\right] \text{KB} \;= 66594 \text{ KB}$$

Q5 **Consider a Unix I–node, which maintains 32 direct disk block addresses, one single indirect, one double indirect and one triple indirect disk block address. The size of a disk block and disk block address is 2 KB and 32 bits. What is the maximum file size possible in the system? ________ (in GB) (rounded off up to two decimal places)**

Sol: **Range: 256.50 – 256.50**

Maximum file size possible –

$$\left[\underset{\uparrow}{32} + \left(\frac{\text{DB Size}}{\text{DBA}}\right) + \left(\frac{\text{DB Size}}{\text{DBA}}\right)^2 + \left(\frac{\text{DB Size}}{\text{DBA}}\right)^3\right] * \text{Disk Block Size}$$

Direct

DBAS'

Where DB : Disk Block Address = 32 bits = 4B.

$$= \left[32 + \frac{2\,\text{KB}}{4\,\text{B}} + \left(\frac{2\,\text{KB}}{4\,\text{B}}\right)^2 + \left(\frac{2\,\text{KB}}{4\,\text{B}}\right)^3\right] * 2\,\text{KB}$$

= $[2^5 + 2^9 + (2^9)^2 + (2^9)^3]$ * 2 KB
= $2^5 (1 + 2^4 + 2^{13} + 2^{22})$ * 2 KB
= 268,960,832 KB
= 256.50 GB (divided by 2^{20})

Q6 **Consider a disk with FAT system which has FAT entry size of 32 bits. Alice is using a system with 64 GB hard disk. Block size is known to be 32 KB. What is the minimum size (in MB) of file that Alice can store in the hard disk?**

Sol: **Range: 65,528 - 65,528**

Hard disk size = 64 GB = 2^{30} * 26 = 2^{36} Bytes
Number of blocks in disk = 2^{36} / 2^{15} = 2^{21} blocks
For each block, there will be a FAT entry → (We want to maximize FAT table size)
FAT size = 2^{21} * (32/8) = 2^{21} x 2^2 = 2^{23} Bytes = 8MB
Minimum size of file = (2^{36} / 2^{20}) – 8 = 2^{16} – 8 = 65,528 MB

Rack Your Brain

Consider the Unix I–node, which maintains 14 direct pointers, one single indirect pointer and one double indirect pointer. The disk block offset is 12,768 bits, and the disk block address is 64 bits long. The maximum file size possible with double indirect is _____ GB.

Free space management:

- As there is limited disk space, we need to reuse the deleted space for the new files.
- There is a free-space list that is managed by the system to keep the track of all the disk space that is free.
- To keep track of all the disk space that is free, a free-space list is managed by the system.

1) Bit vector:

The empty space list is implemented as a bit map. A block is represented using a bit

If bit=1, block is free

If bit=0, block is allocated

Suppose 1,2,4,6,7...blocks are free. So, the free space bit map would be look like.

0 1 1 0 1 0 1 1 ....

Advantages:

i) It is simple to understand.

ii) It's very efficient in finding the first free block.

2) Linked list:

Free space management on the disk can also be done using a linked list; in this method, a free disk block has a pointer to the next free block on the disk, and so on. A pointer to the first free block is stored at a designated location on the disk, and it is cached in memory.

3) Grouping:

In this approach, the addresses of the first 'n' free blocks on the disk are stored in the first free block. In these n blocks, except the last block n-1 blocks are actually free. The last block holds the address of the next 'n' free blocks on the disk.

Main advantage of this method → group of free disk blocks can be found easily.

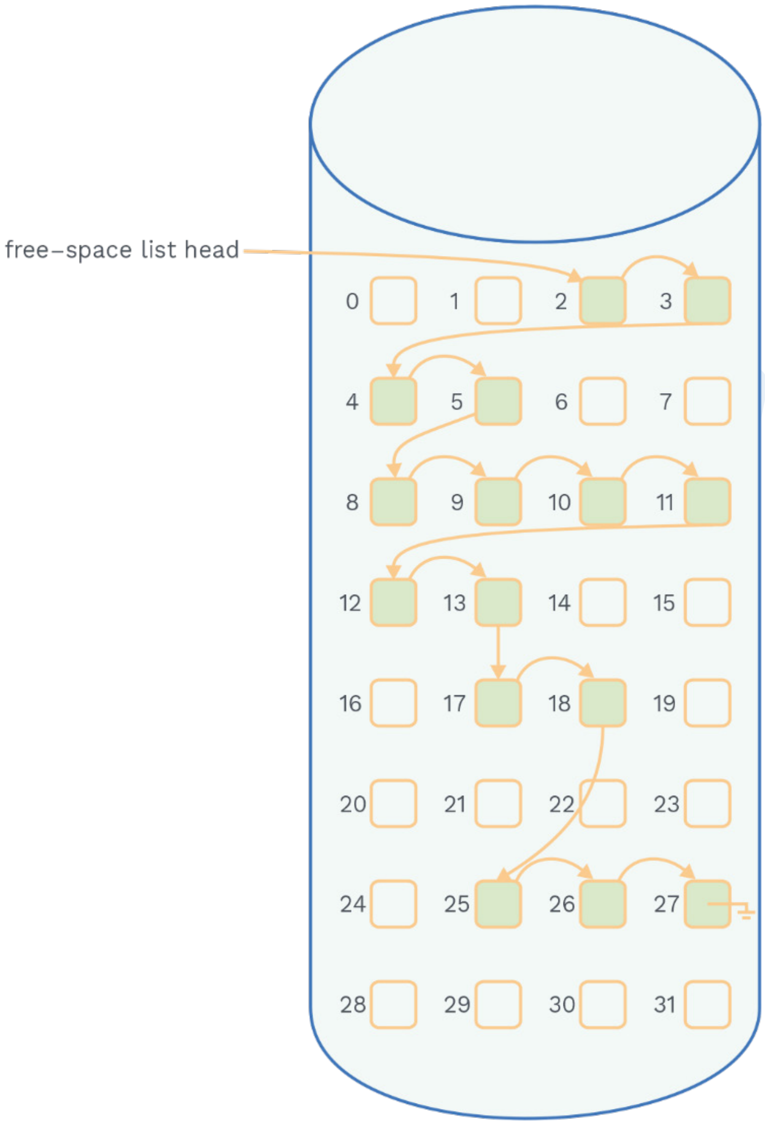


Fig. 6.13 Linked Free–Space List on Disk

Disk storage structures (DSS):

Basics of DSS

Magnetic disks

Bulk secondary storage is provided by magnetic disks for a modern computer systems. Each disk platter has a flat circular shape like CD.

1) The two–surface of the platter is covered with magnetic material. Each surface is divided into tracks.

- Each track is further divided into sectors. Outer tracks are bigger than inner tracks. They have the same number of sectors and storage capacity, so storage density is high in the inner tracks and low in the outer.
- Disk head (R–W head) rotates over the rotating hard disk. This head performs all read–write operations on the disk. Position of the head is the major concern as we need to put R–W head to the position where we want in read/write.

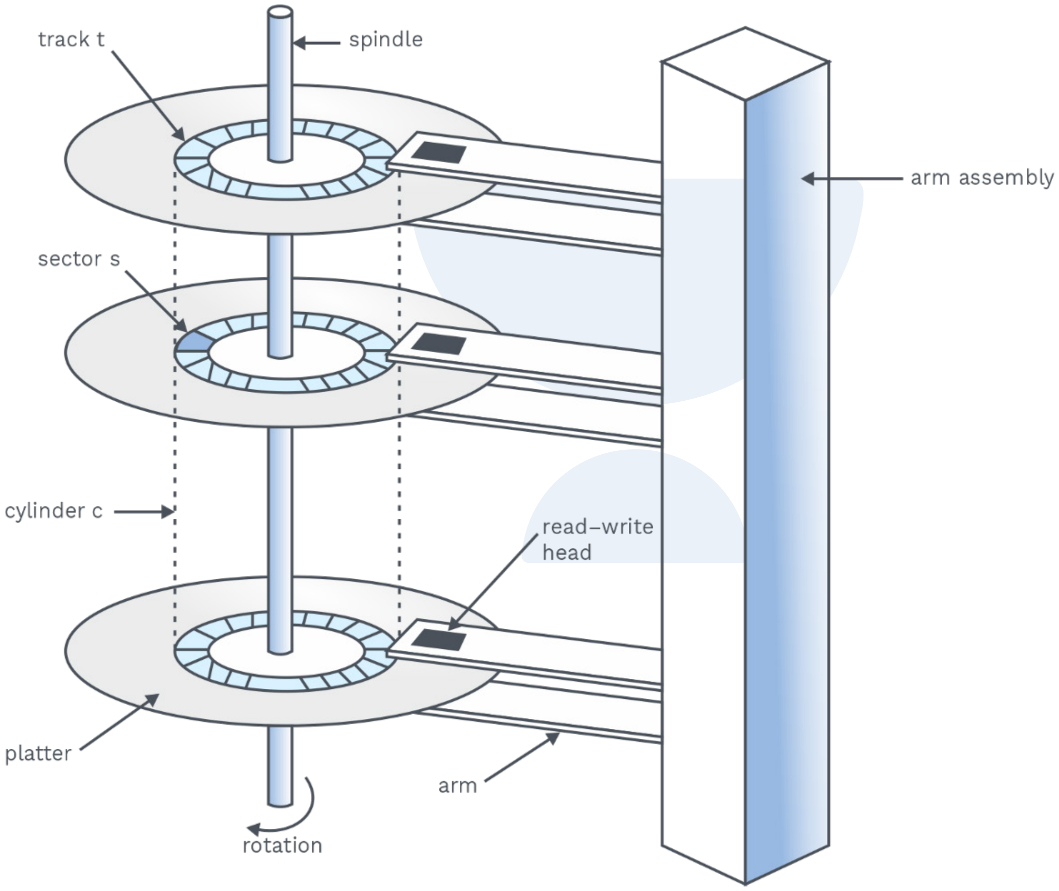


Fig. 6.14 Moving–Head Disk Mechanism

Some terms are:

1) **Seek time:** Time is taken by the read-write head to reach from one track to another one is known as seek time.
2) **Rotational latency:** The amount of time taken by the sector to rotate in such a way that it can be accessed by the read-write head.

3) **Data transfer time:** Time taken to transfer the required amount of data.

4) **Average access time:** = Average rotational Latency + Data transfer + Seek time.

Average
Rotational Latency = ½ * Time taken for
one full rotation

Rack Your Brain

Consider the following parameters
Number of surfaces = 32
Number of tracks/surface= 512
Number of sectors/track = 512
Number of bytes/sector = 1 KB
The number of bits required to specify a particular sector in the disc are?

Disk Scheduling:

Benefits of disk scheduling:

- Even though multiple processes requests for I/O operation, at a time, only a single I/O request is served by the disk controller. The remaining I/O requests need to wait until they get scheduled.
- The greater disk arm movement happens when two or more requests are far from each other.
- HDD is one of the slowest (secondary devices) parts of the computer system. It needs to be accessed in an efficient manner.

1) **FCFS scheduling (first come – first serve):**
It is considered as the easiest disk scheduling algorithm.
In this method, the disk requests are satisfied in the order they arrive in the disk queue.

Example: A disk queue with requests for I/O to blocks on a cylinder.
90, 178, 42, 133, 10, 140, 76, 79 and consider head starts at 50 in a disk system with 200 cylinders.

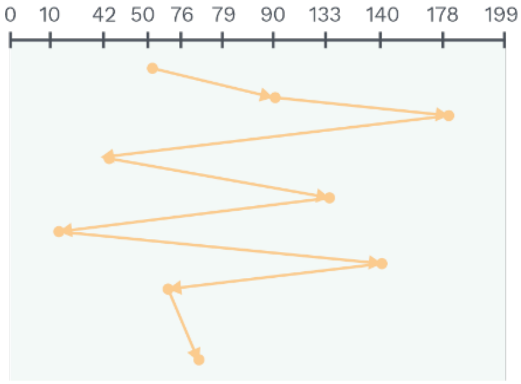


Fig. 6.15 FCFS Algorithm

Total seek time =
(90 - 50) + (178 - 90) + (178 - 42) + (133 - 42) + (133 – 10) + (140 – 10) + (140 – 76) + (79 – 76) = 675

2) SSTF scheduling (shortest seek time first):

It is reasonable to serve the requests near to present head position before serving far away disk requests. This scheduling selects the request with the least time from the current head position.

With this method, average response time decreases and throughput increases.

Example: A disk queue with requests for I/O to blocks on the cylinder. 82, 170, 43, 140, 24, 16, 190 and consider SSTF scheduling and current head start at 50 in a disk system with 200 cylinders.

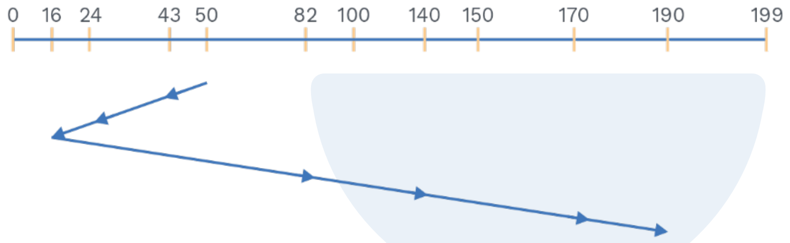


Fig. 6.16 SSTF Algorithm

Total seek time

= (50 – 43) + (43 – 24) + (24 – 16) + (82 – 16) + (140 – 82) + (170 – 140) + (190 – 170) = 208.

Major disadvantages are:

- An overhead of prior seeks time calculation.
- Starvation is possible for a request with a higher seek time compared to the next requests.
- The high variance of response time as SSTF favours only some requests.

3) SCAN scheduling:

The disk arm initiates at one end of the disk and moves towards, the other end of the disk, serving all the requests falling in between. After that, the disk arm changes its direction and serves all the requests in that direction in order. Head scans across the disk, back and forth. This scheduling is also called the elevator algorithm.

Example: A disk queue with requests for I/O to blocks on the cylinder. 90, 178, 42, 133, 10, 140, 76, 79 and consider scan algorithm and head starts at 50 in a disk system with 200 cylinders and it is scanning in left direction.

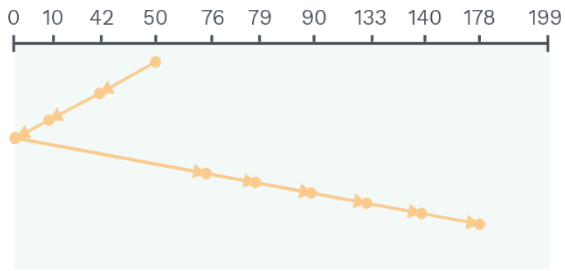


Fig. 6.17 SCAN Scheduling

Total seek time= (50 - 42) + (42 - 10) + (10 - 0) + (76 - 0) + (79 - 76) + (90 - 79) + (133 - 90) + (140 - 133) + (178 - 140) = 228

4) C–SCAN scheduling

C-SCAN is an extension of SCAN. It tries to ensure a uniform waiting time for all requests. In this method, the disk head moves from one end of the disk to the other like SCAN, fulfilling requests on the way. After the head reaches the other end of the disk, it returns to the beginning of the disk, by passing any requests on the way back.

The cylinders are treated as a circular list by the C–SCAN scheduling, which wraps around from the last cylinder to the first.

Example: A disk queue with request for I/O to the cylinder are

90, 178, 42, 133, 10, 140, 76, 79 .

Consider C–SCAN scheduling and head starts at 50 in a disk system with 200 cylinder, and is moving in the right direction.

Sol:

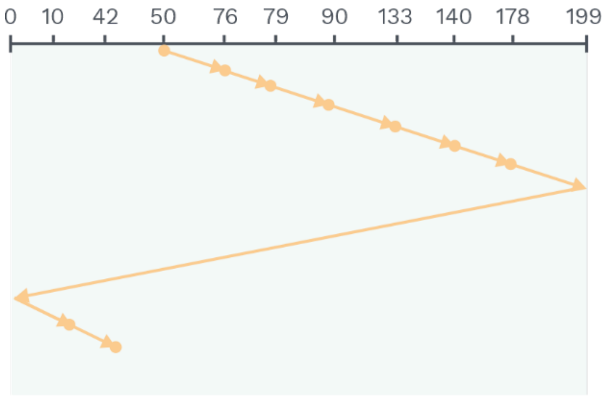


Fig. 6.18 C-SCAN Scheduling

Total seek time = (76 - 50) + (79 - 76) + (90 - 79) + (133 - 90) + (140 - 133) + (178 - 140) + (199 - 178) + (199 - 0) + (10 - 0) + (42 - 10) = 390

5) **LOOK scheduling:**

It is identical to the SCAN disk scheduling algorithm, except that instead of going to the end of the disk, the disk arm only goes to the last request to be handled in front of the head and then reverses its direction from there. As a result, the extra time caused by unnecessary traversal to the disk's end is avoided.

Example: A disk queue with request for I/O to the cylinder are

90, 178, 42, 133, 10, 140, 76, 79.

Consider LOOK scheduling and head starts at 50 in a disk system with 200 cylinders and is moving in the right direction.

Sol:

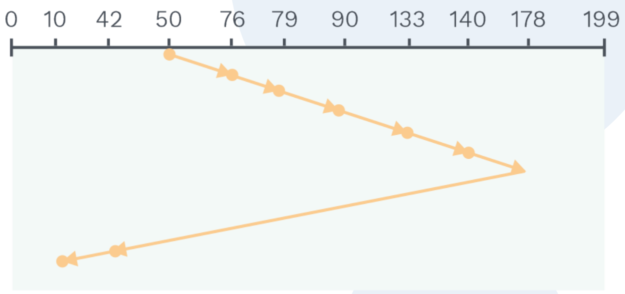


Fig. 6.19 LOOK Scheduling

Total time = (76 - 50) + (79 - 76) + (90 - 79) + (133 - 90) + (140 - 133) + (178 - 140) + (178 - 42) + (42 - 10) = 296

6) **C–LOOK scheduling:**

Circular-LOOK (C-LOOK) is similar to C–SCAN scheduling. In this approach, the disk arm moves in a direction serving all requests till the last request in the same direction. After that, the disk arm changes its direction and serves the last request nearest to other ends. Again the disk arm changes the direction and serves the remaining requests in the direction in order. In the given example, the disk arm goes from 178 – 10 and not to 199 and then 0 unlike the C-SCAN. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Example: A disk queue with request for I/O to the cylinder are

90, 178, 42, 133, 10, 140, 76, 79.

Consider LOOK scheduling and head starts at 50 in a disk system with 200 cylinders and is moving in the right direction.

Sol:

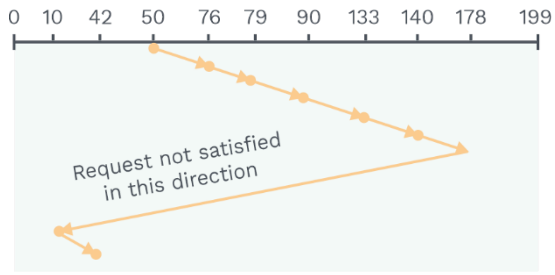


Fig. 6.20 C-LOOK Scheduling

Total seek time = (76 - 50) + (79 - 76) + (90 - 79) + (133 - 90) +
(140 - 133) + (178 - 140) + (178 - 10) + (42 - 10)
= 328

Rack Your Brain

Consider a disk system with 200 cylinders (0–199) and reading data from track 100. Track sequence is given 45, 8, 10, 11, 110, 50, 186, 176, 192. Suppose the FCFS scheduling is used and it takes 1 ms to move from one cylinder to adjacent one. Then total seek time is _______ ms.

SOLVED EXAMPLES

Q7 **Consider a disk drive with 3000 cylinders, numbered in the range [0, 2999]. The disk arm is serving the cylinder 125 now, and it is moving upwards. The pending requests in order are given as: 85, 1390, 900, 1765, 729, 1512, 1025, 1699, 128.**

The total distance (in cylinders) that the disk arm moves from its current position to serve all the pending requests using C–SCAN is ______

Sol: **Range: 5958 – 5958**

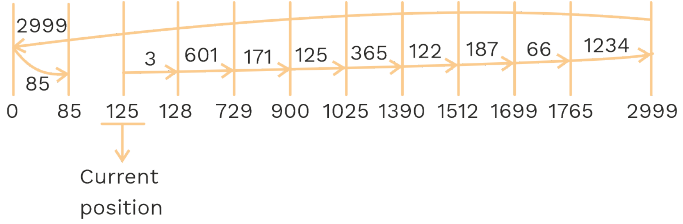


Fig. 6.21 C-SCAN Scheduling

Total distance = 3 + 601 + 171 + 125 + 365 + 122 + 187 + 66 + 1234 + 2999 + 85 = 5958

Q8 **Consider the tracks request sequence of a disk with 200 tracks is 72, 160, 33, 130, 14, 6, 180. Initially, R/W head is at track 50. R/W head moves in an upward direction. How many additional head movements will be traversed by the R/W head when the C-SCAN method is used compared to Shortest Seek Time First (SSTF) method?**

a) 200

b) 163

c) 180

d) 120

Sol: **Option: b)**

SSTF

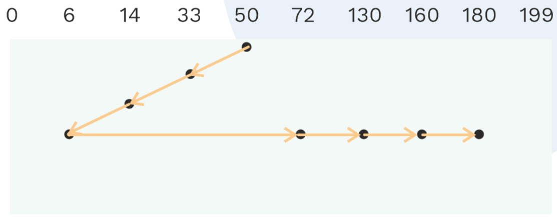


Fig. 6.22 SSTF Scheduling

No. of head movements = (50 - 6)+(180 - 6) = 44 + 174 = 218

C–SCAN

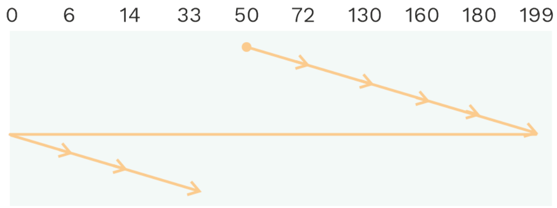


Fig. 6.23 C-SCAN Scheduling

Number of head movements = (199 - 50) + (199 - 0) + (33 - 0) = 149 + 199 + 33 = 381

Additional head movement are: = 381 – 218 = 163

IMPORTANT FORMULAE

Secondary memory:

No. of blocks in secondary memory = (size of the secondary memory / size of a block)

Contiguous allocation:

The X^{th} block of the file, which starts at block ‘p’ can be obtained at $(p + X)^{th}$ block.

File allocation table (FAT):

No. of entries in FAT = No. of blocks in secondary memory
Size of one entry = Bits required to identify a block
FAT Size = No. of entries in FAT * FAT Entry Size

UNIX I-node:

UNIX I-Node file system with n_0 direct disk block addresses, n_1 single indirect disk block addresses, n_2 doubly indirect DBAs and n_3 triple indirect DBAs. Then,
then maximum file size for triple indirect disk block addresses is

$$= \frac{\text{Disk Block Size}}{\text{Disk Block Address}} * \text{Disk Block Size}$$

Size of the whole file system is give as:

$$\left\{ n_0 + n_1 \times \left\{ \frac{\text{Disk Block Size}}{\text{Disk Block Address}} \right\} + n_2 \times \left\{ \frac{\text{Disk Block Size}}{\text{Disk Block Address}} \right\}^2 + n_3 \left\{ \frac{\text{Disk Block Size}}{\text{Disk Block Address}} \right\}^3 \right\} * \text{Disk Block Size}$$

Chapter Summary

- Basics of file system – The mechanism for online storage and access to both data and programs.
- File attributes – File has various identifiers to uniquely identify it in file system, such as name, size, type, etc.
- File operations –
  1) Creating a file
  2) Writing a file
  3) Reading a file
  4) Repositioning within a file
  5) Deleting a file
  6) Truncating a file
- File types – It represents the type of file and the operating system can recognize it and take appropriate action accordingly.
- Directory – A directory contains various files and folders.
- Types of directory –
  1) Single level
  2) Two level
  3) Tree structured
  4) Acyclic graph
- File access methods –
  Sequential access
  Direct access
  Index sequential
- File allocation methods –
  Contiguous allocation
  Linked allocation
  Indexed allocation
  I-node
  File allocation table
- Free space management –
  Bit vector
  Linked list
  Grouping
- Disk scheduling –
  FCFS
  SSTF
  SCAN
  C–SCAN
  LOOK
  C–LOOK