Unit-5 ; File Management

- Ankit Pangeni

DATE

* (File Overview)

A file is a named collection of related information that is recorded in the secondary storage such as magnetic disks, magnetic tapes and optical disks.
In general, a file is a sequence of bits, bytes line or records whose naming is defined by the files creator and user.

Most Common file systems in Windows are: NTFS, FAT, EXT, etc. These file systems can differ between OS, such as Windows, MACOS & Linux based OS. A file system stores & organizes data & can be thought as a type of index for all the data contained in storage device. File systems specify conventions for naming files, including the maximum number of characters in a name, which characters can be used and, in some systems, how long the file name suffix can be. In many file systems, file names are not case sensitive.

* (File Naming)

When a process creates a file, it gives the file name; while process terminates, the file continue to exist and can be accessed by other processes. A file is named, for the convenience of its human users, and is referred to by its name. Many OS support two-part file names; separated by period;

PAGE

DATE

The part following the period is called the file extension & usually indicates something about the file. (eg: xyz.c is a C program source file).

The name of each file must be unique within the directory where it is stored. This ensures that the file also has a unique path name in the file system. File naming guidelines are:

- A file name can be up to 255 characters long & can contain letters, numbers, & underscores
- File names should be as descriptive & meaning as possible
- Directories follow the same naming conventions as files
- Avoid using characters like / \ " * ; - ? ( ) ~ ! # @. These have special meaning in OS

Some file extensions with their meanings

File extension	File Meaning.
xyz.bak	indicates backup file.
xyz.gif	indicates graphic image format file
xyz.jpg	indicates jpg format image file

(File Structure)

A File Structure should be according to a required format that the OS can understand

- A file has a certain defined structure according to its type
- A text file is a sequence of characters organized into lines
- A source file is a sequence of procedures & functions

classmate

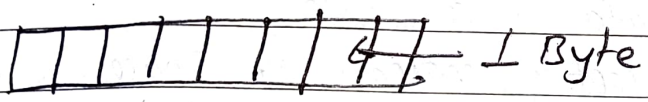
PAGE

DATE

When an OS defines different file structures, it also contains the code to support these file structure. Common file structures are:-

1) Unstructured Sequence of bytes (Byte sequence)

It is just an unstructured sequence of bytes. The OS doesn't know or care what is in the file. All it sees are bytes. Any meaning must be imposed by user-level-programs. Both UNIX & Windows 98 use this.



2) Record Sequence/structures

Here, a file is a sequence of fixed length records, each with some internal structure. The read operation returns one record and the write operation overwrites one record.

3) Tree structure

Here, a file consists of a tree of records, not necessary all the same length, each containing a key field in a fixed position in the record. The tree is sorted on the key field, to allow rapid searching for a particular key.

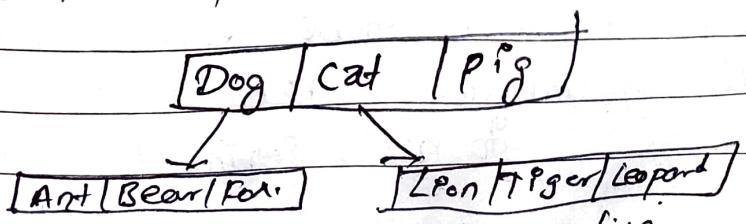


fig: Tree str of a file.

classmate

PAGE

DATE

* (Basic File operations)

There are many file operation that can be performed by the computer system some of them are listed below:

- Create: find space for a file & make an entry in the directory
- Open: find file & determine if it has already been opened
- Close: close the file
- Read: Read data from file.
- Write: Write data to a file.
- Delete: Search directory, release file space & erase directory entry
- Reposition: reposition the file position pointer. (seek)
- Truncate: delete contents of a file, but keep file properties
- Lock: to lock a file to ensure only one writer can modify a file.

* (File Types)

Many OS supports several types of files

- Regular Files: Contains user information, are generally ASCII or binary
- Directories: System files for maintaining the structure of the file system.
- Block Special files: used to model disks
- ASCII files: Consists of line of text. Can be displayed & printed as it can be edited with ordinary text editor
- Character Special files: related to I/O of computer to model serial I/O device like terminals

classmate

PAGE

DATE

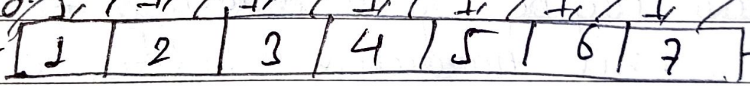
- Binary files: Consists of sequence of byte only They have some internal structure known to programs that use them. (e.g. executable files).

* (File access)

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 as-

- Sequential access

This is the simplest access method. Information in the file is accessed sequentially one by one in order, starting with the beginning, but could not skip around and read them out of order. It is convenient when storage medium is magnetic tape. It is used in many early systems. Ex. If we want to read a document from beginning to end, we typically start at the beginning and read page by page until we reach the end.



Advantages:

- Simple & easy to understand
- Sequential files are easy to organize & maintain
- Loading or reading a record requires only record key
- Inexpensive; efficient & economical.

Disadvantages

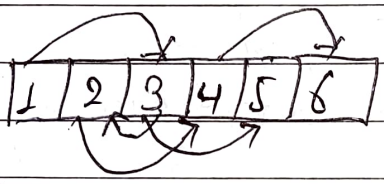
- Can't support modern technologies that require fast access
- Speed is slow. ie. time consuming.
- requirement that all records be of same size is sometimes difficult to enforce.

PAGE

DATE

o Direct access.

Files whose information can be readd in any order is called direct access. It is based on disk model of file, since disks allow random access to any block. It is mostly used for immediate access to large amount of informatio. If we want to read any information from a particular block, we can read that block directly. Ex: If we only want to read page no 3, it makes sense to seek (reposition) to page no 3 & read the page



Advantages:

→ Quick retrival of records
→ The records can be of different sizes

Disadvantages:

→ Data may be erased accidently if special precautions aren't taken
→ Expensive hardware & software resources are required.
→ Relatively complex when programming

* (File Attributes)

File attributes are the settings associated with the computer files that grant or deny certain rights to how a user or os can access that file. Some of the attributes of a file are:

classmate

PAGE

DATE

- Name: Only information in human-readable form
- Identifier: file is identified by a unique tag in file system
- Type: needed for systems that support diff. types of files
- Location: pointer to file location on device.
- Size: Current size of the file.
- Protection: Controls & assigns power of reading, writing, executing.
- Read-only: allows a file to be read only, but nothing can be written to the file or changed
- System: System file.
- Archive: tells windows Backup to backup the file.
- ~~System~~ Hidden: file won't be shown when doing a regular dir from DOS.

* (Directory Structure)

A directory is a node containing information about files. It is also called a folder. The directory may store some or the entire file attributes ~~To keep~~ Directory is defined as the file list on the disk. A harddisk is divided into number of partition of different size. The partition are called volume or hard disk. Each partition have at least one directory.

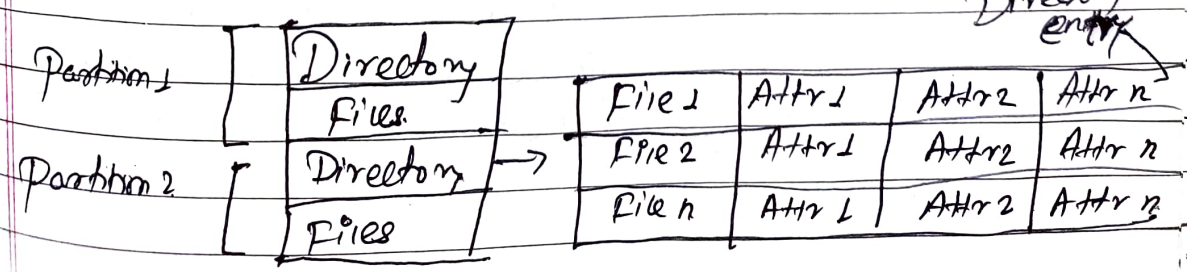


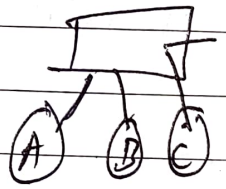
fig: Directory Structure

DATE

Directories can have different structures

o Single-Level-Directory:

It is the simplest form of directory system having one directory containing all the files. It is also called root directory. The entire system will contain only one directory which is supposed to mention all the files present in the file system. The directory contains one entry per each file present on the file system. An example is:-



Advantages:-

i) Easy to support and understand
ii) Implementation is very simple.
iii) If sizes of files are very small then the searching becomes faster.
iv) File creation, searching, deletion is very simple since we have only one directory.

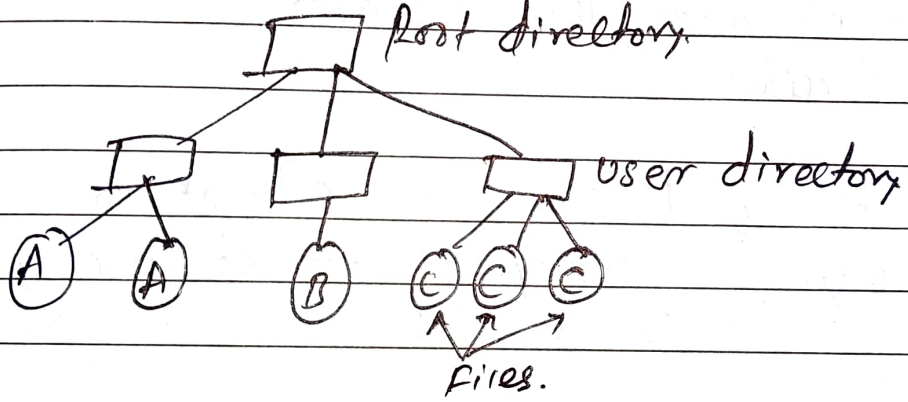
Disadvantages

i) Cannot have two files with the same name
ii) Protection can't be implemented for multiple users
iii) There are no ways to group same kind of files.
iv) Difficult to manage large amount of files and to manage different users.

DATE

Two-Level-Directory.

To avoid conflicts caused by different users choosing the same name for their own files, in the two level directory systems, we can create a separate directory for each user. There is one master directory which contains separate directories dedicated to each user. For each user, there is a different directory present at the second level, containing group of user's file. The system doesn't let a user enter in the other user's directory without permission.



Characteristics:

- Each file has a path name as /user-name/directory-name
- Different users can have the same file name
- Searching becomes efficient as only one user's list needs to be traversed
- Same kind of files can't be grouped into a single directory for a particular user.

DATE

o Hierarchical - Directory

It is also known as tree structure. Any directory can have sub directory or files. The similar types of files can be grouped in one directory. so, It overcomes the drawbacks of two level directory system

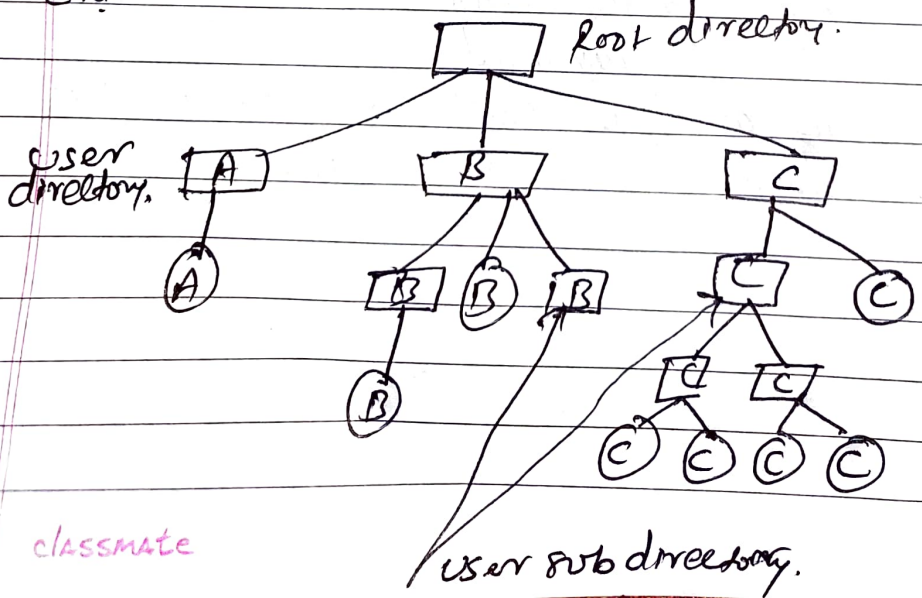
Each user has its own directory & it can't enter in the other user's directory. However, the user has the permission to read the root's data but he can't write or modify this. Only admin has complete access to root directory

Searching of files is more efficient in hierarchical directory. The concept of current working directory is used. A file can be accessed by two types of path:-

absolute path:- path of the file with respect to the root directory

relative path:- path with respect to the current working directory of the system

Eg:-



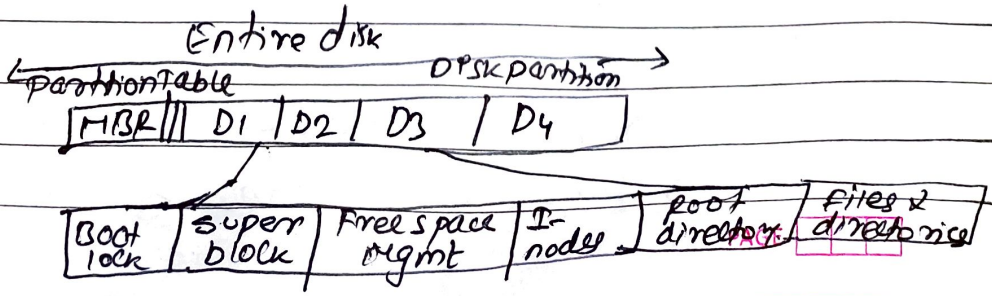
DATE

(File-System Layout).

A file system layout is a set of directories, folders and files. It maintains information & identifies where a file data is located on the disk. In addition to files and directories, file system contain a boot block, a superblock, bitmaps, and one or more allocation groups.

The file system layout consists of following info.

- MBR : It stands for Master Boot Record. This is used to boot the computer.
- Partition Table :- It is present at the end of MBR. This table gives the starting & ending address of each partition of disks.
- Boot Lock :- When the computer is booted, the BIOS read in & execute the MBR. The 1st thing the MBR program does is locate the active partition read in 1st block, which is called boot lock and execute it
- Super block :- It contains all the key parameters about the file system & is read into memory when the computer is booted or the file system is 1st touched



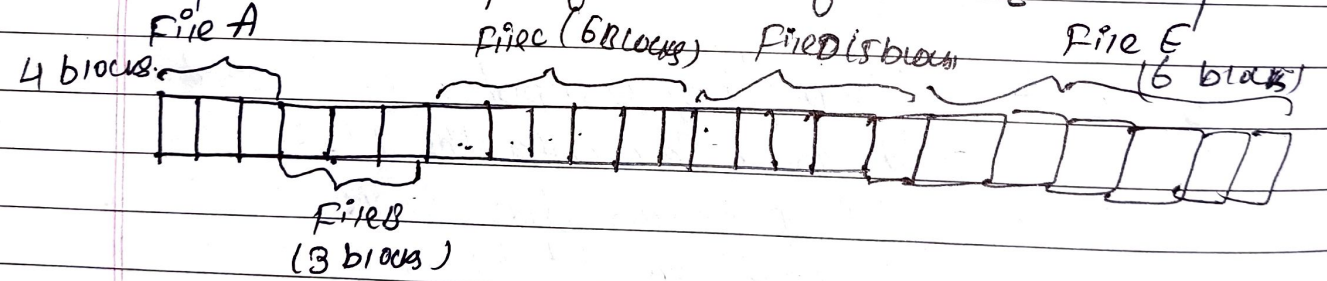
DATE

* (Implementing Files)

Implementing files mean defining how the files are stored in the disk blocks. There are three main disk space or allocation methods

1) Contiguous Allocation

In this allocation a contiguous set of blocks is associated to a file at a time of file creation. The file allocation table consists of single entry of file which shows starting block and length of file. This method is suitable for sequential file. This is a pre-allocation strategy, using variable size portions Multiple blocks can be read in at a time to improve I/O performance for sequential processing



Advantages

- Both the sequential & direct access is supported by this
- This is extremely fast since the number of seeks are minimal because of contiguous allocation of file blocks

Disadvantage

- This method suffers from both internal & external fragmentation
- This is inefficient in terms of memory utilization

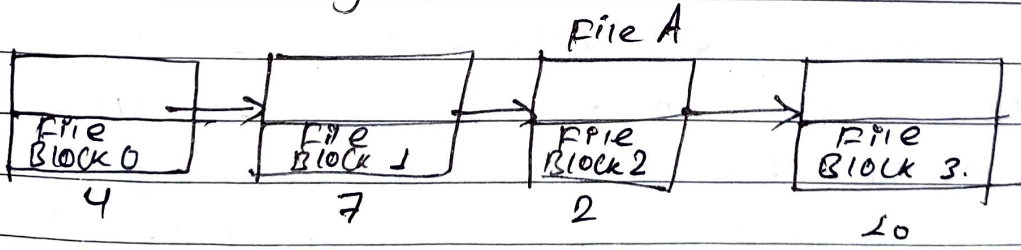
DATE

-) With pre-allocation, it is necessary to declare the size of the file at the time of creation.
→ Increasing file size is difficult.

2) Linked List Allocation

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

Each file is a linked list of disk blocks which need not be contiguous. The disk blocks can be scattered anywhere on the disk. There is no external fragmentation because only one block is needed at a time but there can be internal fragmentation but it exists only in the last disk block of file.



Advantages
→ Flexible in terms of file size. File size can be increased easily since the system doesn't have to look for contiguous chunk of memory
→ Does not suffer from external fragmentation

→ Internal fragmentation exists in last disk block of file only

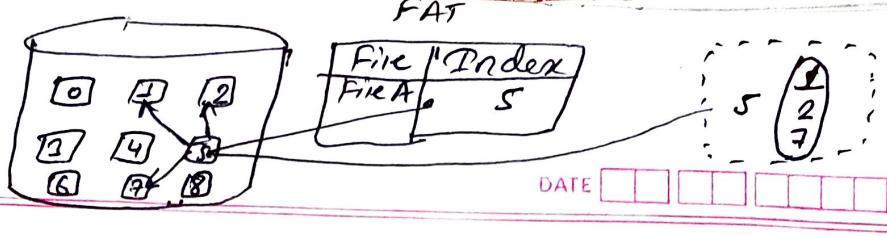
Disadvantages

→ Linked allocation is slower because a large no. of seeks are needed to access every block individually since file blocks are distributed randomly on the disk

→ It doesn't support random or direct access.
→ Pointers required in the linked allocation incur some extra overhead.

3) Index Allocation :-

It address many of the problems of contiguous and chained allocation. Here, the file allocation table contains a separate one-level index for each file. The index has one entry for each block allocated to the file. Allocation may be on the basis of fixed-size blocks or variable-sized blocks. Allocation by blocks eliminates external fragmentation, whereas allocation by variable-size blocks improve locality. It supports both sequential & direct access to the file & thus is the most popular form of file allocation.
Here, a special block known as Index block contains the pointers to all the blocks occupied by a file. Each file has its own index block. The i^{th} entry in the index block contains the disk address of i^{th} block.



DATE

Advantages:
- Flexible in terms of file size.
- No external fragmentation.
- Provides fast access to the file blocks. as it supports direct access to the blocks occupied by the file.

Disadvantage:
-> Pointer overhead for indexed allocation is greater than linked allocation
-> Searching is difficult.

4) Linked List Allocation Using Table in Memory/ File Allocation table

The main disadvantage of linked list allocation is that Random access to a particular block is not provided. In order to access a block, we need to access all its previous blocks. File allocation table overcomes this drawback of linked list allocation. Here, a FAT is maintained, which gathers all the disk block links. The table has one entry for each disk block and is indexed by block number.

It simply accesses the fat, read the desired block entry from there and access that block. This is the way by which the random access is accomplished by using FAT. It is used by MS-Dos & pre-NT windows version.

PAGE

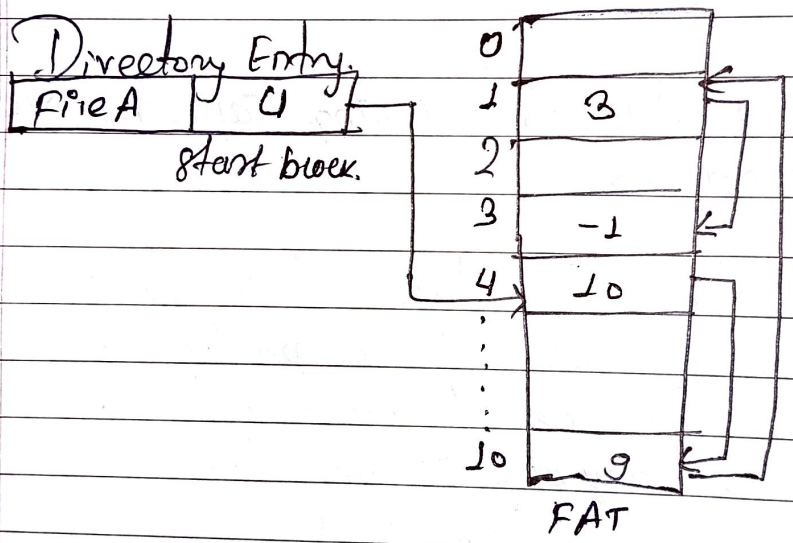
DATE

Advantages

→ Uses the whole disk block for data
→ A bad disk block doesn't cause all successive blocks to be lost
→ Only FAT needs to be traversed in each file operation.

Disadvantages

→ Each disk block needs a FAT entry.
→ FAT size may be very big depending upon the number of FAT entries.
→ Reducing no of FAT entries increases Internal Fragmentation



Numerical

Q. Consider a disk of 100 KB. If block size is 2KB, calculate the size of FAT assuming that each entry in FAT takes 4 bytes

→ Size of disk = 100 GB = 100×2^{20} KB = 104857600 KB
Size of block = 2 KB
Thus, no. of blocks = 104857600 / 2 = 52428800
→ No. of entries in FAT = 52428800, size of an entry = 4 byte.
Size of FAT = 52428800 × 4 byte = 200 MB

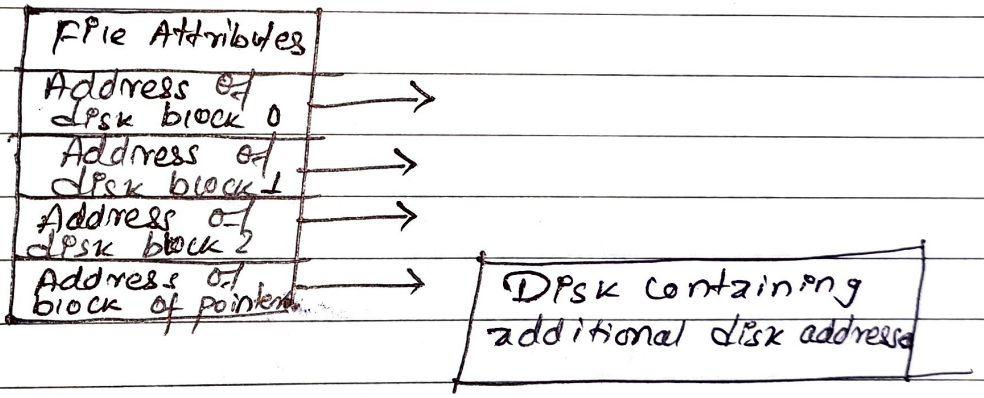
PAGE

DATE

* (I-nodes)

I-node (Index node) is a data structure which is used to identify which block belongs to which file. It contains the attributes & disk addresses of the file's blocks. Unlike the in-memory table, the i-node need to be in memory only when the corresponding file is open.

It lists the attributes and disk address of the block. All the attributes for the file are stored in an I-node entry, which is loaded onto memory when the file is opened. The i-node also contains a no. of direct pointers to disc blocks.



In UNIX based OS, each file is indexed by an I-node. I-node is the special disk block which is created with the creation of the file system. The no. of files or directories in a file system depends on the no. of I-nodes in the system. I-node contains attributes of file, single indirect pointer which points to an index block & a double indirect pointer which points to a disk block & a triple index pointer which also points to a disk block that is a collection of pointers.

PAGE

classmate

DATE

Advantages

→ ~~Entire blo~~ Supports direct access to blocks occu
by the file so it provides fast access.

→ Total memory occupied by the i-node is only kn

Disadvantage

→ It only have room for a fixed number of disk
addresses. Problem occurs when a file grows beyond
the limit.

DATE

* (Directory)

Directory is a place/location where a set of file(s) will be stored. It is a folder which contains details about files, file size and time when they were created and last modified. To keep track of files, file systems normally have directories or folders which are themselves files

* (Directory operations)

→ Create: A directory is created. It is empty except for dot & dot dot., which are put there automatically by system.

→ Delete: A directory is deleted. Only an empty one can be deleted

→ Opendir: Directory can be read

→ Closedir: When a directory is read, it should be closed to free up empty space.

→ Rename: Used for renaming the directories.

→ Link: Linking is a technique that allows a file to appear in more than one directory

→ Unlink: A directory entry is removed. If the file being is present in one directory, it is removed from the file system

DATE

* (Path Names)

When the file system is organized as a directory tree, some way is needed for specifying file names. Two different methods are commonly used

1. Absowte Path Name

It is the path name starting from root directory to the file. E.g in UNIX: /usr/user1/bin/lab2. path separated by / in UNIX & \ in Windows

2. Relative path Name

It is the path name based on the concept of working directory (also called working directory) A user can designate one directory as the current working directory, in which case all path names not beginning at the root directory are taken relative to the working directory
Ex: bin/lab2.

* (Directory Implementation)

The selection of directory-allocation & management algorithms significantly affects the efficiency performance, and reliability of the file system We implement the directory by two methods:-

DATE

1) 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.

To create a new file, we add a new entry at the end of the directory. To delete a file, we search the directory for the named file, then release the space allocated to it. The real disadvantage of a linear list of directory entries is that finding a file requires a linear search. A sorted list allows a binary search and decreases the average search time.

Note: When a new file is created, then the entire list is checked whether the file name is existing already or not. And, the list needs to be traversed in case of every operation (creation, deletion, updating) on files, so systems become inefficient.

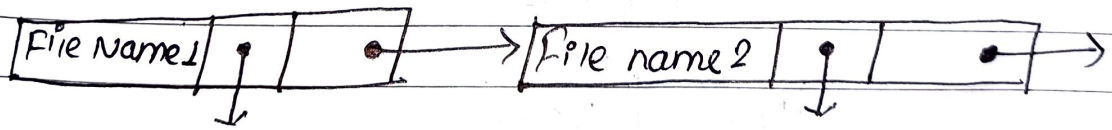
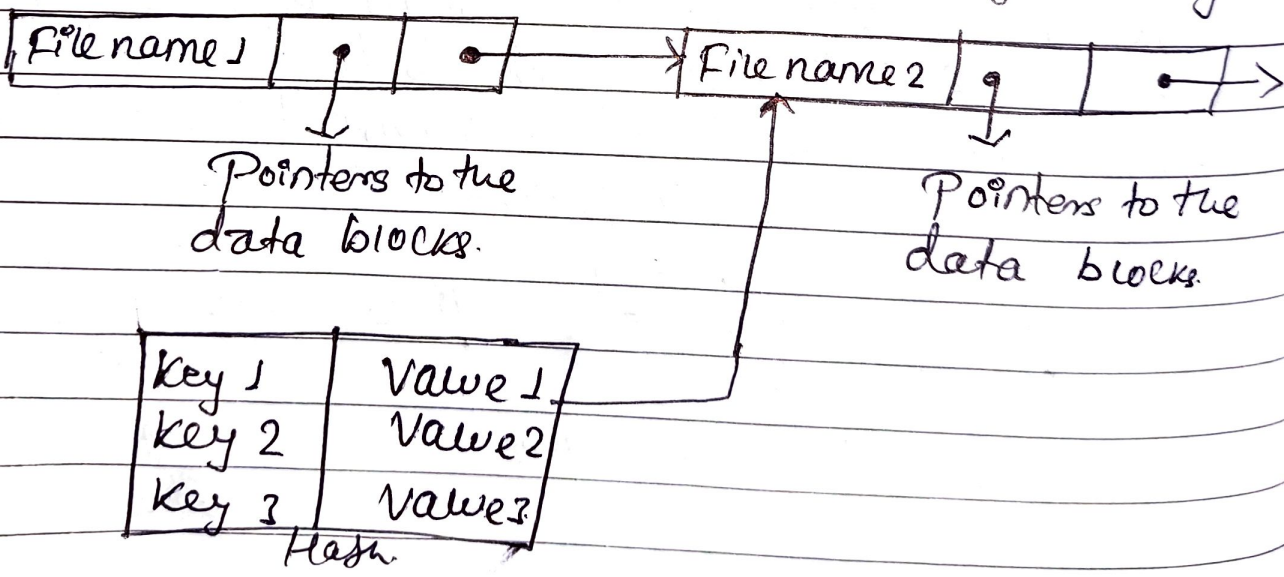


Fig: Linear List.

2) Hash Table.

Another data structure that has been used for a file directory is a hash table. It consist linear list with hash table. The hash table takes a value computed from the file name & returns a pointer to the file name in the linear list. So, it decreases the directory search time. Insertion and deletion is also fairly straight forward, although Sometimes you have to take care of collisions that occur when two file names hash to same location.

The key value that is stored in the table is generated by applying the hash function. The key value corresponds file stored in the directory. Now, Searching becomes efficient due to the fact that now, entire list will not be searched on every operation. Only hash table entries are checked using the key.



Advantages: greatly decreases the file search time
Problems: It has ~~con~~fixed size & dependance of the hash function on that size.

* (Shared files)

In many situations, we need to share files between users. Ex, when several users are working together on a project, they often need to share files. It is often convenient for a shared file to appear simultaneously in different directories belonging to different users. Thus, it is better to represent file system by using directed acyclic graph (DAG) rather than tree structure as shown:

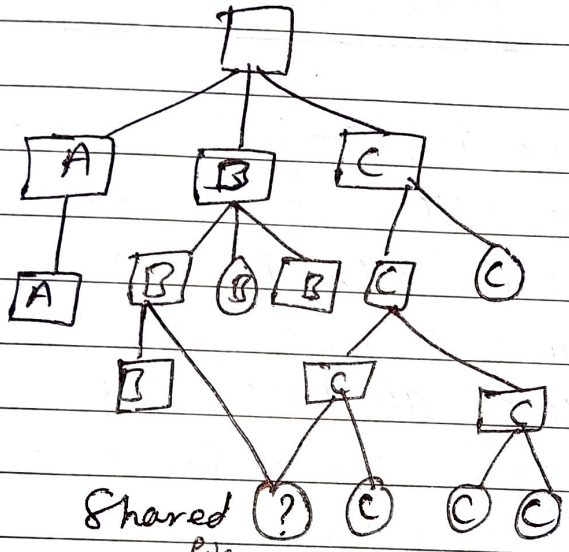


fig: File representation using DAG.

It is not a good idea to make a copy of a file if it is being shared. If directories contain disk addresses problem may occur in synchronizing the change made to the file by multiple users. If one user appends new block in the file, new block will be listed only in the directory of the same user only. To solve this:- two approaches are used:

- Directory entry that only points to i-nodes : here disk blocks aren't listed in directories rather directory entry point to the little data structure associated with the file itself.

DATE

- Directory entry that points to link file i.e. here, when another user B want to share the file owned by user C, then system creates a link file & then make entry of the link's file in B's directory.

* (Free Space Management)

Since disk space is limited, we need to reuse the space from deleted files for new files, if possible. To keep track of free space, the system maintains a free space list. The free space list records all free disk blocks those not allocated to some file or directory. There are mainly two approaches by which the free blocks on the disk are managed:

1. Bitmap Free Space Management

In this approach, free space list is implemented as a bit map vector. It contains the number of bits where each bit represents each block. If the block is empty then the bit is 1 otherwise it is 0. Initially, all the blocks are empty therefore each bit in the bitmap vector contains 1. A disk with n blocks require a bitmap with n bits.

classmate

PAGE

DATE

Ex: Let's take block list: {2, 3, 4, 5, 9, 10, 13}

⇒

Blocks →	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Bits →	0	0	1	1	1	1	0	0	0	1	1	0	0	1

Advantages:- Simple & efficient in finding first free block, or n consecutive free blocks.

Problems:- Inefficient until the entire bitmap is kept in main memory. Keeping in main memory is possible only for small disk, when disk is large, the bitmap would be large.

2. Linked List Free Space Management

It is another approach for free space management. This approach suggests linking together all the free blocks and keeping a pointer in cache which points to the first free block. So, all the free blocks on the disk will be linked together with a pointer. When a block gets allocated, its previous free block will be linked together to its next free block. We would keep a pointer to block 2, as the first free block. Block 2 would contain a pointer to block 3 which would point to block 4, which would point to block 5 & so on. However, this scheme is not efficient to traverse the list, we must read each block, which requires substantial I/O time

DATE

Advantages: Only one block is kept in memory
Problem: last sentence of previous paragraph.

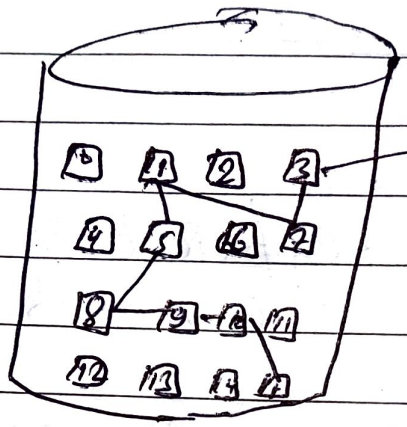


Fig: Linked list free space in the disk

Q Consider a 40MB disc with 2k blocks. Calculate the number of blocks needed to hold the disc bitmap. If we require 16-bit to hold a disc block number (i.e. disk block number range from 0-65535). What will be the number of blocks needed by linked list of free blocks?

=) For Bitmap

Disk Size = 40MB = 40 x 1024 KB

∴ No. of blocks = $\frac{40 \times 1024}{2}$ = 20 x 1024.

Since every block needs 1-bit in bitmap

∴ Size of Bitmap: 20 x 1024 bit = 20480 bit = $\frac{20480}{8}$ = 2560 byte

= 2.5 kB

Since block size is 2kB

∴ 2 blocks are used to hold the bitmap

For free list

Block Size: 2KB = 2 x 1024 byte.

No. of bits needed to store block no. = 16 bit = 2 byte.

Thus, no. of blocks that can be stored in a block

= (2 x 1024)/2

= 1024

Since, one of the addresses is used to store address of the next block in the free list

∴ No. of blocks that can be stored in a block = 1024 - 1

= 1023

We know, size of disk = 40MB = 40 x 1024 kB

Since, size of block = 2k

∴ No. of blocks in disc = (40 x 1024)/2 = 20 x 1024

Thus, no. of blocks needed to store free list = $\frac{20 \times 1024}{1023} \approx 20$

Questions asked from this Chapter.

Q. What is meant by file attributes? Discuss any one technique of implementing directories in detail. (2078 - 5 marks)

Q. What approaches are used for managing the free disk spaces? Explain linked list approach with example. (2078 - 5 marks) (2076 - 5 marks)

Q. Describe the methods for implementing files (2075 - 5 marks)

Q. Describe the methods for implementing directories with examples. (2074 - 5 marks)

Q. Define file and directories. Explain about protection mechanism. (2071 - 5 marks)

Q. Explain how file allocation table (FAT) manage the files. Mention the merits and demerits of using FAT (2071 - 5 marks) (2070 - 5 marks) (2066 - 5 marks).

Q. A 200 GB disk has 1 KB block size, calculate the size of FAT if each entry of the table has to be 8 bytes. (2066 - 5 marks)