File management Operations on files

Any file system provides not only a means to store data organized as files, but a collection of functions that can be performed on files. File manager must be able to perform following functions on files. These functions are divided into two categories as follow:

- Basic File Operations
- Typical Operations

Basic File Operations

- Create: Create a new file
- Delete: Delete an existing file
- Open: Open an existing or new file in read ,write or append mode
- Close: Close an open file
- Read : Read data from a file
- Write: Write data to a file

Typical Operations

- *Retrieve All*: To Retrieve all the records of a file. This type of operation will be required for an application that must process all the information stored in the file at one time.
- *Retrieve One*: retrieve_one operation requires the retrieval of a single record from file at a time . transaction-oriented and Interactive applications need this operation
- *Retrieve Next:* This operation requires the retrieval of the next record which is in some logical sequence to the most recently retrieved record.
- *Retrieve Previous :* retrieve next is same as retrieve next, but in this case the record that is –previous‖ to the currently accessed record is retrieved.
- *Insert_ One :* Insert a new record into the file at a particular place in the file and it is necessary that the new inserted record preserve a sequencing of the file.
- *Delete_One :* delete an existing record. This also needs to preserve the sequencing of the file.
- *Update_One :* Retrieve a record from file , update required fields, and rewrite updated record back into the file.
- *Retrieve_Few :* Retrieve required number of records. For example, if an application

needs to retrieve all records that satisfy a certain criteria.

Minimal Set of Requirements for File Users

- Each user should be able to create, delete, read, write and modify files.
- Each user may have controlled access to other users‘ files.
- Each user may control what type of accesses are allowed to the users‘ files.
- Each user should be able to restructure the user‘s files.
- Each user should be able to move data between files.
- Each user should be able to back up and recover the user‘s files in case of damage.
- Each user should be able to access the user‘s files by using symbolic names.

Information Types

It is useful to categorize the information stored in a file. type of information stored in a file, depend upon the type of a file. Whether it is a text file, document file executable file, object file or a source file of a program ie C,C++ etc. Depending upon the type of a file a request can be processed. Suppose executable programs, which contains machine code, sent to the printer will give garbled results. Print spooler may refuse to print the file executable program file. Following are the major schemes to identify the type of a file:

a) File name extension: An important part of a file name is its extension, which is a short string of additional characters appended to the file name. In most systems it is separated from the name by a period. File extensions are used to indicate the file‘s type. For example, myfile.txt would indicate that the file is a text tile, while myprog.bin would indicate that this file is an executablebinary program (a load module). A typical file extension is between one and four characters in length. Older systems, e.g., MS-DOS, are more rigid, requiring an extension of one to three characters, while Unix, Linux, and many other more recent systems allow more flexibility. The file type implies the internal format and the semantic meaning of the file‘s contents. by including the extension with the file name, type of information contained in the file can be identified. The file is a sequence of ASCII characters with no other format imposed on it. Such files are generally called text files and carry the extension .txt. A file containing a source program generally carries the extension indicating the programming language. For example, main.c might be a C program while main.f77 would be a Fortran 77 program. W hen compiled, the resulting files are object modules, denoted by the extension .o or .obj. A linker then takes an object module and produces the corresponding load module, a binary file, either annotated with the extension .bin, .com, or, as in the case of Unix, left without an extension.

In case of DOS operating system it only support .com or .exe file extension to be an executable file .following table lists the file type ,extension of that particular file and functions.

File Type	Usual Extension	Function
Executable	exe, com, bin	Read 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 language
Text	txt, doc	Textual data, documents

b) **Magic Number** magic number is a sequence of numbers at a particular place in a file. Operating system didn't recognize the magic number it depend upon and individual using or reading a file. Unix operating system is the only operating system in which Unix kernel recognize magic number for executable file to run executable files.

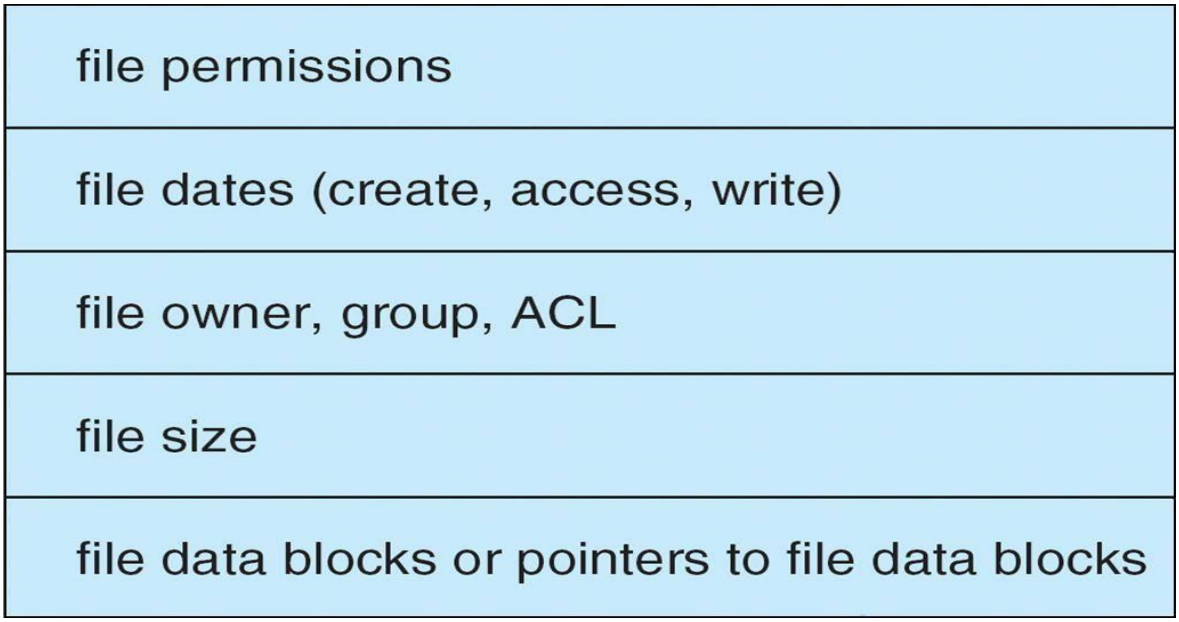
c) The third scheme is operating system dependable, as when file is being created with create file operation then operating system maintains ―type of file‖ as an attribute of file information structure while saving other attributes.

File information structure consist of following information which is stored in the File Control Block:

- the file's type (plain file, directory, etc),
- the file's size, in bytes and/or blocks,
- any limits set on the file's size,
- the primary owner of the file,
- information about other potential users of this file,
- access constraints on the owner and other users,
- dates and times of creation, last access and last modification
- dates and times of last backup and recovery, and
- event triggers when the file changes

Above three schemes can be used to recognize the file and the information stored on the file.

Structure of File Control Block:



Structure of File Control Block

Introduction to File System Architecture

In most of the operating systems, ―Everything is a file; if something is not a file, it is a process‖. In order to keep track of which information is stored on the hard disk, information is bundled into logical units called as Files. Files are stored on Secondary storage with a unique name. Files can be organized into hierarchy of directories and subdirectories. To manage these files and directories file system of operating system manage various task about files. File system architecture is divided into layers, where each layer performs particular set of functions. For accessing information from the files various access methods such as direct, sequential and indexed can be used. For the security and protection of files access rights are used to restrict the access on files which are stored in the List known as Access control list.

File-System Structure

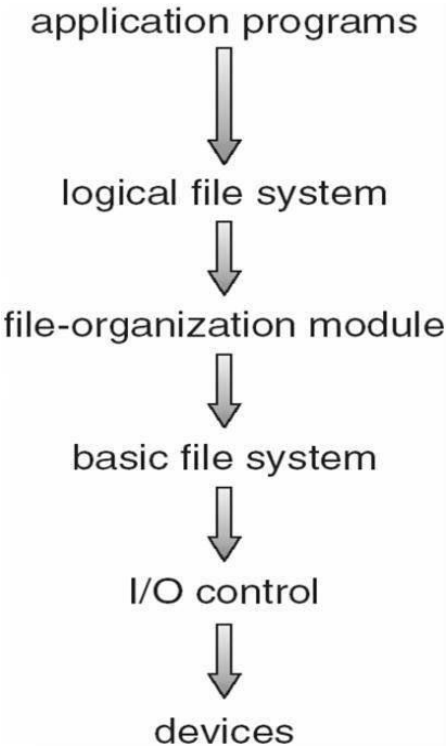
Hard disks are most suitable devices for the secondary storage because they have following two properties:

(1) Blocks of data can be read from the Disk then one can modify the data and it can be written back to the same place after that.
(2) Any block of data to be accessed directly with only (relatively) minor movements of the disk heads and rotational latency.

- Disks are accessed in physical blocks, rather than a byte at a time. Block sizes may range from 512 bytes to 4K or larger.

- File systems organize files into logical hierarchical structures with directories, soft links and so on held in blocks on physical device, and can be viewed as a layered design as follow(fig 12.1):

- At the devices layer, lowest layer of the hierarchy there are the physical devices, consisting of the magnetic media, motors & controls with electronics connected to them and controlling them.the Modern disk put more of the electronic controls directly on the disk drive to minimize work for the disk controller card to perform.
-



Layered Architecture of File System

- ***I/O Control*** consists of ***device drivers***, a **device driver** commonly referred to as simply a **driver** is a computer program that operates or controls a particular type of device that is attached to a computer . A driver provides a software interface to hardware devices, enabling operating systems and other computer programs to access hardware functions without needing to know precise details of the hardware being used.

- The ***basic file system*** level works directly with the device drivers in terms of retrieving and storing raw blocks of data, without any consideration for what is in each block. Depending on the system, blocks may be referred with a single block number, or it can be referred with head-sector-cylinder combinations.

- The ***file organization module*** knows about files and logical blocks comprises the files, and how the logical blocks map to physical blocks on the disk. In addition to transforming from the logical to physical blocks, A list of free blocks is maintained by the file organization module and when required allocates free blocks to files.

- The ***logical file system*** deals with all the meta data associated with files stored on the secondary storage i.e. everything about a file except the data of the file itself. The directory structure and the mapping of file names to ***file control blocks,are managed by this level***, file control block contain all of the meta data as well as block number information for finding the data on the disk.

- The key idea of layered approach for the file systems is, for the different file systems,much of the code can be used uniformly and only certain layers need to be a particular filesystem specific. Common file systems in use are the Berkeley Fast File System, FFS, Windows systems FAT, FAT32, NTFS, CD-ROM systems ISO 9660, UNIX file system, UFS and for Linux the extended file systems ext2 and ext3.

File Access Methods

An access method is a function of operating system that enables access to data on disk, tape or other external devices. Access methods provide an application programming interface (API) for programmers to transfer data to or from device.

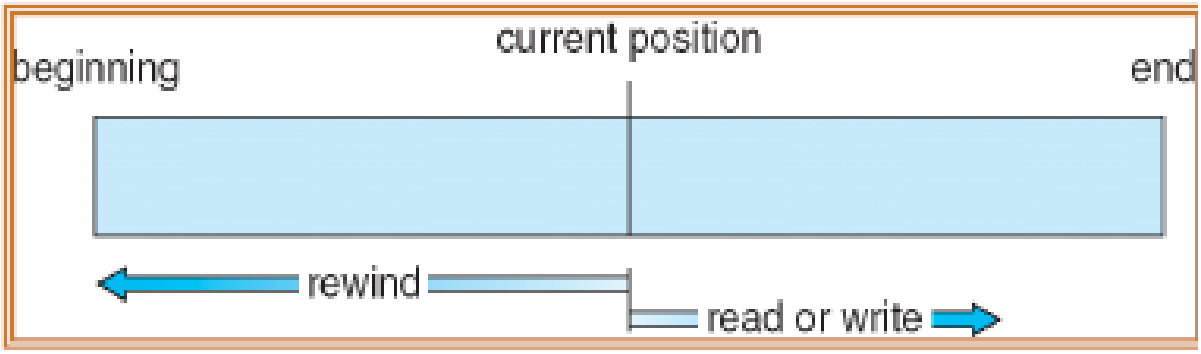
Information is kept in files. Files reside on secondary storage. When this information is to be used, it has to be accessed using access methods and brought into primary main memory. Information in files could be accessed in many ways. It is usually dependent on an application.

There are several ways that the information in the file can be accessed. Some systems provide only one access method for files. On other systems, many different access methods are supported, and choosing the right one for a particular application is a major design problem

Sequential Access

Information in the file is processed in order, one record after the other. This is by far the most common mode of access of files. For example, computer editors usually access files in this fashion.

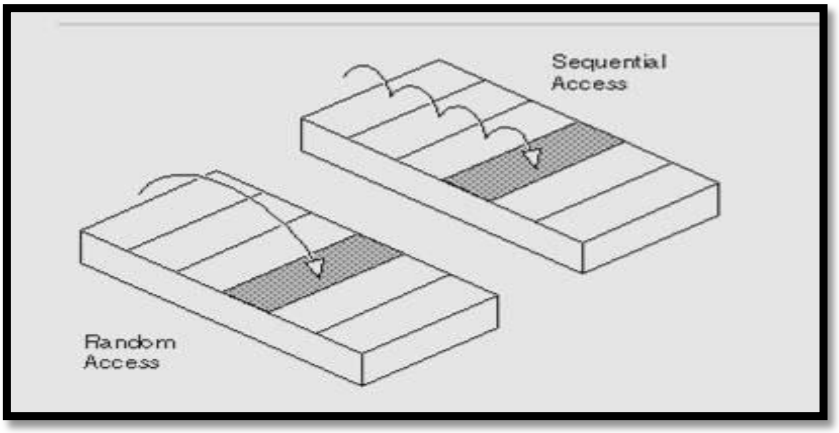
A read operation reads the next portion of the file and automatically advances the file pointer. Similarly, a write appends to the end of the file and the file pointer. Similarly, a write appends to the end of the file and the file pointer. Similarly, a write appends to the end of the end of the file and advances to the end of the newly written material (the new end of file). Such a file can be reset to the beginning, and, on some systems, a program may be able to skip forward or backward n records, for some integer n. This scheme is known as sequential access to a file. Sequential access is based on a tape model of a file.



Sequential File Access

Direct Access

Direct access is based on a disk model of a file. For direct access, the file is viewed as a numbered sequence of block or records. A direct-access file allows arbitrary blocks to be read or written. Thus, after block 28 has been read, block 67 could be next, and then block 13. There are no order of reading and writing for a direct access file. Direct access files are of great use when intermediate access to large amounts of information is required.



direct Access

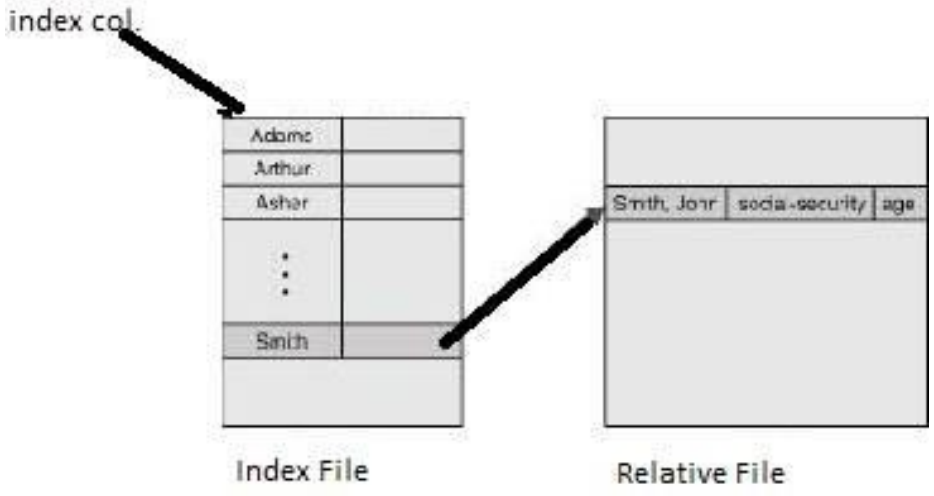
The Direct Access file operations as compared to Sequential access must be modified to include the block number as a parameter. so, we have to use "read n", where n is the block number, not "read next", and "write n", rather than "write next".

Not all OS support both sequential and direct access for files. Some systems allow only sequential file access; others allow only direct access. Some systems require that a file be defined as sequential or direct when it is created; such a file can be accessed only in a manner consistent with its declaration.

Indexed Sequential Access

This access method is the modification of the direct access method. Basically It is a combination of both the sequential access as well as direct access. The main concept is to access a file with direct access method first and then access the file in sequential manner from that point onwards. In this access method an index is being maintained by file system which is a pointer to a block.

To read/write a record from file, a direct access of the index is made, which is further used to access the file. The direct access to a file will give the block address and within the block the record is accessed sequentially. Large indexes may decrease the performance of the system. To overcome this problem hierarchical indexes are built in which one direct access of an index leads to info to access another index directly and so on till the actual file is accessed sequentially for the particular record. The main advantage in this type of access is that both direct and sequential access of files is possible.



Index Sequential File.

Other Access Methods

Using direct access method Other access methods can be built.These methods involve the creation of an index for a file. The index contains pointers to the various blocks. To find a record in the file, the index is searched first and the pointer is then used to access the file directly to find the desired Record.

In case index file is large file, the index itself may become too large to be kept in memory. One solution is to create an index for the index file. The primary index file would contain pointers to secondary index files, which would point to the actual data items.

For example, IBM uses indexed sequential access method (ISAM). ISAM uses a small master index that points to disk blocks of a secondary index. The secondary index blocks in turn point to the actual file blocks. The file is kept sorted on a defined key. To find a particular item, binary search of the master index is perfomed, which provides the block number of the secondary index. This block is read in, and again a binary search of the block is used to find the block containing the desired record. At last the block is searched sequentially.

Access control

Access control scheme in case of a file system is the set of operations to be performed. So in order to limit the access on the file, owner of the file should limit the set of operation allowed

on file. Following are the set of operations that can be performed on a file.

Read: to read, what information is stored in the file.

Write: in order to wite new information in the file,**Append:** to write new information at the end of a file.

Delete: to delete a file and space occupied by the file is also released to reuse it by another purpose.

List : to list, which files are contained in a particular directory.

Execute: to execute a file is to load file‘s contents into main memory and to create a process to execute a particular file.

Change Access : to change access rights of a user in order to give controlled rights on the file.

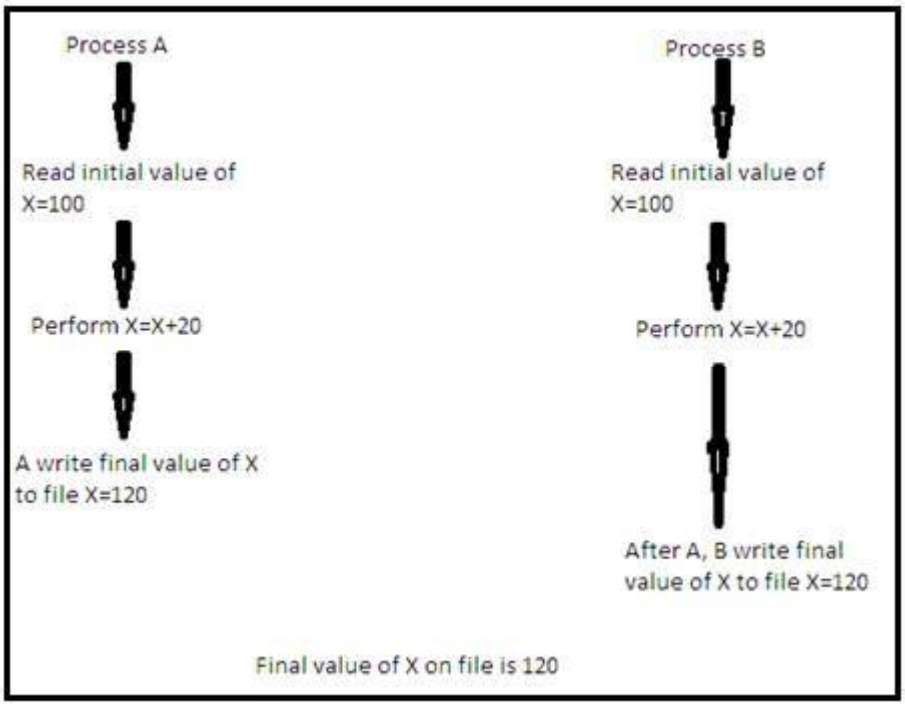
Access Control List (ACL)

An access control list (ACL) is a list of set of permissions attached to an object. An ACL identifies, on a given object what operations are allowed to which users or system processes.

A Filesystem **ACL** is a data structure (usually a table) containing entries that specify individual user or group rights to specific system objects such as programs, processes, or files. These entries are known as access control entries (ACEs) in all the operating systems e.g. Unix-like, Windows NT, and Mac OS X operating systems. Each object stored in the Hard disk contains an identifier to its ACL along with the privileges or permissions which determines specific access rights (**Read, write, Execute)** on a particular object.

File Locking

File locking is a way of ensuring and maintaining the integrity of files. It allows many processes to share a file in a safe way by allowing only one user or process access at any specific time. There is no need to follow an integrity mechanism if only one process is working on the file, and then there is no need to worry about anybody else changing it. However, when more than one user or processes trying to access a single file to modify it , conflicts can arise, and some sort of mechanism is to be follow to prevent classic *interceding update* scenario as shown in following fig 12.5. The most common mechanism is file locking.


Process A
Read initial value of X=100
Perform X=X+20
A write final value of X to file X=120
Process B
Read initial value of X=100
Perform X=X+20
After A, B write final value of X to file X=120
Final value of X on file is 120

Blocking Methods

The records in a data set have one of the following formats:

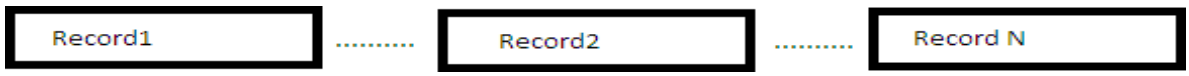
Fixed-length
Variable-length
Undefined-
length.

If there is requirement, Depending upon the various records lengths , Records can be blocked. The operating system will de-block fixed-length and variable-length records. but for the undefined records types one has to give implementation code to de-block it.

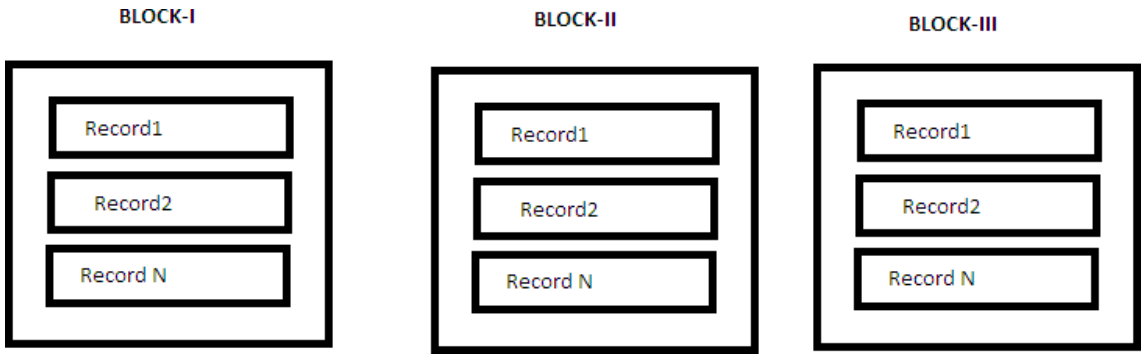
Fixed-length records

Fixed-length records can be specified as follow:
F : Fixed-length,
unblocked FB : Fixed-
length, blocked

In fixed-length records, all records have the same length. In case of blocked records, each block usually contains an equal number of fixed-length records(shown in Fig 12.7). If the records are unblocked, then each record constitutes a block(shown in Fig 12.6).



Fixed Length Unblocked Records



Fixed Length Blocked Records

When blocking and de-blocking on a constant record length is done then the operating system

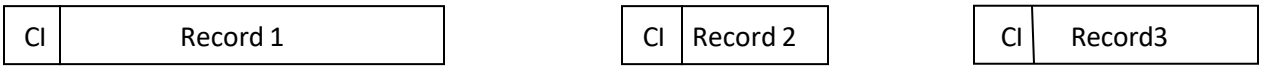
processes fixed-length records faster than variable-length records.

Variable-length records

Variable-length Records can be specified as follow:
VB/V Variable-length,Blocked/
unblocked VS Variable-length,
Spanned

VB/V Variable-length,Blocked/ unblocked

Variable-format allows both records of variable-length and variable-length blocks. Each block and record in this case will be prefixed by 4 bytes which contains control information(CI) to be used by the operating system as shown in the Figure 12.8. Due to 4 bytes prefix format, variable-length records cannot be read backward.



Variable Length unblocked/Blocked Records

When unblocked variable-length records exist then each record is treated as a block containing only one record. Here first 4 bytes will be used as block control information and next four will be used as record control information.

VS Variable-length, Spanned

Spanned Records: when the size of a record exceeds the size of a block then the record is divided into small segments and stored in two or more consecutive blocks by specifying the record format .This type of records are known as spanned records. Segmentation of records and their assembly is handled by the operating system.The main advantage of this type of record is ,it allows you to select a block size, independently of record length, that will result into optimum use of auxiliary storage with maximum efficiency of transmission.

Record1	Record 2	Free Block	Record3	Record3	Free Block	Record4	Record 5

Spanned Records

VBS-format differs from VS-format in that each block contains complete records as it can accommodate; each block is, therefore, app. the same size (although there can be a variation of up to 4 bytes, since each segment must contain at least 1 byte of data).

Undefined-length records

U-format allows the processing of records that do not conform to F- and V-formats. The operating system and the compiler treat each block as a record; your program must perform any required blocking or deblocking.

File Allocation Methods

File allocation method is the method that the basic file system uses to place the logical file blocks onto the physical Storage medium for permanent storage e.g into the disk‘s tracks and sectors following are the three major methods of storing files on disks:

- Contiguous File Allocation
- Linked Allocation
- Indexed Allocation.

Contiguous File Allocation

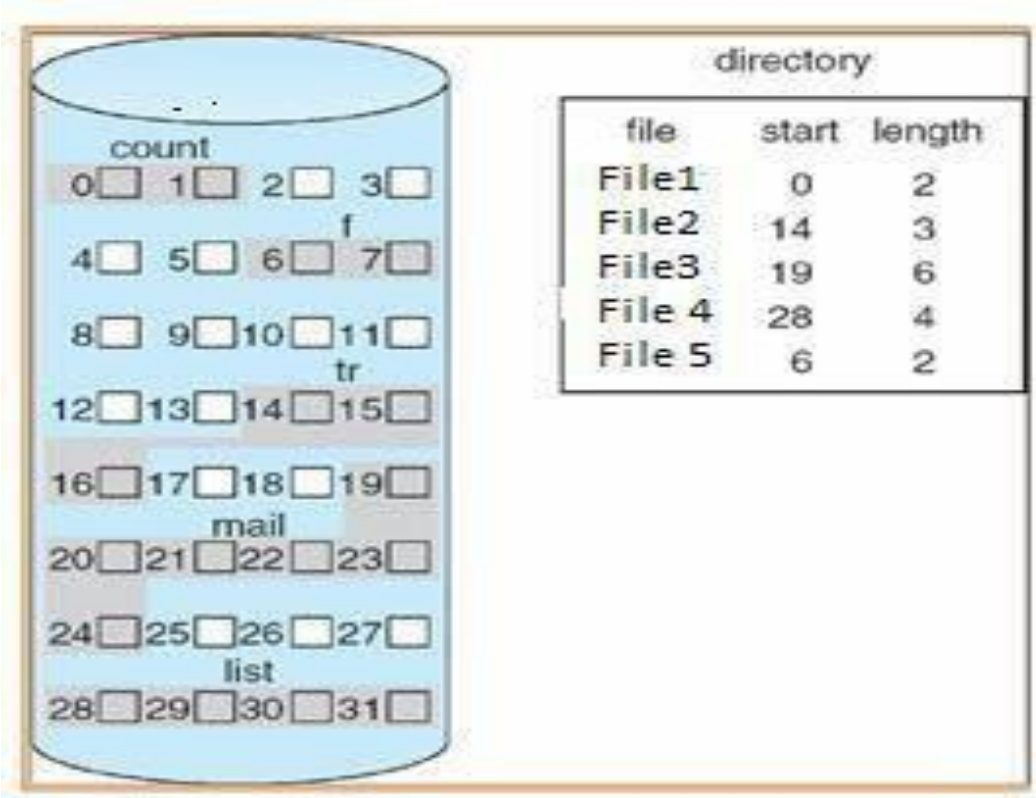
The simplest policy is the fixed or contiguous allocation of free space to the file. This type of allocation requires that a file‘s maximum size be known in advance ,when the file is created, and the size of file cannot exceeds beyond maximum space allocated to file.The file allocation table (FAT) stores starting block and length for each file. As this is simple method for allocation, but this also suffers from both internal fragmentation and external fragmentation. In order to minimize the disk space fragmentation, a compaction scheme is required.

Following are the key features of *Contiguous Allocation*

- ***In Contiguous Allocation*** all blocks of a file should be Contiguous.
- In case of contiguous allocation , generally ,there is no movement of the disk heads, or at most it requires one step to move to the next adjacent cylinder. These improve the performance in terms of fast access time.

- If the size of a file is not known in advance then following problems can arise if files grow:
    - First, Over-estimation of size of file can leads to external fragmentation and wastes disk space.
    - Second, Under-estimation of file size may require that a file may need to be moved to another location in order to accommodate the file data. Second case may be that a process may be aborted if the file grows beyond its originally allocated space.
    - Third problem is If a file grows slowly over a long period of time and the space allocated to file is very large, then a lot of space becomes unusable before the file fills the space.

- A variation is to allocate file space in large contiguous chunks, called ***extents.*** When a file

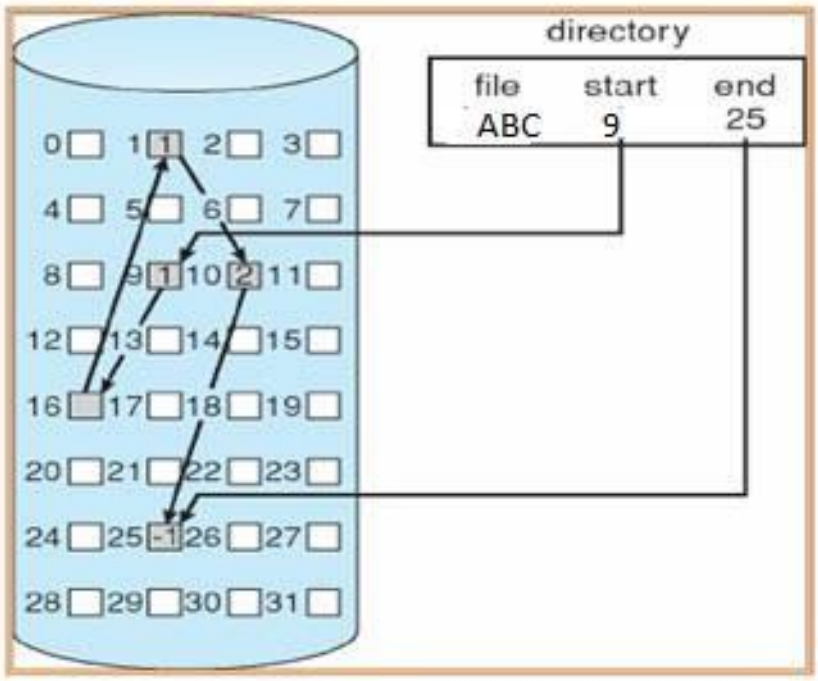


outgrows its original extent, then an additional one is allocated.

Contiguous File Allocation

Chained or Linked File Allocation

The opposite Scheme to contiguous allocation is chained allocation. In this method, all the blocks allocated to a file form a linked list and to extend the file‘s length, a new block is allocated and linked to the last block allocated to the file. Within each block a small pointer is allocated to indicate the address of next block allocated to the same file.the size of this pointer can be 32 or 64 bits. Thus to read the full file follow the pointers pointing to next block allocated to file. The main advantage of this approach is new blocks may be allocated from any free block available on the disk.



Linked Allocation

Following are the key features of *Linked Allocation*

- This approach is expensive than contiguous allocation in terms of disk storage as the storage space consumed by each link E.g. a block may be 508 bytes instead of 512.
- The main advantage of Linked allocation are
  1. that there is no external fragmentation
  2. it does not require pre-known file sizes.
  3. It Allows files to grow dynamically at any time when required.
- Unfortunately linked allocation is not efficient in case of random access as random access requires starting at the beginning of the list for each new location access.
- Cluster Allocation will reduce the space wasted by pointers, at the cost of internal fragmentation.
- Another big problem with this approach is reliability. if a pointer is lost or damaged then link to the next block will be lost. So to minimize this risk doubly linked lists can provide some protection, at the cost of additional overhead and wasted space.

Indexed File Allocation Method

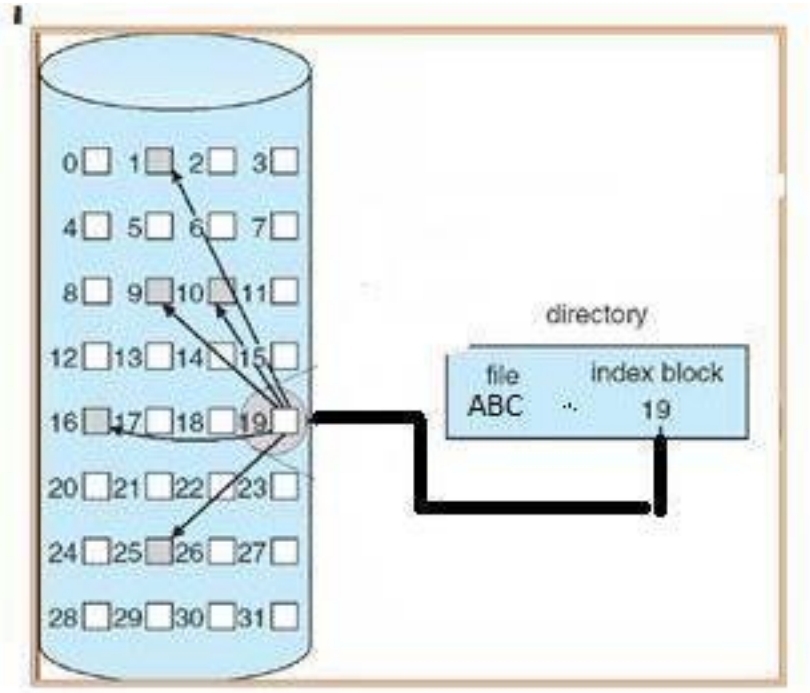
In this approach multilevel index is used for each file pointer in FAT table. Indirection blocks are bring in, each time the total number of blocks –overflows‖ the previous index allocation. The indices are stored neither with the FAT Table nor with the file, and are kept in memory when the file is opened. Both in Unix and Windows This file allocation method is used. In 1966 Multics

operating system was the big supporter of indexed file allocation.

Key features of *Indexed Allocation*

- ***In case of Indexed Allocation indexes for accessing each file are combined into a common block instead of*** spreading them all over the disk or storing them in a FAT table as single entries.

- Small portion of disk block space is wasted as compared to linked lists or FAT tables because a full index block must be allocated for each file, regardless of how many data blocks the file contains.



Indexed file Allocation Performance

- The optimal allocation method is different for sequential access files than for random access files, and is also different for small files than for large files.

- Some systems support more than one allocation method, which may require specifying how the file is to be used (sequential or random access) at the time it is allocated. Such systems also provide conversion utilities.

- Some systems have been known to use contiguous access for small files, and automatically switch to an indexed scheme when file sizes surpass a certain threshold.

- And of course some systems adjust their allocation schemes (e.g. block sizes) to best match the characteristics of the hardware for optimum performance.