ISSN:0975-9646

Directory Structure and File Allocation Methods

Mandeep Kaur, Sofia Singh, Rupinder Kaur

Assistant Professor,
PG Department of Computer Science and Applications,
GHG Khalsa College Gurusar Sadhar, Ludhiana, Punjab, India

Abstract: Today computer is an integral part of human life. The storage of large amount of data permanently in computer system files is used. In this research paper we discuss the file that is a collection of records or information stored on secondary storage such as hard disk. In computing a file system is used to control how data is stored and retrieved. File system control the files starting and ending locations. The information present in the file can be accessed using access methods. In file any time data is failure with hardware problem for solution file system provide protection with access privileges of users. In files secondary storage space is allocated using file allocation methods. These allocated space in such a manner so that disk space is utilized effectively and files can be accessed quickly. Directory structure is use symbol table of files that stores all the related information about the file it holds with the contents.

Keywords: File System, File Protection, File Access Methods, File Allocation Methods, Directory Structure.

I. INTRODUCTION

File is a logical collection of information stored on secondary storage such as hard disk. It is a collection of records. Physically, a file is smallest allotment of secondary storage device for example disk. Logically, a file is a sequence of logical records such as a sequence of bits and bytes. Files can be used to contain the data and programs (both source and object programs). Data files can be numeric, alphabetic, alphanumeric or binary. A file has various attributes like name, type, location, size, protection, time and date of creation etc. Computers can store information in several physical forms, depending on which storage device is used. Disks and drums though are the most common devices for this purpose. Since each device has its own characteristics and physical organization, information may be stored in several ways and therefore different views of information are created. To unify all these views of information in the system, a uniform logical view of it was created. This logical view is called a file. It is the job of the OS to map this sequence of words into physical devices. The part of the OS responsible for this is the file system. It is clear that the main objective of the file systems is to free the users of the details of storing the information in the physical devices. That is, when the storage device is changed, from disk to drum for example, the user still sees the same information as before the change. If this is allowed in the system, then we can say that the file system is device dependent.

II. FILE SYSTEM

In computing a file system is used to control how data is stored and retrieved. Without a file system, information placed in a storage area would be one large body of data with no way to tell where one piece of information stops and the next begins. By separating the data into individual pieces, and giving each piece a name, the information is easily separated and identified. Taking its name from the way paper-based information systems are named, each group of data is called a "file". The structure and logic rules used to manage the groups of information and their name is called a "file system". There are many different kinds of file systems. Each one has different structure and logic, properties of speed, flexibility, security, size and more. File systems can be used on many different kinds of storage devices. Each storage device uses a different kind of media. The most common storage device in use today is hard device whose media is a disc that has been coated with a magnetic film. The film has ones and zeros 'written' on it sending electrical pulses to a magnetic "read-write" head. Other media that are used are magnetic_tape, optical disc and flash memory. The file system manages access to both the content of files and the metadata about those files. It is responsible for arranging storage space; reliability, efficiency, and tuning with regard to the physical storage medium are important design considerations.

III. ACCESS METHODS OF FILE

Files are used to store data. The information present in the file can be accessed by various methods. Thus, the way of retrieving data from a file is known as access methods. Different systems use different access methods. The various access methods used are:

1. Sequential access
2. Direct access
3. Indexed access

1. **Sequential access:-** It is the simplest and most commonly used access method. In this information in the file is accessed in the order it is stored in the file one record after the other. The various records are read sequentially one after the other in an order, starting at the beginning to the end of the file. The various records cannot be read randomly out of order we can not skip any record in between. For example reading of 34 record followed by 5 record and then 1 record is not possible in sequential access. 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 access is convenient when the storage medium is magnetic tape, rather then a disk.

2. **Direct Access:-** In direct access method it is possible to access the records of a file in any order. For example, if we are reading block 13, we can read block 46 after this and then block 20. Various records are read or write randomly. 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 18 has been read, block 57 could be next, and then block 3. There are no restrictions on the order of reading and writing for a direct access file. Direct access files are of great use for intermediate access to large amounts of information. The file operations must be modified to include the block number as a parameter. Thus, we have "read n", where n is the block number, rather than "read next", and "write n", rather that "write next". An alternative approach is to retain "read next" and "write next" and to add an operation; "position file to n" where n is the block number. Then, to effect a "read n", we would issue the commands "position to n" and then "read 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. Direct access method is important for many applications, for example database system.
3. **Indexed Access:-**In this method, an index is created for the file. This index contains pointer for various blocks of a file, just like an index in the back of the book. If we want to find a record of a file, first the index is searched and then the pointer from index is used to access that file. In this way, a required record is found. This access method is a slight modification of the direct access method. It is in fact a combination of both the sequential access as well as direct access. The main concept is to access a file direct first and then sequentially from that point onwards. This access method involves maintaining an index. The index is a pointer to a block. To access a record in a file, a direct access of the index is made. The information obtained from this access is used to access the file. For example, the direct access to a file will give the block address and within the block the record is accessed sequentially. Sometimes indexes may be big. So hierarchies of 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.

IV. DIRECTORY STRUCTURE

Directory is a symbol table of files that stores all the related information about the file it hold with the contents. Directory is a list of files. Each entry of a directory define a file information like a file name, type, its version number, size ,owner of file, access rights, date of creation and date of last backup.

LOGICAL STRUCTURE OF DIRECTORY

The directories can be structured in the following ways:-

1. Single level directory
2. Two level directory
3. Tree structured directory
4. Acyclic graph directory
5. General graph directory

l. Single level directory: In a single level directory system, all the files are placed in one directory. This is very common on single-user OS's. A single-level directory has significant limitations, however, when the number of files increases or when there is more than one user. Since all files are in the same directory, they must have unique names. If there are two users who call their data file "test", then the unique-name rule is violated. Although file names are generally selected to reflect the content of the file, they are often quite limited in length. Even with a single-user, as the number of files increases, it becomes difficult to remember the names of all the files in order to create only files with unique names shown in Fig1.

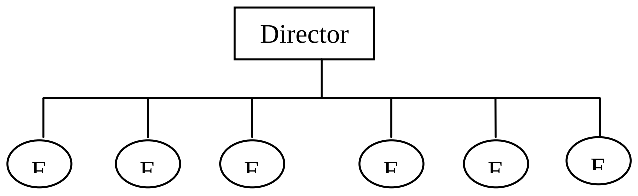


Fig1

2. Two level directory: In the two-level directory system, the system maintains a master block that has one entry for each user. This master block contains the addresses of the directory of the users. There are still problems with two-level directory structure. This structure effectively isolates one user from another. This is an advantage when the users are completely independent, but a disadvantage when the users want to cooperate on some task and access files of other users. Some systems simply do not allow local files to be accessed by other users shown in Fig 2.

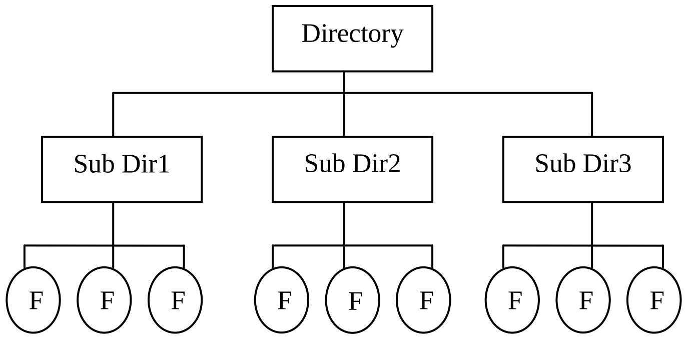


Fig 2

3. Tree structured directory: In the tree-structured directory, the directory themselves are files. This leads to the possibility of having sub-directories that can contain files and sub-subdirectories. An interesting policy decision in a tree-structured directory structure is how to handle the deletion of a directory. If a directory is empty, its entry in its containing directory can simply be deleted. However, suppose the directory to be deleted id not empty, but contains several files, or possibly sub-directories. Some systems will not delete a directory unless it is empty. Thus, to delete a directory, someone must first delete all the files in that directory. If these are any sub-directories, this procedure must be applied recursively to them, so that they can be deleted also. This approach may result in a insubstantial amount of work shown in fig 3

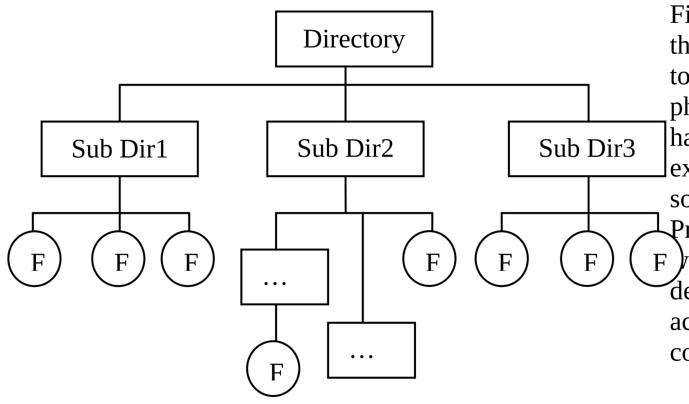


Fig. 3

4. Acyclic graph directory: The acyclic directory structure is an extension of the tree-structured directory structure. In the tree-structured directory, files and directories starting from some fixed directory are owned by one particular user. In the acyclic structure, this prohibition is taken out and thus a directory or file under directory can be owned by several users shown in fig 4.

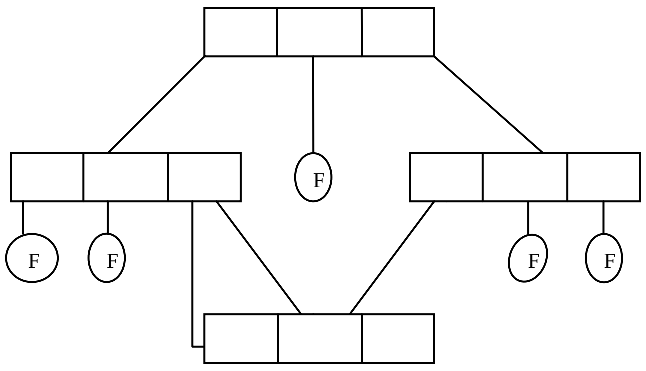


Fig 4

5. General graph directory: The general graph directory is formed by adding links into an existing tree structure. It overcomes the problem of acyclic graph by allows the cycles in a directory. Thus it avoids the searching of a component twice in a subdirectory in fig 5.

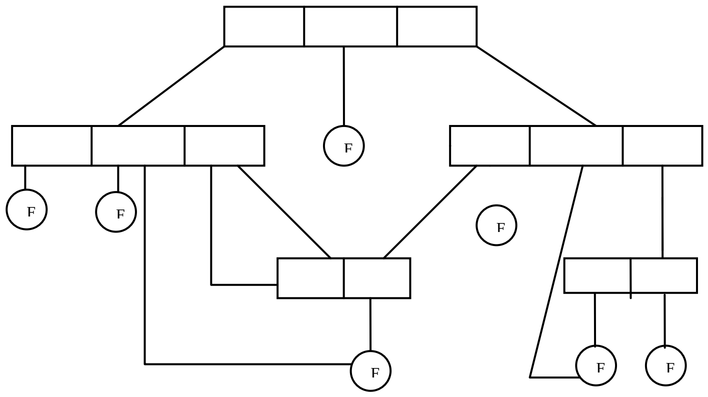


Fig 5

V. FILE PROTECTION

Files often contain information that is highly valuable to their users. One of the major functions of the file system is to protect this information against unauthorized access and physical damage. Physical damage may occur because of hardware problems, power failure, head crashes, dirt and extreme temperatures. In order to prevent such damage some systems performs backup at regular intervals. Protection is achieved by limiting the type of file access which can be made. Access is permitted or denied depending upon several factors, one of which is the type of access requested. Several operations on files can be controlled. Some of these are:

- **read** - read a file
- **write** - write a file
- **execute** - load and execute a file
- **append** - append information at the end of a file
- **delete** - free the space allocated to a file
- **update** – modifying, deleting and adding to a file
- **copy-** copy the contents of a file
- **list-** listing the name of the file

The most common implementation of the file systems allow the owners of the file to do operations 1-5, whereas other users can only invoke those operations that do not modify the file, e.g., file read. However, in some systems, e.g., UNIX, the user can change the access control of a file such that he can let anybody access (modification allowed) the file or he can completely deny any user (including himself) access to a file. Files use different access rights such as:

1. Access Control:- It is the most common approach to protect the files and directories depending upon the identify of the users. Access control limits who can access files and how they can access them. Users and group of users are granted certain access rights to a file. An access list is associated to each file or directory. The access list contains information on the type of users and accesses that they can do on a directory or file. An **example** is the following access list associated to a UNIX file or directory: drwxrwxrwx

The d indicates that this is an access list for a directory, the first rwx indicates that it can be read, written, and executed by the owner of the file, the second rwx is an access information for users belonging to the same group as the owner (somewhere on the system is a list of users belonging to same group as the owner), and the last rwx for all other users. The rwx can be changed to just r-- indicating that it can only be read, or -w- for write-only, --x for execute only.

2. Password Protection:- Another approach to protect a file from an unauthorized access is to use a password with each file. This scheme associates a password to each file. If a user does not know the password associated to a file then he cannot access it. This is a very effective way of protecting files but for a user who owns many files, and constantly changes the password to make sure that nobody accesses these files will require that users have photographic memories.

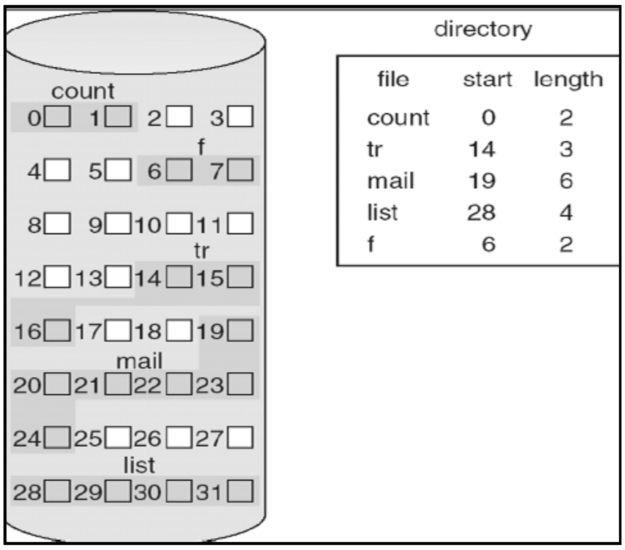
3. File Naming:- This depends upon the inability of a user to access a file he cannot name. This can be implemented by allowing only users to see the files they have created. But since most file systems allow only a limited number of characters for filenames, there is no guarantee that two users will not use the same filenames. A name is attached to every file so as to uniquely identify it and access it through its name. The exact rules for naming file vary from system but all the operating systems allow string of one to eight letters as legal filename.

VI. ALLOCATION METHODS OF FILE

Allocation refers to the process of assigning secondary storage space in files. The files should be allocated space in such a manner so that disk space is utilized effectively and files can be accessed quickly. The allocation method is responsible for mapping a file's logical blocks into the actual physical blocks on the secondary storage device. In most operating systems, the size of a physical block is a power of 2 between 512 and 4096. There are three major methods of allocating disk space to files:

1. Contiguous allocation
2. Linked allocation
3. Indexed allocation

1. Contiguous Allocation:- The contiguous allocation method requires each file to occupy a set of contiguous address on the disk. Disk addresses define a linear ordering on the disk. Notice that, with this ordering, accessing block b+1 after block b normally requires no head movement. When head movement is needed (from the last sector of one cylinder to the first sector of the next cylinder), it is only one track. Thus, the number of disk seeks required for accessing contiguous allocated files in minimal, as is seek time when a seek is finally needed. Contiguous allocation of a file is defined by the disk address and the length of the first block. If the file is n blocks long, and starts at location b, then it occupies blocks b, b+1, b+2, …, b+n-1. The directory entry for each file indicates the address of the starting block and the length of the area allocated for this file. The difficulty with contiguous allocation is finding space for a new file. If the file to be created is n blocks long, then the OS must search for n free contiguous blocks. First-fit, best-fit, and worst-fit strategies (as discussed in Chapter 4 on multiple partition allocation) are the most common strategies used to select a free hole from the set of available holes. Simulations have shown that both first-fit and best-fit are better than worst-fit in terms of both time storage utilization. Neither first-fit nor best-fit is clearly best in terms of storage utilization, but first-fit is generally faster. These algorithms also suffer from external fragmentation. As files are allocated and deleted, the free disk space is broken into little pieces. External fragmentation exists when enough total disk space exists to satisfy a request, but this space not contiguous; storage is fragmented into a large number of small holes. The operating system that uses contiguous allocation is IBM VM/CMS. For example file count starts from 0 and length is 2 so end is 1, file tr starts from 14 and length is 3 so end is 16, file mail is starts from 19 and end 24, file list is starts from 28 and ends with 31 and file f is starts from 6 and end with 8.



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

Contiguous Allocation

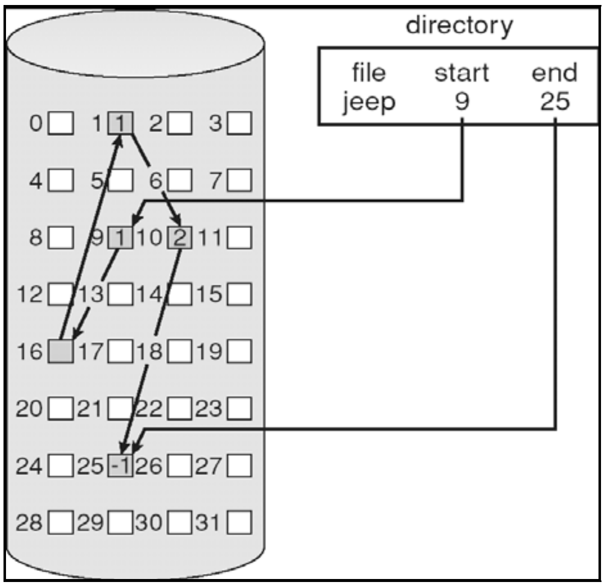
Advantages of Contiguous Allocation:

- It is simple to implement because keeping track of where a file's blocks are reduced to remembering only one number.
- Performance is good because entire file can be read from the disk in a single operation.
- In this scheme, number of disk seeks required for accessing the file is minimal. Disk addresses define linear ordering on the disk, accessing block b+1 after block b requires no head movement. As a result number of disk seeks required for a file s are less.
- It is the best from the point of view of the individual sequential file.

Disadvantages of Contiguous Allocation:

- This method from the problem of external fragmentation. As files are allocatd and deleted, the free disk space is broken into little pieces. External fragmentation exists whenever free space is broken into chunks. However, compaction be applied as a solution to this problem, because compaction of the disk is expensive.

1. Linked Allocation:- The problems in contiguous allocation can be traced directly to the requirement that the spaces be allocated contiguously and that the files that need these spaces are of different sizes. These requirements can be avoided by using linked allocation. In linked allocation, each file is a linked list of disk blocks. The directory contains a pointer to the first and (optionally the last) block of the file. For example, a file of 5 blocks which starts at block 9, might continue at block 16, then block 1, block 10 and finally block 25. Each block contains a pointer to the next block and the last block contains a NIL pointer. The value -1 may be used for NIL to differentiate it from block 0. With linked allocation, each directory entry has a pointer to the first disk block of the file. This pointer is initialized to nil (the end-of-list pointer value) to signify an empty file. A write to a file removes the first free block and writes to that block. This new block is then linked to the end of the file. To read a file, the pointers are just followed from block to block. There is no external fragmentation with linked allocation. Any free block can be used to satisfy a request. Notice also that there is no need to declare the size of a file when that file is created. A file can continue to grow as long as there are free blocks. Linked allocation, does have disadvantages, however. The major problem is that it is inefficient to support direct-access; it is effective only for sequential-access files. To find the ith block of a file, it must start at the beginning of that file and follow the pointers until the ith block is reached.



Linked Allocation

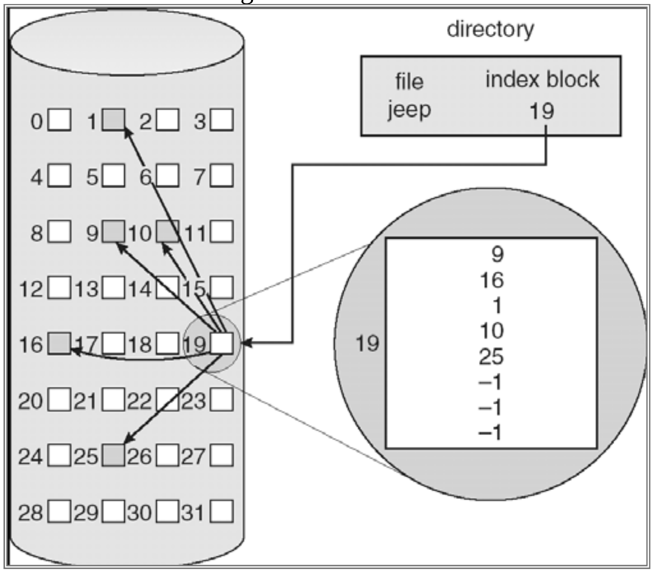
Advantages of Linked Allocation:

- Unlike contiguous allocation every free disk block can be utilized.
- It does not suffer from the problem of external fragmentation.
- There is no need to declare the size of a file at the time of its creation. A file can grow as long as free blocks are available.
- There is no need to perform the compaction.

Disadvantages of Linked Allocation:

- Linked allocation can be used effectively only for sequential files. It is inefficient to support a direct access. In order to find the ith block of a file, we must start at the beginning of that file and follow the pointers until we get to the ith block. Each access to a pointer disk read and a disk seek.
- Pointer takes up space in each disk block. It consumes some portion of a block that can be used for storing information.

2. Indexed Allocation: - The indexed allocation method is the solution to the problem of both contiguous and linked allocation. This is done by bringing all the pointers together into one location called the index block. Of course, the index block will occupy some space and thus could be considered as an overhead of the method. In indexed allocation, each file has its own index block, which is an array of disk sector of addresses. The ith entry in the index block points to the ith sector of the file. The directory contains the address of the index block of a file. To read the ith sector of the file, the pointer in the ith index block entry is read to find the desired sector. Indexed allocation supports direct access, without suffering from external fragmentation. Any free block anywhere on the disk may satisfy a request for more space. For example set the index value is 19 all blocks are linked to that index values provide the references to all blocks. Index method follow the direct method using index block.



Indexed Allocation

Advantages of Indexed Allocation:

- It is the most popular from of file allocation and support both sequential and direct access to the file.
- Any free block on the disk can be used for allocation.
- Allocation of space an the basis of individual block eliminates external fragmentation.
- Allocation of space on the basis of variable size portions improves locality.

Disadvantages of Indexed Allocation:

- If the index block is small, it will not be able to hold enough pointers for a large file.
- The entire index or table will have to be kept in main memory for all the times to make it work.
- Looking up for an entry in a large index is a time consuming process.

CONCLUSION

This paper discusses how the Modify-on-Access file system efficiently extends the capabilities of conventional file systems. It demonstrates how an active file system can simplify both applications and system usage by performing computations on behalf of processes. Furthermore, the paper describes the structure of files and directories. The file system is the first component of a suite of system software designed for a collaborative memory system in which intelligent peripheral devices collaborate with a host processor to accomplish tasks. These implementations are similar to the Active Page and Active Disk simulations described in related work. This provides an extensible environment in which a privileged user implements common, time-critical operations within the kernel and an unprivileged user safely implements user-defined operations outside of the kernel.

REFERENCES

Web Links

- https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/11_FileSystemImplementation.html
- https://en.wikipedia.org/wiki/File_system
- http://web.cs.wpi.edu/~cs3013/c07/lectures/Section10-File_Systems.pdf
- http://www.tutorialspoint.com/operating_system/os_file_system.htm
- http://zerofiles.8k.com/fileaccess.html
- https://en.wikipedia.org/wiki/Access_method
- http://www.tutorialspoint.com/operating_system/os_file_system.htm
- http://www.cse.nd.edu/~ssr/papers/linc99/node16.html
- http://shodhganga.inflibnet.ac.in/bitstream/10603/9868/6/conclusion.pdf

Books

[1] Principles of Operating Systems: Design & Applications By Brian L. Stuart

[2] Operating Systems By Sibsankar Haldar, Alex Alagarsamy Aravind