FILES ORGANIZATION AND ACCESS MECHANISM

Files Organization and Access Mechanism:

When a file is used then the stored information in the file must be accessed and read into the memory of a computer system. Various mechanism are provided to access a file from the operating system.

i. Sequential access
ii. Direct Access
iii. Index Access

Sequential Access:
It is the simplest access mechanism, in which informations stored in a file are accessed in an order such that one record is processed after the other. For example editors and compilers usually access files in this manner.

Direct Access:
It is an alternative method for accessing a file, which is based on the disk model of a file, since disk allows random access to any block or record of a file. For this method, a file is viewed as a numbered sequence of blocks or records which are read/written in an arbitrary manner, i.e. there is no restriction on the order of reading or writing. It is well suited for Database management System.

Index Access:
In this method an index is created which contains a key field and pointers to the various block. To find an entry in the file for a key value , we first search the index and then use the pointer to directly access a file and find the desired entry.

With large files , the index file itself may become too large to be keep 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.

File Allocation Method:

1. **Contiguous Allocation**
2. **Linked List Allocation**
3. **Linked List Allocation Using a Table in Memory**
4. **I-Nodes**

Contiguous allocation:

It requires each file to occupy a set of contiguous addresses on a disk. It sore each file as a contiguous run of disk blocks. Thus on a disk with 1-KB blocks, a 50-KB file would be allocated 50 consecutive blocks. Both sequential and direct access is supported by the contiguous allocation method.
Contiguous disk space allocation has two significant advantages.

1. First, i**t is simple to implement** because keeping track of where a file's blocks are is reduced to remembering two numbers: the disk address of the first block and the number of blocks in the file. Given the number of the first block, the number of any other block can be found by a simple addition.
2. Second, **the read performance is excellent** because the entire file can be read from the disk in a single operation. Only one seek is needed (to the first block). After that, no more seeks or rotational delays are needed so data come in at the full bandwidth of the disk.

Thus contiguous allocation is simple to implement and has high performance.
Unfortunately, contiguous allocation also has a major drawback: in time, the disk becomes fragmented, consisting of files and holes. It needs compaction to avoid this.

Example of contiguous allocation: CD and DVD ROMs

Linked List Allocation:

keep each file as a linked list of disk blocks as shown in the fig. The first word of each block is used as a pointer to the next one. The rest of the block is for data.

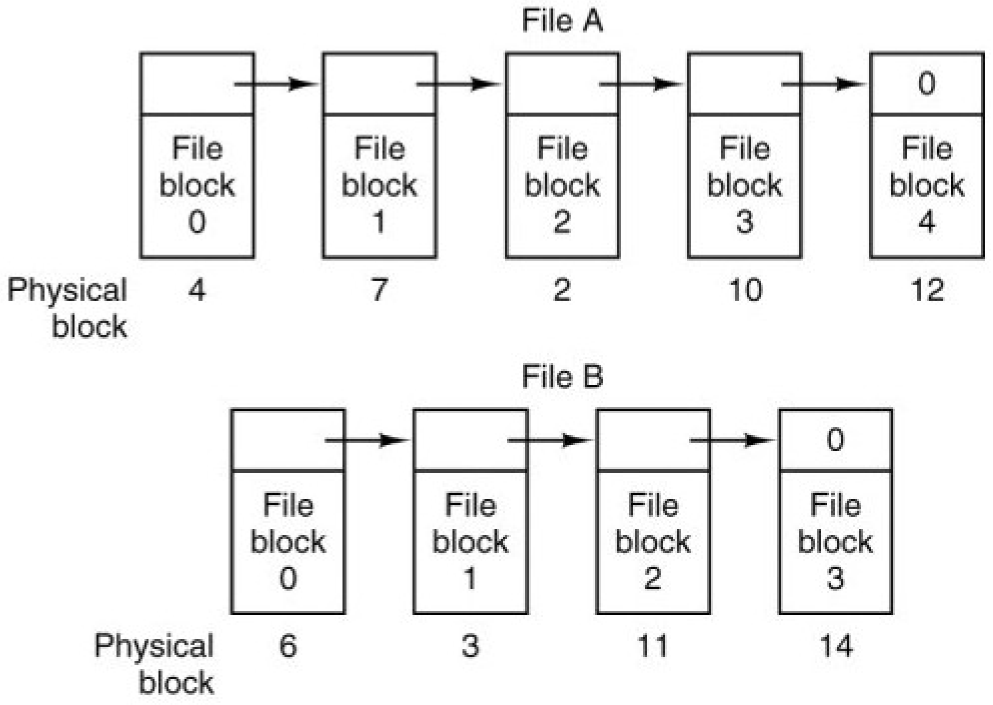


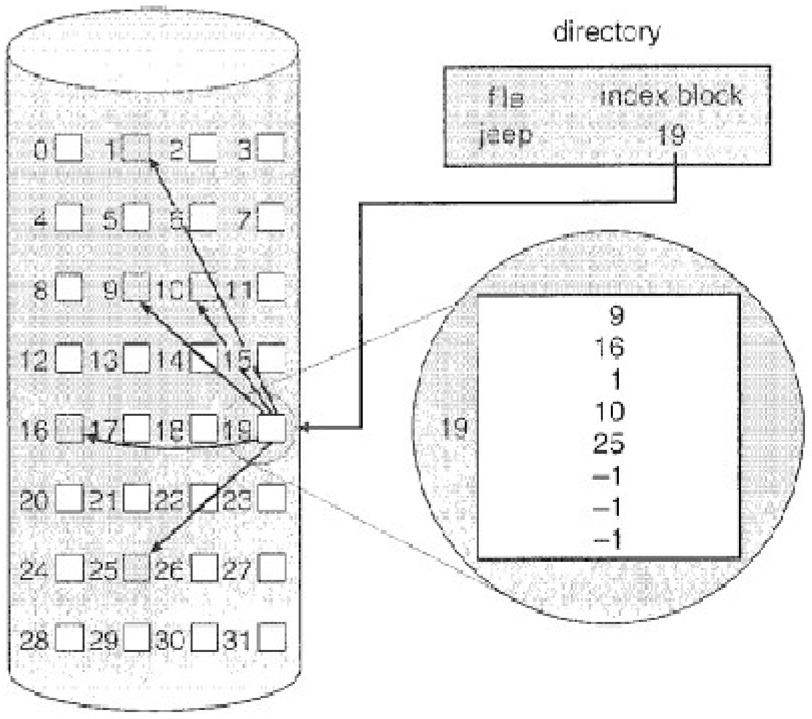
Fig:Storing a file as a linked list of disk blocks.

Unlike contiguous allocation, every disk block can be used in this method. No space is lost to disk fragmentation. The major problem with linked allocation is that it can be used only for sequential access files. To find the i^{th} block of a file, we must start at the beginning of that file, and follow the pointers until we get the i^{th} block. It is inefficient to support direct access capability for linked allocation of files.

Another problem of linked list allocation is reliability. Since the files are linked together with the pointer scattered all over the disk. Consider what will happen if a pointer is lost or damaged.

Indexed allocation (I-Nodes):

It solves the external fragmentation and size declaration problems of contiguous allocation. In this allocation all pointers are brought together into one location called **Index block.**
Each file has its own index block, which is an array of disk-block addresses. The ith entry in the index block points to the ith block of the file. The directory contains the address of the index block.



Inex alloction of Disk sapce

To read the ith block, we use the pointer in the ith index block entry to find and read the desired block. This scheme is similar to the paging scheme.

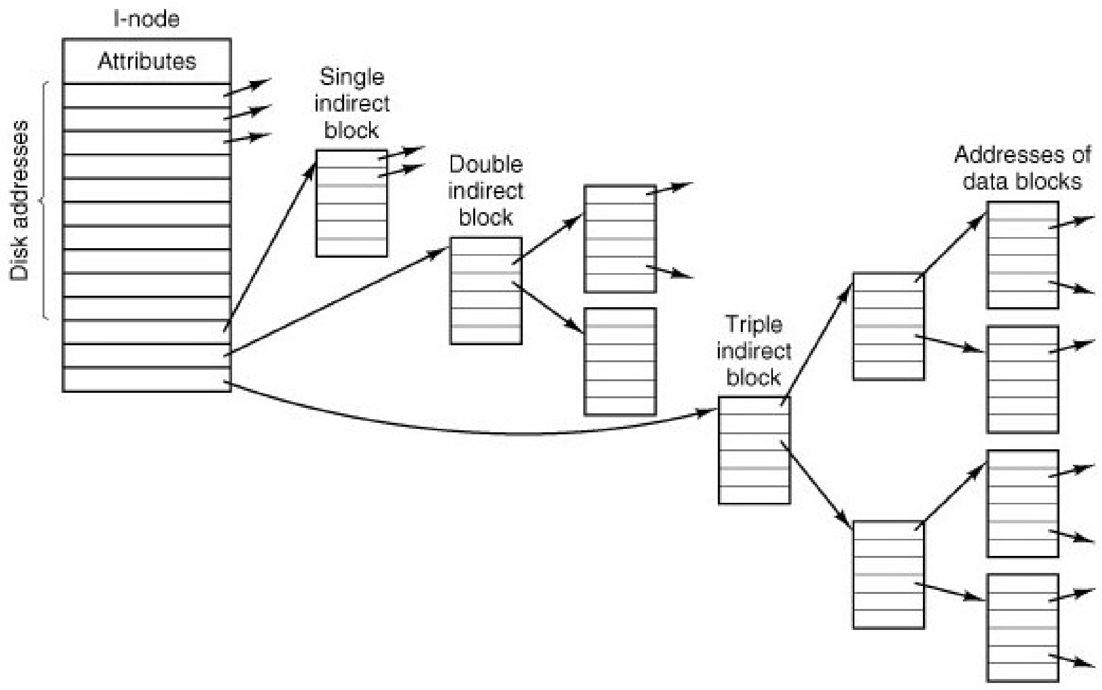


Fig: An i-node with three levels of indirect blocks.

Source : http://dayaramb.files.wordpress.com/2012/02/operating-system-pu.pdf