4.9 DIRECTORY IMPLEMENTATION

1. Linear List

- The simplest method of implementing a directory is to use a linear list of file names with pointer to the data blocks.
- A linear list of directory entries requires a linear search to find a particular entry.
- This method is simple to program but time- consuming to execute. To create a new file, we must first search the but time – consuming to execute.
- The real disadvantage of a linear list of directory entries is the linear search to find a file.

2. Hash Table

- In this method, a linear list stores the directory entries, but a hash data structure is also used.
- The hash table takes a value computed from the file name and returns a pointer to the file name in the linear list.
- Therefore, it can greatly decrease the directory search time.
- Insertion and deleting are also fairly straight forward, although some provision must be made for collisions – situation where two file names hash to the same location.
- The major difficulties with a hash table are its generally fixed size and the dependence of the hash function on that size.

4.10 ALLOCATION METHODS

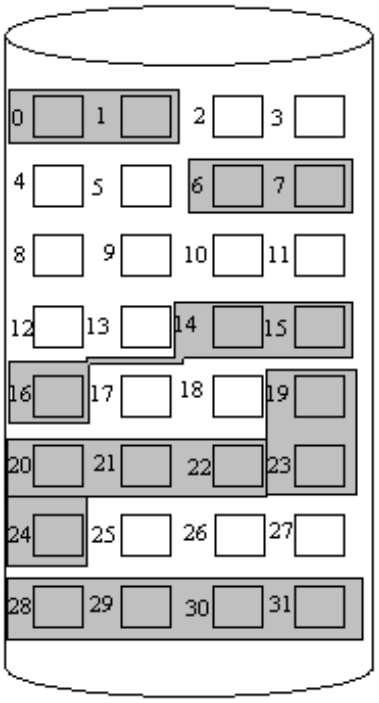
Allocation methods deals with how to allocate space to the files so that disk space is utilized effectively and files can be accessed quickly.

There are there major methods of allocating disk space:

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 blocks on the disk.



Directory

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

- Contiguous allocation of a file is defined by the disk address and length (in block units) 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 allocated for this file.

Disadvantages:

1. Finding space for a new file.

It suffers from the problem of external fragmentation. Storage is fragmented into a number of holes, no one of which is large enough to store the needed data.

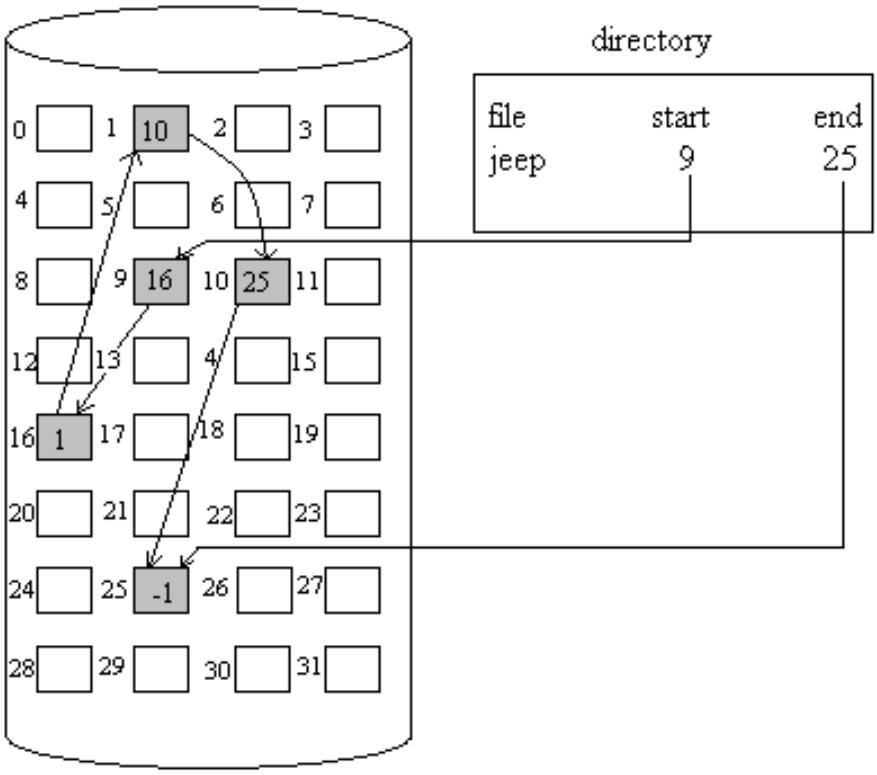
2. Determining how much space is needed for a file.

- When the file is created, the total amount of space it will need must be specified. It is not possible for all files.
- Even if the total amount of space needed for a file is known in advance pre-allocation may be inefficient.
- A file that grows slowly over a long period (months or years) must be allocated enough space for its final size therefore has an internal fragmentation.

To overcome these disadvantages: Use a modified contiguous allocation scheme, in which a contiguous chunk of space called as an **extent** is allocated initially and then, when that amount is not large enough another chunk of contiguous space an extent is added to the initial allocation.

2. Linked Allocation

- Linked allocation solves all problems of contiguous allocation.
- With linked allocation, each file is a linked list of disk blocks, the disk blocks may be scattered anywhere on the disk.
- The directory contains a pointer to the first and last blocks of the file. For example, a file of five blocks might start at block 9, continue at block 16, then block 1, block 10, and finally bock 25.
- Each block contains a pointer to the next block. These pointers are not made available to the user.
- There is no external fragmentation with linked allocation, and any free block on the free space list can be used to satisfy a request.
- The size of a file does not need to be declared when that file is created.



Disadvantages:

1. Used effectively only for sequential access files.

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. It is inefficient to support a direct- access capability for linked allocation files.

2. Space required for the pointers

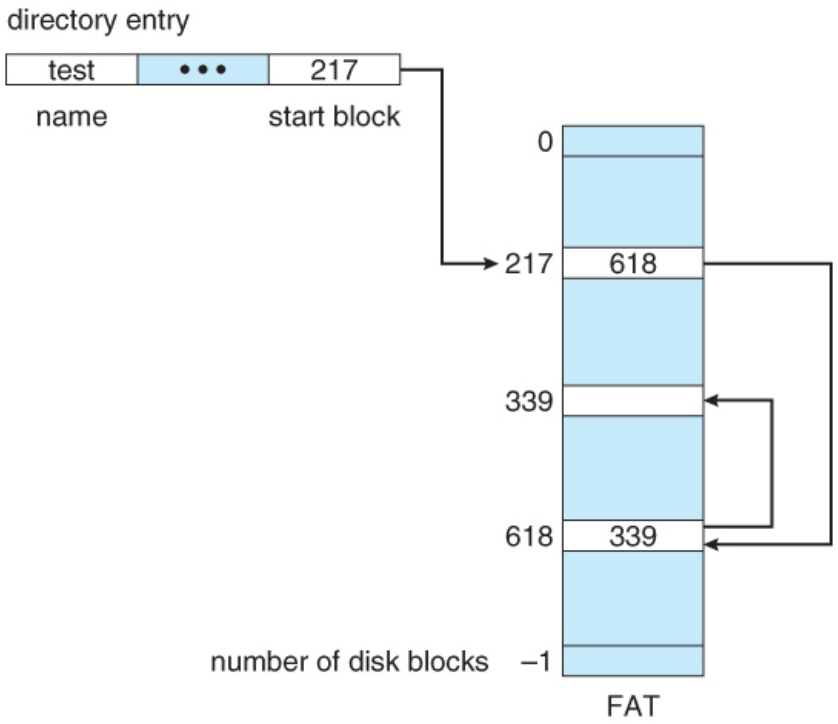
If a pointer requires 4 bytes out of a 512-byte block, then 0.78 percent of the disk is being used for pointers, rather than for information. Solution to this problem is to collect blocks into multiples, called **clusters**, and to allocate the clusters rather than blocks.

3. Reliability

Since the files are linked together by pointers scattered all over the disk hardware failure might result in picking up the wrong pointer. This error could result in linking into the free-space list or into another file.

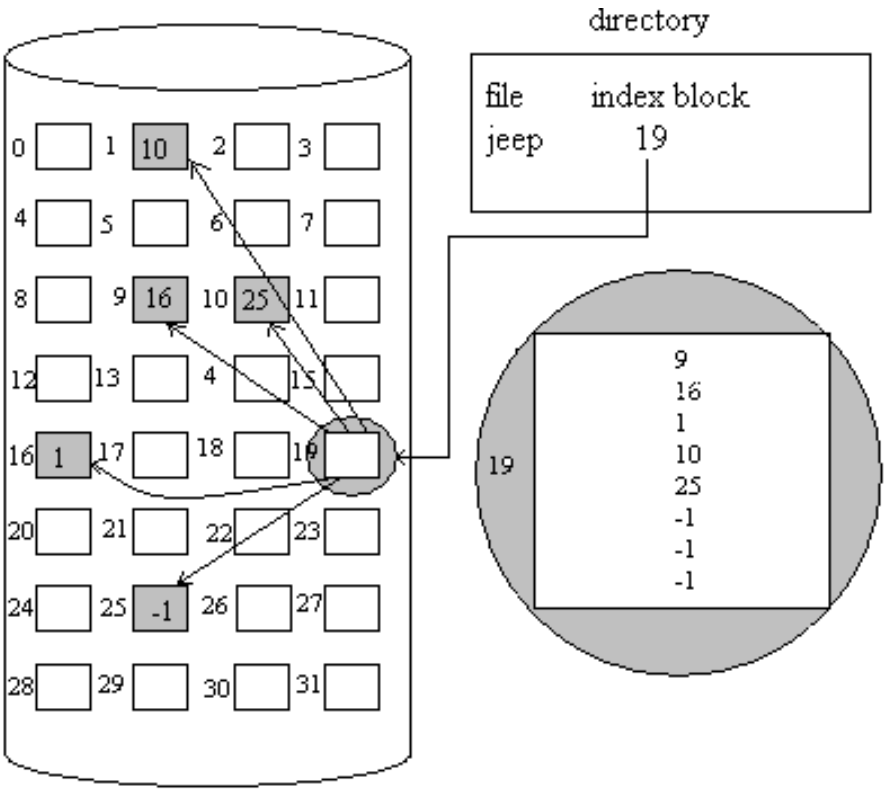
File Allocation Table(FAT)

- An important variation on the linked allocation method is the use of a file allocation table(FAT).
- This simple but efficient method of disk- space allocation is used by the MS-DOS and OS/2 operating systems.
- A section of disk at beginning of each partition is set aside to contain the table.
- The table has entry for each disk block, and is indexed by block number.
- The FAT is much as is a linked list.
- The directory entry contains the starting block number of the file.
- The table entry indexed by that block number contains the block number of the next block in the file.
- This chain continues until the last block which has a special end – of – file value as the table entry.
- Unused blocks are indicated by a 0 table value.
- Allocating a new block file is a simple matter of finding the first 0 – valued table entry, and replacing the previous end of file value with the address of the new block.
- Example the FAT structure for a file consisting of disk blocks 217,618, and 339.



3. Indexed Allocation

- Linked allocation solves the external – fragmentation and size- declaration problems of contiguous allocation.
- Linked allocation cannot support efficient direct access, since the pointers to the blocks are scattered.
- Indexed allocation solves this problem by bringing all the pointers together into one location: the **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 .
- 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.



Disadvantages

1.Pointer Overhead

Indexed allocation suffers from wasted space. The pointer over head of the index block is generally greater than the pointer over head of linked allocation.

2. Size of Index block

If the index block is too small, however, it will not be able to hold enough pointers for a large file.

Linked Scheme: An index block is normally one disk block. Thus, it can be read and written directly by itself. To allow for large files, we may link together several index blocks.

Multilevel index: A variant of the linked representation is to use a first level index block to point to a set of second – level index blocks.

Combined scheme:

- Another alternative, used in the UFS, is to keep the first, say, 15 pointers of the index block in the file's inode.
- The first 12 of these pointers point to direct blocks; that is for small (no more than 12 blocks) files do not need a separate index block
- The next pointer is the address of a single indirect block.

- The single indirect block is an index block, containing not data, but rather the addresses of blocks that do contain data.
- Then there is a double indirect block pointer, which contains the address of a block that contains pointers to the actual data blocks. The last pointer would contain pointers to the actual data blocks.
- The last pointer would contain the address of a triple indirect block.

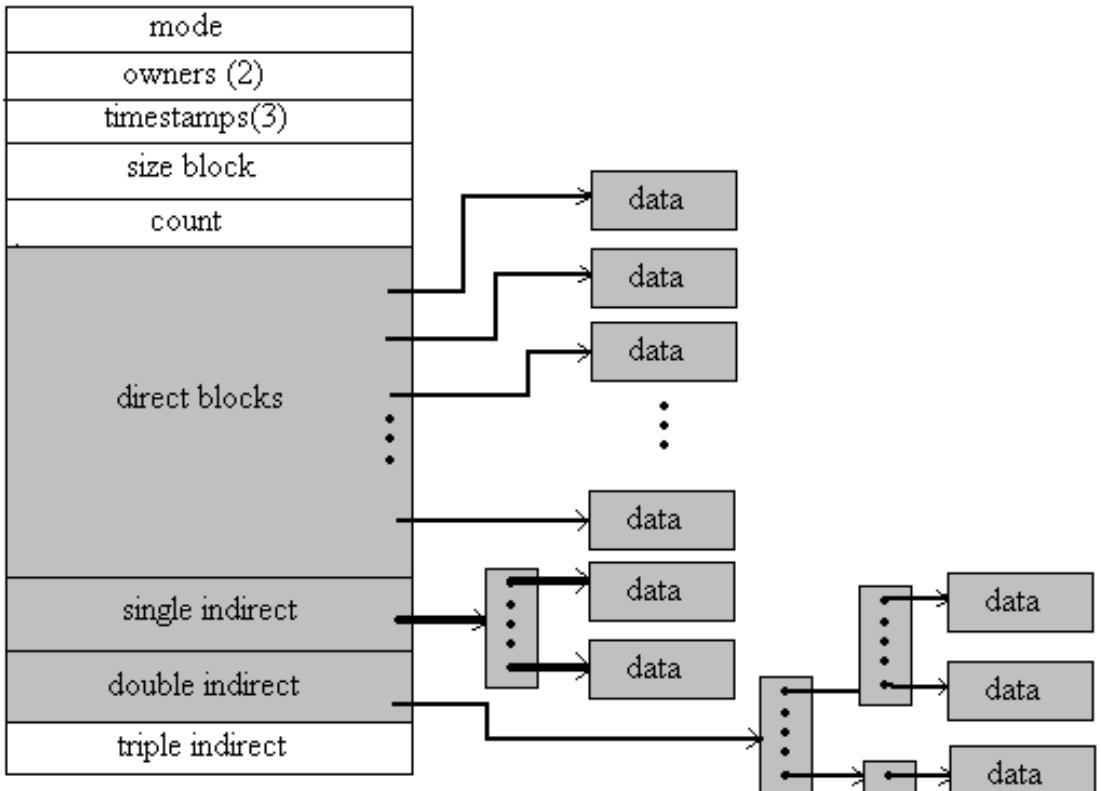


Fig 5.8 The UNIX inode