File allocation methods

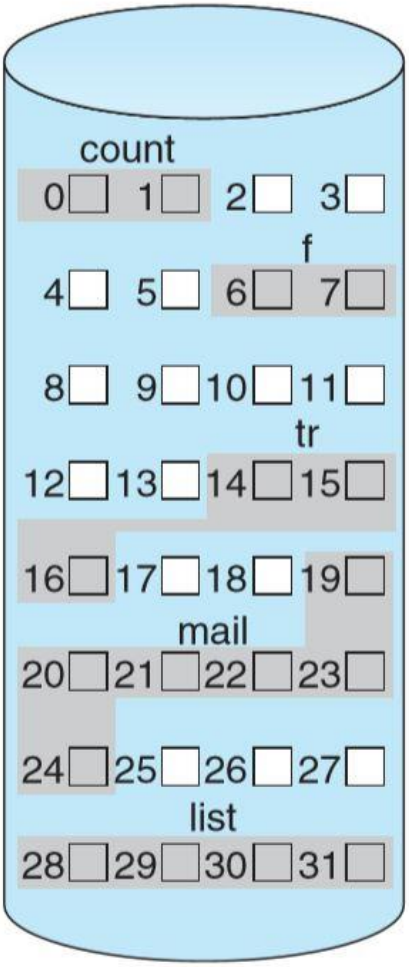
The main aim of file allocation problem is how disk space is utilized effectively and files can be accessed quickly. Three major methods of allocating disk space are in wide use:

- **Contiguous**
- **Linked**
- **Indexed**

Each method has advantages and disadvantages. Although some systems support all three, it is more common for a system to use one method for all files.

Contiguous Allocation

- Contiguous allocation requires that each file occupy a set of contiguous blocks on the disk. Disk addresses define a linear ordering on the disk. With this ordering, assuming that only one job is accessing the disk, accessing block b + 1 after block b normally requires no head movement.



directory

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

- Advantage
  - Accessing a file that has been allocated contiguously is easy. Thus, both sequential and direct access can be supported by contiguous allocation.
  -
- Disadvantage
  - Suffer from the problem of external fragmentation.
  - Suffer from huge amount of external fragmentation.
  - Another problem with contiguous allocation file modification

Q Suppose there are six files F1, F2, F3, F4, F5, F6 with corresponding sizes 150 KB, 225 KB, 75 KB, 60 KB, 275 KB and 65 KB respectively. The files are to be stored on a sequential device in such a way that optimizes access time. In what order should the files be stored? **(NET-NOV-2017)**

A) F5, F2, F1, F3, F6, F4

b) F4, F6, F3, F1, F2, F5

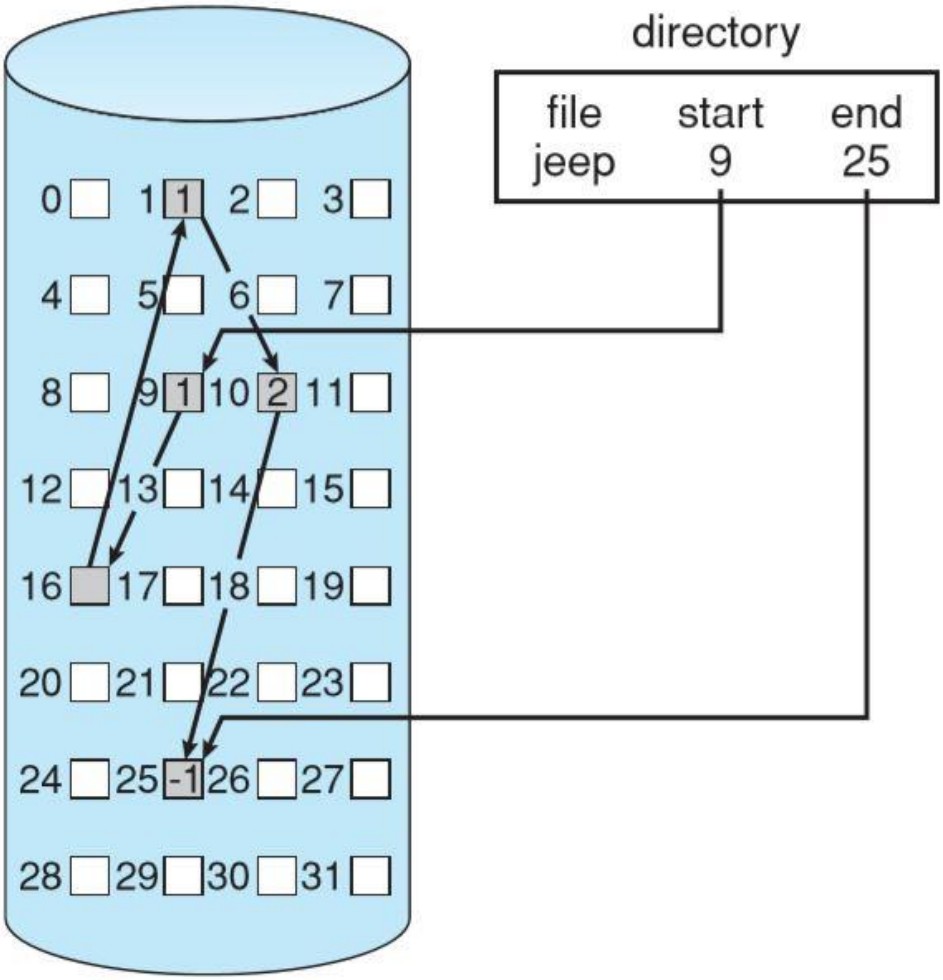
c) F1, F2, F3, F4, F5, F6

d) F6, F5, F4, F3, F2, F1

Answer: (B)

Linked Allocation

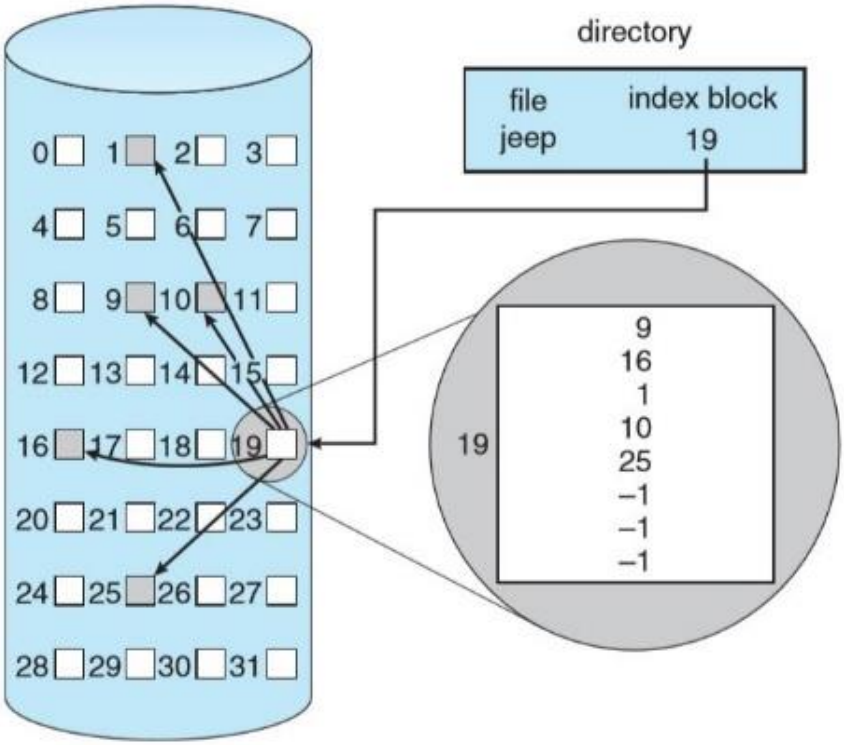
- Linked allocation solves all problems of contiguous allocation. With linked allocation, each file is a linked list of disk blocks; the disk blocks may be scattered anywhere on the disk. The directory contains a pointer to the first and last blocks of the file.

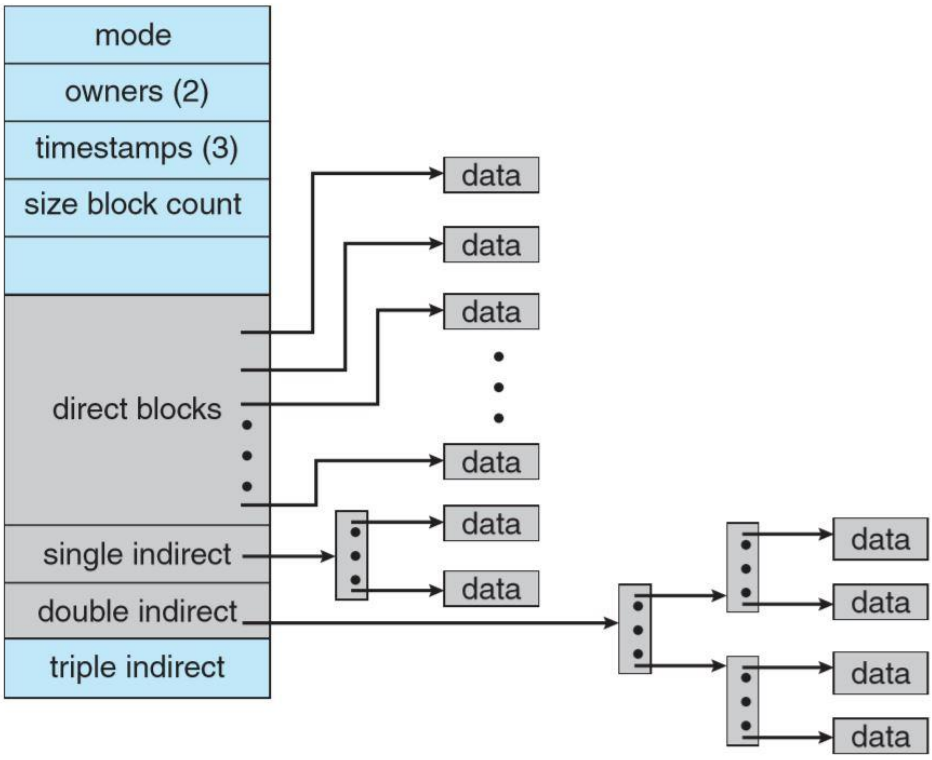


- Advantage: -
    - To create a new file, we simply create a new entry in the directory. With linked allocation, each directory entry has a pointer to the first disk block of the file. To read a file, we simply read blocks by following the pointers from block to block. The size of a file need not be declared when the file is created. A file can continue to grow as long as free blocks are available.
    - There is no external fragmentation with linked allocation, and any free block on the free-space list can be used to satisfy a request.
- **Disadvantage: -**
    - 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 to the i^{th} block. Each access to a pointer requires a disk read.
    - Another disadvantage is the space required for the pointers, so each file requires slightly more space than it would otherwise.
    - Yet another problem is reliability. Recall that the files are linked together by pointers scattered all over the disk, and consider what would happen if a pointer were lost or damaged. One partial solution is to use doubly linked lists, and another is to store the file name and relative block number in each block. However, these schemes require even more overhead for each file.

Indexed Allocation

- Indexed allocation solves problems of contiguous and linked allocation, 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 i^{th} entry in the index block points to the ith block of the file. The directory entry contains the address of the index block. To find and read the ith block, we use the pointer in the ith index-block entry.
- When the file is created, all pointers in the index block are set to null. When the ith block is first written, a block is obtained from the free-space manager, and its address is put in the ith index-block entry.
- This point raises the question of how large the index block should be. Every file must have an index block, so we want the index block to be as small as possible. If the index block is too small, however, it will not be able to hold enough pointers for a large file.
- Linked scheme: To allow for large files, we can link together several index blocks. For example, an index block might contain a small header giving the name of the file and a set of the first 100 disk-block addresses. The next address (the last word in the index block) is null (for a small file) or is a pointer to another index block (for a large file).
- Multilevel index. A variant of linked representation uses a first-level index block to point to a set of second-level index blocks, which in turn point to the file blocks. To access a block, the operating system uses the first-level index to find a second-level index block and then uses that block to find the desired data block. This approach could be continued to a third or fourth level, depending on the desired maximum file size.
- Combined scheme. Another alternative, used in UNIX-based file systems, 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, they contain addresses of blocks that contain data of the file. Thus, the data for small files do not need a separate index block. The next three pointers point to indirect blocks. The first points to a single indirect block, which is an index block containing not data but the addresses of blocks that do contain data. The second points to a double indirect block, which contains the address of a block that contains the addresses of blocks that contain pointers to the actual data blocks. The last pointer contains the address of a triple indirect block.

- Advantage
  - Indexed allocation supports direct access, without suffering from external fragmentation, because any free block on the disk can satisfy a request for more space.
- Disadvantage
  - Indexed allocation does suffer from wasted space, however. The pointer overhead of the index block is generally greater than the pointer overhead of linked allocation.

Q In a file allocation system, which of the following allocation scheme(s) can be used if no external fragmentation is allowed? **(GATE-2017) (1 Marks)**

I. Contiguous **II.** Linked **III.** Indexed

(a) I and III only **(b)** II only **(c)** III only **(d)** II and III only

Ans: d

Q Consider a file currently consisting of 50 blocks. Assume that the file control block and the index block is already in memory. If a block is added at the end (and the block information to be added is stored in memory), then how many disk I/O operations are required for indexed (single-level) allocation strategy? **(NET-NOV-2016)**

A) 1 **b)** 101 **c)** 27 **d)** 0

Ans: a

Q Match the following: **(NET-JUNE-2014)**

List-I	List-II
a) **Contiguous allocation**	**i)** This scheme supports very large file sizes.
b) **Linked allocation**	**ii)** This allocation technique supports only sequential files.
c) **Indexed allocation**	**iii)** Number of disks required to access file is minimal.
d) **Multi-level indexed**	**iv)** This technique suffers from maximum wastage of space in storing pointers.

Codes:

	(a)	(b)	(c)	(d)
(a)	(iii)	(iv)	(ii)	(i)
(b)	(iii)	(ii)	(iv)	(i)
(c)	(i)	(ii)	(iv)	(iii)
(d)	(i)	(iv)	(ii)	(iii)

Answer: (B)

Q The index node (inode) of a Unix-like file system has 12 direct, one single-indirect and one double-indirect pointers. The disk block size is 4 kB, and the disk block address is 32-bits long. The maximum possible file size is (rounded off to 1 decimal place) _____ GB. **(GATE-2014) (2 Marks)**

Answer: 4

Q A file system with 300 Gbyte disk uses a file descriptor with 8 direct block addresses, 1 indirect block address and 1 doubly indirect block address. The size of each disk block is 128 Bytes and the size of each disk block address is 8 Bytes. The maximum possible file size in this file system is **(GATE-2012) (2 Marks)**

(A) 3 Kbytes
(B) 35 Kbytes
(C) 280 Bytes
(D) Dependent on the size of the disk

Answer: (B)

Q The data blocks of a very large file in the Unix file system are allocated using **(GATE-2008) (1 Marks)**

(A) contiguous allocation
(B) linked allocation
(C) indexed allocation
(D) an extension of indexed allocation

Answer: (D)

Q A Unix-style i-node has 10 direct pointers and one single, one double and one triple indirect pointer. Disk block size is 1 Kbyte, disk block address is 32 bits, and 48-bit integers are used. What is the maximum possible file size? **(GATE-2004) (2 Marks)**

(A) 2^{24} bytes **(B)** 2^{32} bytes **(C)** 2^{34} bytes **(D)** 2^{48} bytes

Answer: (C)

Q In the index allocation scheme of blocks to a file, the maximum possible size of the file depends on **(GATE-2002) (1 Marks)**

(a) the size of the blocks, and the size of the address of the blocks.

(b) the number of blocks used for the index, and the size of the blocks.

(c) the size of the blocks, the number of blocks used for the index, and the size of the address of the blocks.

(d) None of the above

Answer: (C)

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 disk 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. To create a file, we search the free-space list for the required amount of space and allocate that space to the new file. This space is then removed from the free-space list. When a file is deleted, its disk space is added to the free-space list.

Q How many disk blocks are required to keep list of free disk blocks in a 16 GB hard disk with 1 kB block size using linked list of free disk blocks? Assume that the disk block number is stored in 32 bits. **(NET-DEC-2014)**

(A) 1024 blocks **(B)** 16794 blocks **(C)** 20000 blocks **(D)** 1048576 blocks

Bit Vector

- Frequently, the free-space list is implemented as a bit map or bit vector. Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.
- For example, consider a disk where blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, and 27 are free and the rest of the blocks are allocated. The free-space bit map would be 001111001111110001100000011100000 ...
- The main advantage of this approach is its relative simplicity and its efficiency in finding the first free block or n consecutive free blocks on the disk.
- Unfortunately, bit vectors are inefficient unless the entire vector is kept in main memory. Keeping it in main memory is possible for smaller disks but not necessarily for larger ones.
- A 1.3-GB disk with 512-byte blocks would need a bit map of over 332 KB to track its free blocks.
- A 1-TB disk with 4-KB blocks requires 256 MB to store its bit map. Given that disk size constantly increases, the problem with bit vectors will continue to escalate as well.

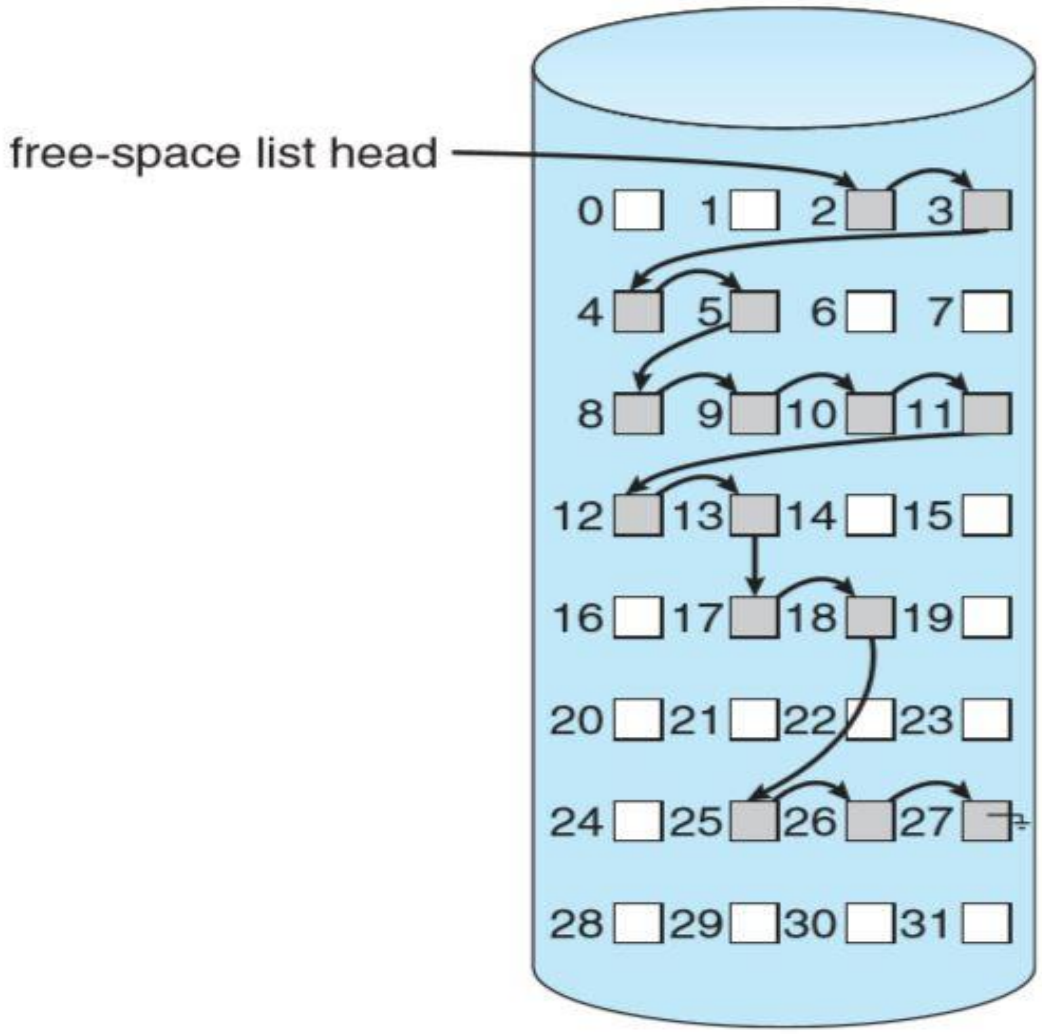
Q How much space will be required to store the bit map of a 1.3 GB disk with 512 bytes block size? (**NET-DEC-2013)**

(A) 332.8 KB **(B)** 83.6 KB **(C)** 266.2 KB **(D)** 256.6 KB

Ans: a

Linked List

- Another approach to free-space management is to link together all the free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory.
- This first block contains a pointer to the next free disk block, and so on. Recall our earlier example, in which blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, and 27 were free and the rest of the blocks were allocated.
- In this situation, 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, which would point to block 8, and so on.
- This scheme is not efficient; to traverse the list, we must read each block, which requires substantial I/O time.



- Fortunately, however, traversing the free list is not a frequent action. Usually, the operating system simply needs a free block so that it can allocate that block to a file, so the first block in the free list is used.

Q A FAT (file allocation table) based file system is being used and the total overhead of each entry in the FAT is 4 bytes in size. Given a 100 x 10^6 bytes disk on which the file system is stored and data block size is 10^3 bytes, the maximum size of a file that can be stored on this disk in units of 10^6 bytes is _____. **(GATE-2014) (2 Marks)**
Answer: 99.55 to 99.65

Q An experimental file server is up 75% of the time and down for 25% of the time due to bugs. How many times does this file server have to be replicated to give an availability of at least 99%? **(NET-NOV-2016)**
a) 2 **b)** 4 **c)** 8 **d)** 16