Q.1 Explain File Access Method.

Ans

File: A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

File Access Method:

The way that files are accessed and read into memory is determined by Access methods. Usually a single access method is supported by systems while there are OS's that support multiple access methods.

There are several ways to access files –

1. Sequential Access
2. Direct Access
3. Indexed Sequential Access

1. Sequential Access

- Data is accessed one record right after another is an order.
- Read command cause a pointer to be moved ahead by one.
- Write command allocate space for the record and move the pointer to the new End Of File.
- Such a method is reasonable for tape.

2. Direct Access

- This method is useful for disks.
- The file is viewed as a numbered sequence of blocks or records.
- There are no restrictions on which blocks are read/written, it can be dobe in any order.
- User now says "read n" rather than "read next".
- "n" is a number relative to the beginning of file, not relative to an absolute physical disk location.

3. Indexed Sequential Access

- It is built on top of Sequential access.
- It uses an Index to control the pointer while accessing files.

Q2. Explain File Allocation method.

Ans

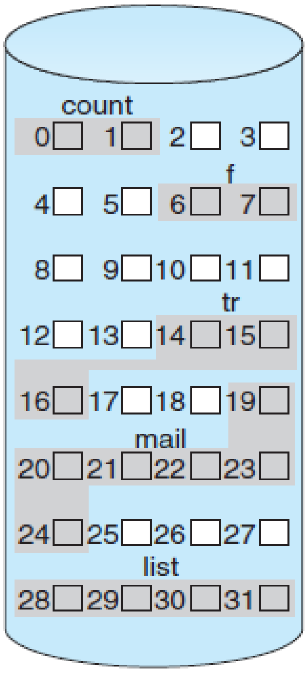
File Allocation Method:

Files are allocated disk spaces by operating system. Operating systems deploy following three main ways to allocate disk space to files

1. Contiguous Allocation
2. Linked Allocation
3. Indexed Allocation

Contiguous Allocation

- Each file occupies a contiguous address space on disk.
- Assigned disk address is in linear order.
- Easy to implement.
- External fragmentation is a major issue with this type of allocation technique.



directory

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

Figure 12.5 Contiguous allocation of disk space.

Linked Allocation

- Each file carries a list of links to disk blocks.
- Directory contains link / pointer to first block of a file.
- No external fragmentation
- Effectively used in sequential access file.
- Inefficient in case of direct access file.

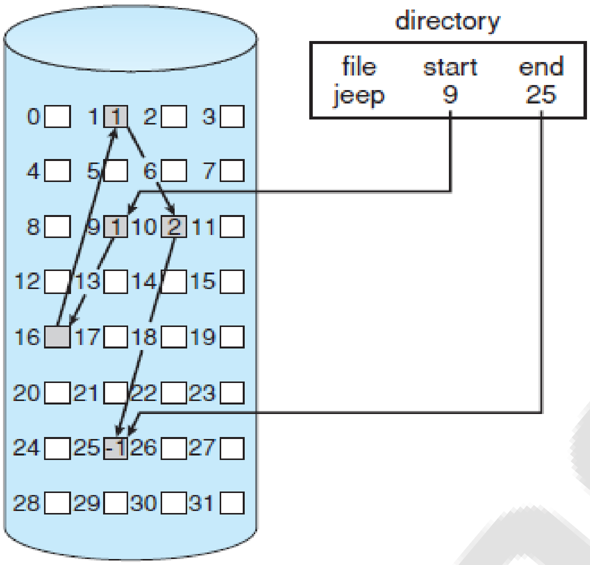


Figure 12.6 Linked allocation of disk space.

Indexed Allocation

- Provides solutions to problems of contiguous and linked allocation.
- A index block is created having all pointers to files.
- Each file has its own index block which stores the addresses of disk space occupied by the file.
- Directory contains the addresses of index blocks of files.

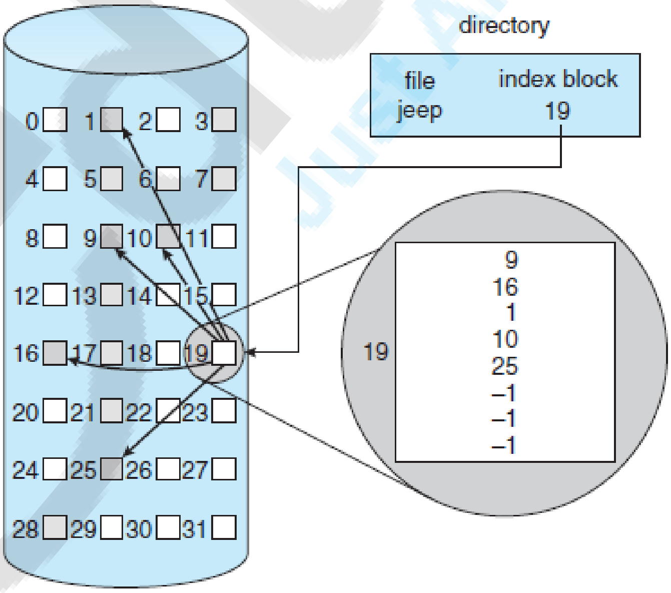
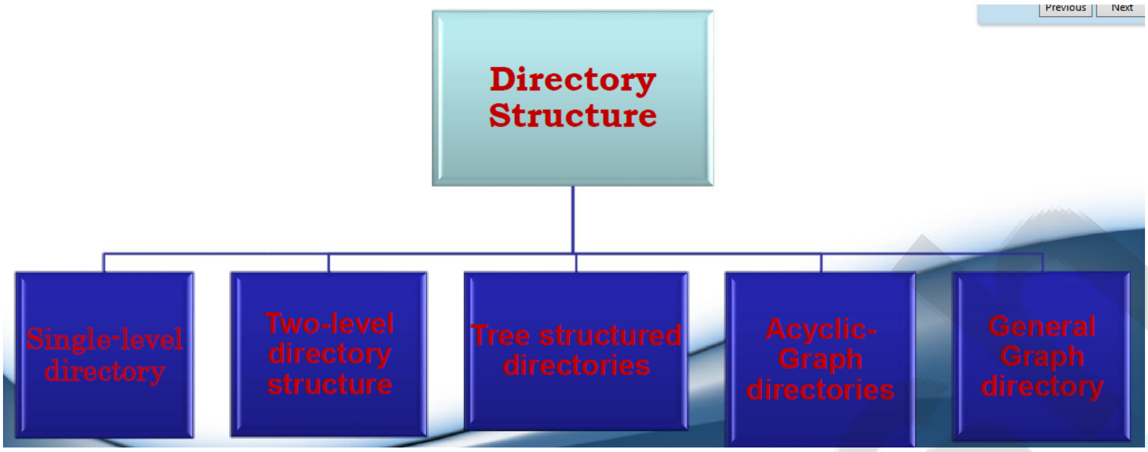


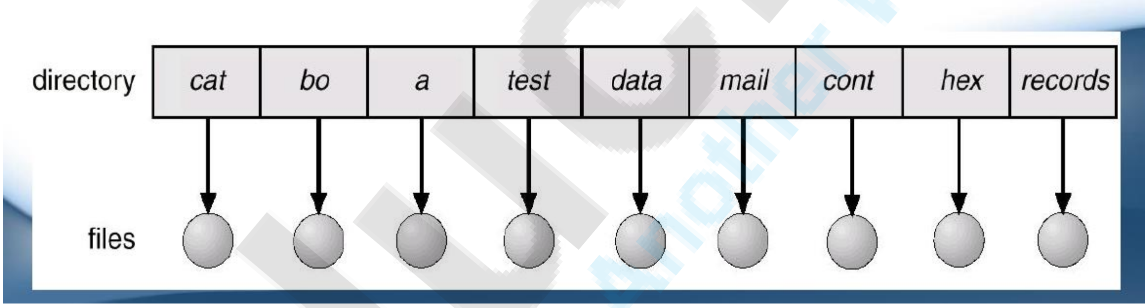
Figure 12.8 Indexed allocation of disk space.

Q.3	What is Directory?
Ans.	**Directory:** Information about files is maintained by Directories. A directory can contain multiple files. It can even have directories inside of them. In Windows we also call these directories as folders. Following is the information maintained in a directory : • Name : The name visible to user. • Type : Type of the directory. • Location : Device and location on the device where the file header is located. • Size : Number of bytes/words/blocks in the file. • Position : Current next-read/next-write pointers. • Protection : Access control on read/write/execute/delete. • Usage : Time of creation, access, modification etc. • Mounting : When the root of one file system is "grafted" into the existing tree of another file system its called Mounting.
Q.4	Directory structure or Types of Directory
	Directory systems :- **Directory structure:** The file system of computers can be extensive. Some systems store thousands of file on disk. To manage all these data, we need to organize them. The organization is done in 2 steps. The file system is broken into partitions. Each partition contains information about file within it. **Operation on a directory:** • **Search for a file:** We need to be able to search a directory for a particular file. • **Create a file:** New files are created & added to the directory. • **Delete a file:** When a file is no longer needed, we may remove it from the directory. • **List a directory:** We should be able to list the files of the directory. • **Rename a file:** The name of a file is changed when the contents of the file changes. • **Traverse the file system:** It is useful to be able to access every directory & every file within a directory. **Structure of a directory:** The most common schemes for defining the structure of the directory are:


Previous
Next
Directory Structure
Single-level directory
Two-level directory structure
Tree structured directories
Acyclic-Graph directories
General Graph directory

1. **Single level directory:** It is the simplest directory structure. All files are present in the same
directory. So it is easy to manage & understand.

Limitation: A single level directory is difficult to manage when the no. of files increases or when there is more than one user. Since all files are in same directory, they must have unique names. So, there is confusion of file names between different users.



2. **Two level directories:**

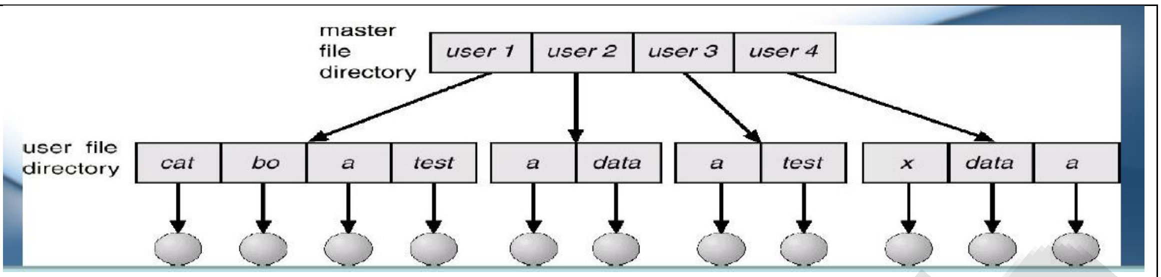
The solution to the name collision problem in single level directory is to create a separate directory for each user. In a two level directory structure, each user has its own user file directory.

When a user logs in, then master file directory is searched. It is indexed by user name & each entry points to the UFD of that user.

Limitation:

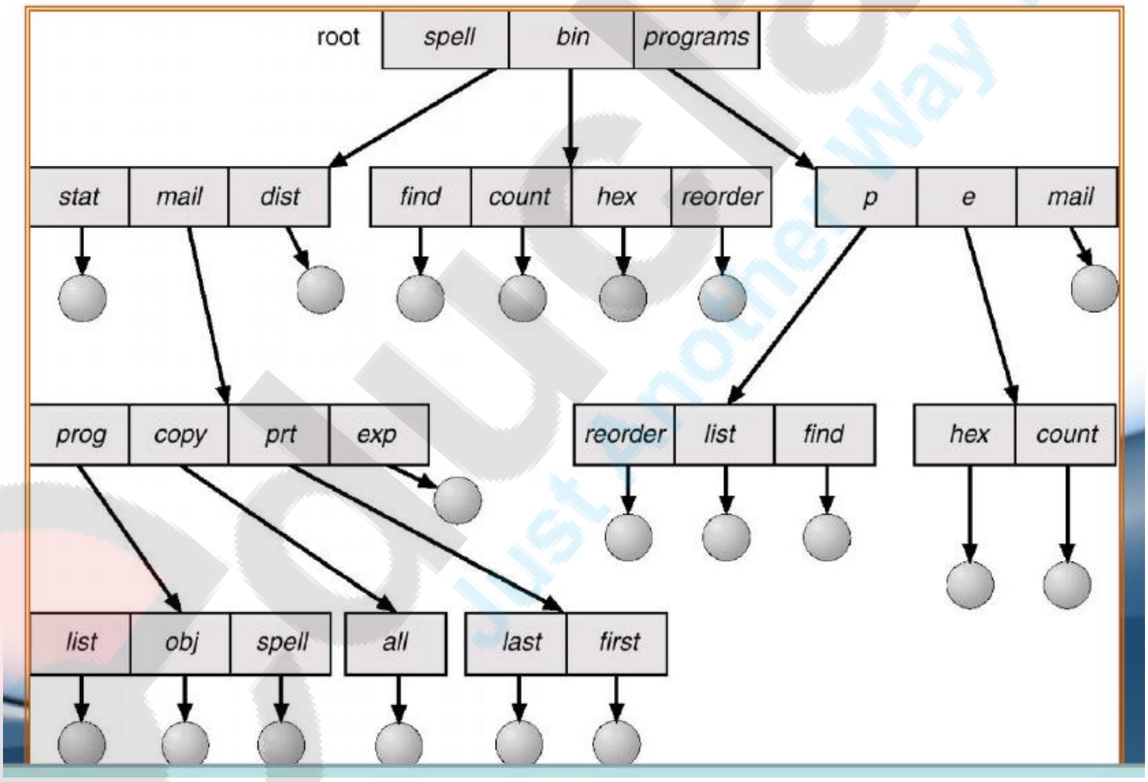
It solves name collision problem. But it isolates one user from another.

It is an
advantage when users are completely independent. But it is a disadvantage when the users need to access each other’s files & co-operate among themselves on a particular task.



4. Tree structured directories:

- It is the most common directory structure. A two level directory is a two level tree. So, the generalization is to extend the directory structure to a tree of arbitrary height.
- It allows users to create their own subdirectories & organize their files. Every file in the
- system has a unique path name.
- It is the path from the root through all the sub-directories to a specified file. A directory is simply another file but it is treated in a special way. One bit in each directory entry defines the entry as a file (O) or as sub- directories.



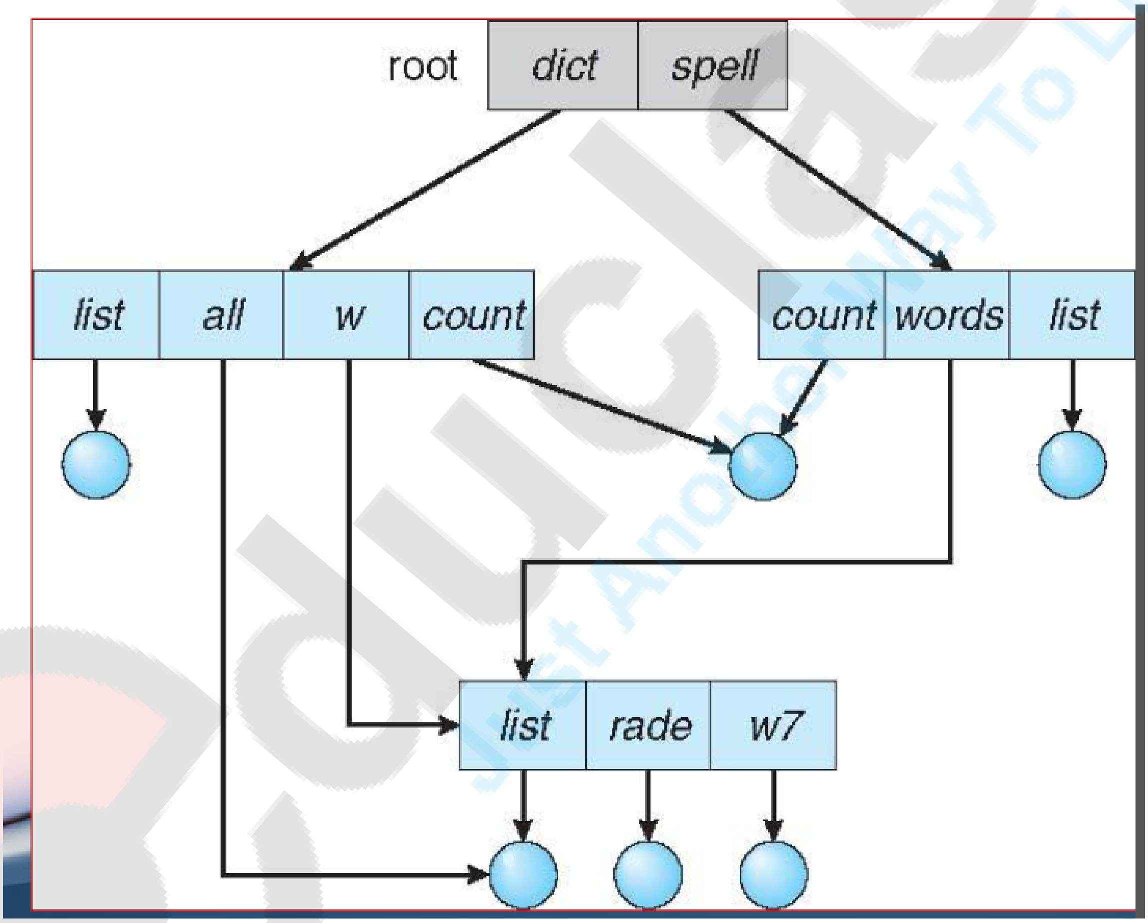
5. A cyclic graph directory:

- It is a generalization of tree structured directory scheme.
- An a cyclic graph allows directories to have shared sub-directories & files.
- A shared directory or file is not the same as two copies of a file. Here a programmer can view the copy but the changes made in the file by one programmer are not reflected in the other's copy.
- But in a shared file, there is only one actual file. So many changes made by a person would be immediately visible to others.

- This scheme is useful in a situation where several people are working as a team. So, here all the
- files that are to be shared are put together in one directory.

Limitation:

Now a file may have multiple absolute path names. So, distinct file names may refer to the same file. Another problem occurs during deletion of a shared file. When a file is removed by any one user. It may leave dangling pointer to the non existing file. One serious problem in a cyclic graph structure is ensuring that there are no cycles. To avoid these problems, some
systems do not allow shared directories or files. E.g. MS-DOS uses a tree structure rather than a cyclic to avoid the problems associated with deletion. One approach for deletion is to preserve the file until all references to it are deleted. To implement this approach, we must have some mechanism for determining the last reference to the file. For this we have to keep a list of reference to a file. But due to the large size of the no. of references. When the count is zero, the file can be deleted.

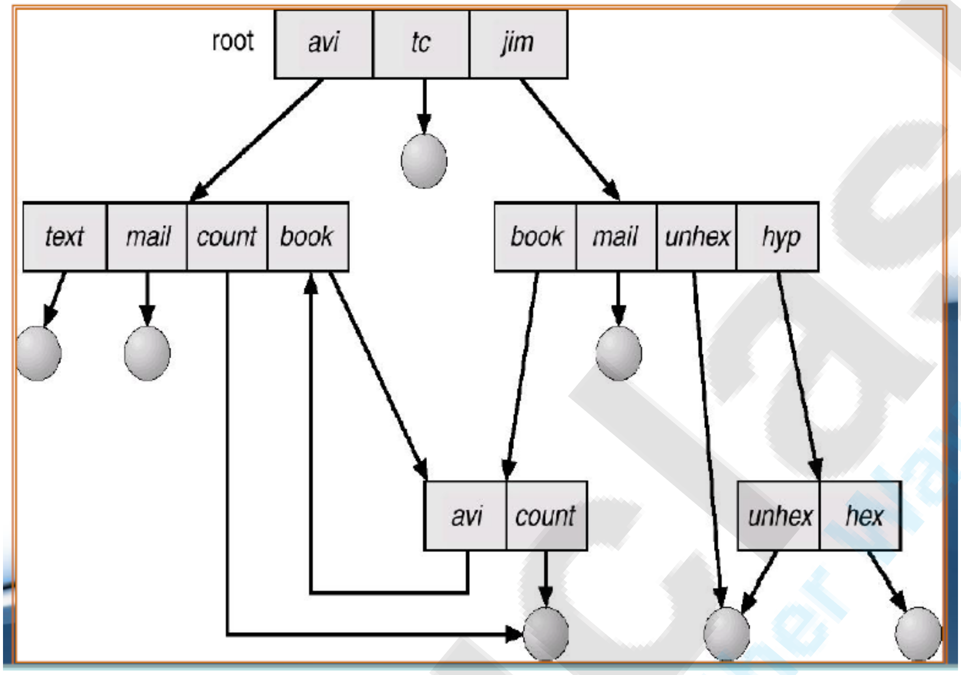


5. **General graph directory:** When links are added to an existing tree structured directory, the
tree structure is destroyed, resulting in a simple graph structure. Linking is a technique that allows a file to appear in more than one directory. The advantage is the simplicity of algorithm to
transverse the graph & determines when there are no more references to a file. But a similar problem exists when we are trying to determine when a file can be deleted. Here also a value zero in the reference count means that there are no more references to the file or directory & the
file can be deleted. But when cycle exists, the reference count may be non-zero even when there

are no references to the directory or file. This occurs due to the possibility of self referencing (cycle) in the structure. So, here we have to use garbage collection scheme to determine when the last references to a file has been deleted & the space can be reallocated. It involves two steps:

- Transverse the entire file system & mark everything that can be accessed.
- Everything that isn't marked is added to the list of free space.

But this process is extremely time consuming. It is only necessary due to presence of cycles in the graph. So, a cyclic graph structure is easier to work than this.



Q.5 Swap-space Management

Ans.

Free Space Management:-

Since there is only a limited amount of disk space, it is necessary to reuse the space from the deleted files. To keep track of free disk space, the system maintains a free space list. It records all the disk blocks that are free i.e. not allocated to some file or dictionary. To create a file, we search the free space list for the required amount of space and allocate it 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.

Implementation:

There are 4 ways to implement the free space list such as:

- **Bit Vector:** The free space list is implemented as a bit map or bit vector. Each block is represented as 1 bit. If the block is free, the bit is 1 and if it is allocated then 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 & 27 are free and rest of the blocks are allocated. The free space bit map would be 0011110011111100011000000111……………………..

The main advantage of this approach is that it is simple and efficient to find the first free block or n consecutive free blocks on the disk. But bit vectors are inefficient unless the entire vector is kept in main memory. It is possible for smaller disks but not for larger ones.

- **Linked List:** Another approach is to link together all the free disk blocks and keep a pointer to the first free block. The first free block contains a pointer to the next free block and so on. For example, we keep a pointer to block 2 as the free block. Block 2 contains a pointer to block which points to block 4 which then points to block 5 and so on. But this scheme is not efficient.

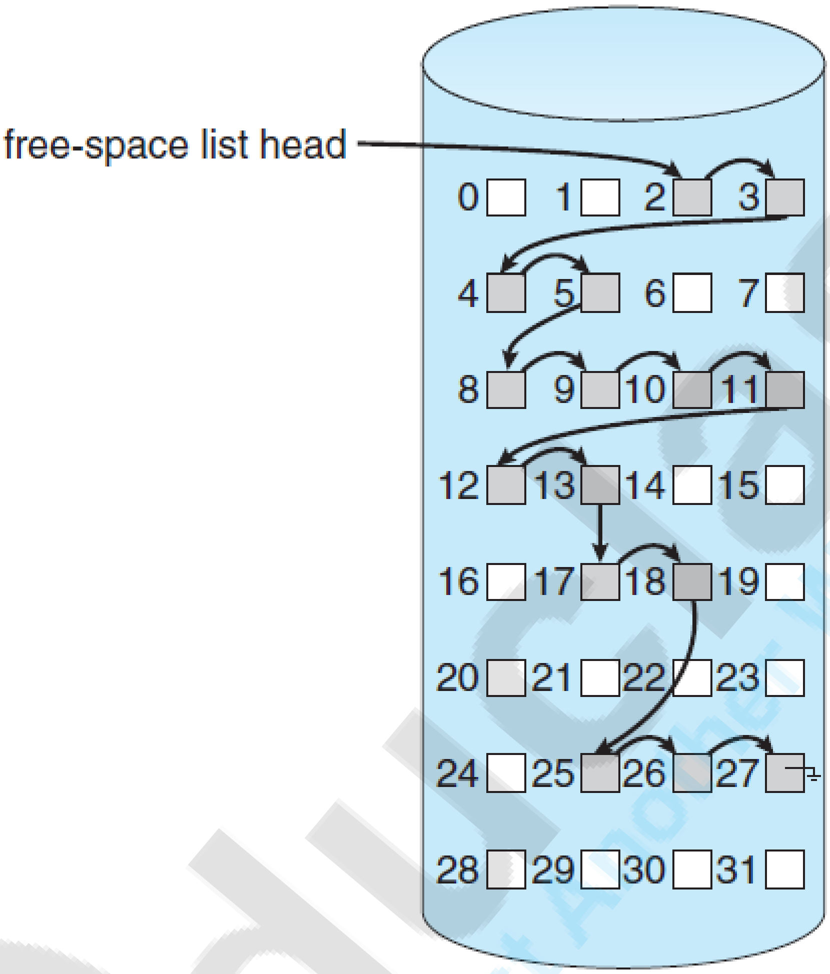


Figure 12.10 Linked free-space list on disk.

To traverse the list, we must read each block which require a lot of I/O time.

- **Grouping:** In this approach, we store the address of n free blocks in the first free block. The first n-1 of these blocks is actually free. The last block contains the address of another n free blocks and so on. Here the addresses of a large number of free blocks can be found out quickly.
- **Counting:** Rather than keeping a list of n free disk block addresses, we can keep the address of the first free block and the number of free contiguous blocks. So here each entry in the free space list consists of a disk address and a count.

Q. 6	Stable storage implementation.
Ans	
Q.7	Disk Reliability

Q. 8	What is Deadlock Prevention.
Q. 9	Topic left…
	Concurrency control: concurrency and race condition Mutual Exclusion requirements Software and hardware solution
Q. 10	
	•
Q.11	
	•
Q.12	
Q.13	
Q.14	
Q.15	