Unit – 7
File Management

Prof. Firoz Sherasiya

☎ 9879879861

✉ firoz.sherasiya@darshan.ac.in

Dedicated Faculty Committed Education

Darshan

Institute of Engineering & Technology

Topics to be covered

- File concept
- Access methods
- File types
- File operation
- Directory structure
- File System structure
- Allocation methods (contiguous, linked, indexed)
- Free-space management (bit vector, linked list, grouping)
- Directory implementation (linear list, hash table)

File

- A file is a unit of storing data on a secondary storage device such as a hard disk or other external media.
- Every file has a name and its data.
- Operating system associates various information with files. For example the date and time of the last modified file and the size of file etc….
- This information is called the file's attributes or metadata.
- The attributes varies considerably from system to system.

File attributes

- Below attributes tell who may access it and who may not:
  1. **Protection** - Who can access the file and in what way.
  2. **Password** - Password needed to access the file.
  3. **Creator** - ID of the person who created the file.
  4. **Owner** - Current owner.

File attributes

- Flags are bits or short fields that control or enable some specific property.
  1. **Read only flag** - 0 for read/write, 1 for read only.
  2. **Hidden flag** - 0 for normal, 1 for do not display the listings.
  3. **System flag** - 0 for normal, 1 for system file.
  4. **Archive flag** - 0 for has been backed up, 1 for needs to be backed up.
  5. **Random access flag** - 0 for sequential access only, 1 for random access.
  6. **Temporary flag** - 0 for normal, 1 for delete file on process exit.
  7. **Lock flag** - 0 for unlocked, 1 for locked.

File attributes

- Below attributes are only present in files whose record can be looked up using the key. They provide the information required to find the keys.
  1. **Record length** - Number of bytes in a record.
  2. **Key position** - Offset of the key within each record.
  3. **Key length** - Number of bytes in key field.

File attributes

- Various attributes keeps times track of when the file was created, most recently accessed and most recently modified.
  1. **Creation time** - Date and time the file was created.
  2. **Time of last access** - Date and time of file was last accessed.
  3. **Time of last change** - Date and time of file was last changed.

File attributes

- Various attributes keeps track of file size.

1. **Current size** - Number of bytes in the file.
2. **Maximum size** - Number of bytes the file may grow to.

File operations

- Create
  - The purpose of the call is to announce that the file is coming and to set some attributes.
- Delete
  - When the file is no longer needed, it has to be deleted to free up disk space.
- Open
  - The purpose of the open call is to allow the system to fetch the attributes.
- Close
  - When all accesses are finished, the attributes and disk addresses are no longer needed, so the file should be closed to free up table space.

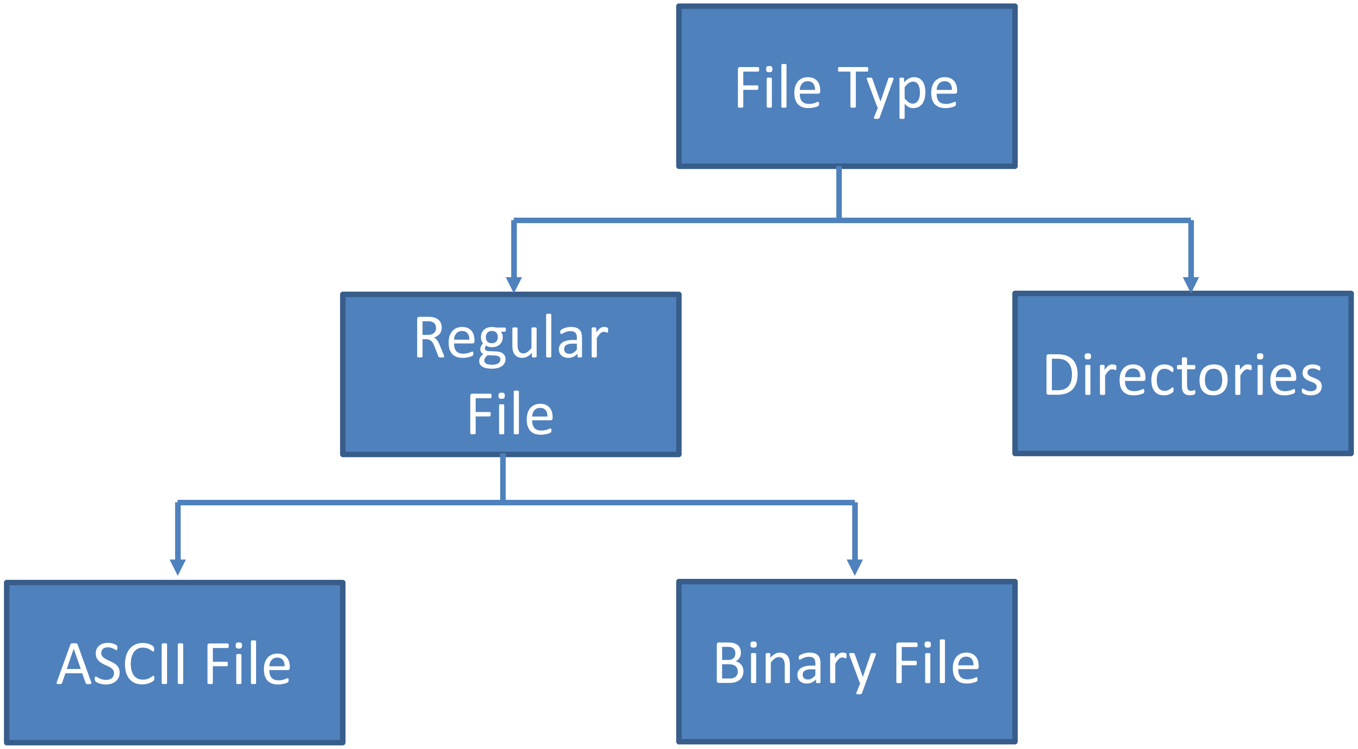
File operations

- Read
  - Data are read from file. Usually the bytes come from the current position.
- Write
  - Data are written to the file, usually at the current position.
- Append
  - This call is restricted form of write. It can only add data to the end of file.
- Seek
  - For a random access files, a method is needed to specify from where to take the data.
  - Seek repositions the file pointer to a specific place in the file.

File operations

- Get attributes
  - Processes often need to read the file attributes to do their work.
- Set attributes
  - Some of the attributes are user settable and can be changed after the file has been created. This system call makes that possible.
- Rename
  - It frequently happens that a user needs to change the name of an existing file. This system call makes it possible.

Types of files



File Type

Regular File

Directories

ASCII File

Binary File

Regular file Vs Directories

- Regular File
  - Regular files are the ones that contain user information.
  - Regular file, as a randomly accessible sequence of bytes, has no other predefined internal structure.
  - Application programs are responsible for understanding the structure and content of any specific regular file.
- Directories
  - Directories are system files for maintaining the structure of the file system.
  - To keep track of files, file systems normally have directories or folder.

Different types of files

1. ASCII files
   - ASCII file consists of line of text.
   - Advantage of ASCII files is that they can be displayed & printed as it is & they can be edited with ordinary text editor.
   - If number of programs use ASCII files for input and output, it is easy to connect the output of one program to the input of another.
   - C/C++/Perl/HTML files are all examples of ASCII files.

Different types of files

1. Binary Files
   - Binary files contain formatted information that only certain applications or processors can understand.
   - Binary files must be run on the appropriate software or processor before humans can read them.
   - Executable files, compiled programs, spreadsheets, compressed files, and graphic (image) files are all examples of binary files.

Different types of files

3. Device Files
    - Under Linux and UNIX each and every hardware device is treated as a file. A device file allows to accesses hardware devices so that end users do not need to get technical details about hardware.
    - In short, a device file (also called as a special file) is an interface for a device driver that appears in a file system as if it were an ordinary file.
    - This allows software to interact with the device driver using standard input/output system calls, which simplifies many tasks.

Different types of files

4. Character Special Files
    - It is a type of device file which talks to devices in a character by character (1 byte at a time).
    - Character Special files are related to input/output and use to model serial I/O devices, such as terminals, printers, and networks.
5. Block Special Files
    - It is a type of device file which talks to devices 1 block at a time (1 block = 512 bytes to 32KB).
    - Block special files are used to model disks, DVD/CD ROM, and memory regions etc.

Files access methods

- Sequential File Access
  - In Sequential access, process could read all the bytes or records from a file in order, starting at the beginning, but could not skip around and read them out of order.
  - Sequential files could be rewound, however, so they could be read as often as needed.
  - These files were convenient when the storage medium was magnetic tape or CD-ROM.

Files access methods

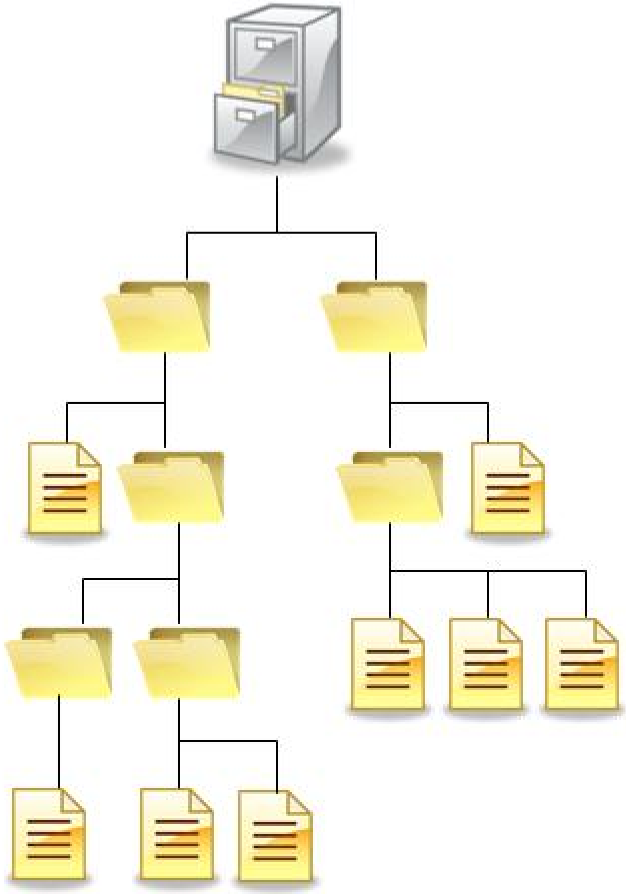
- Random File Access
  - Files whose bytes or records can be read in any order are called random access files.
  - Random access files are essentials for many applications, for example, data base systems.
  - Example: If an airline customer calls up and wants to reserve a seat on a particular flight, the reservation program must be able to access the record for that flight without having to read the records for thousands of other flights.

Directory structure

- Single Level Directory system
  - The simplest form of directory system is having one directory containing all the files. This is some time called as root directory.
- Here directory contain three files.
- The advantages of this scheme are its simplicity and the ability to locate files quickly there is only one directory to look, after all.
- The single level is used for simple dedicated application, but for modern user with thousands of files, it would be impossible to find anything if all files were in a single directory.

Hierarchical Directory system

- In this system user can create an arbitrary number of subdirectories to organize their work.
- When the file system is organized as a directory tree, some way is needed for specifying file names.
- Two methods are used commonly
  1. Absolute path name
  2. Relative path name



Absolute path name

- An absolute path name consisting of the path from the root directory to the file.
- As an example, the path /usr /ast/mailbox means that the root directory contains a subdirectory usr, which in turn contains a subdirectory ast, which contain the file mailbox.

Relative path name

- Relative path name is used in conjunction with the concept of the working directory.
- User can designate one directory as the current working directory, in which case, all path names not beginning at the root directory are taken relative to working directory.
- For example if the current working is /usr/ast, then the file whose absolute path is /usr/ast/mailbox can be referenced simply as mailbox.
- Most operating systems that support a hierarchical directory system have two special entries in every directory,
  1. "." or "dot" refers to current directory
  2. ".." or "dotdot" refers to parent directory

Directory operations

- Create
  - A directory is created. It is empty except for dot and dotdot, which are put there automatically by the system.
- Delete
  - A directory is deleted. Only an empty directory can be deleted.
  - A directory containing only dot and dotdot is considered as an empty directory.

Directory operations

- Opendir
  - Directories can be read.
  - For example, to list all the files in a directory, a listing program opens the directory to read out the names of all the files it contains.
  - Before a directory can be read, it must be opened.
- Closedir
  - When a directory has been read, it should be closed to free up space.

Directory operations

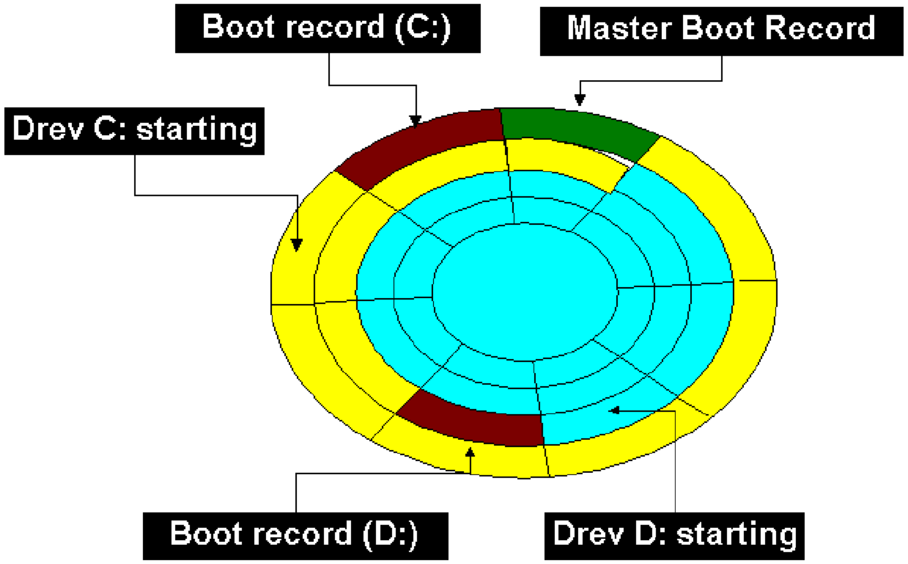
- Readdir
  - This call returns the next entry in an open directory.
  - It was possible to read directories using the usual read system call, but that approach has the disadvantage of forcing the programmer to know and deal with the internal structure of directories.
  - In contrast, readdir always returns one entry in a standard format, no matter which of the possible directory structure is being used.
- Rename
  - In many respects, directories are just like files and can be renamed the same way files can be.

Directory operations

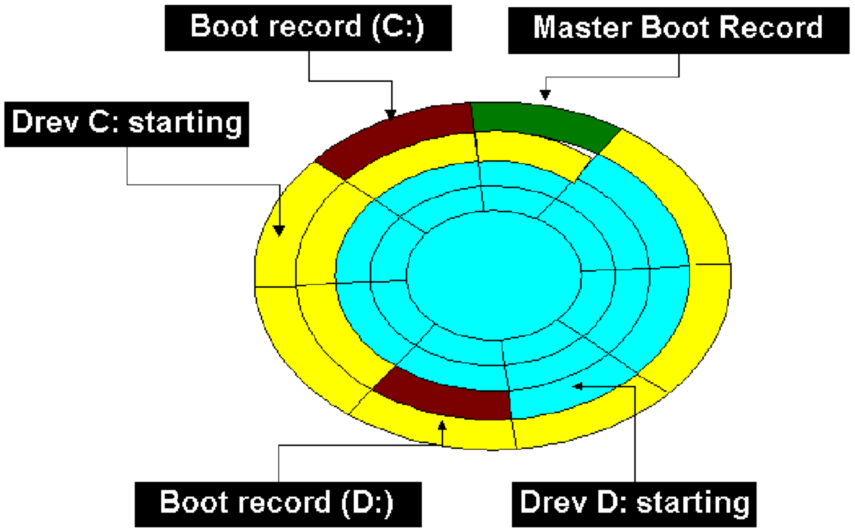
- Link
    - Linking is a technique that allows a file to appear in more than one directory.
    - This system call specifies an existing file and a path name, and creates a link from the existing file to the name specified by the path.
    - In this way, the same file may appear in multiple directories.
- Unlink
    - A directory entry is removed.
    - If the file being unlinked is only present in one directory (the normal case), it is removed from the file system.
    - If it is present in multiple directories, only the path name specified is removed. The others remain.

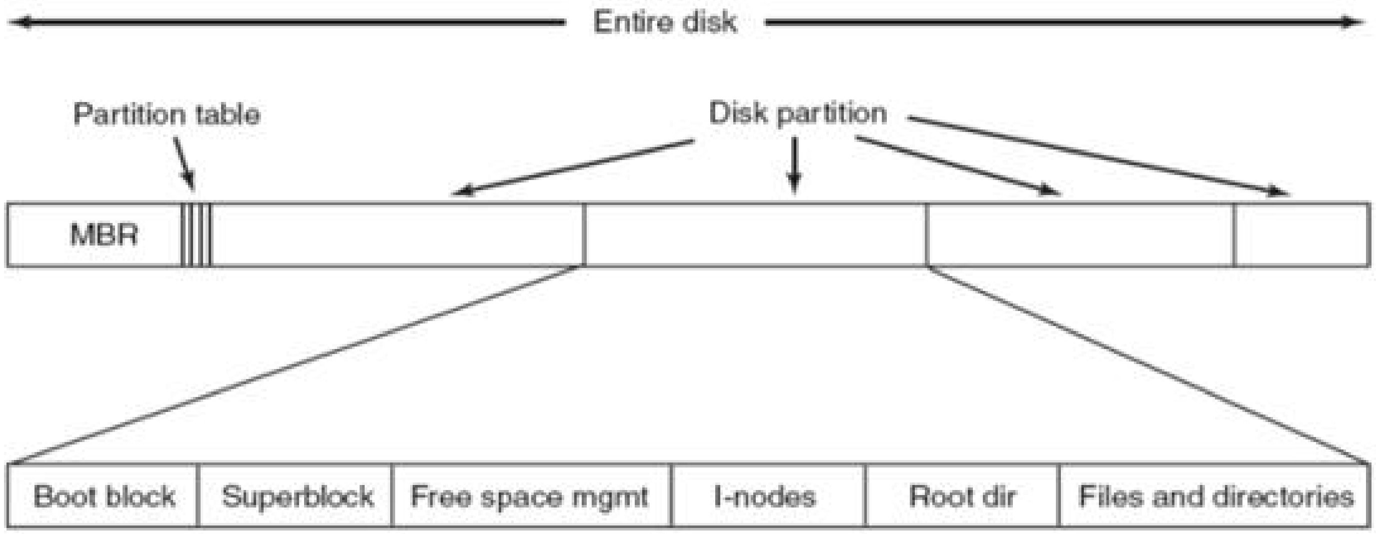
MBR (Master Boot Record)

- The Master Boot Record (MBR) is the information in the first sector (sector 0) of hard disk.
- It identifies how and where an operating system is located so that it can be boot (loaded) into the computer's main storage or random access memory (RAM).
- It is used to boot the computer.
- The end of the MBR contains the partition table which gives the starting and ending addresses of each partition



MBR (Master Boot Record)





File system

- It is group of files and information regarding them.
- The disk space allotted to file system is made up of blocks, each of which are 512 bytes.
- All the blocks belonging to file system are logically divided into:
  1. Boot block
  2. Super block
  3. Inode table
  4. Data block

Boot block

- It represent the beginning of file system.
- It contains a program “bootstrap loader”.
- This program is executed when we boot the machine.
- All the file system contain one boot block.

Super block

- It describe the state of file system.
  - How large file system is
  - How many maximum files it can accommodate
  - How many more files can be created
  - List of free and allocated blocks
  - Modification time of the file system

Inode table

- The information related to all file (not the content) is stored in an inode table on the disk.
- For each file, there is an inode entry in table.
- Each entry is made up of 64 bytes and contain the detail for that file.
- These details are:
  - Owner of file
  - Group to which the owner belongs
  - Types of file
  - File access permission
  - Date and time of last access
  - Date and time of last modification
  - Size of file
  - Address of blocks where the file is physically present

Data block

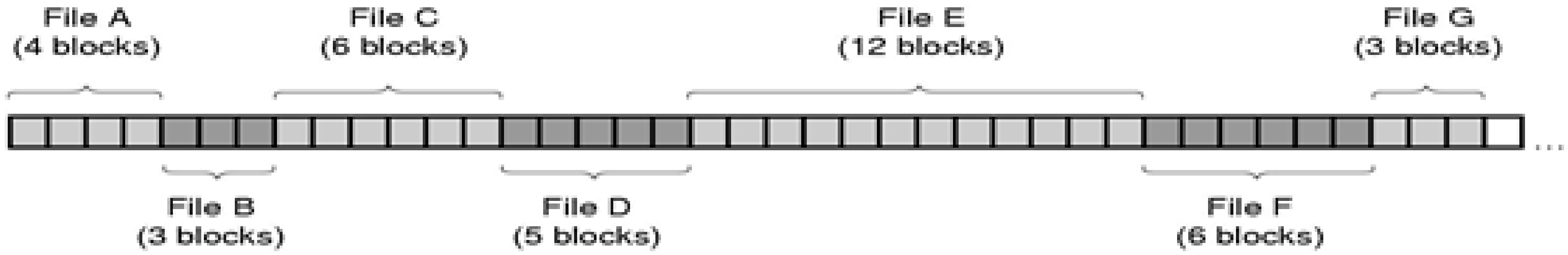
- It contains actual file content.
- An allocated block can belong to only one file in the file system.
- This block can’t be used for storing any other file’s content unless the file to which it originally belonged is deleted.

File implementation

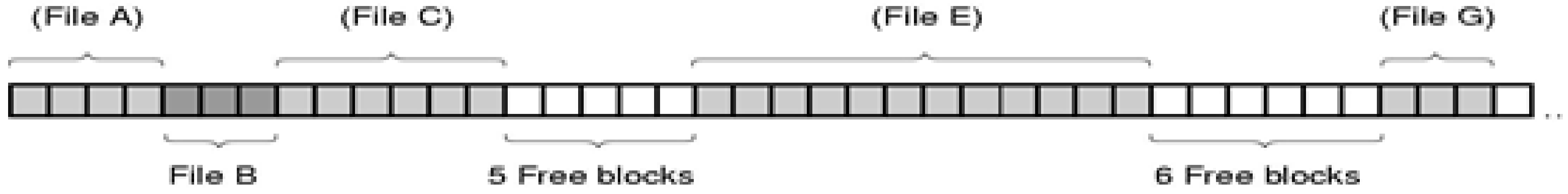
- Various methods to implement files are listed below,
  1. Contiguous Allocation
  2. Linked List Allocation
  3. Linked List Allocation Using A Table In Memory
  4. I-nodes

Contiguous Allocation

- The simplest allocation scheme is to store each file as a contiguous run of disk block.



State of the disk after files D and F have been removed

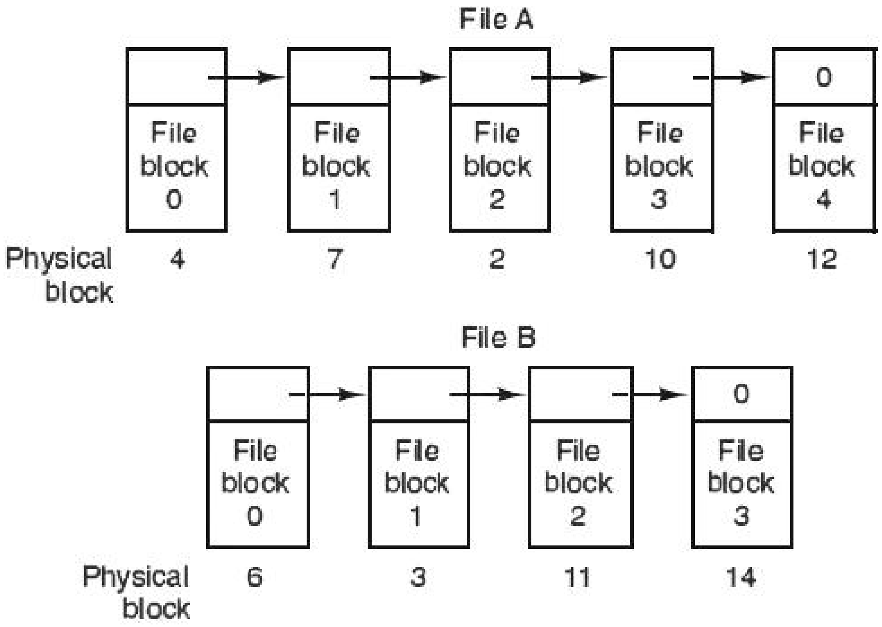


Contiguous Allocation

- It is simple to implement because we need to keep track of only the first block and the number of blocks in the file.
- 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).
- Once the disk is filled, reusing the space requires maintaining a list of hole.
  - However, when a new file is to be created, it is necessary to know its final size in order to choose a hole of the correct size to place it in.

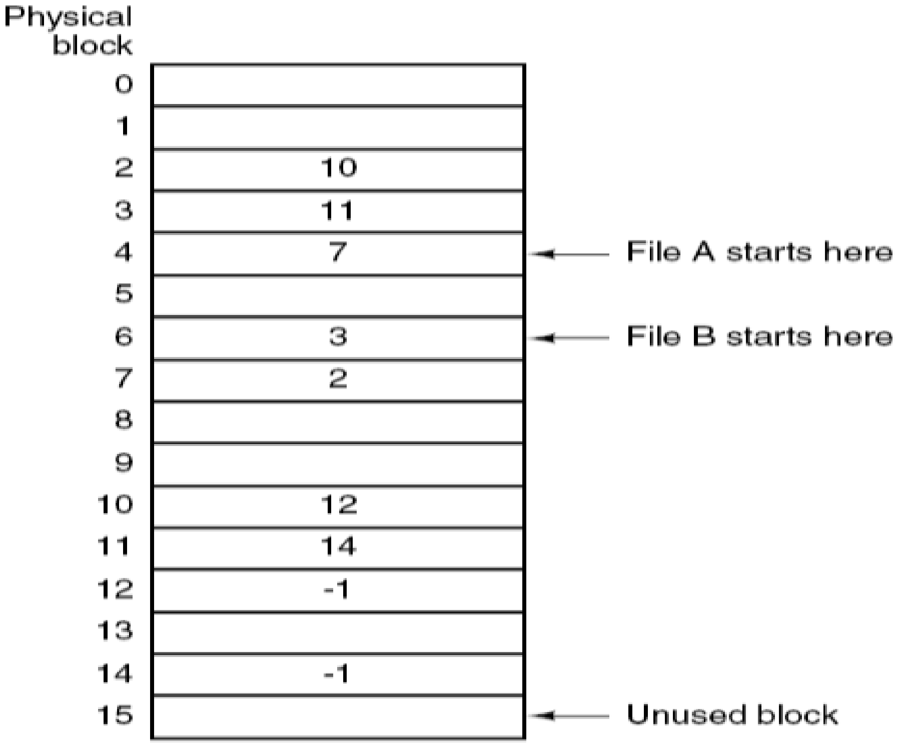
Linked List Allocation

- Another method for storing files is to keep each one as a linked list of the disk blocks.
- The first word of each block is used as a pointer to the next one. The rest of the block is for data.
- No space is lost to disk fragmentation.
- Reading a file is extremely slower than Contiguous Allocation.
- Less data occupied in block.



Linked List Allocation Using A Table In Memory

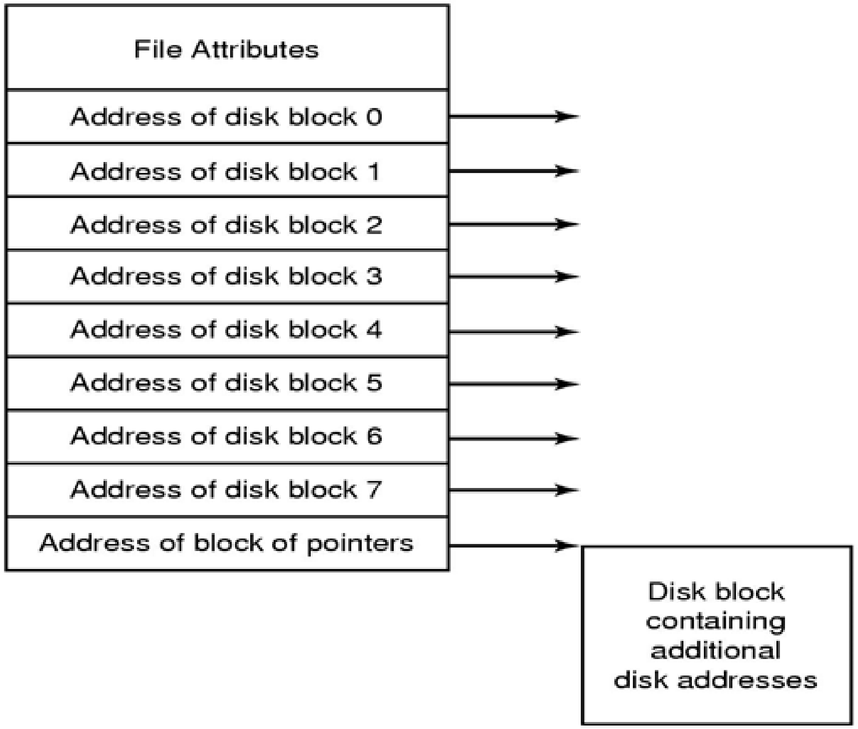
- In this method instead of a pointer, a table in memory called file allocation table is used.
- Take the pointer word from each block and put it in a table in memory.
- Advantage of this method is that random access of files is faster.
- The only disadvantage of this method is that the entire table should always be in the memory.



Physical block		
0		
1		
2	10	
3	11	
4	7	← File A starts here
5		
6	3	← File B starts here
7	2	
8		
9		
10	12	
11	14	
12	-1	
13		
14	-1	
15		← Unused block

I-nodes

- I-node (Index-node) is a data structure used for storing attributes and disk address of a block in memory.
- A method for keeping track of which blocks belong to which file is to associate with each file a data structure called an i-node (index-node), which lists the attributes and disk addresses of the file's blocks.
- Here, only the i-node should be in the memory at all times.



Free-space management

- Techniques to manage free space are:
  1. bit vector
  2. linked list
  3. grouping

Bit vector

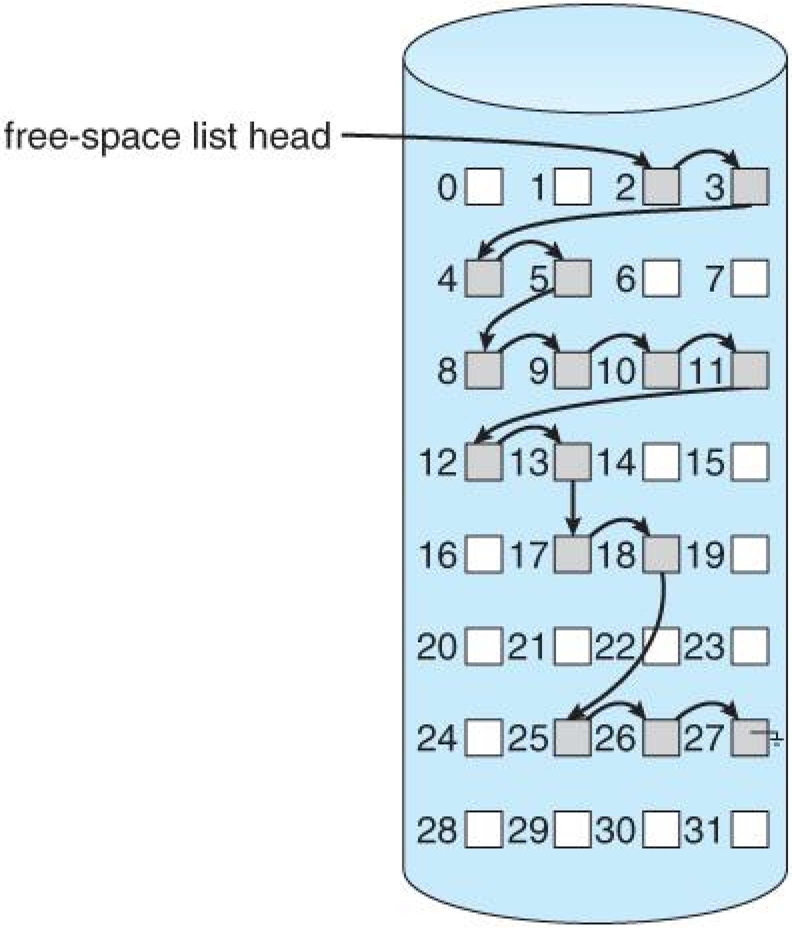
- In this method free space list (block) is implemented as a bit map or bit vector.
- If the block is free the bit is set to 1.
- If the block is allocated the bit is set to 0.
- Example:
  - Consider a disk, where blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 27 are free and rest of blocks are allocated.
  - The free space bit map would be:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0	0	1	1	1	1	0	0	1	1	1	1	1	1	0	0	0	1	1	0	0	0	0	0	0	1	0	1

- Advantages
  - Relatively simple and efficient
  - Easy to get contiguous files

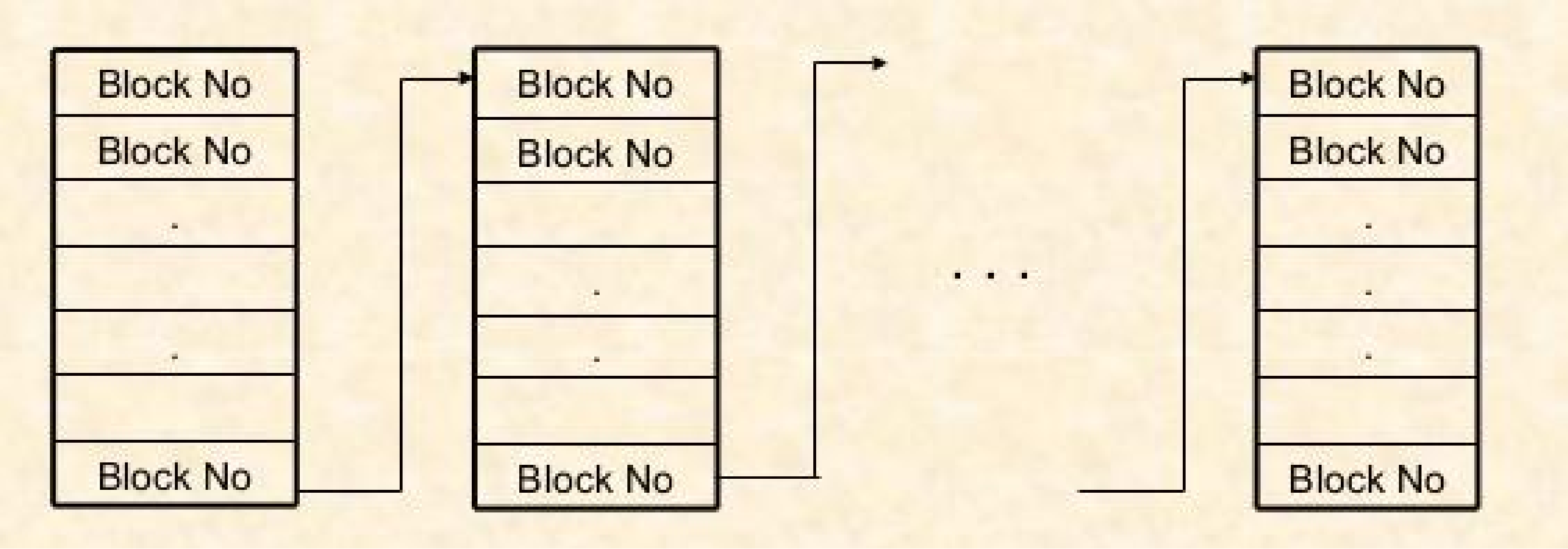
Linked list

- It links all free disk blocks together.
- Each block contain a pointer to next free blocks, and so on
- In figure there is a pointer to block 2, as the first free block.
- Block 2 would contain a pointer to block 3.
- Which would point 4,5,8,9,10,11,12,13,17,18,25,26,27
- Can’t get continuous space easily.
- No waste of space.
- Not efficient for faster access.



Grouping

- It stores addresses of n free blocks in the first free block.



- First n-1 of these block are address of free blocks.
- Last block contains the pointer of another free block.
- Large number of free blocks can be found quickly.

Directory implementation

1. Linear list
    - Linear list of file names with pointer to the data blocks
    - It is simple to program
    - It is time-consuming to execute
2. Hash table
    - It uses hash table – linear list with hash data structure
    - It decreases directory search time
    - It may cause collisions – situations where two file names hash to the same location

Efficiency and performance

- Efficiency depends on:
  - Disk allocation and directory algorithms
  - Type of data kept in file
    - Example: keep a last write date or last access date
    - Thus whenever a file is read, a field in the directory structure must be written to.
- Performance:
  - Keep disk cache – a separate section of RAM in the disk controller for frequently used blocks.
  - Some system maintain a separate section of main memory for a buffer cache.
  - Improve the performance of system by allocating dedicated section of memory as virtual disk or RAM disk.

Issues in file backup

1. First should the entire file system be backup or only part of it? It is usually desirable to back up only specific directories and everything in them rather than the entire file system.
2. It is wasteful to back up files that have no change since the previous backup which leads to the idea of incremental dumps.
3. Third since huge amount of data are typically dumped, it may be desirable to compress the data before writing them to tape.
4. It is difficult to perform a backup on an active file system.
5. Making backups introduces many nontechnical problems into an organization.

Physical dump vs Logical dump

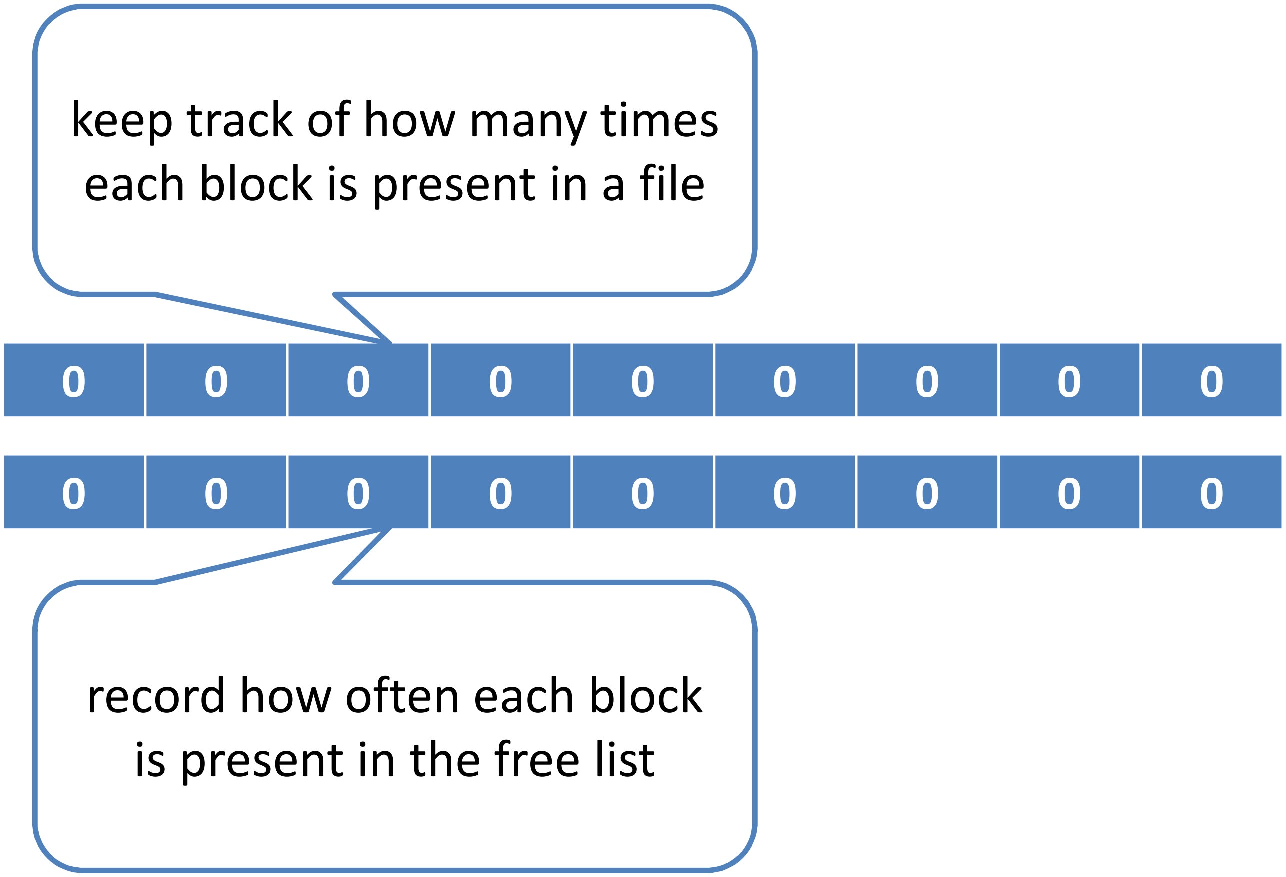
- Physical dump:
  - It is started at block 0 of the disk, writes all the disk blocks onto the output tape in order, and stops when it has copied the last one.
  - Such program is so simple that it can probably be made 100% bug free.
  - Main advantages of physical dumping are simplicity and great speed.
- Logical dump:
  - It is started at one or more specified directories and recursively dumps all files and directories found there that have changed since some given base date.
  - In logical backup individual files can be backed up and restored as per the requirements.

File System Consistency

- Two kinds of consistency checks can be made:
  1. Blocks
  2. Files

Block consistency

- To check for block consistency, the program builds two tables, each one containing a counter for each block, initially set to 0.



keep track of how many times each block is present in a file

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

record how often each block is present in the free list

Block consistency

- If the file system is consistent, each block will have a 1 either in the first table or in the second table.

0	1	0	0	1	1	1	0	1
1	0	1	1	0	0	0	1	0

Block consistency

- Block number 2 do not occur in either table.
- It will be reported as being a missing block.

0	1	0	0	1	1	1	0	1
1	0	0	1	0	0	0	1	0

- The solution to missing blocks is straightforward: the file system checker just adds them to the free list.

0	1	0	0	1	1	1	0	1
1	0	1	1	0	0	0	1	0

Block consistency

- Block number 2 occurs twice in the free list.

0	1	0	0	1	1	1	0	1
1	0	2	1	0	0	0	1	0

- The solution here is also simple: rebuild the free list.

0	1	0	0	1	1	1	0	1
1	0	1	1	0	0	0	1	0

Block consistency

- Block number 4 occurs twice in two or more files.

0	1	0	0	2	1	1	0	1
1	0	1	1	0	0	0	1	0

- The appropriate action for the file system checker to take is to allocate a free block, copy the contents of block 4 into it, and insert the copy into one of the files.

0	1	0	0	1	1	1	1	1
1	0	1	1	0	0	0	0	0