File Management

Module-4

* **Files:**

- Data collection created by users

* **Properties of Files:**

(1) Long term existence:

- Files are stored on disk or other secondary storage and do not disappear after a user logs off.

(2) Sharable between processes:

- Files have names and can have associated access permissions that permit controlled sharing.

(3) Structure:

- Depending upon file system, file can have internal structure.
- Files can be organized into hierarchical (more complex structure to reflect relationship among files.

* **File systems:**

- Provide a means to store data organized as files as well as collections of functions that can be performed on files.
- Maintain a set of attributes with the file.
- Operations:

1. Open 2. Close 3. Read 4. Write 5. Create 6. Delete.

* **File Structure:**

- File can be structured as a collection of records or a sequence of bytes.
- UNIX, Linux, Windows consider files as a sequence of bytes.
- Other OS, IBM manufacturers adopt a collection of records approach.

* Structure Terms:

(1) Field: - Basic element of data.
- Contains a single value.
- eg: employee's last name.
- characterized by data type.
- fixed or variable length

(2) File:
- collection of similar records
- treated as a single entity by users.
- may be referenced by name.
- access control restrictions apply at file level.

(3) Record:
- Collection of related fields that can be treated as a unit by some application of a program.
- One field of record is the key - unique identifier.
- eg: An employee record would contain fields: name, social security number, job classification, date of hire and so on.
- Depending on design, records may be of fixed length or variable length.

(4) Database:
- collection of one or more file types.
- Collection of related data.
- Relationship among elements of data is explicit
- Designed for use by a number of different applications.

* File Management system objectives:
- A file management system is a set of system software that provides services to users and applications in the use of files.
- The only way a user can get access to a file is through filer management system.
- Meet data management needs of user.
- Optimize performance.
- Guarantee that data in the ~~field~~ ~~if~~ file are valid.

- Provide I/O support for variety of storage device types.
- Minimize potential for lost or destroyed data
- Provide a standardised set of I/O interface routines to user process.
- Provide I/O support to multiple users in a multi-user system.

* File System Architecture.

1. File Formats
2. Logical I/O
3. Basic I/O
4. Basic File System
5. Device Drivers.

* Device Drivers:

- Lowest level
- ~~Connect~~ Communicates with peripheral devices
- Responsible for starting I/O operations on devices
- Processes the completion of an I/O request.
- Considered to be a part of the operating system.

* Basic File System:

- Also called physical I/O level.
- Primary interface is with the environment outside the system.
- Deals with blocks of data that are exchanged with disk or other storage devices.
- X { Placement of blocks on ~~sec~~ secondary storage.
  Buffering blocks in ~~in~~ memory.
- Considered to be a part of OS.

* Basic I/O Supervisor
- Responsible for file I/O initiation and termination.
- Selects the device on which I/O has to be performed.
- Control structures that deal with device I/O, scheduling and file status are maintained.
- Part of Operating System.

* Logical I/O
- ~~Enabl~~ This level acts as an interface between logical commands sent by the program and physical details required by the disk.
- Enables the user and applications to access records.
- Provides general I/O capability
- Maintains data about file.

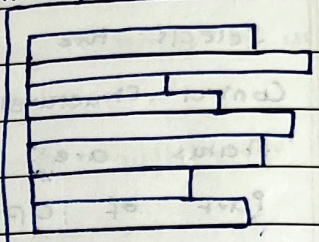
* Access Methods:
- Level of file system to closest to the user. file system
- Provides a standard interface between application and ~~files~~ and devices that holds the data.
- Different access methods give different file structures and different ways of accessing and processing the data.

* File organization and Access:
- File organization is the logical structuring of the file records as determined by the way they are accessed.
- In choosing a file organization different criteria are important:
(1) short access time (4) simple maintenance
(2) ease of update. (5) reliability.
(3) economy of shortage
- Relative priority of these criteria depends on the application that will use the file.

* File Organisation Types:

(1) Pile: - Least complicated way of organising files.

- Data are collected as soon as they arrive.
- Purpose is to simply accumulate the mass and save it.
- Every record contains one burst of data.
- Records may have different fields or same fields in different order.
- Since, there is no structure to the pile file, record access is by exhaustive search.
- If we wish to access a particular record in a file with particular value, it is necessary to examine each record in pile until the record is found or entire file has been searched.



Variable- length records
Variable set of fields
chronological order.

(2) Sequential File: - Most common file form of file structure.

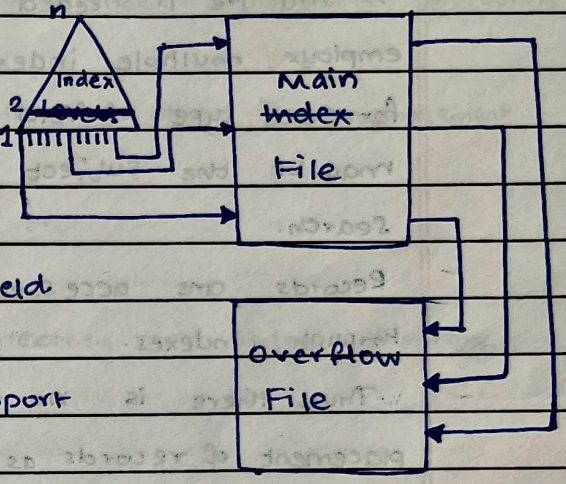
- A fixed format is used for all records.
- All records are of the same length.
- Records have the same number of fixed length fields in a particular order.
- one particular field usually the first field in every record is taken as the 'key' field.
- The 'key' field uniquely identifies the record.
- Thus, key values are different for different records.
- Records are stored in key sequence: alphabetical order for text key and numerical order for numeric key.

- Fixed length records
- Fixed set of fields in a fixed order.
- Sequential order based on key field.

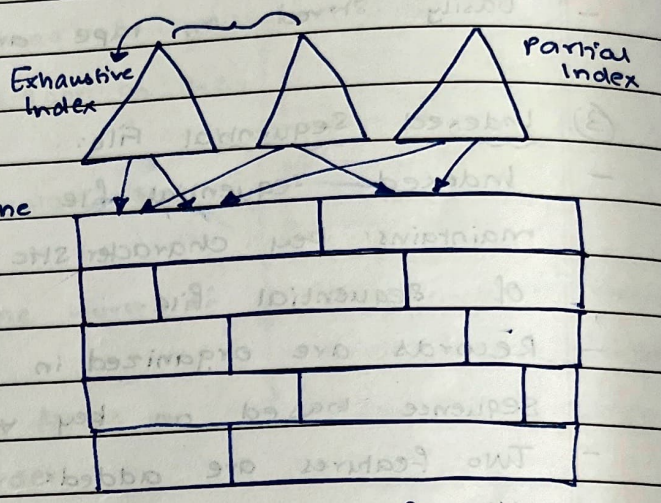
- used in batch applications, if they involve processing of all records.
- Easily stored on tape and disk.

(3) Indexed Sequential File:

- Indexed sequential file maintains key characteristic of sequential file.
- Records are organized in sequence based on key ~~value~~ field
- Two features are added:
  - Adds an index to the file to support random access
  - Overflow file.
- Each record in the index file has two fields:
  - A key ~~record~~ field which is same as the key in the main file.
  - A pointer ~~which~~ to the main file.
- To find a specific ~~key~~ field, index is searched to find highest key value that is equal or just precedes the key value.
- The search continues in the ~~to~~ mainfile as indicated by the pointer.
- Consider 1 million records in a sequential file.
- To search for a key value will require us to access one half million records on an average.
- Now suppose that an index with 1000 entries is constructed with keys more or less evenly distributed over main file.
- The average search length is reduced.
- Each file in the main field ~~can~~ contains an additional file that is not visible to the application ~~also called as~~ which is a pointer to overflow file.
- When a new record is added to the ~~main~~ file, it is inserted to the overflow file. The record in the main file that immediately precedes the new record in a logical sequence is updated to contain a



Pointer to the new record in overflow file



④ Indexed File:

- A structure is needed that employs multiple indexes fo one for each type of field that may be the subject of the search.
- Records are accessed only through indexes.
- Thus, there is no restriction on placement of records as long as at least one pointer in index refers to one record.
- Records with variable length are also employed. (variable length records)
- Two types of indexes:
  - Exhaustive index: It contains one entry for every record in the main file.
  - Partial index: It contains entries of only those records where field of interest exists.
- with variable-length records, not every record will contain all fields.
- Indexed file is used in situations where timeliness is critical and data is rarely processed exhaustively.

eg: Airline reservation system.

- When new record is added, indexes must be updated.

⑤ Direct or Hashed Files:

- Direct file makes use of hashing.
- No concept of sequential ordering.
- Often used where:
  - Very rapid access is required
  - fixed length records
  - Records are accessed one at a time.

eg: directories, pricing tables, schedules & name lists.

* File Directory Information:

- Associated with any file management system and collection of files is called a directory.
- The directory contains information about the files including attributes, location and ownership.
- The directory itself is a file accessible by other file management routines.

* Operations performed:

Search, create files, delete files, list directory and update directory.

* Two-level scheme:

- There is one directory for users called user directory and a master directory.
- A master directory contains an ~~list of~~ entry for each user directory providing ~~access~~ address and access control information.
- User directory is a simple list of files that user uses.
- Names must be unique within a collection files of a single user.
- File system can easily enforce access restriction on directories.
- Each user directory may have sub-directories and files as entries.

* File sharing:

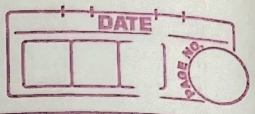
- Two issues arise when allowing files to be shared among a number of users:
  - Access rights
  - Management of simultaneous access.

* Access Rights:

(1) None: user is not allowed to read the directory that includes the file.

(2) Knowledge: The user can determine that file exists and who it's owner is and can petition the owner to provide extra ~~acc~~ access rights.





(3) Execution: user can load and execute a program, but not copy it.

(4) Reading: User can read file for any purpose: including copying & execution.

(5) Appending: User can add data to file but cannot ~~add~~ or modify or delete file's contents.

(6) updating: User can modify, delete and add to the file's data.

(7) Changing protection: The user can change the access rights granted to other users.

(8) Deletion: user can delete any file.

* User Access Rights:

(1) owner: Initial creator of file. Has full rights. Grants rights to others.

(2) Specific users: Specific users who are designated by their user ID.

(3) User groups: set of individual users who are not defined.

(4) All: All users have access to system. Consists of public files.

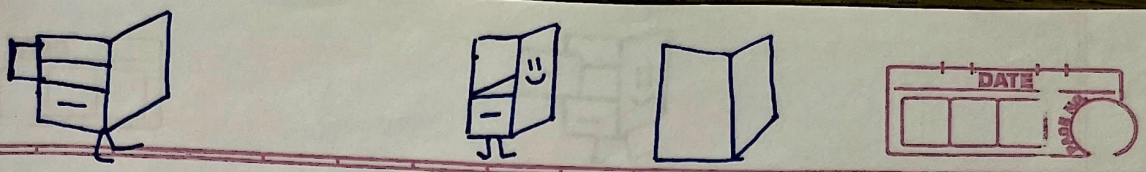
* File Allocation:

- Disks are divided into physical blocks.
- Files are divided into logical blocks (subdivisions of files)
- Logical block size = multiple of physical block size.
- Operating system or file management system is responsible for allocating blocks to files.
- Space is allocated to file as one or more portion (contiguous set of allocated disk blocks). Portion is logical block size.
- FAT Table:

A data structure used to keep track of portions assigned to a file.

* Pre-Allocation V/s Dynamic Allocation:

- Pre-allocation: Maximum size of file must be declared at the time of file creation request.



- For many applications, it is difficult to estimate reliably the maximum potential size of file. Tends to be wasteful because users and programmers over estimate the size.
- Dynamic Allocation: Allocates space to file as portions as and when needed.
- While determining portion size, there is trade-off b/w efficiency from the point of view of a single file versus overall system efficiency.
- We must have variable size of small fixed-size allocations portions to minimise wastage of unused storage due to over allocation.
- Another alternative is to have variable, large contiguous portions
- And instead of having small fixed portions which may require large tables for allocations we can use blocks.

* FAT Table: SEE FROM ppt CV. (imp)

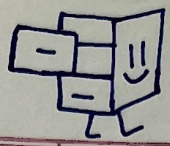
* Contiguous File Allocation: See dig.
- Single contiguous set of blocks are allocated at the time of file creation.
- Thus it is a pre-allocation strategy.
- Along with pre-allocation strategy, it can also use a variable-size portions.
- The file allocation needs single entry of file, showing starting block and length of the file.
- Contiguous allocation is best from the point of view of an individual sequential file.
- Multiple blocks can be read at a time to improve I/O performance for sequential processing.
- It is easy to retrieve a block.
- If a file starts at block 'b' and ith block of file is also wanted, its location of secondary memory is b+i-1

- Problems: External fragmentation can occur, making it difficult to find contiguous blocks of sufficient length.
- Soln: Perform compaction algorithms to free up space on the disk.

* Chained Allocation: see dig.
- Allocation is done on individual block basis.
- Each block contains a pointer to the next block in the chain.
- File allocation table contains a single entry of each file, showing starting point of the block and length of file.
- Although, pre allocation is possible, it is easier to simply allocate blocks as and when needed.
- Selection of block is simple. Any empty / free block can be added to the chain.
- There is no external fragmentation, because only one block at a time is needed. Better for sequential files.
- One consequence is- no accomodation of the principle of locality. To overcome this, some system periodically consolidate the files. - No direct access., pointer overhead

* Indexed Allocation with Block Portions: see dig.
- File allocation table contains a separate one-level index for each file.
- The index has one ~~en~~ entry for each file that has been allocated.
- File index is kept in a separate block and the entry for the file in the FAT points to that block.

* Indexed Allocation with Variable Length Portions: (see dig.)
- No external fragmentation.
- Direct / Random Access Available.
- disadvantages: Pointer overhead, multilevel index



** Free space management (See ppt)

- Disk allocation table.
- Unallocated space should be managed too.

~~To perform~~

- Bit Tables (Bit Vectors)
  - Vector containing one bit for each block on disk
  - Enter 0 if it is a free block
  - Enter 1 if block is used.
  - Advantages: Works with any allocation method, as small as possible, finding one contiguous group of free blocks becomes easy.

* UNIX File Management

- Six types of files distinguished:

(1) Regular or ordinary File:

- Regular file contains information entered by a user, an application program or a system utility program.
- Ordinary file contains data, text or program instructions.
- Contains arbitary data in zero or more data blocks.
- File system does not impose an internal structure but treats as a stream of bytes.

(2) Directory:

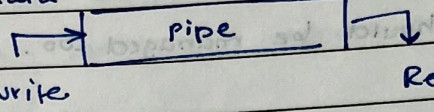
- Hierarchically organized.
- List of files names and pointers to associated to index nodes.

(3) Special:

- used to map physical devices to file names
- File names are used to access peripheral devices like terminals
- Special files are used for I/O operations on UNIX / UNUX systems.

(4) Named Pipes: inter process communications facility

- Pipe buffers data received in input. Process that reads data from output gets it on FIFO basis.



(5) Socket: special file that allows advanced inter process-communication. used in client-server applications.

(6) Symbolic Links: Data file that contains name of file it is linked to. Also called soft link.

* INODES (See ppt)

- Inode is a control structure that contains key information needed by OS for a particular file.

* Disk Scheduling:

→ What is Hard-disk?

Hard disk / drive is one electro-mechanical data storage device that stores and retrieves digital data using magnetic storage and one or more rigid rapidly rotating platters covered with magnetic material.

Platters are paired with magnetic heads, moving on an actuator arm which can read & write data to platter surfaces.

Data is accessed in random manner, individual blocks of data can be stored and retrieved in any order.

Imp. Terms:

(1) Seek Time: - Seek Time is time taken by to move disk arm to a specified ~~sector~~ track containing desired sector where data is to be read or written. Disk Scheduling algo give minimum avg. Seek time.

(2) Rotational latency: is the time taken by the ~~disk sector~~ desired sector of disk to rotate to the head of the disk i.e a position where it can access read/write heads.
The disk scheduling algo that gives min. avg. rotational latency is better.

(3) Disk bandwidth: $\frac{\text{Total no. of bytes transferred}}{\text{Time b/w first request for service \& completion of last transfer.}}$

(4) Transfer time: Time to transfer data. Depends on rotating speed & no. of bytes to be transferred.

(5) Disk Access Time: Seek Time + Rotational Latency + Transfer Time.

(6) Disk Response Time: Avg. of time spent by a request waiting to perform I/O operation.

* Disk scheduling Algos:

(1) FCFS

- Simplest form of disk scheduling.
- Instrinsically fair
- Does not provide fastest service.
- Advantages: (1) Every request gets a fair chance
  (2) No ~~td~~ indefinite postponement
- Disadvantages: (1) Does not optimize seek time.
  (2) High Response Time
  (3) Not the best service.

② SSTF: -Service all the requests close to current head position before moving the head far to the service other requests.

- Select the request with shortest seek time from current head position.
- It may cause starvation of requests
- Disadvantages: ① ↓
  ② Not optimal
  ③ Indefinite postponement
  ④ Only favors some requests.
  ⑤ Overhead to calculate seek time in advance.
- Advantages: ① Avg. Response Time decreases.
  ② Throughput increased. (seeks are short)

③ SCAN: -Also called elevator algorithm. (L → R → end → stop)

- First service all the requests going up and reversing to service requests the other way.
- If a request arrives in front of head, it will be serviced almost immediately. if it arrives behind the head, it will have to wait until the arm moves to the end, reverses it direction and comes back.

(practically not used)

- Advantages: High Throughput, Low variance response time (better than FCFS & SSTF)
  Avg. Response Time
- Disadvantages: • Long waiting time for requests for locations just visited by disk arm.

④ C-Scan: circular Scan. (R→end then direct →end and then service)

- moves from one end of disk to another, servicing requests as it goes.
- When it reaches the other end, it immediately returns to the beginning without servicing any requests on return trip.
- More uniform wait time than SCAN.
- Treats cylinder as a circular list wraps around from the last to first