(X) Create file
like adding new file to the system
- first check the folder where the file should go.
- by looking through the list of existing files in that folder.
- If file with some name exists stoping with an error.
- then reserving the space for a record of this file.
- creating an entry that represents this file in the system (just recording it with its name & some metadata like timestamps & size).
- Adding some kind of link between folder & this file record. so the system know this file is part of that folder.
- making it as an empty file At this point, the file exists but has no content allocated yet.

(X) Writing into a file
adding content to existing file.
① Find the file's record
looking for the file by its
name to find its entry in system

○ Breaking the file content into
units which are managable.
like by some fixed data block
size, creating file that many
chunks.

○ Finding space to store these
chunks
locating free spots in storage
and plan where to place each
piece of data.

○ Writing contents chunks into
the choosen locations.

○ Updating the file record
Adjust size & timestamps to
reflect new content.

(1) Read a file
Retrieving data from the file.
- locating the file
  locating it in the folder by its name
- figuring out where its data is stored
  determining which storage units hold the requested bytes.
- loading the data pieces in right order
- returning the requested portion

(2) Delete a file
completely removing a file & its storage from a system.
- finding the file in its folder
- removing its storage
  identifying all storage pieces used by this file & clear them for reuse
- erasing the file record & folder reference

Date
Page

(*) need of system
- keep a catalog of all files & their properties.
- knowing where the data is physically stored.
- organizing storage into fixed-size chunks.
- efficiently finding free space to store new data.
- Allow folders to contain multiple files
- Track metadata like timestamps, permissions, etc.

(*) Need A Global index of the filesystem layout
The system needs a map of where all its structure are
- where does metadata start?
- where does storage start?
- How many blocks do we have?

soln
- creating a fixed block structure let's say named superblock to store
  - total number of blocks
  - total number of file record
  - and more metadata informations
  - The IO of root directory

(*) A way to store metadata about each file
-> File needs
- size
- permission (not necessary if only one user)
- time of creation / modification / accessed
- which storage block hold its data.

Sol^n
-> defining a fixe structure of size some K to hold, named Inode
- A unique record of each file.
- contains metadata & pointers to storage blocks.
- separating filename from actual data.

(*) Need a way to map file name to their Records
- the user knows the file by its name, not its ID

so, we need structure that links filename -> inode number.

- Folders themselves should behave like special files that contain their mappings.

Sol^n. defining a directory entry structure
- A small record holding.
  - File name
  - Inode number.

A directory will be just file containing multiple

directory entries.

④ system need to find free space quickly
- when creating or writing a file, we must know which slots are free for
  -> new inodes
  -> new data blocks

-1 sol^n
-> creating bitmaps:
inode_bitmap[]: track which inode slot are free or used.
block_bitmap[]: track which storage blocks are free or used.

⑤ need a way to store actual file content.
- file content is stored in fixed-size data blocks.
- each inode points to the block where its file data is stored.

sol^n:
implementing a data block area contiguous space divided into fixed-size blocks.
- this will be where all file content lives.

(X) Also need a way to store All inodes.
- All file metadata (inode) should be stored in a dedicated space for quick access.

Ans.
An Inode table : a fixed-size array of inode records, one for each file or directory.