# Lecture 22 — the file abstraction / file system

# manage (access) store / index data (information).

- file — abstraction for a data object / item.

- identifier (filename,
- size
- on-disk location information | persistence
  in-memory | volatile (ramdisk)
- access control
- statistics — timestamps created, last updated
- # links

(1) How to locate elements with an abstraction?

linear contiguous sequence of bytes.

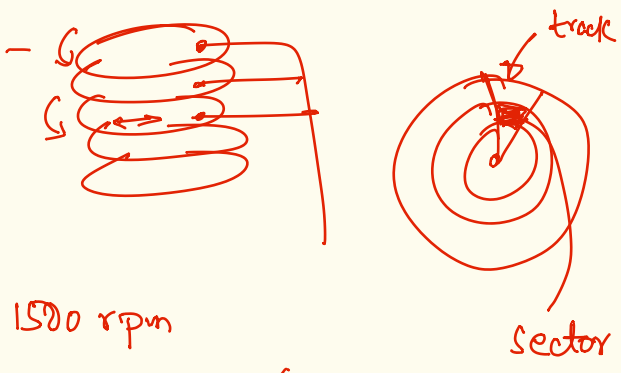
file id, offset

interface: open read seek
close write

---

(2) How view storage

## Hard disk



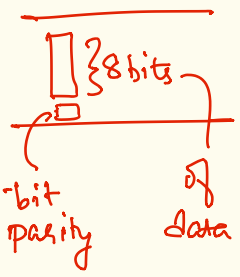

- 1500 rpm

access ⇒ ~~block~~ (platter no, track no, sector no.

## Flash storage

- semi-conductor no moving parts
- NAND / NOR flashes.
- ~~fixed~~ fixed # write per location

## Tapes



- magnetization on the platter to encode 0/1s.
- moving parts. (electro mechanical tech.)

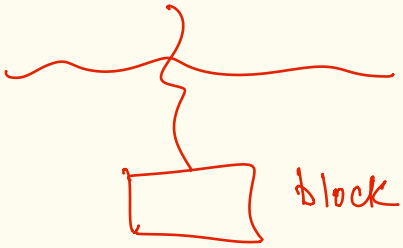
- blocks of pages
- all R/Ws are at page granularity.

h/w interface

(P, t, S)

bank page#

offset on the tape



block interface. (abstraction)

provided disk controller

Assume

storage = # blocks.

size per block — bytes — typically ~512

User

Appln

open, read, write

Kernel

user foo.txt, 23

OS, fd, 23

OS2 inode, fd (23)

OS3

<fd, offset>

FS

<fs info of fd, offset>

~~Device~~ driver OS

<disk, block>

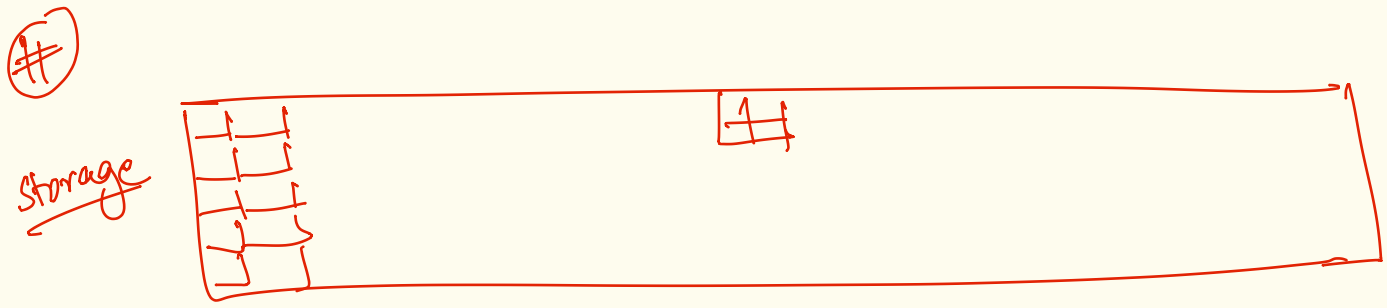
disk controller

HDD.

DD

DD

USB storage




two types of metadata

① device-related.

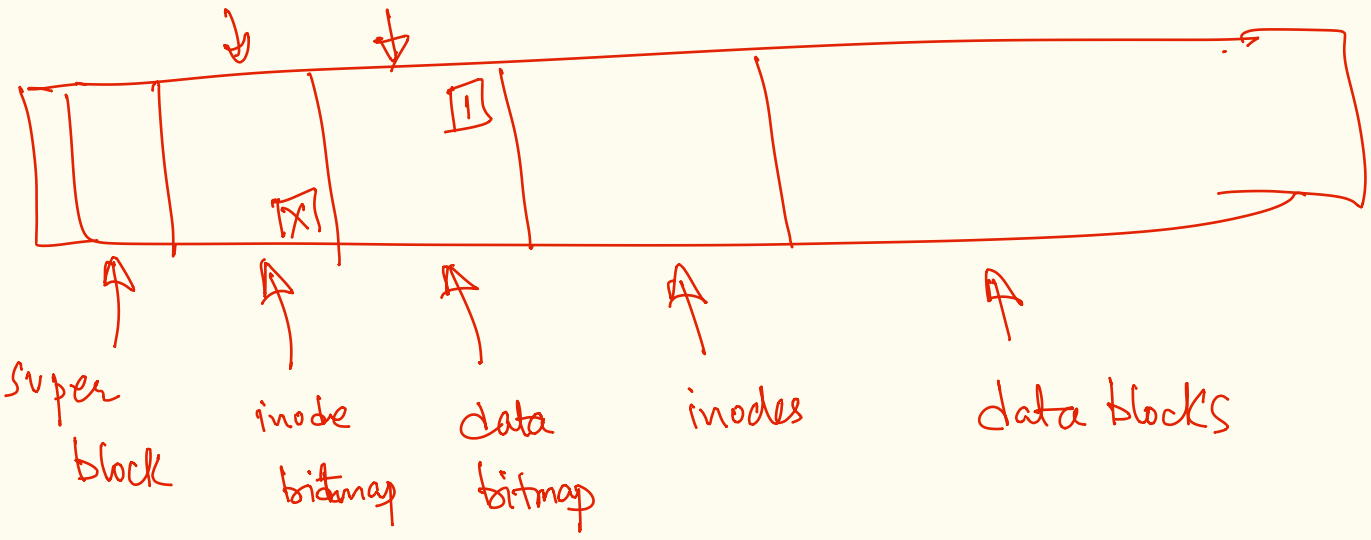
data bitmap — bitmap of used/free datablocks.

per-file related.

inode (index node)

- size
- timestamps
- permissions
- links
- identifier
- set of datablocks sequence

② How should a disk be organized?

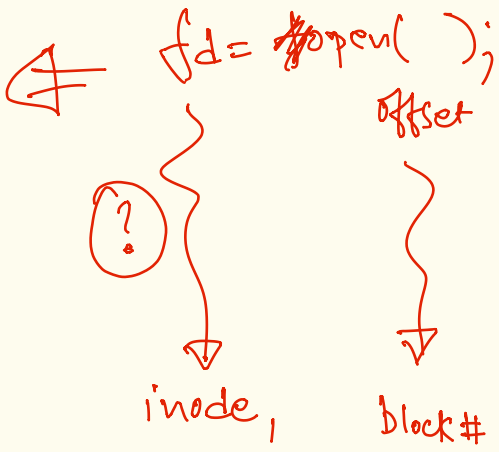


# create a file:

① – lookup inode bitmap
(find free inode)

– mark / update inode to not-free

– write to inode file details.




② write to a file

– fd → inode number.

– read inode.

– use inode to for offset → block.

↳ check offset < size.
↳ check permissions

– read / write from datablock * ✓

– update data bitmap * ✓

– update inode * ✗