File System

l.sharifi@urmia.ac.ir

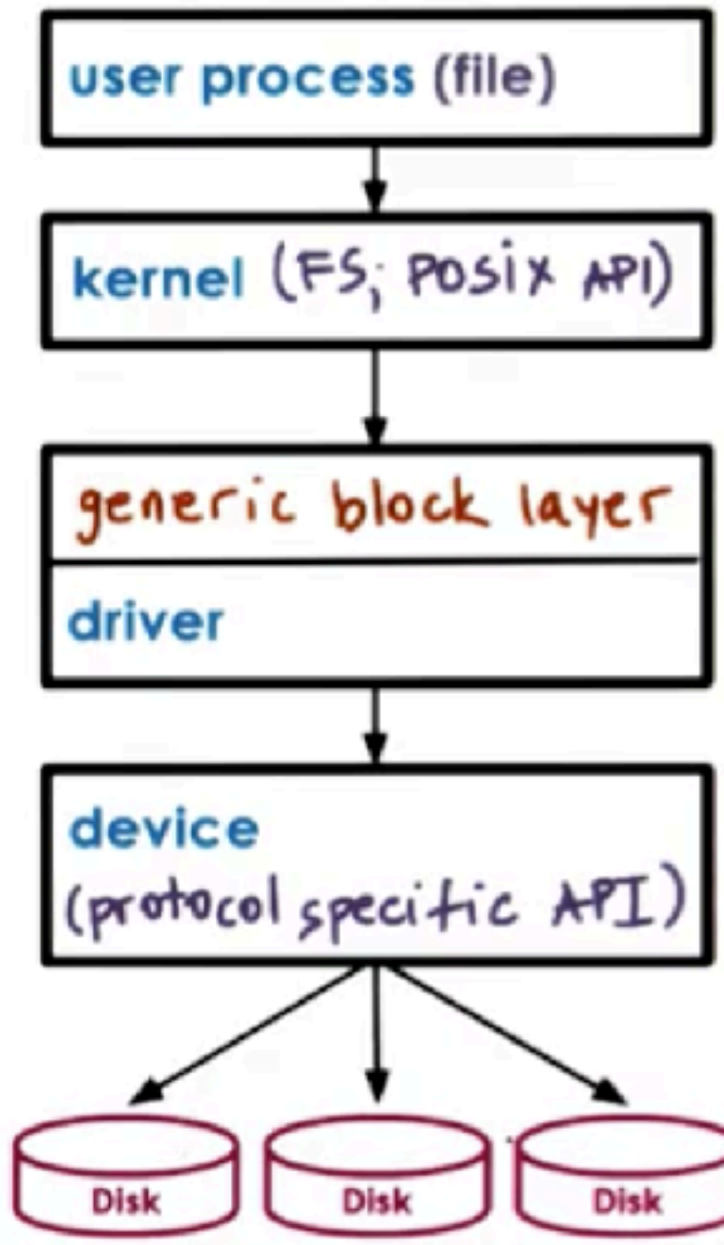
Block Device Stack

processes use files => logical storage unit

kernel file system (FS)
- where, how to find and access file
- OS specifies interface

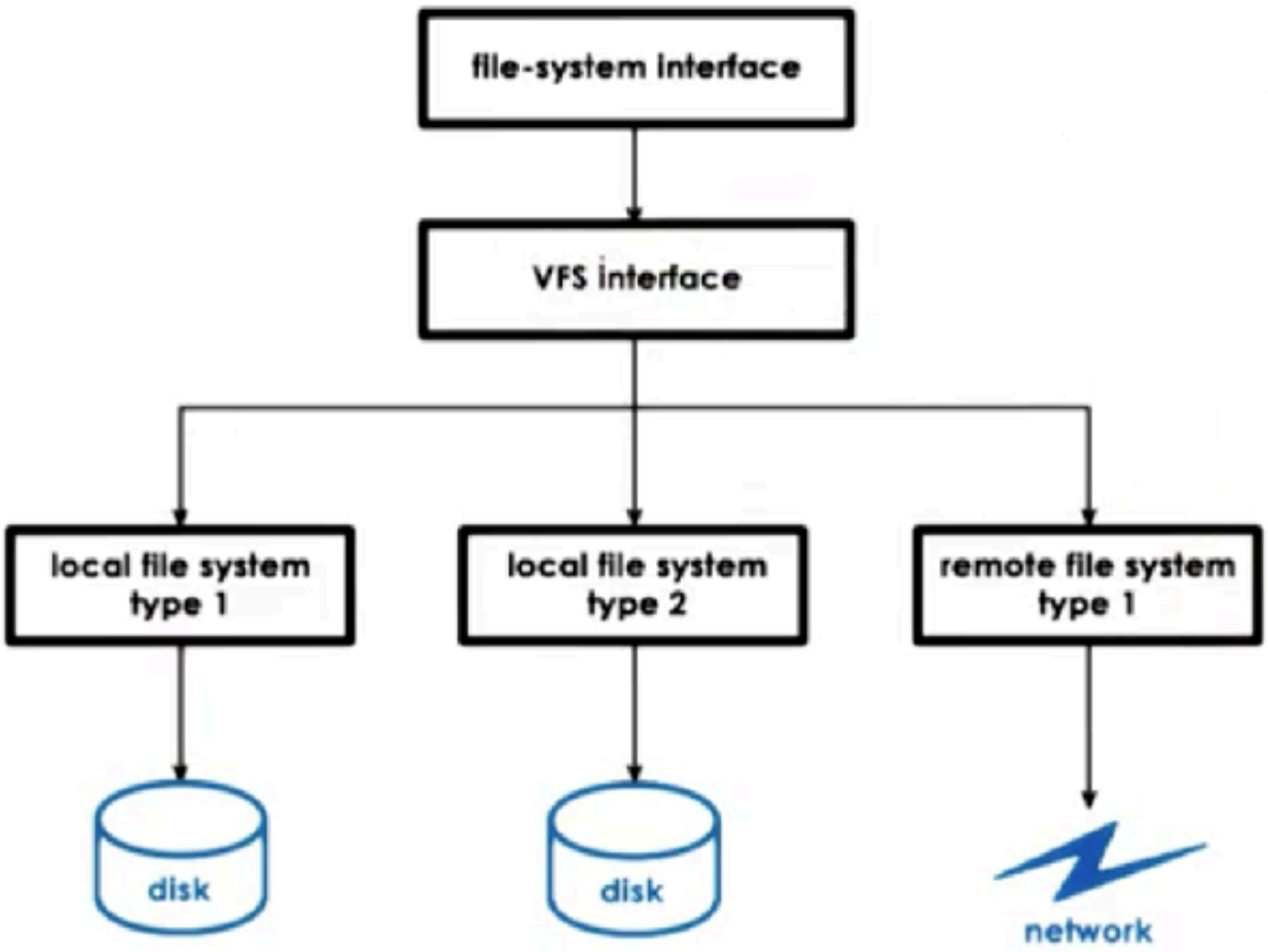
generic block layer
- OS standardized block interface

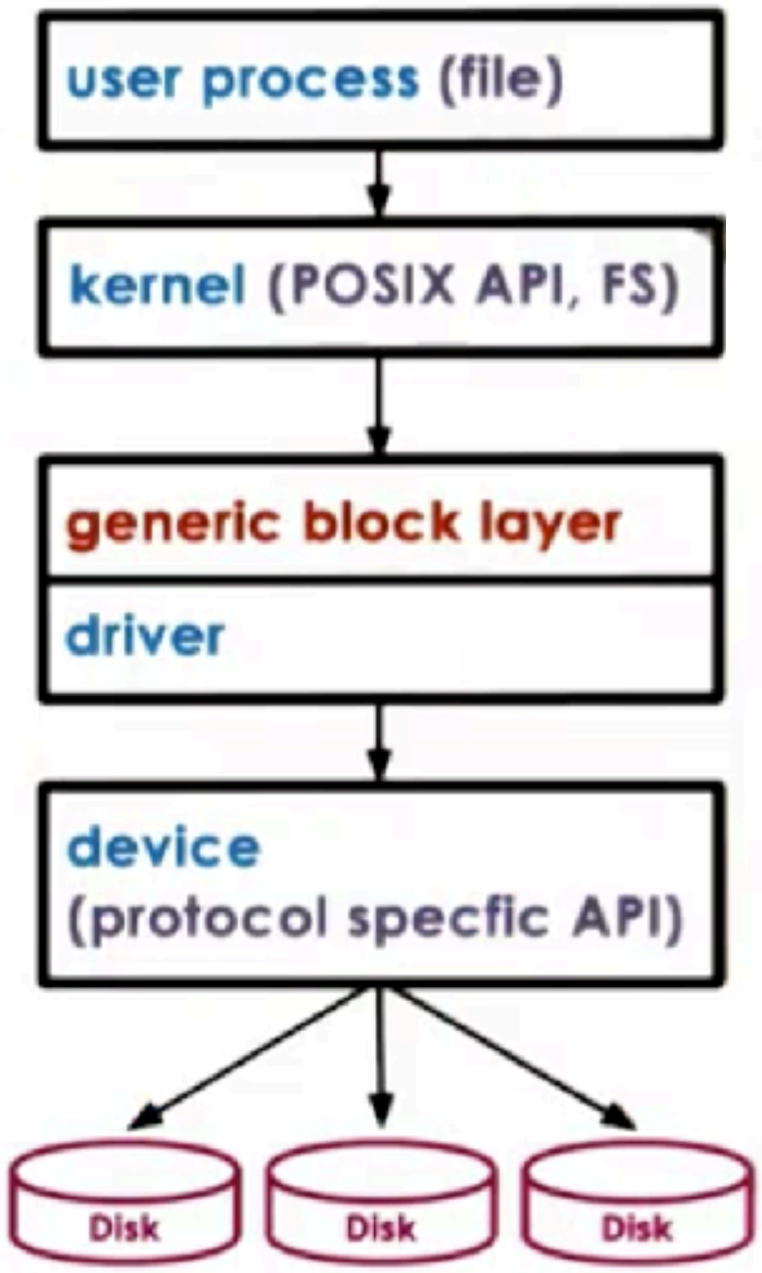
device driver



block device
typical storage
for files

Virtual File System





Virtual File System Abstractions

file == elements on which the VFS operates

file descriptor == OS representation of file

- open, read, write, sendfile, lock, close ...

inode == persistent representation of file "index"

- list of all data blocks
- device, permissions, size, ...

dentry == directory entry, corresponds to single path component

- /users/ada => /, /users, /users/ada
- dentry cache

superblock == filesystem-specific information regarding the FS layout

VFS on Disk

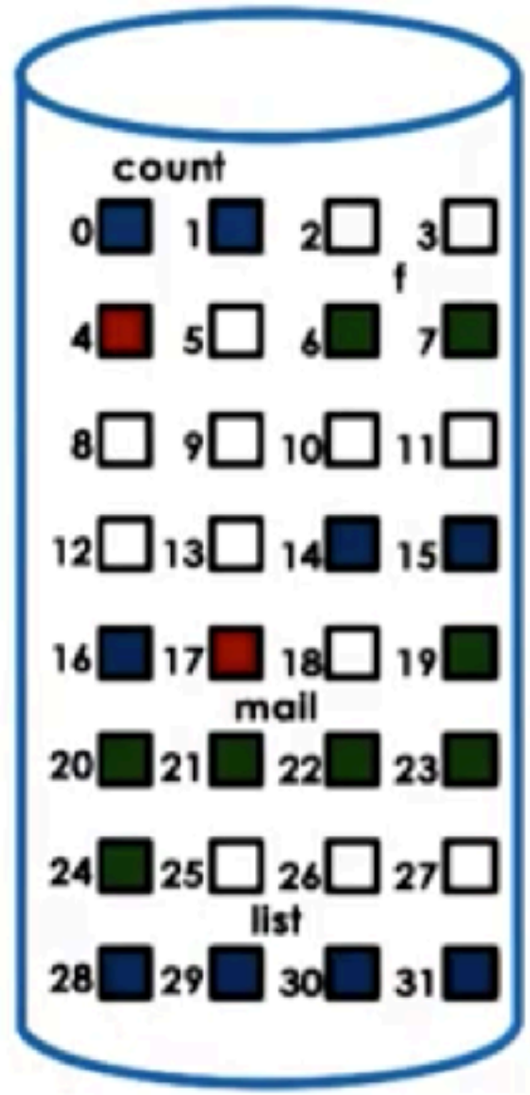
file => data blocks on disk

inode => track files' blocks

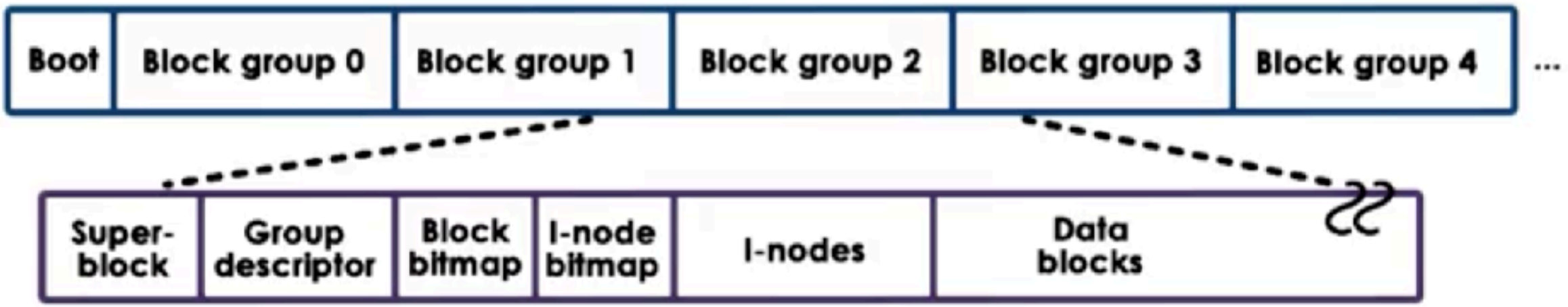
- also resides on disk in some block

superblock => overall map of disk blocks

- inode blocks
- data blocks
- free blocks



ext2: Second Extended Filesystem



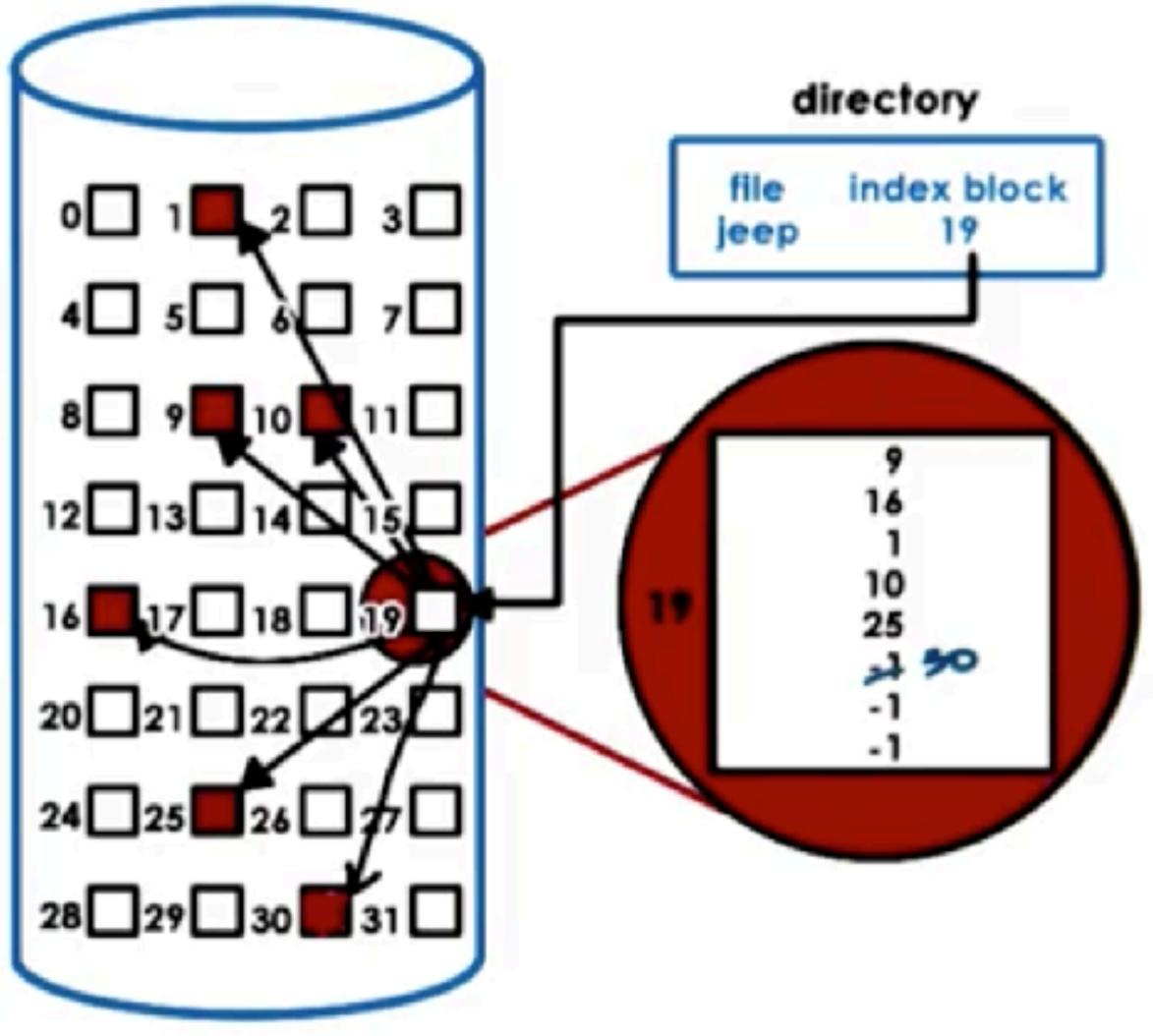
For each block group...

- superblock => #inodes, # disk blocks, start of free blocks
- group descriptor => bitmaps, # free nodes, # directories
- bitmaps => tracks free blocks and inodes
- inodes => 1 to max number, 1 per file
- data blocks => file data

inodes

== index of all disk blocks corresponding to a file

- file => identified by inode
- inode => list of all blocks
  + other meta data



⊕ easy to perform sequential or random access

⊖ limit on file size

e.g., 128 B inode, 4 byte block ptr

=> 32 addressible blocks x 1kB block

=> 32 kB file size

inodes with indirect pointers
== index of all disk blocks corresponding to a file

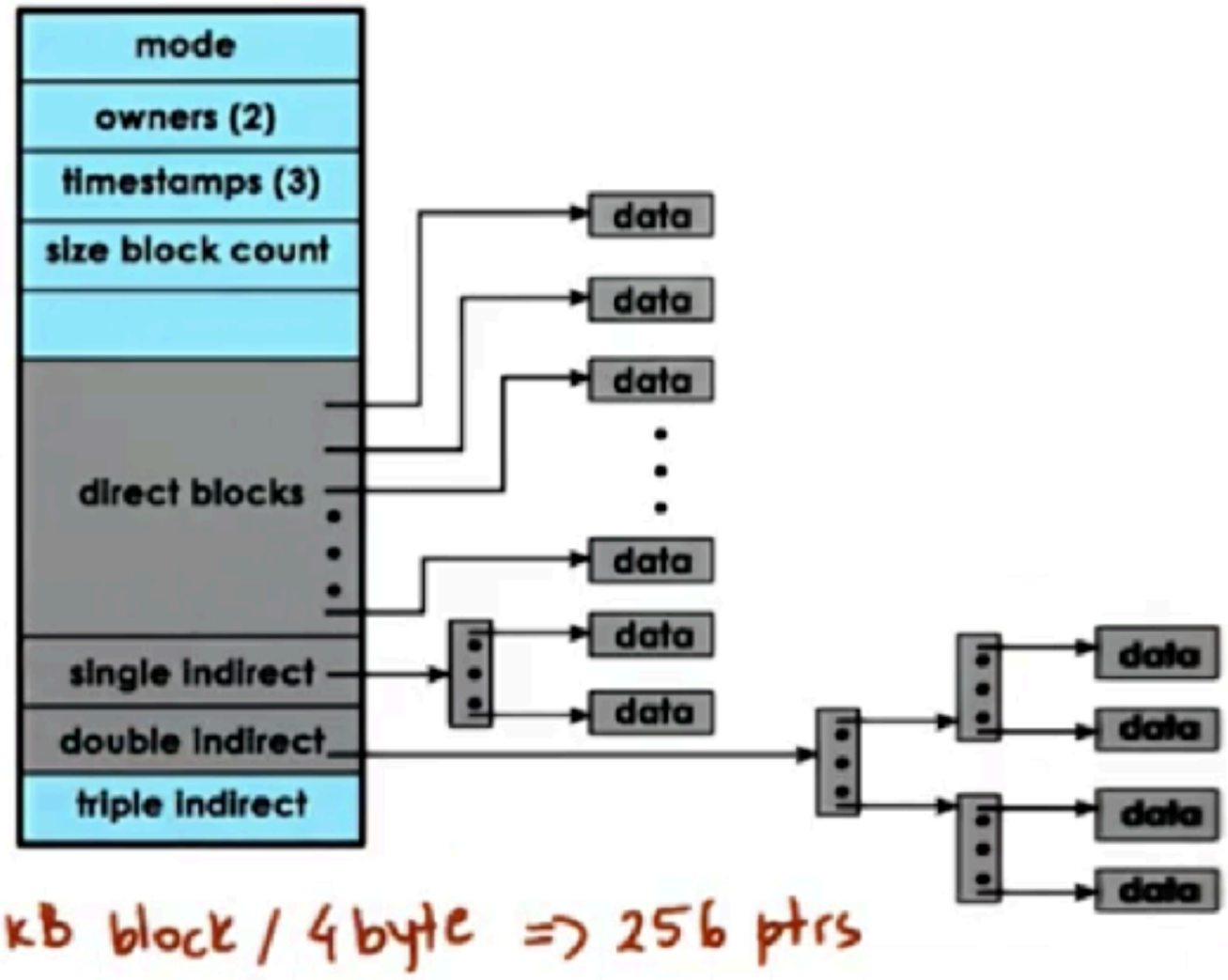
inodes contain...
- metadata
- pointers to blocks

e.g., 4B block ptr, 1kB blocks

Direct pointer "points to data block"
=> 1kB per entry

Indirect pointer "-⊬ block of pointers"
=> 256 kB per entry

Double indirect pointer "-⊬ block of block of pointers"
=> 64 MB



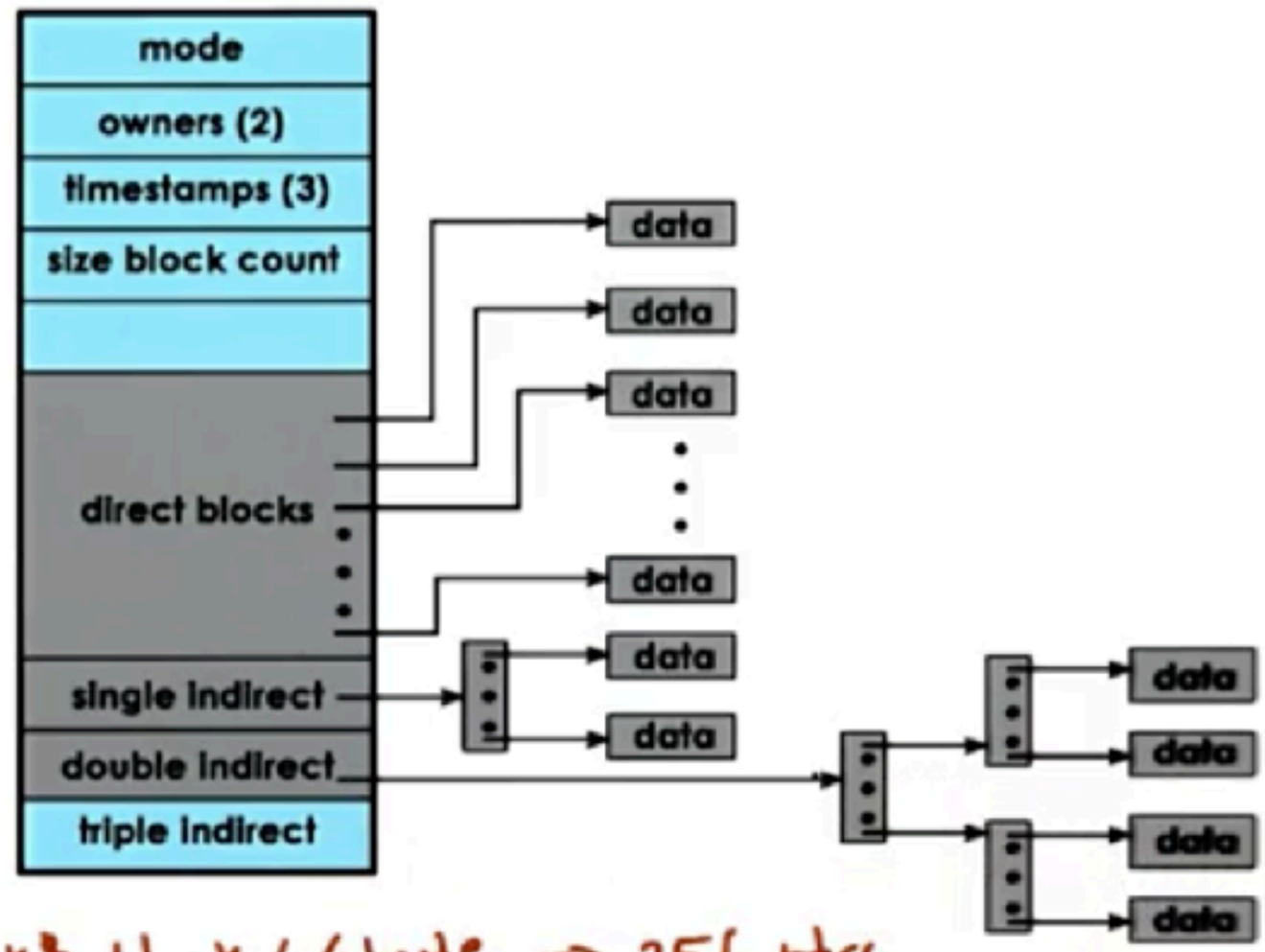
1kB block / 4byte => 256 ptrs

inodes with indirect pointers
== index of all disk blocks corresponding to a file

⊕ small inode ⇒ large file size

⊖ file access slowdown

e.g,
- direct ptr ⇒ 2 disk accesses
- double indirect ptr
  ⇒ up to 4 disk accesses



1kB block / 4byte ⇒ 256 ptrs

inode Quiz

An inode has the following structure:
Each block ptr is 4B.

If a block on disk is 1kB, what is the maximum file size that can be supported by this inode structure (nearest GB)?

1kB -> 256 ptrs

(12 + 256 + 256^2 + 256^3) x 1kB

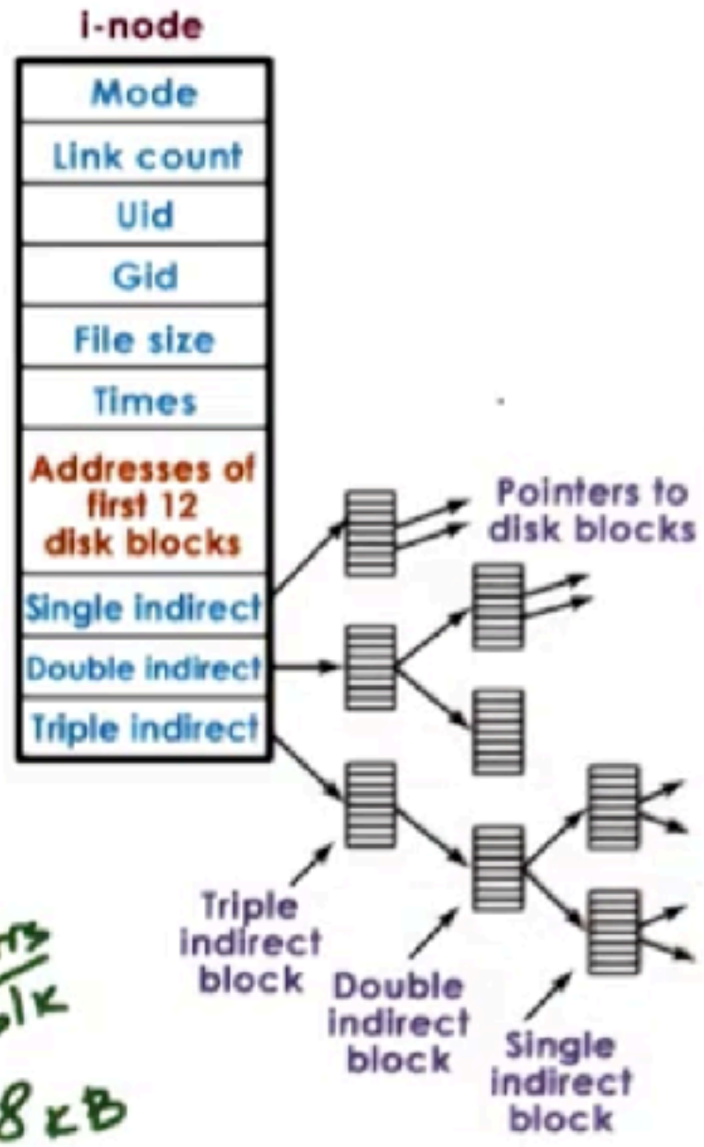
16	GB

What is the maximum file size if a block on disk is 8kB (nearest TB)?

8kB block / 4B ptr size = 2k ptrs/blk

(12 + 2k + 2k^2 + 2k^3) x 8kB

64	TB



Reducing File Access Overheads

caching / buffering => reduce # disk accesses
- buffer cache in main memory
- read / write from cache
- periodically flush to disk - fsync()

I/O scheduling => reduce disk head movement
- maximize sequential vs random access
- e.g., write block 25, write block 17 => write 17, 25

prefetching => increases cache hits
- leverages locality
- e.g., read block 17 => read also 18, 19

journaling / logging => reduce random access (ext3, ext4)
- "describe" write in log: block, offset, value ...
- periodically apply updates to proper disk locations