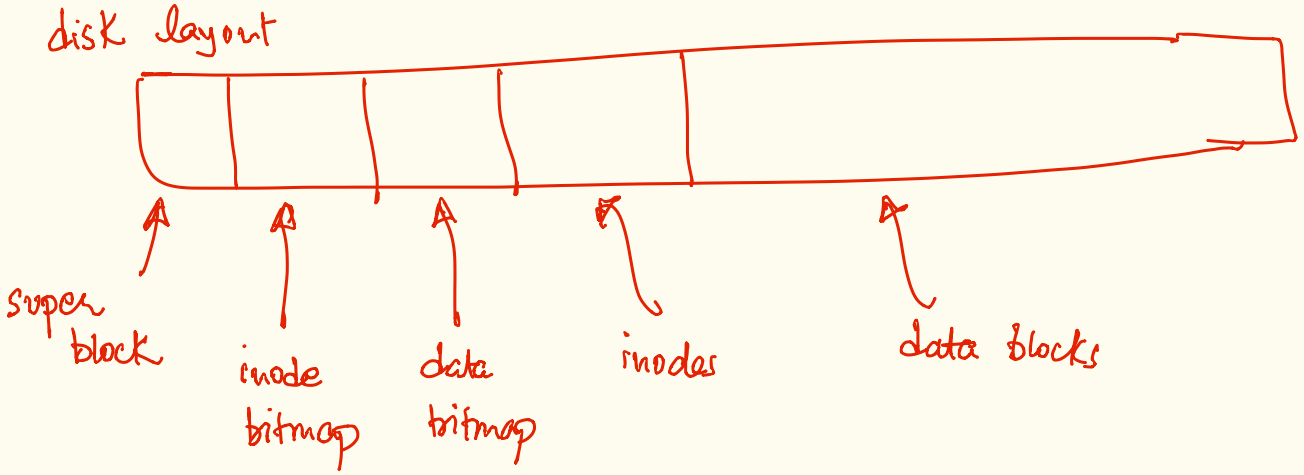
Lecture # 30 (?) CS 347/333 ⊛ some more file systems

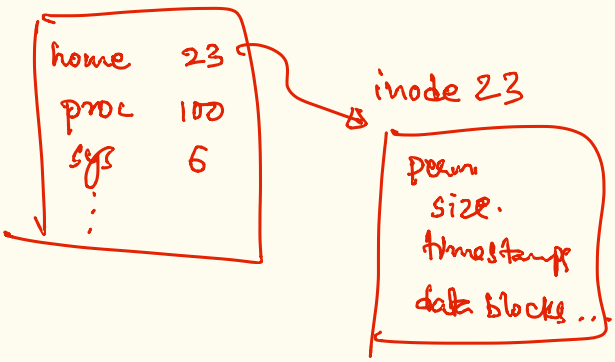


⊛ every file maps/has/represented by an inode.

HW whether an inode object/structure has a filename field in it? (Linux).

Q. What is a directory? — it is a file.?

/home/
/sys
/proc

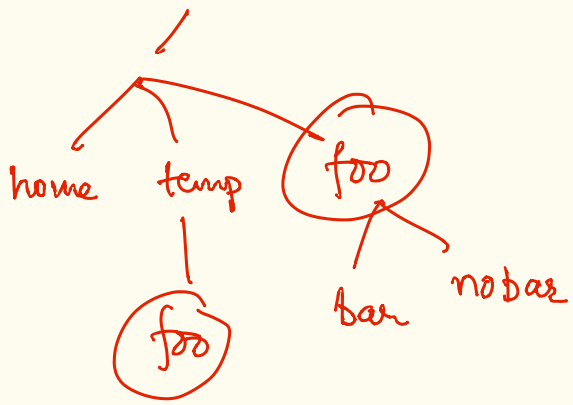


(#) What FS actions are invoked on file accesses via the FILE API.

fd = open("/foo/bar");

(disk) read root inode
read root data block
read foo inode
read foo data
read bar inode

/foo



5 disk accesses to open a file.

(#) CPU > Mem > IO

(#) open("/1/2/3/4/5/.../100/whatreally.txt");

~ 200+ disk accesses!

(#) FS caching techniques

(i) dentry cache (directory entry) } metadata
(ii) inode cache
(iii) page cache (buffer/disk/data cache) } data

(*) (i) dentry cache (in-memory)

maps directory entries to inode objects.

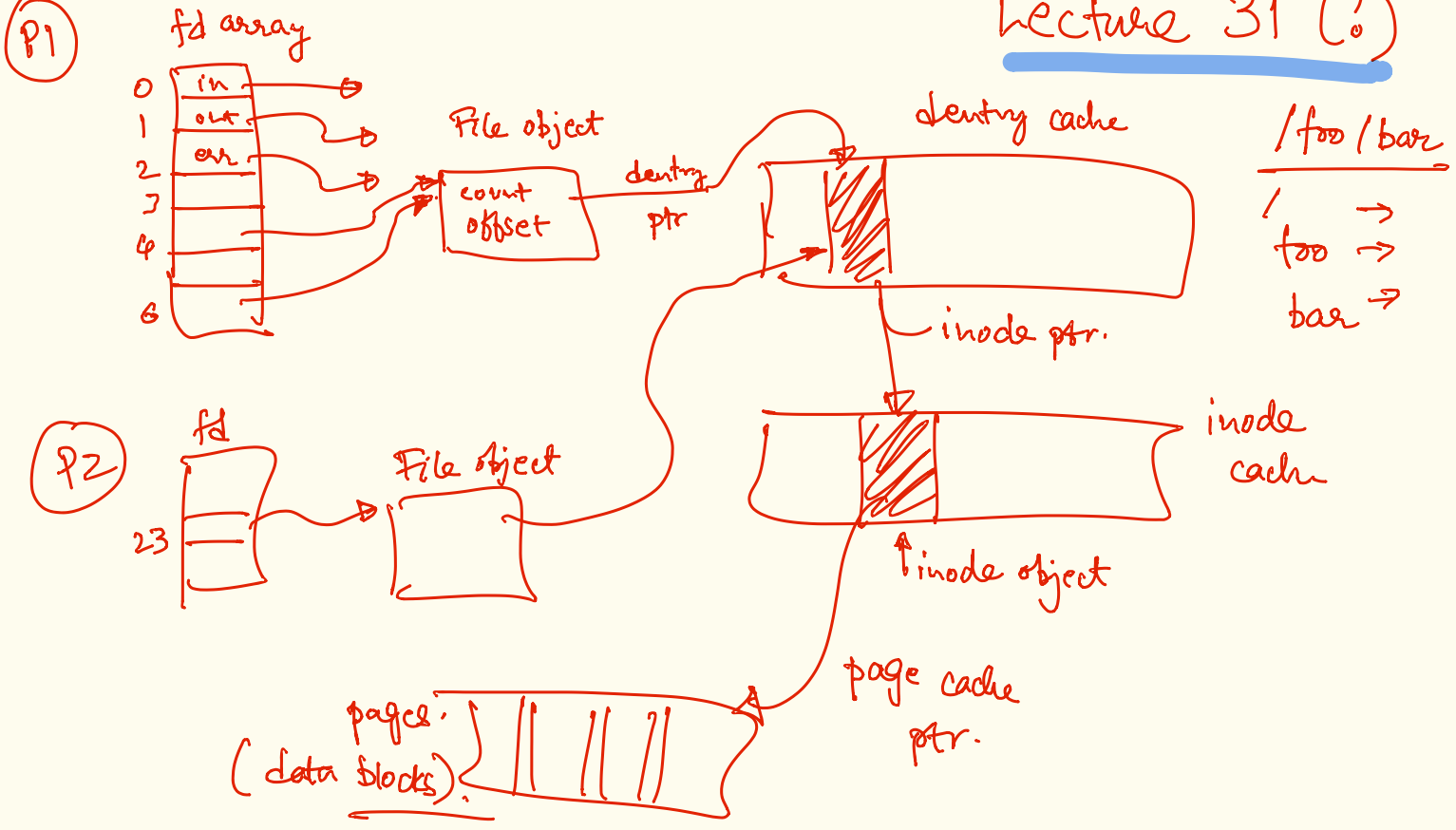
/ foo / bar

/ → inode object 2
foo → 16
bar → 23
} in-memory

(ii) inode cache (in-memory)

maps. inode number to inode objects.

Lecture 31 (?)



(*) pointer for page cache is stored in the in-memory inode object.

(*) - in-memory inode object is not the same / but is built using disk inode object.

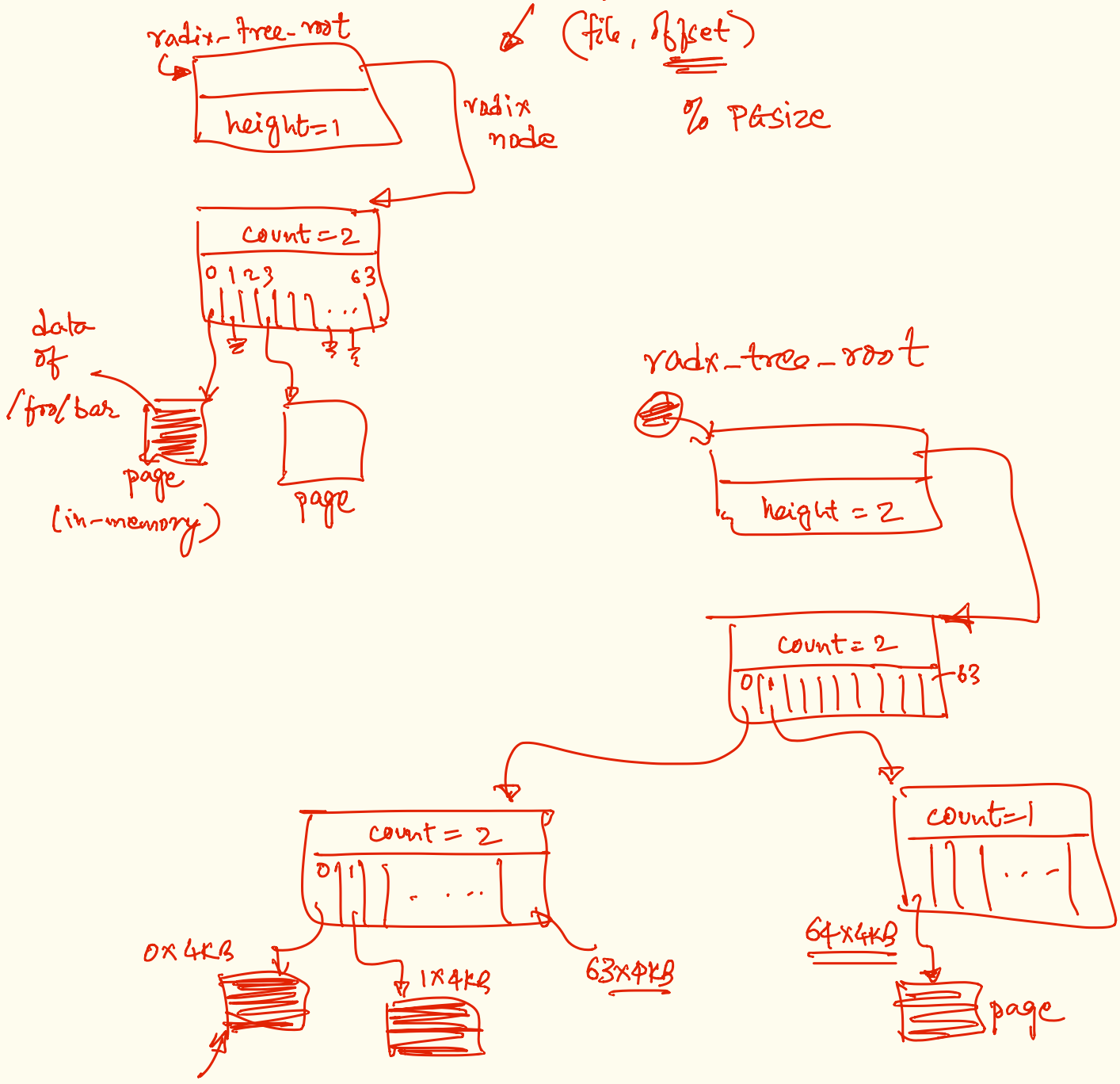
page cache

<fd, offset> ⇒ FS ⇒ (disk, blocknum)

user/app'n process view

page cache ||— in-memory store w/ page granularity objects.

(*) radix-tree page cache org. in Linux.



(#) caches : great for reads. — free (!) access

temporal & spatial locality can be exploited

└ not so much for writes.

issue ⇒ multiple copies of data being updated.

(i) cache invalidation

- invalidate / removes object in cache
- writes to disk

(ii) write through

- writes to cache & to disk synchronously & then returns from write.

(iii) write back

- lazy flush to disk.
- write returns on cache update.

fsync()

(*) fsck

virtualization

Lecture 32 (?) CS 333/CS 347

① file consistency invariant ⇒ always keep the file system "consistent"
- ~ all metadata & data are in-sync
- ~ all or nothing updates

② persistent storage ⇒ what you write is what you get, across restarts (if write succeeds).

~ crash-consistency problem / consistent update problem } ~ many disk writes for a single FS action.
~ crashes are non-deterministic.

① pro-active approach
e.g: journaling

② lazy approach
e.g: fsck file system checker
consistency check
~ let the world go by
~ if crash/restart
- check FS consistency
- fix as best as possible.

fsck.

write to a file. :

	inode	data bitmap	data block	
C:	✓	✓	✓	
C:	x	x	x	
IC:	x	x	✓	←
IC:	✓	x	x	⊛
IC:	x	✓	x	~
(no action) C:	✓	✓	x	
IC:	✓	x	✓	write is lost.
IC:	x	✓	✓	⊛ as data bitmap reset.

cannot identify, no action.

inconsistency detected: data bitmap: data is garbage

detectable: data bitmap: reset

appl", bad inconsistency: FS not detectable

detectable: ~~set~~ data bitmap

(*) fsck. = needs to know file system specifications

- superblock ~ verify integrity of super block.
- free data blocks ~ are all data blocks that are set pointed to by some inode.
- inodes ~ verify type of inode - checks vm
- inode links ~ how many files linked to this inode?
- duplicates ~ data blocks pointed by more than one inode.
- bad blocks - r/w checks
  - block offsets within valid range.
- directory checks — loops
  - "." & ".." are first two entries of each directory file.

(*) VM files / images

- vmdk
  img
  vbx