i] ~~install-trans~~ → commit → i] write_log
(in memory log → disk
bcread (last present in mem
transfer data

ii] write_head → a] read disk log block (header)
b] get pointer to log block
c] fill which blocks will be
written
d] write this block

iii] install_trans → disk log → actual loc^n
(cache → cache)

iv] write_head → clear log.

* new layer

1] balloc
a] read bitmap block
b] find free & mark allocated
c] write bitmap to log
d] return index of new block

2] bfree
a] get bitmap → free bit → write bitmap
b] free blocks

III] Inode → Icache DS
a] iget
find if already present in icache
else find empty make ref = 1 & return
only DS

b] ~~dot~~ iupdate
i] write through every update
i] get on disk inode
ii] modify
iii] write back (to log)

c] ialloc - i] allocate a free inode ~~from disk memory~~
from disk using bitmap
ii] call iget to get cache block (in memory)

d] ilock → copy details from in disk inode
to in memory inode

e] itrunc → free all block (direct + indirect)
using bfree

f] bmap → allocate block using balloc

g] writei, readi → take inode & offset & r/w

V

&

directory entry

client {
~~uint~~ ushort inum; ⇒ inode no. ⇒ 2 bytes
char name [DIRSIZ]; ⇒ 14 - 11
}

1] IDE → DS. idequeue

1. ideinit → does havedisk = 1 setup if FS img presents

2. idewait ⇒ wait for IDE disk to be ready

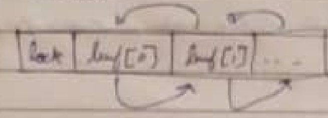
3. idestart ⇒ start reading first buffer in queue

If B_DIRTY the write else read

4. ideintr ⇒ 1] make current block valid, remove dirty list & remove this block from queue
2] wake up process which slept by calling I/O
3] Start I/O for next buffer

5. iderw
1] BVALID should be 0. (because bread calls after checking BVALID
2] If first node → idestart
3] else sleep.

2nd layer ⇒ Buffer cache
DCLL → MRU



head → MRU, prev → LRU

1] bget → If cache, refcnt++ (Traverse MRU)
else pick block with refcnt = 0 & dirty = 0 (Traverse LRU)
only gives bcache block

2] bread - call ~~iderw~~ bget (get buffer) & then call iderw
return this buffer

3] bwrite - write b to disk
1] set B_DIRTY 2] call iderw

4] brelse - 1] remove buffer & add to head.

3rd layer ⇒ logging

in memory log ⟷ disk log block → disk file loc^n
(pointer to buffer cache)

1] initlog ⇒ set pointers, recovery

2] begin op ⇒ sleep if committing or exceed avail. block
else outstanding++
(10 per process max)
(3 process simultaneously)

3] end op ⇒ outstanding--, if 0, commit
wake sleeping (in begin op)
i.e. (4th process)