CS 537 - Lecture 13

1. Final exam – Dec 15th (Fri) – 5:30 pm – 7:30 pm
   Only concurrency & Persistence.
2. P5 – 2 late days (max).
3. 2 worksheets – dropped.
   10 " – graded.

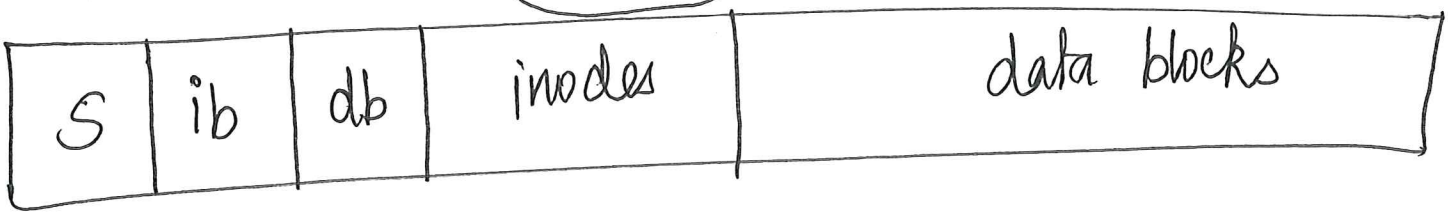
---

1. Review of FS < data structures, access methods
2. Crash consistency.
   - fsck
   - Journaling
3. RAID.

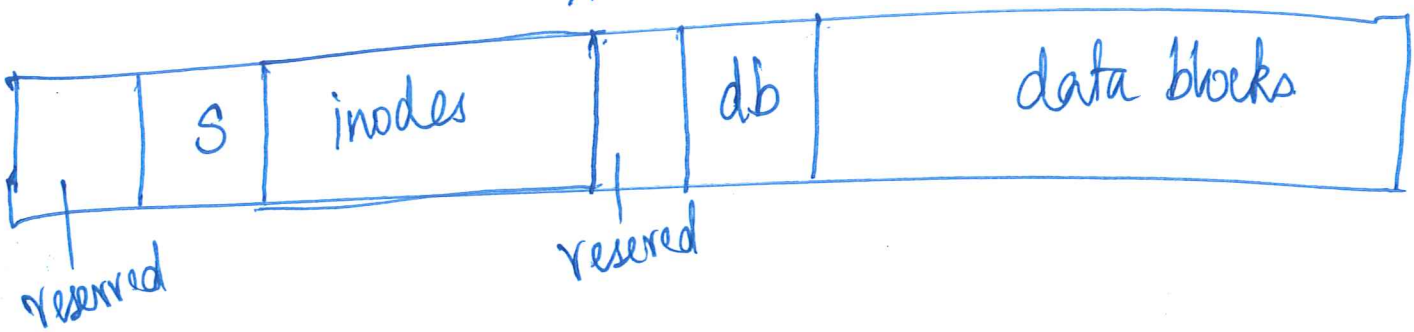
~~HDD~~ ✓ RAID Flash-SSD.

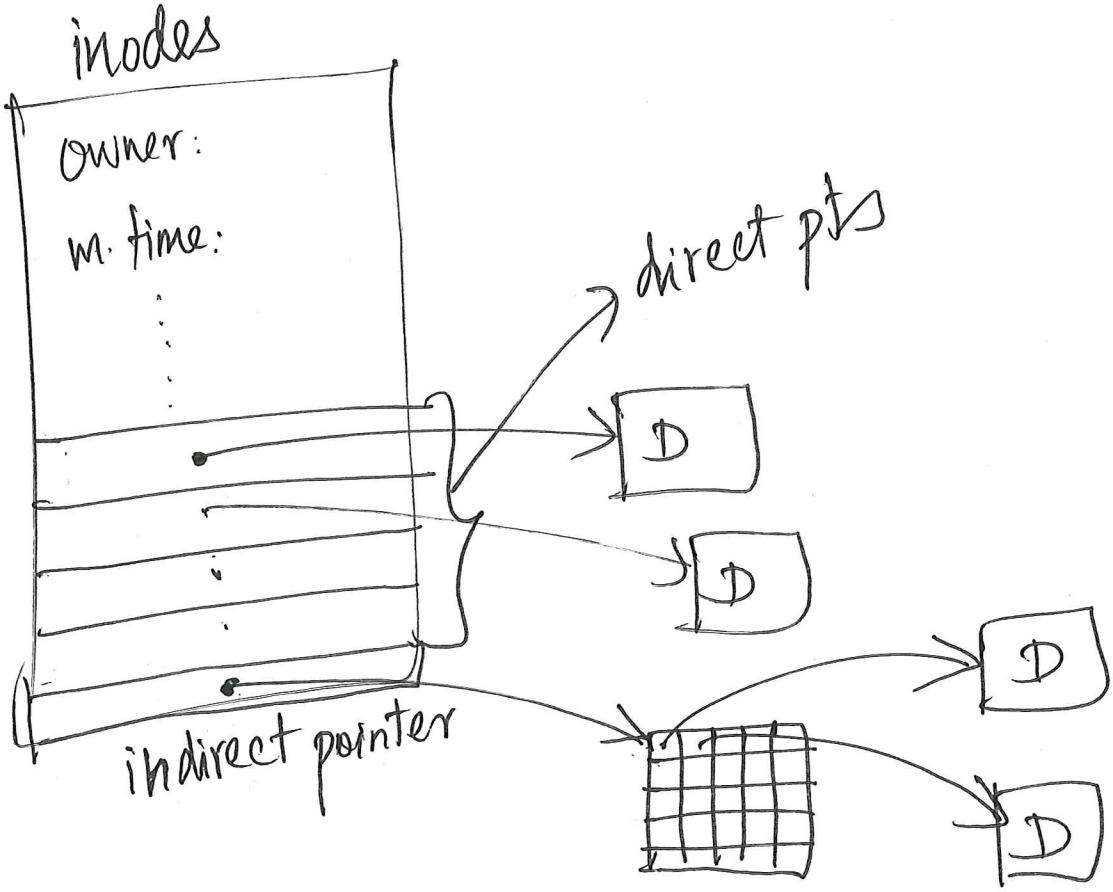
File System

VSFS ✓



XV6





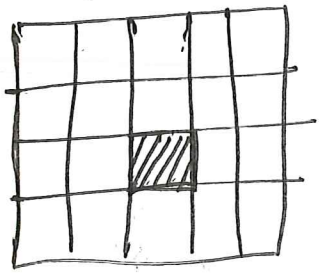
contents of root directory

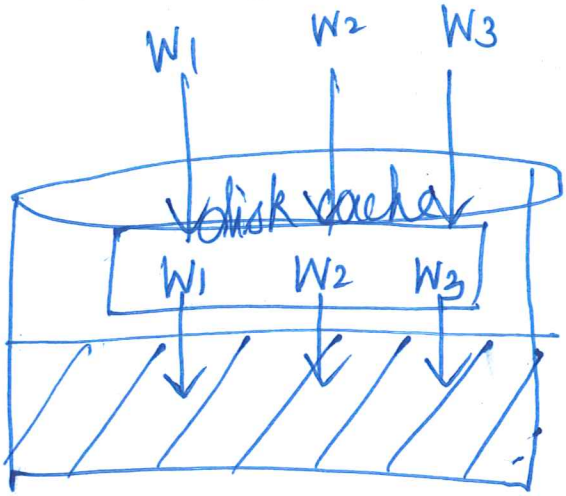
name	inode
.	2
..	2
foo	3

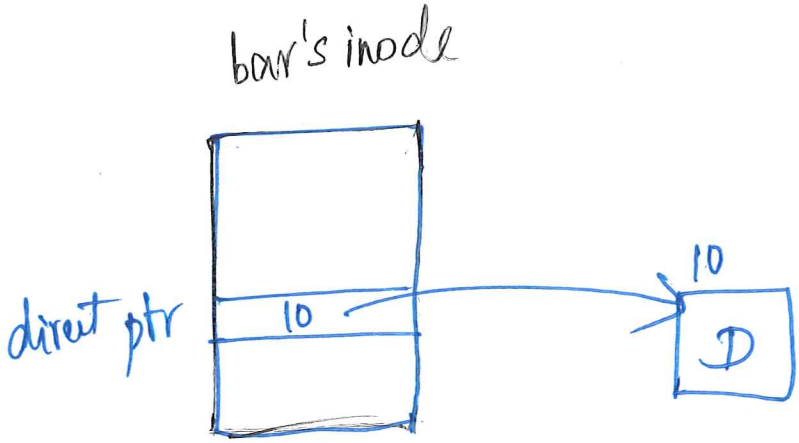
inode # of root = 2.

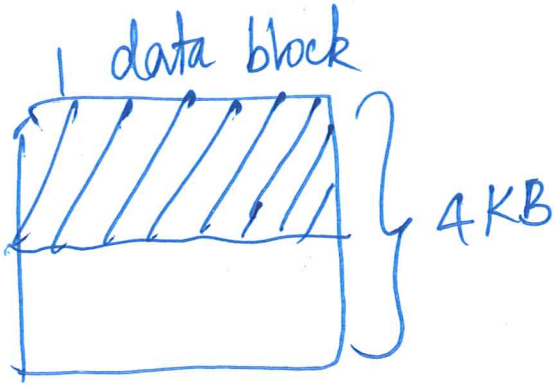
foo's dir contents

.	3
..	2
bar.txt	4





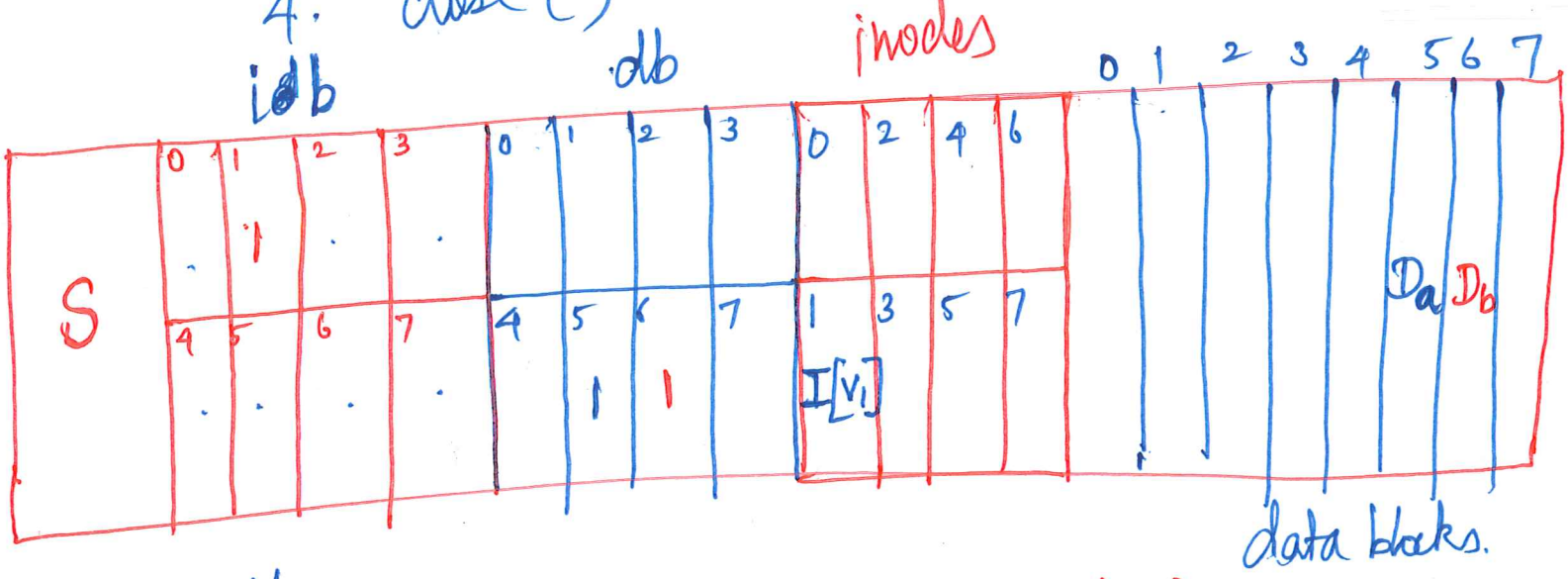




Crash Consistency

Append

1. open()
2. lseek() to end of file
3. issue a 4KB write
4. close()



V1

ib = 0100 0000

db = 0000 0100

Blocks to update?

1. New data block (D_b)
2. Data bitmap.

   0000 0100 → 0000 0110

   db[V1] db[V2]

3. Inode of the file.

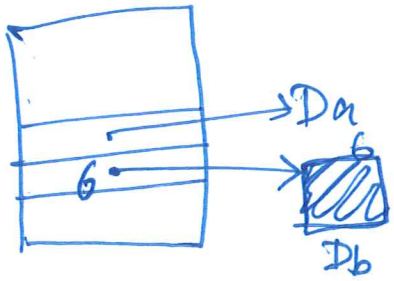
   I[V1] → I[V2]

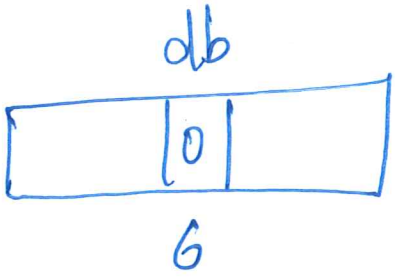
   → 2nd direct ptr
   → # blocks: 1 → 2
   → size: 4KB → 8KB
   → modification time
   :

Crash Scenarios

1. only D_b is written.
   - consistent w.r.t. FS.

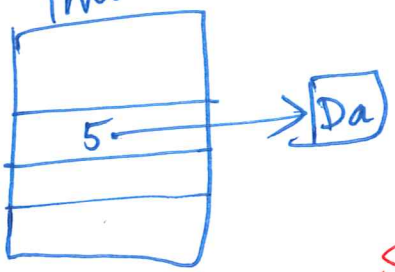
2. only Inode is written.

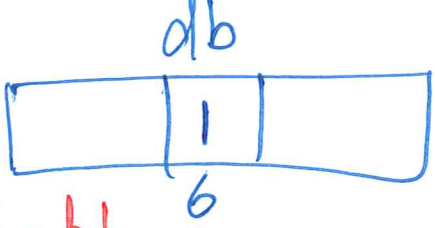




   - inconsistent.

3. only the db[v_2] is written.

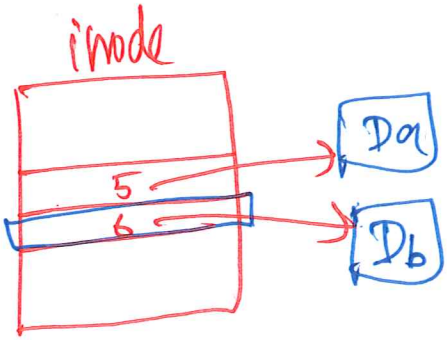


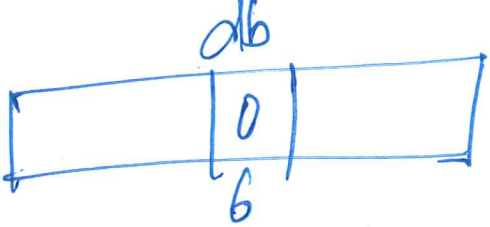


space leak!

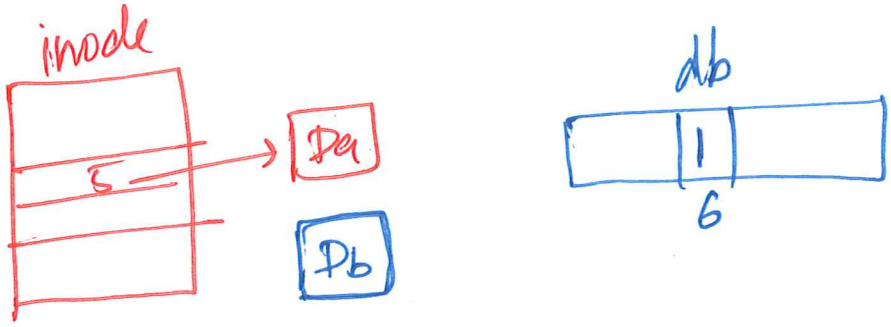
   - inconsistent state

4. D_b and I[v_2]. are written.

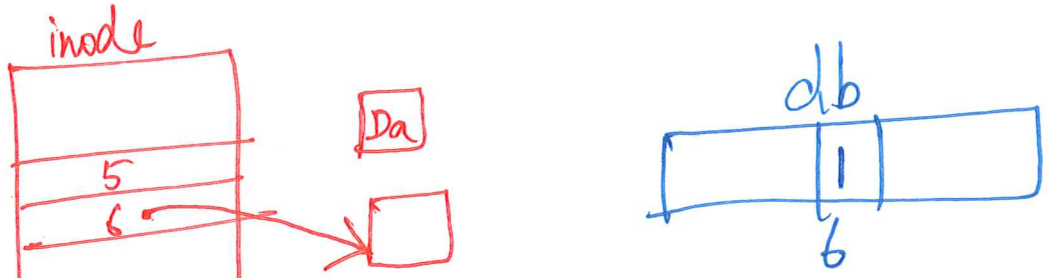




5. D_b and $db[v_2]$.



6. $I[v_2]$ and $db[v_2]$.



Consistent-update problem.

FS —atomically→ FS

FS consistent state 1

FS consistent state 2

FSCK

→ detect & recover.

→ no other FS activity allowed.

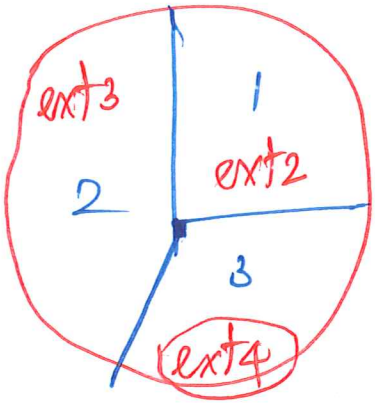
→ fsck - run before mounting the FS.

ASIDE:

mkfs - make FS.

mkfs -t ext2 /dev/sda1

(/dev/sda1: device)



sd → SCSI Device type.

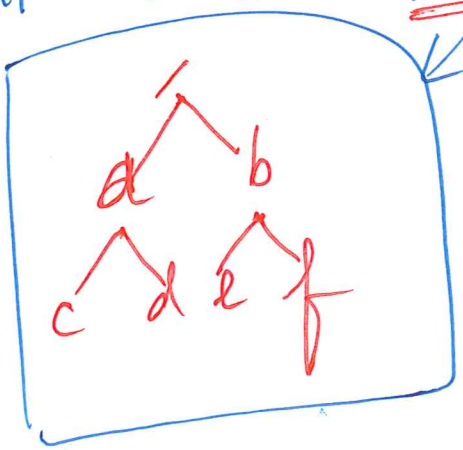
a, b, c, ... Z, Aa, Ab..., Az → disk # (order).

1, 2, 3, ... n → partition # with the disk.

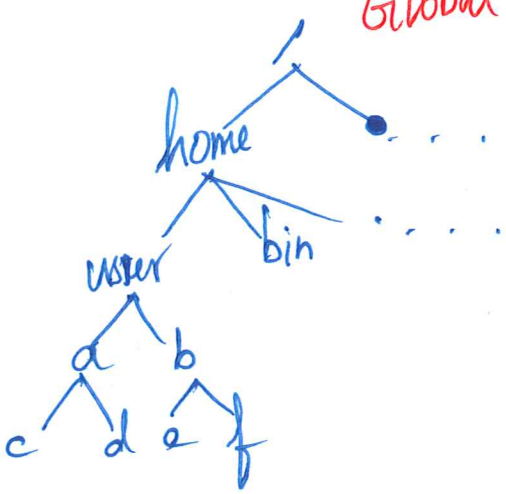
---

mount()

mount -t ext2 /dev/sda1 /home/user



Global FS tree.

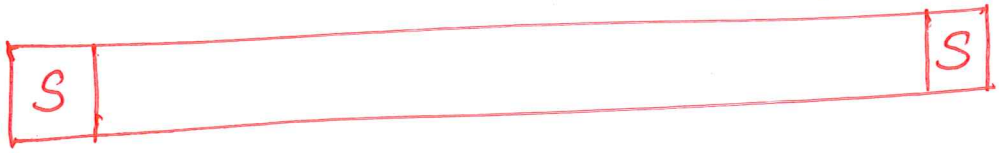


FSCK

1. **Superblock.**

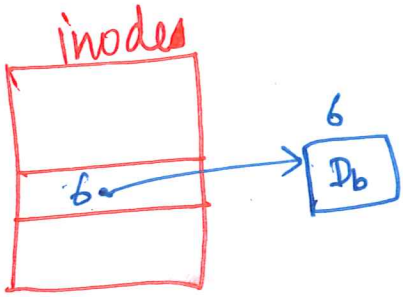
   sanity checks: FS size > # allocated blocks.

   – alternative copy of Superblock.

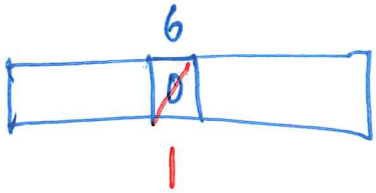


2. **Freeblocks**

   inodes



   bitmaps



   → trust the inode.

3. **Inode State**

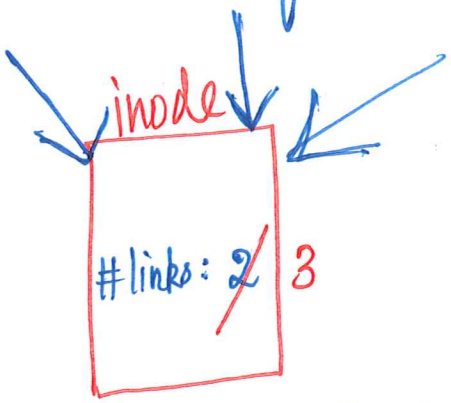
   type: dir, reg file, sym. link.

   ⇒ free the inode

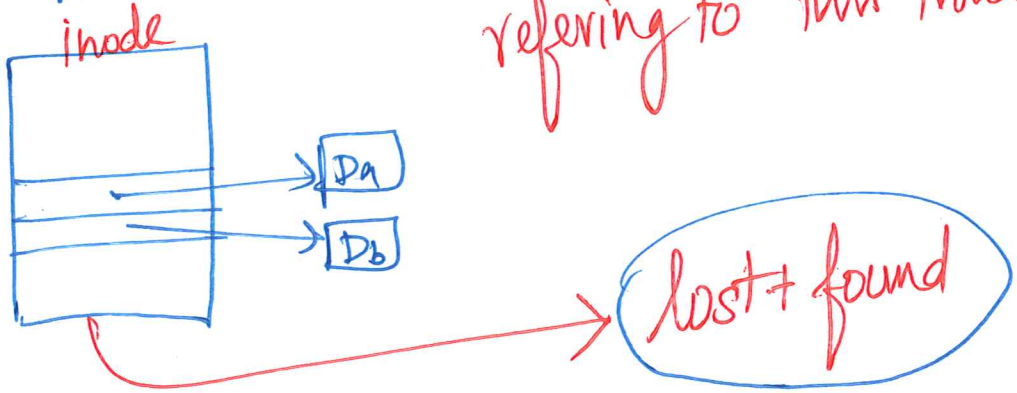
   + update the inode bitmap.

4. Inode links

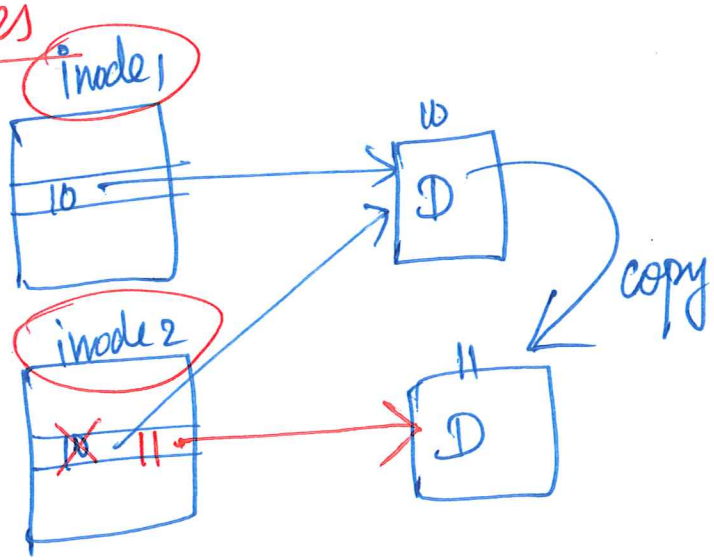
- scan the entire FS and update the link counts for each inode.



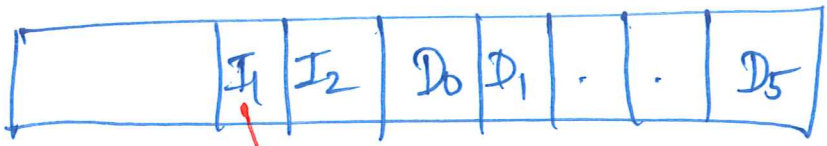
- inode is allocated but no dir is refering to this inode.



5. Duplicates



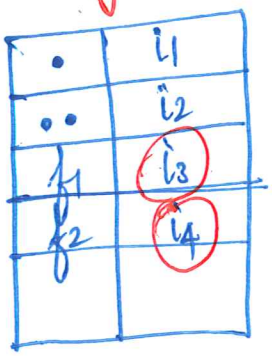
6. Bad blocks



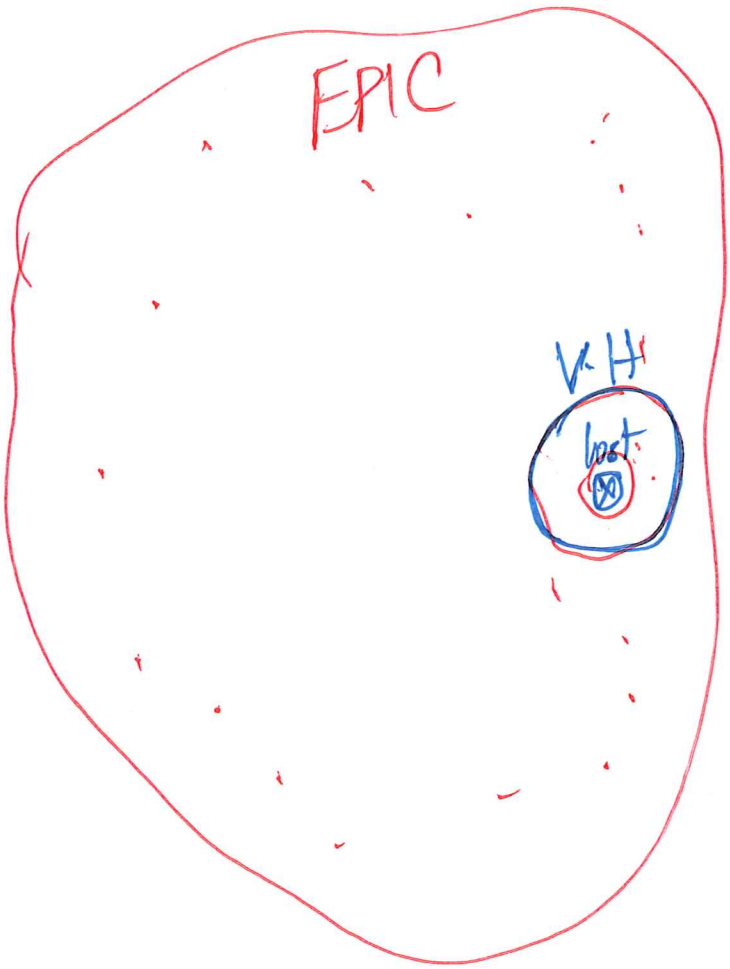
non-existent data block.

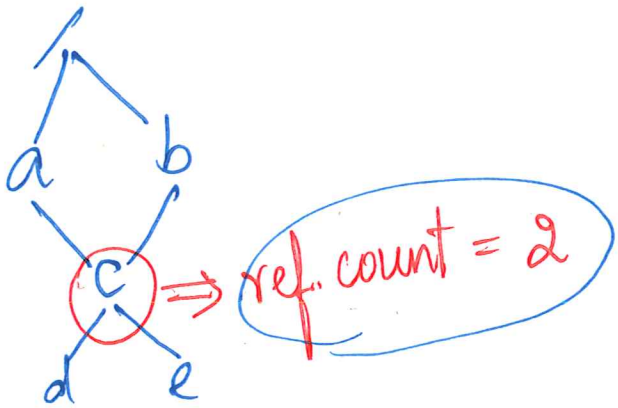
- clear the pointer to the data blak.

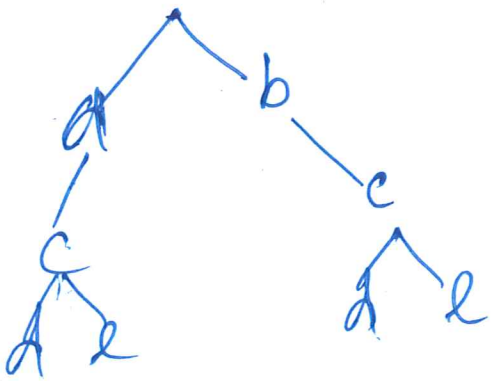
7. Directory checks



- first two entries are . and ..





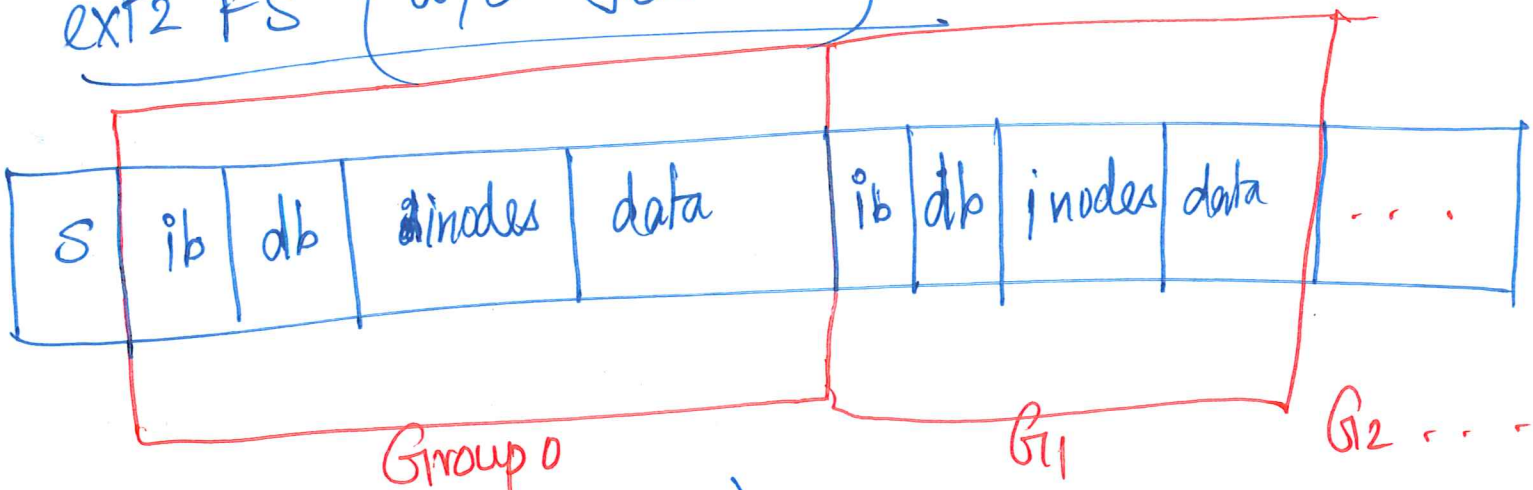


Data Journaling

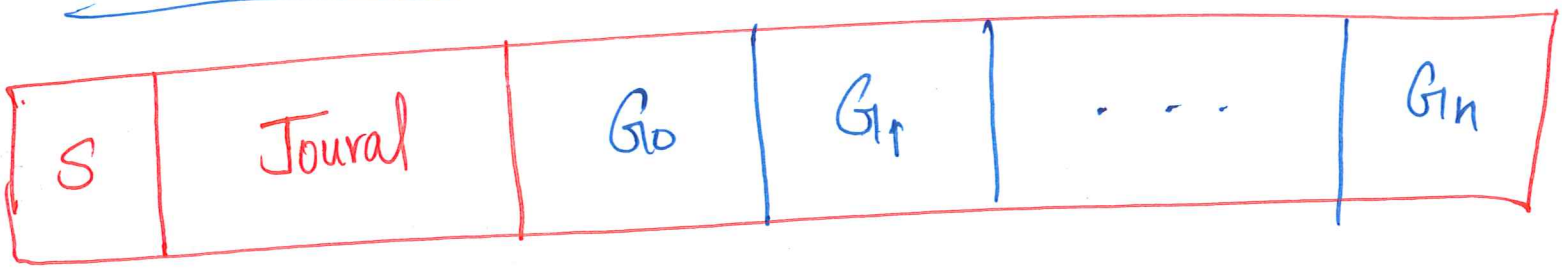
note: write to the journal. → also on disk.

Write to the FS

ext2 FS (w/o Journal)



ext3 FS (w/ Journal)



Workload

1. Write $I[v_2]$
2. " $db[v_2]$
3. " Data D_b.

Journal

Journal

final addrs. of the 3 write + TID

TID

1. Journal Write ✓
2. Checkpoint

write the metadata & data to the FS proper.

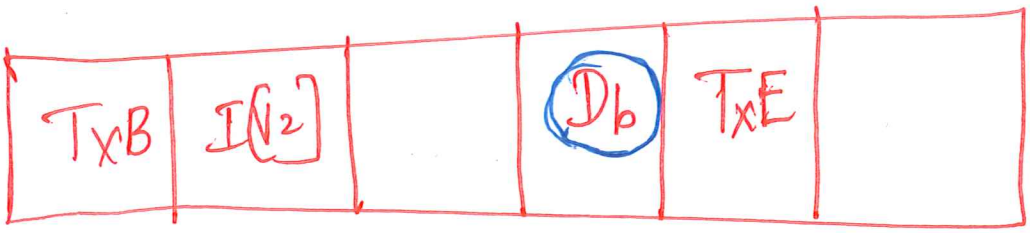
Crash

1) Write I[v2]
2) Write db[v2]
3) " Db

Crash recovery

replaying the tx.

Crash during Journal Write.



- db[v2] is NOT written.
- crash happen.

Protocol

1. Journal Write (TxB, metadata, data). WAIT.
   I[v2], db[v2] Db

WRITE BARRIER.

2. Journal Commit (Write TxE). WAIT.

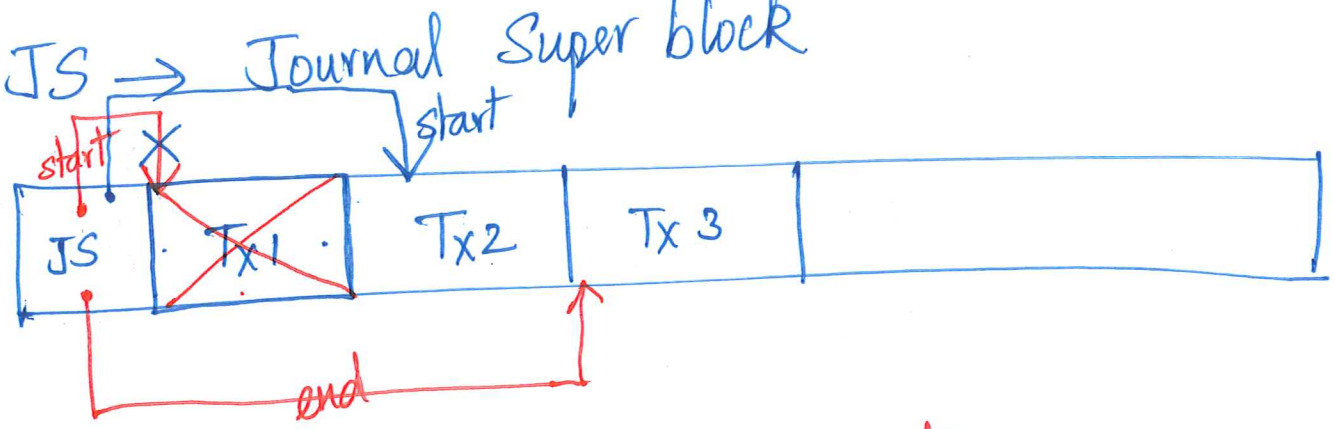
WRITE BARRIER.

3. Checkpoint

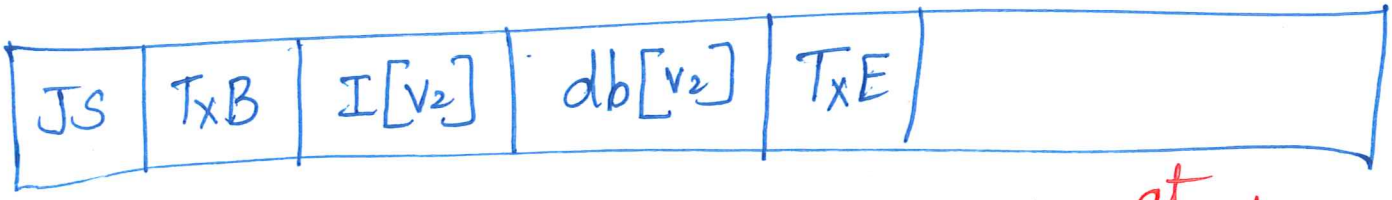
NO WRITE BARRIER

4. Free the transaction on the journal.
   - updating the journal super block.

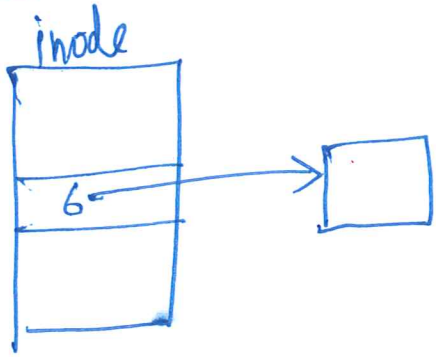
How to make the journal finite?

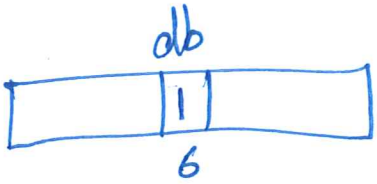


Metadata Journaling



→ Data block (Db) is written only ~~to~~ at its actual location.





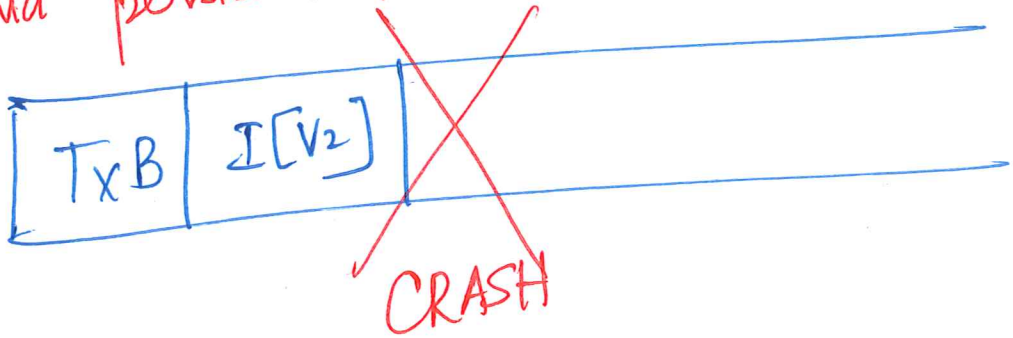
Metadata Journaling

1. Journal Write (TxB, metadata)
2. Joural Commit (TxE)
3. checkpoint
4. Free

When to write the data block?

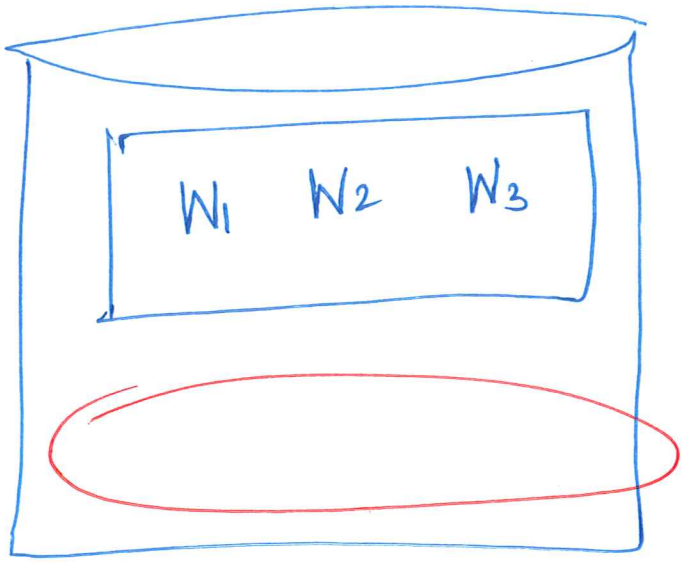
1. Write data block. (D_b). WAIT (optional)
2. Journal metadata write (TxB, metadata). WAIT
3. Journal commit. (TxE). WAIT
4. checkpoint the metadata.
5. Free the Tx from the journal.

→ Data persisted to disk.



fsync(fd);

fd = open( );
write( ); W1
write( ); W2
write( ); W3

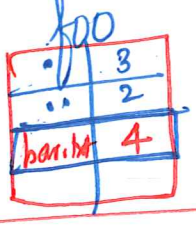


write(fd); W1
fsync(fd);
write(fd); W2
fsync(fd);

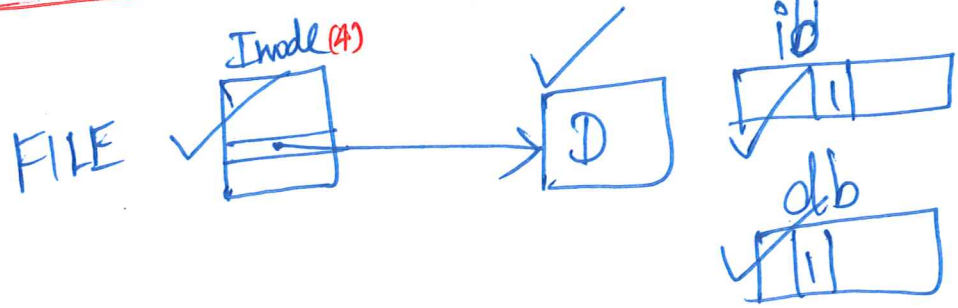
1. create a new file
2. write to the file
3. fsync.

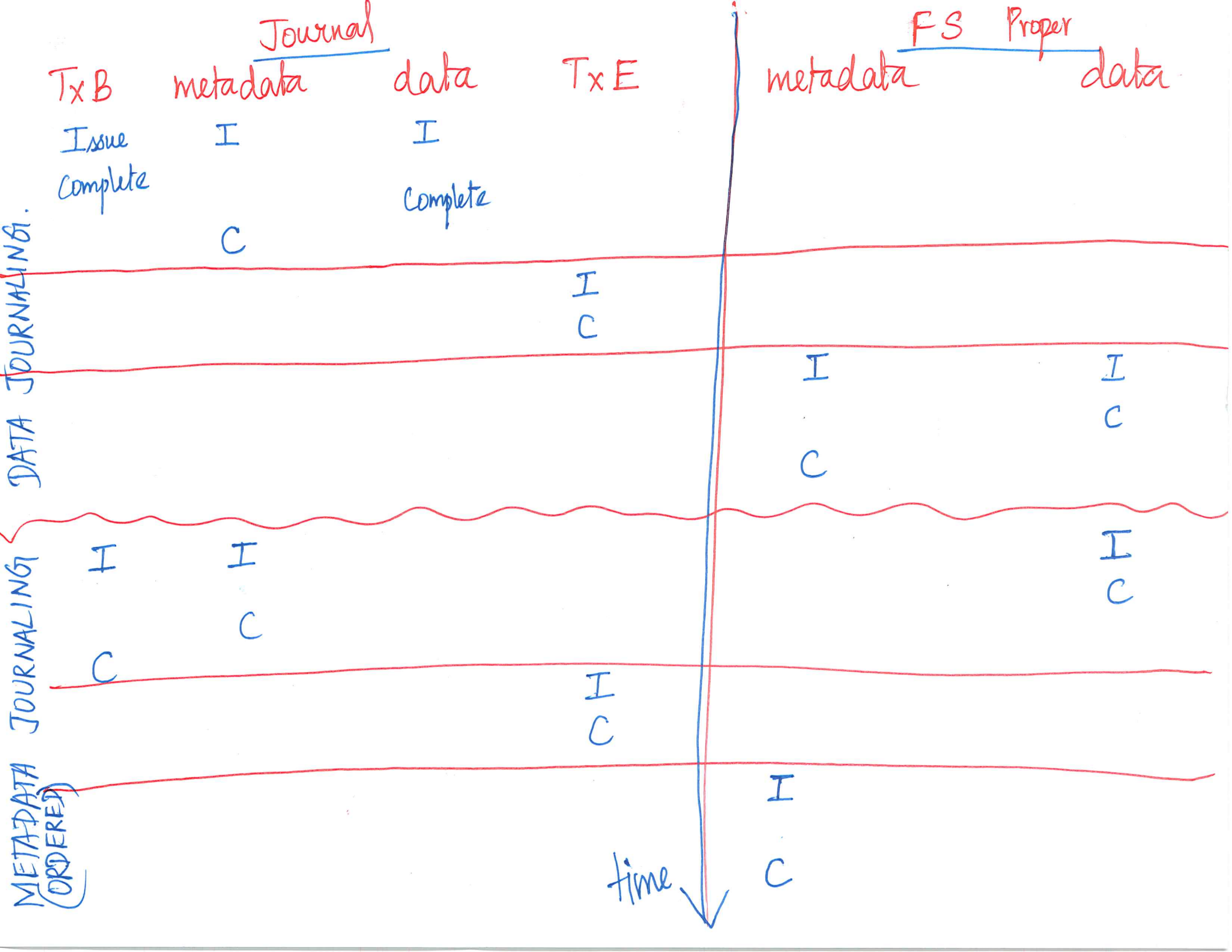
CRASH

DIRECTORY.



/foo/bar.txt




Journal
FS Proper
TxB
metadata
data
TxE
metadata
data
Issue
Complete
I
I
Complete
C
I
C
I
C
I
C
DATA JOURNALING
I
I
C
C
I
C
I
C
METADATA JOURNALING (ORDERED)
time