SYSTEMS GOSSIP MEETUP

Learning large systems using peer-to-peer gossip

Policy Against Harassment at ACM Activities

https://www.acm.org/about-acm/policy-against-harassment

OS Meetup wants to encourage and preserve this open exchange of ideas, which requires an environment that enables all to participate without fear of personal harassment. We define harassment to include specific unacceptable factors and behaviors listed in the ACM’s policy against harassment. Unacceptable behavior will not be tolerated.

Last Time

- Filesystem in Xv6
- Crash recovery in Xv6

Last Time

- Filesystem in Xv6
- Crash recovery in Xv6

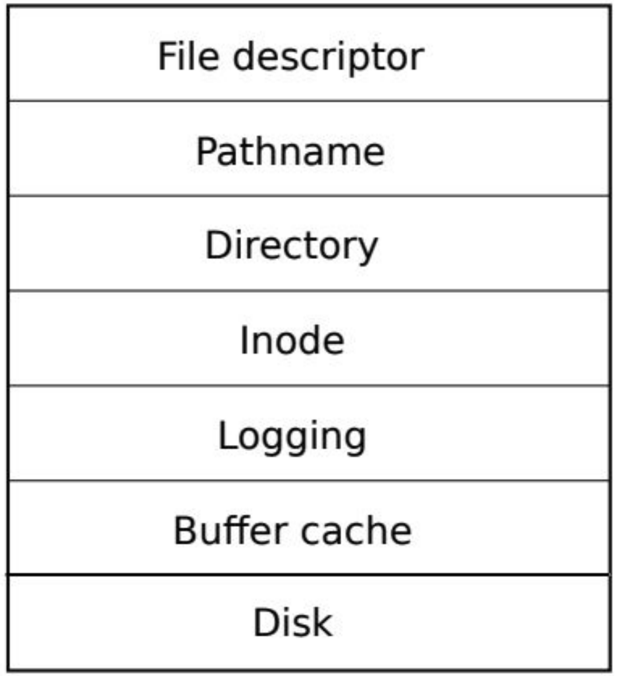


Figure 8.1: Layers of the xv6 file system.

Last Time

- Filesystem in Xv6
- Crash recovery in Xv6

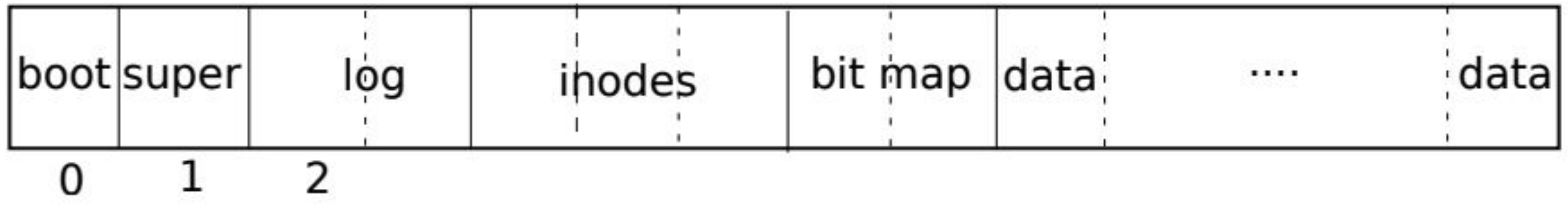


Figure 8.2: Structure of the xv6 file system.

Today

- Unix Ext3 Filesystem

Trade-off

- Xv6 logging:
    - Each system call is a transaction
    - At the end of the syscall
    - Write modified blocks to log on disk
    - Write blocks #s and done to log on disk
    - Write blocks to home location
    - Erase log
- Immediately durable, but slow
    - must commit after every syscall
    - must write to home location after commit

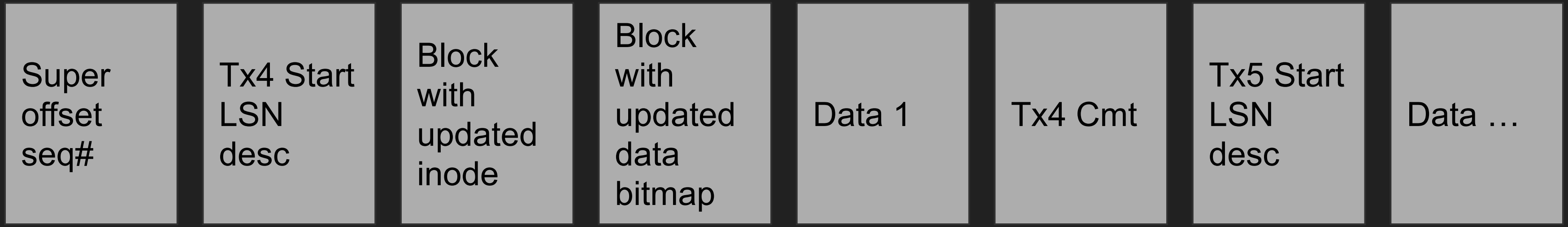
File System Transactions

- Transaction is a series of operations that should be treated as one logical operation. Either all should be committed or none should be committed (All-or-Nothing)
- In Database, transactions are described using ACID
    - Atomicity
    - Consistency
    - Isolation
    - Durability

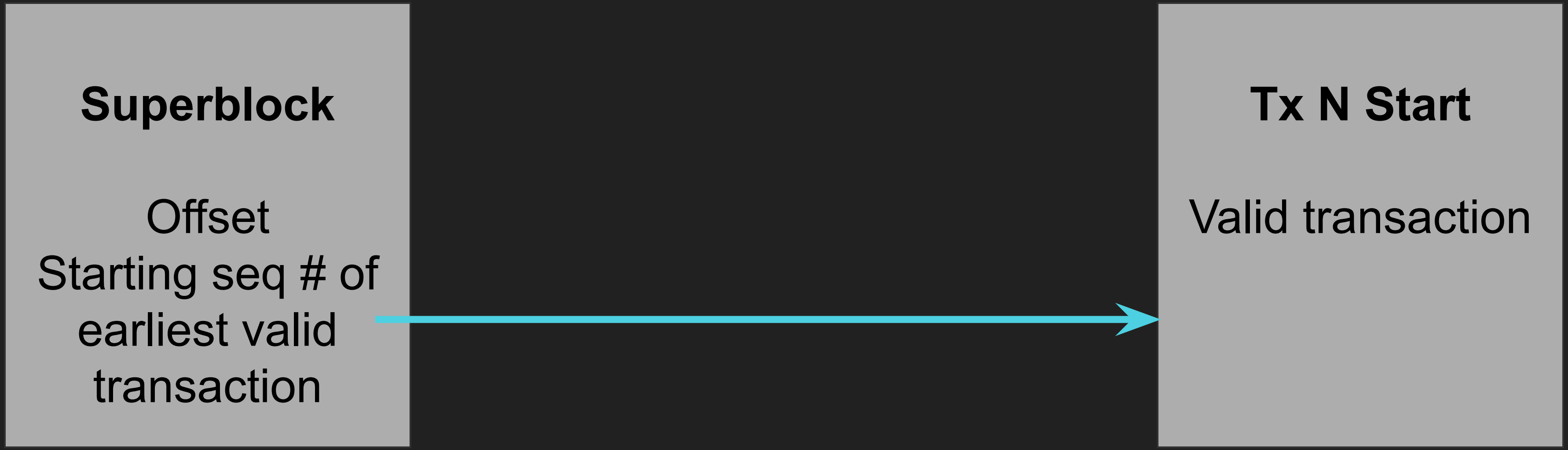
Ext3

- Uses redo write-ahead logging for crash recovery
- In memory:
    - Block cache
    - Per-transaction bookkeeping
- On Disk:
    - Filesystem
    - Circular Log

Ext3 Log


Super offset seq#
Tx4 Start LSN desc
Block with updated inode
Block with updated data bitmap
Data 1
Tx4 Cmt
Tx5 Start LSN desc
Data …

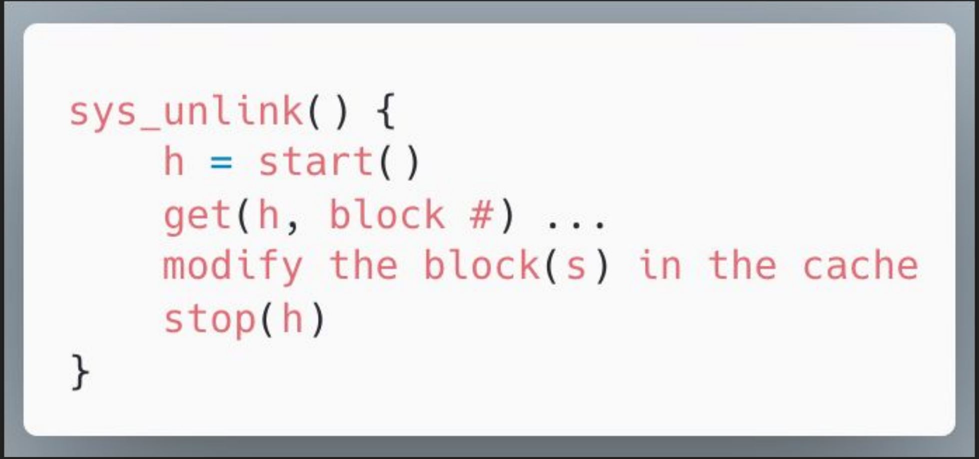
Ext3 Log


Superblock
Offset
Starting seq # of earliest valid transaction
Tx N Start
Valid transaction

Redo Write-ahead Logging

Pre-crash:

- Write blocks that would be updated into the log
- Once all blocks are in the log, txn is committed. Ext3 issues an async writes to home blocks and metadata blocks

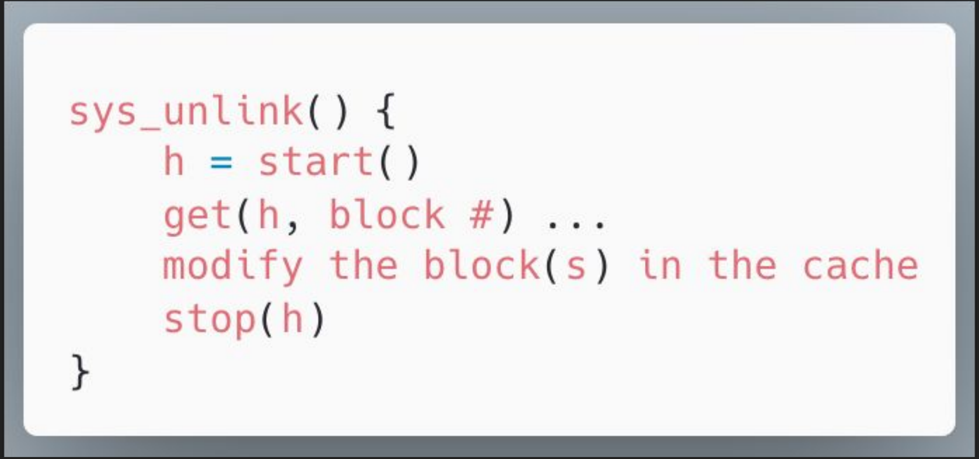


```
sys_unlink() {
    h = start()
    get(h, block #) ...
    modify the block(s) in the cache
    stop(h)
}
```

Redo Write-ahead Logging

Post-crash:

- **Scan the WAL , replay all the committed transactions and ignore uncommitted transactions.**



```
sys_unlink() {
    h = start()
    get(h, block #) ...
    modify the block(s) in the cache
    stop(h)
}
```

Concurrent transactions

- Ext3 allows multiple transactions to concurrent execute
- Some fully committed transactions in the log
- Some doing log writes as part of the commit
- One open transaction thats accepting new syscalls.

WAL Committing

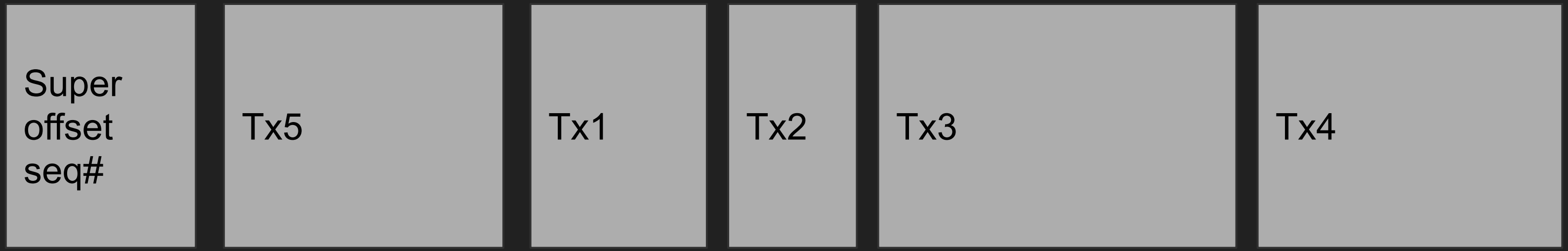
- Block new syscalls
- Wait for in-progress syscalls to `stop()`
- Open a new transaction, unblock system calls
- Write descriptor to log on disk with a list of block numbers
- Write each block from cache to log on disk
- Wait
- Write the commit record
- Wait
- Async write cached blocks to go to their home locations

WAL Committing

Why do we have to issue separately for Commit?

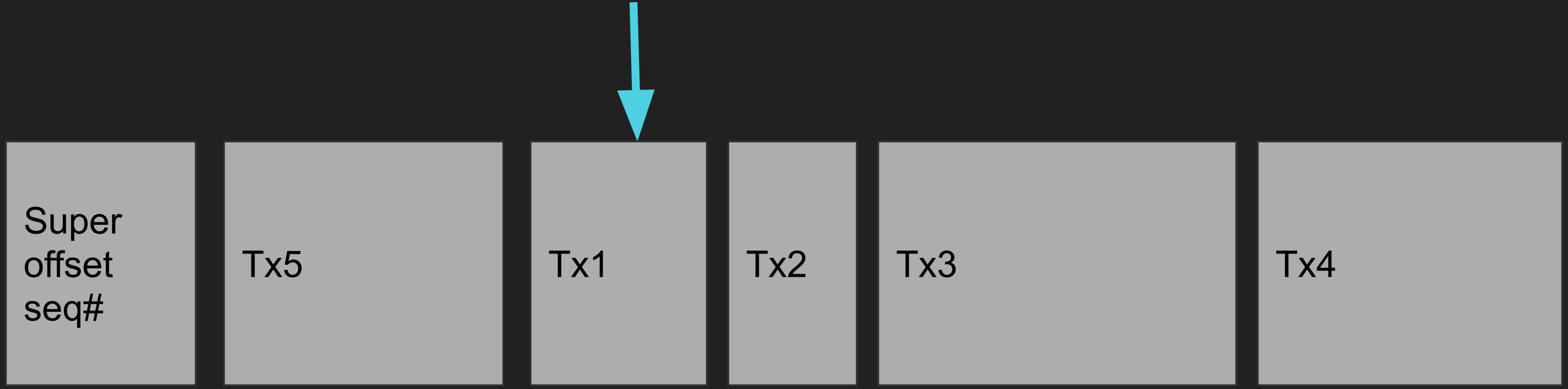
- The kernel will issue all of the log writes at once. The disk can reorder writes, which may cause issue if all records except the data record complete…
- During recovery phase, the filesystem will see that the above transaction has been committed, so a data block will permanently be lost.

WAL Committing



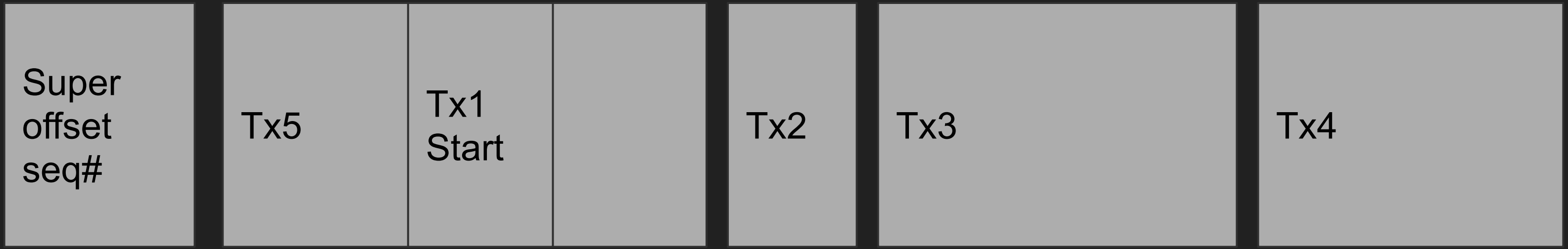
WAL Committing

Can be reused iff Tx1 committed and all blocks are written to home locations



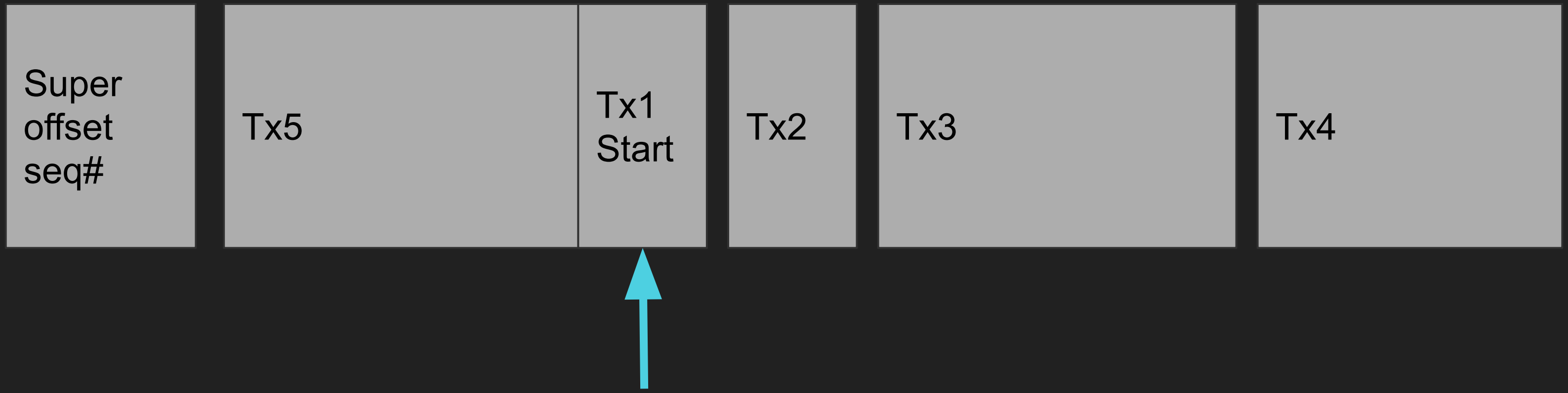
Corner Case A

What if block after last valid commit block looks like a log descriptor (Txn Start)?



Corner Case A

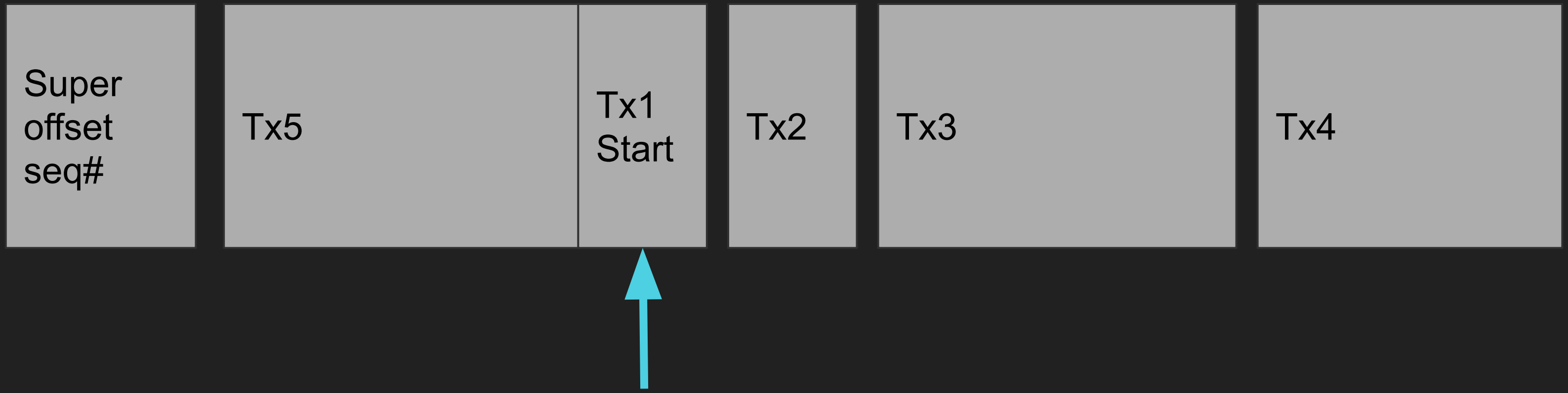
What if block after last valid commit block looks like a log descriptor (Txn Start)?



Magic Number

Corner Case A

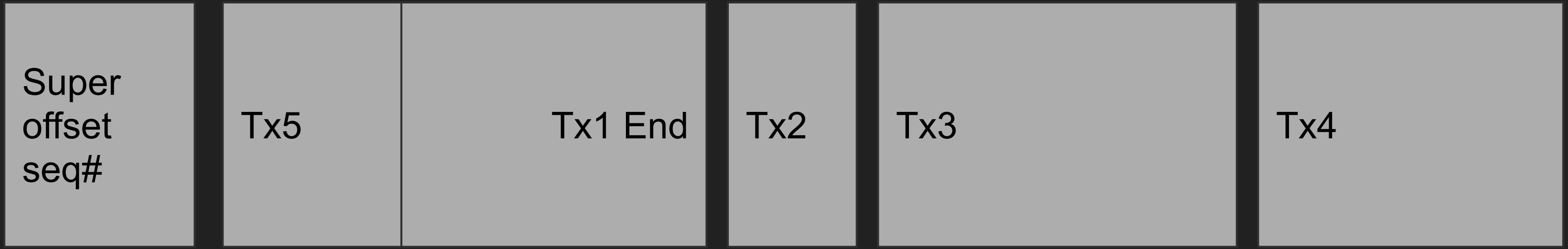
What if block after last valid commit block looks like a log descriptor (Txn Start)?



Magic Number

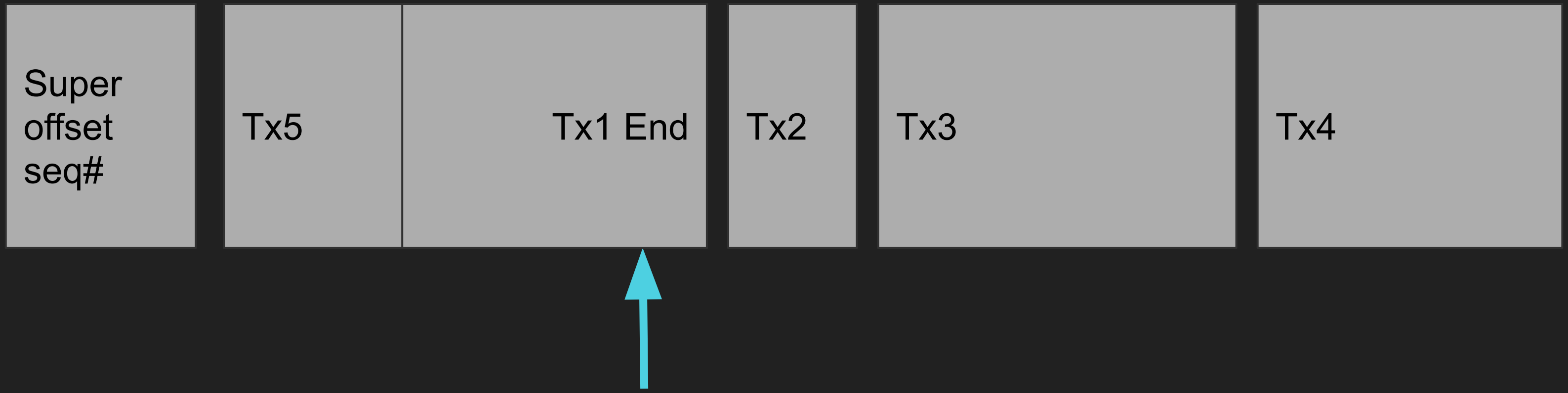
Corner Case B

What if block after last valid commit block looks like a Tx1 End?



Corner Case B

What if block after last valid commit block looks like a Tx1 End?



Log Sequence Number

Summary of ext3 rules (lecture)

- Don’t write metadata block to on-disk FS until committed in on-disk log
- Don’t overwrite a block in buffer cache before it’s in the log
- Don’t free on-disk log transaction until all blocks have been written to FS

Next time

1. Lab 6 Threads is out! Due Nov 12th.
2. Guest Talk!