- [x] 1. Last time
- [ ] 2. Crash recovery
  - [x] intro
  - [x] ad hoc
  - [x] copy-on-write (COW)
  - [x] journaling

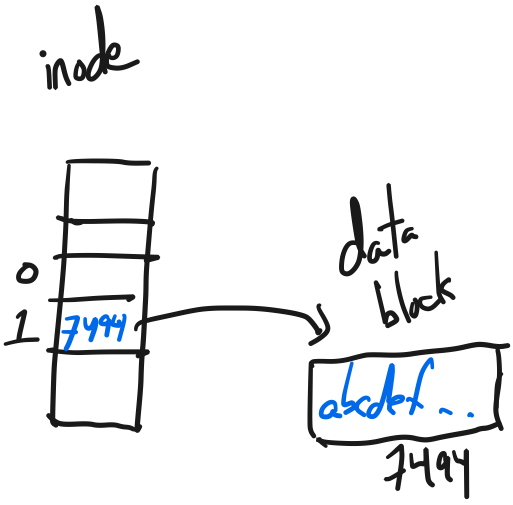
Crash recovery

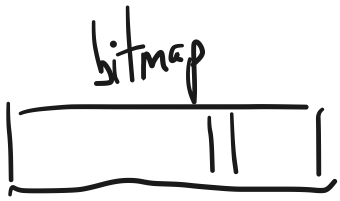
- intro
- ad hoc
- copy-on-write
- journaling

Ex.

C program

```
fd = open();
seek(fd, 4KB);
write(fd, buf, 256);
```





- add to inode
- write data to the block

- update the bitmap

What happens if there is a crash anywhere in this list of operations?

Ad hoc

fsck

Goal: metadata consistency, not data consistency

Approach: send FS updates to the disk in such a way that if there is a crash, fsck can clean up inconsistencies.

Example: file create/write!

- first write data to file blocks on disk

crash?

- then update/write inode

crash?

- then mark inode "allocated" in inode bitmap

crash?

- then mark data blocks "allocated" in bitmap

crash?

- then update directory <name, i#>

crash?

Copy on write

ZFS, btrfs, APFS

Goal: metadata and data consistency

Spend space

Approach: never modify a block, always make a copy

Exception: root block, Uberblock

Journaling

Saltzer-Kaashoek

Golden rule of crash atomicity:
"Never modify the only copy."

Borrow an idea: transactions, from DBs

here, an op is: create a file, delete a file, ...

sub-op: a component of the op

Concept: commit point:

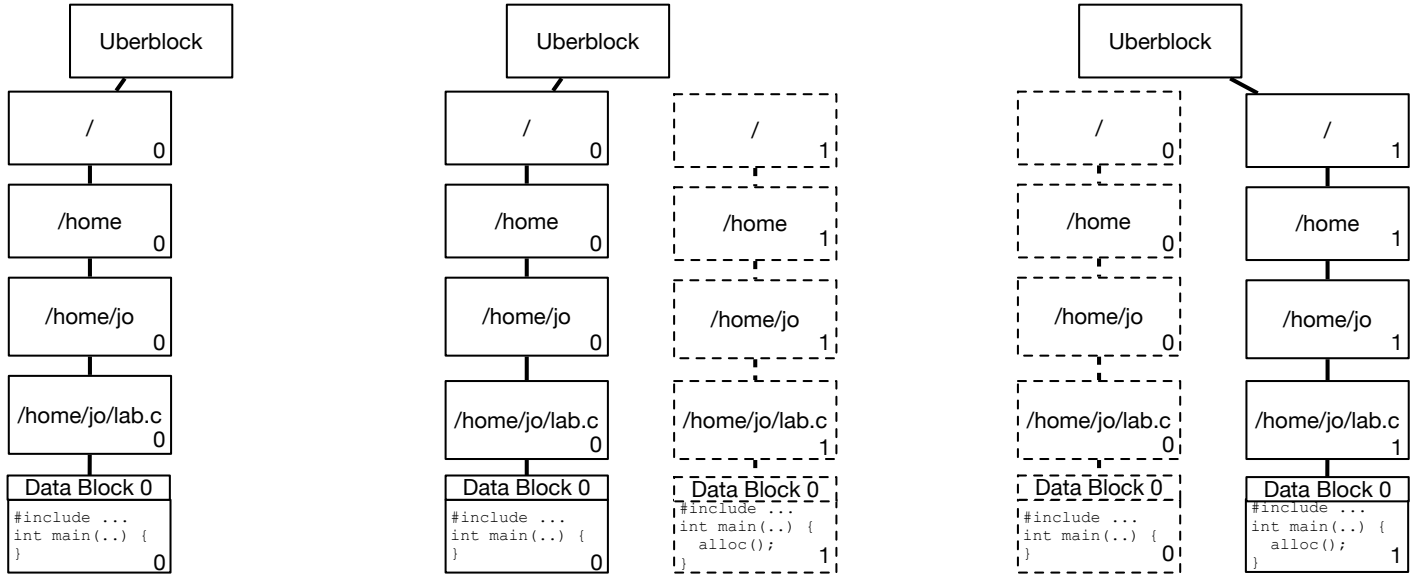
first step
[stuff]
⇒ commit point ⇐
[stuff]
last step

} Can back out, leaving no trace

} operation isn't necessarily finished, but completion is inevitable.

- WAL == redo logging
- undo logging
- redo + undo logging

CS202 Handout 12

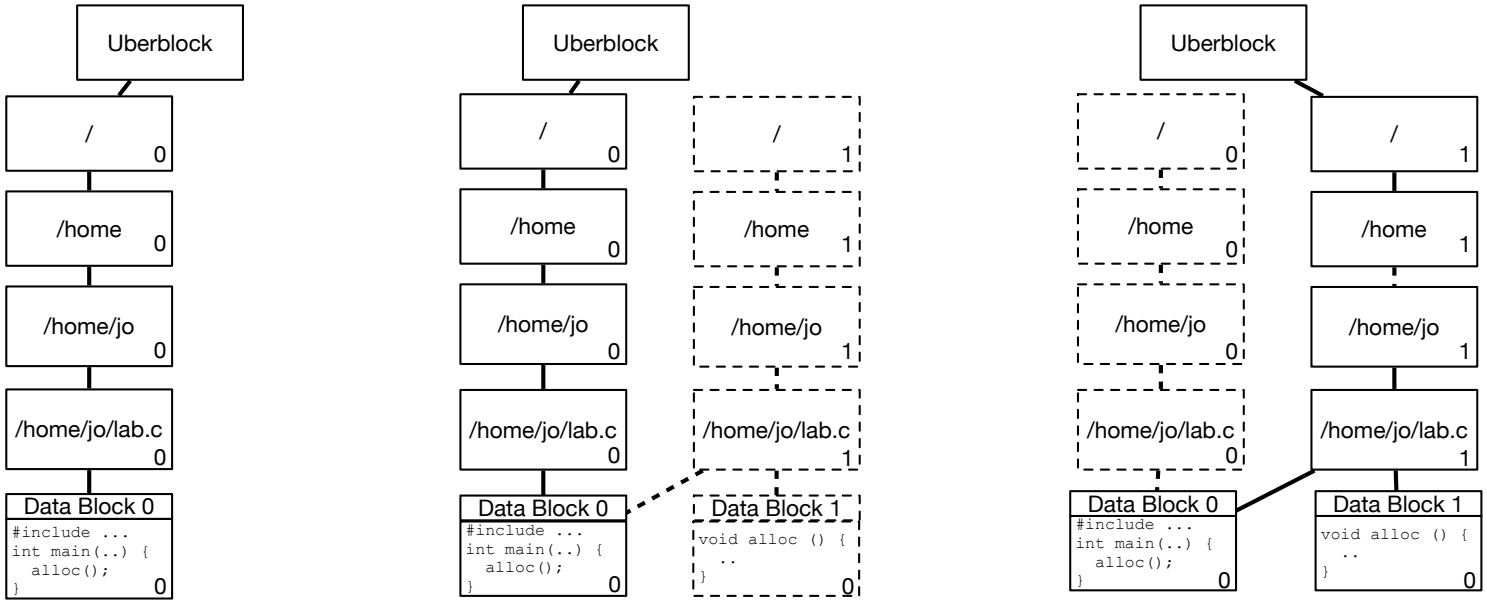


(a) Initial State

(b) System allocates and creates new versions of all modified blocks.

(c) System updates Uberblock to point to new version of blocks.

Figure 1: Copy-on-write filesystem: modifying a data block

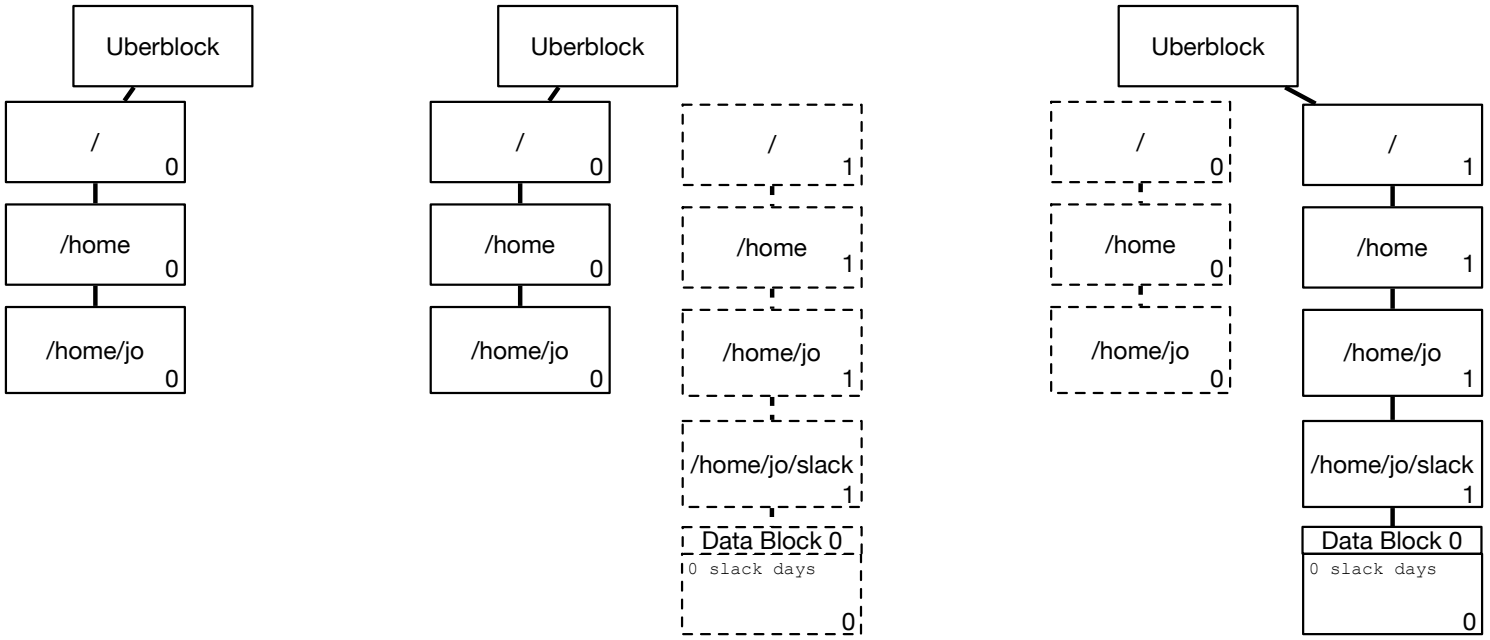


(a) Initial State

(b) System allocates and creates new versions of all modified blocks.

(c) System updates Uberblock to point to new version of blocks.

Figure 2: Copy-on-write filesystem: adding a data block

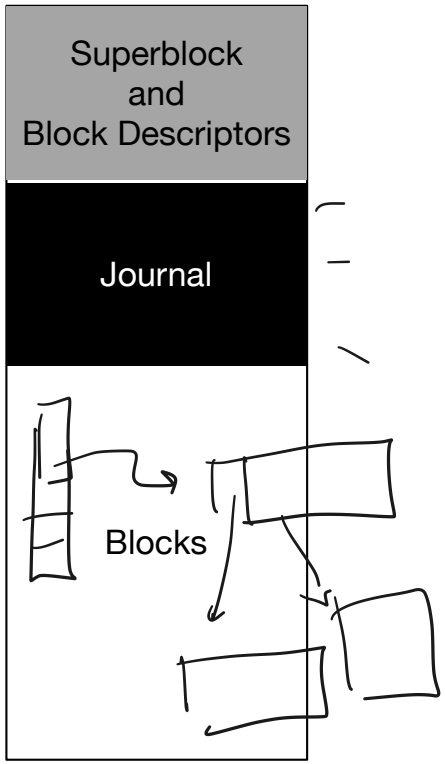


(a) Initial State

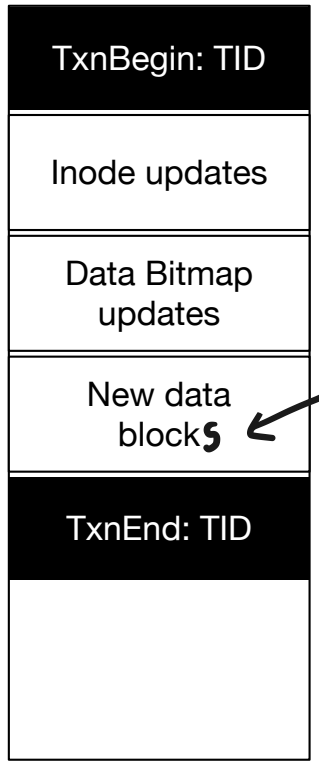
(b) System allocates and creates new versions of all modified blocks.

(c) System updates Uberblock to point to new version of blocks.

Figure 3: Copy-on-write filesystem: creating a file



ext3 disk layout



ext3 journal layout

Figure 4: Redo logging in a filesystem

Undo logging

Recovery

① What would change
② Where in the log we can write it
③ log changes || apply them to the on disk data structures

(of course logging a single entry before the modifications)

④ write Tx End