- [x] 1. Last time
- [ ] 2. Crash recovery
  - [x] Intro
  - [x] Ad hoc
  - [x] Copy on write
  - [ ] Journaling

"consistency"

"atomicity": all-or-nothing w.r.t. a crash

("fsck")

Ad hoc

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 blocks "allocated" in bitmap

- then mark data

crash?

- then update directory

<name, #>

crash?

lost+found

Copy on write ZFS btrfs APFS

Goal: metadata + data consistency

Spend space

Approach: never modify a block, always make a copy.

Exception: root block, Uberblock

Journaling Saltzer-Kaashoek

crash

Unit of atomicity:

Golden rule of [illegible]

"Never modify the only copy."

Borrow an idea: transactions, from the DBs

op: create a file, delete a file, --

sub-op: a component of the op

concept: commit point:

first step
[stuff] } can back out, leaving no trace

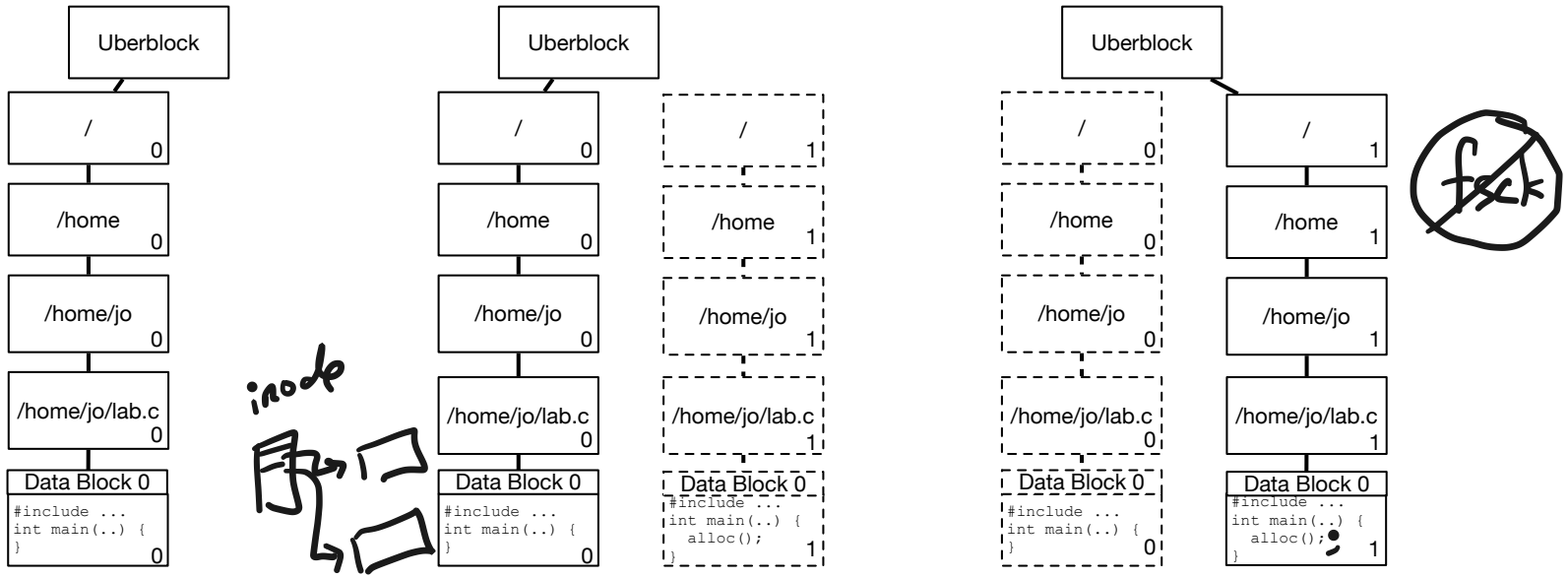
commit point
[stuff] } completion is inevitable

last step

Journaling / logging

- [x] WAL == redo logging
- [ ] undo logging
- [ ] redo + undo logging

CS202 Handout 13

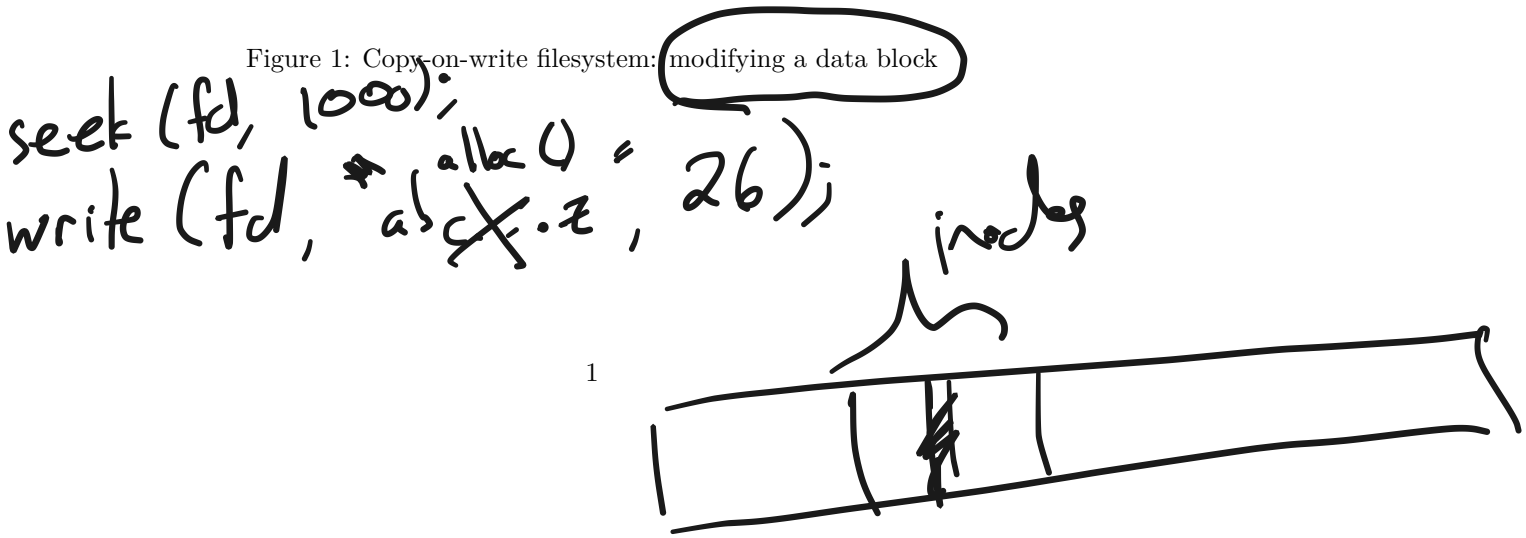


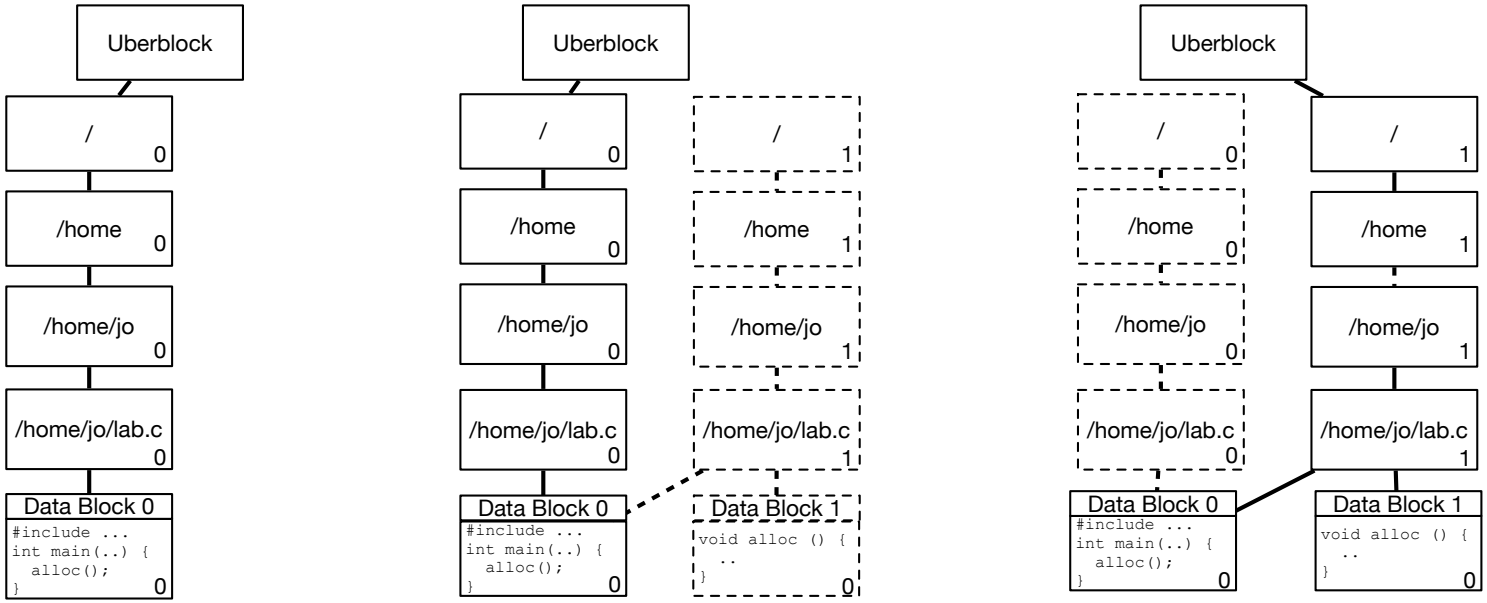
(a) Initial State

(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



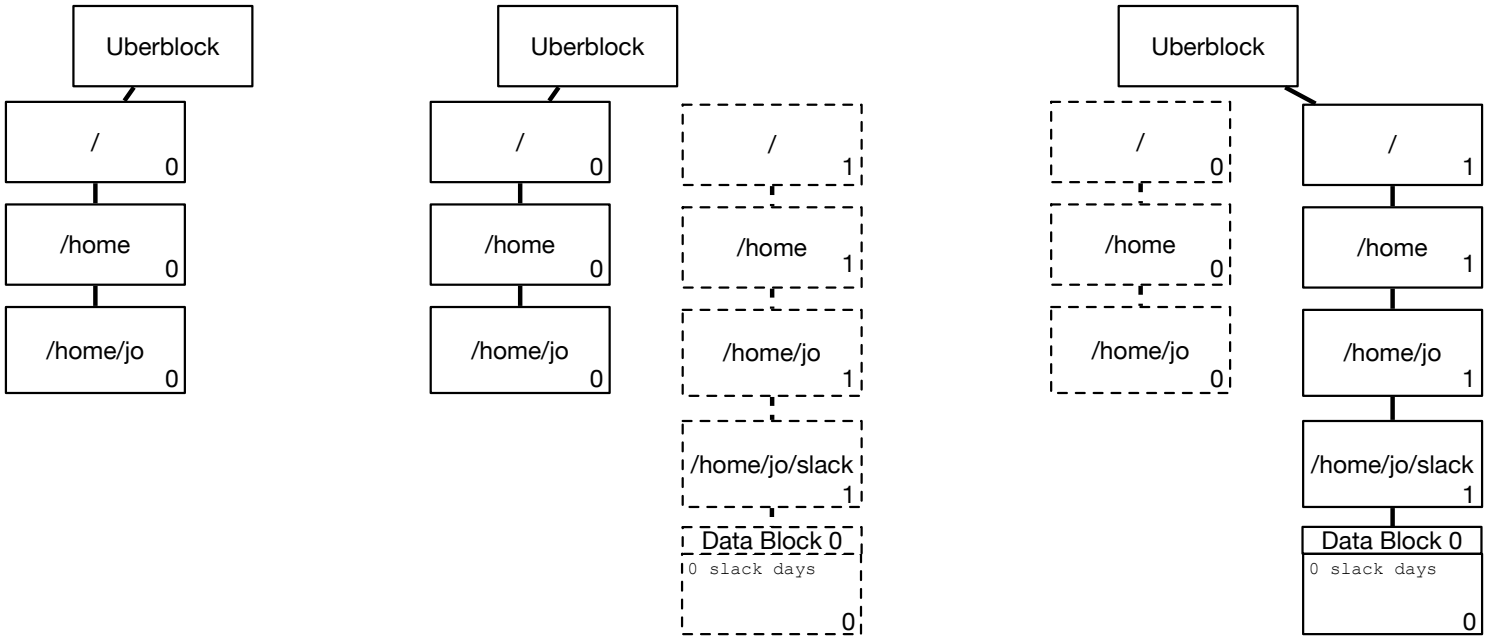


(a) Initial State

(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



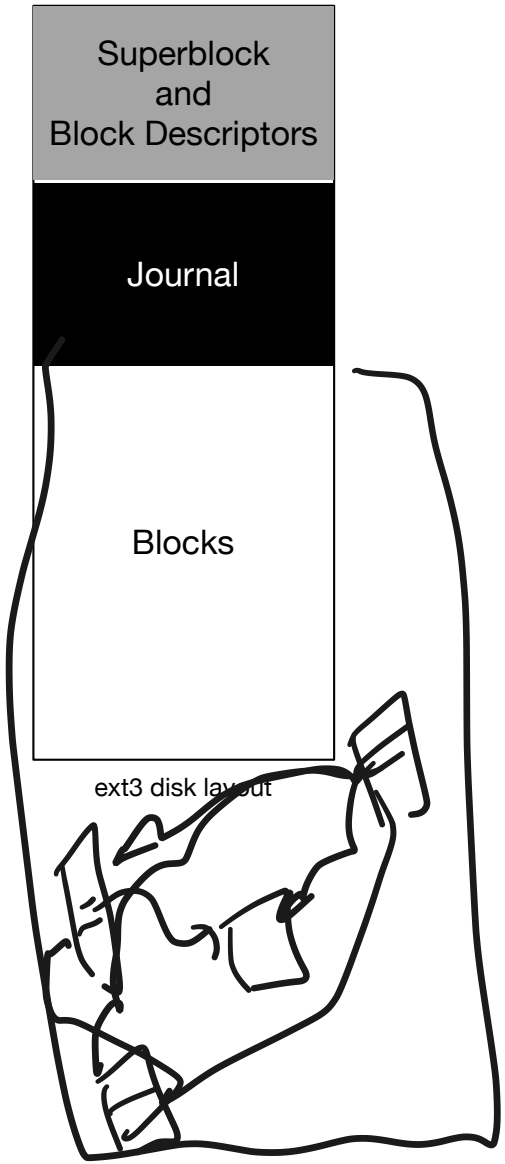
(a) Initial State

(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

= checkpointing =



log == journal

WAL
write-ahead logging

idempotent

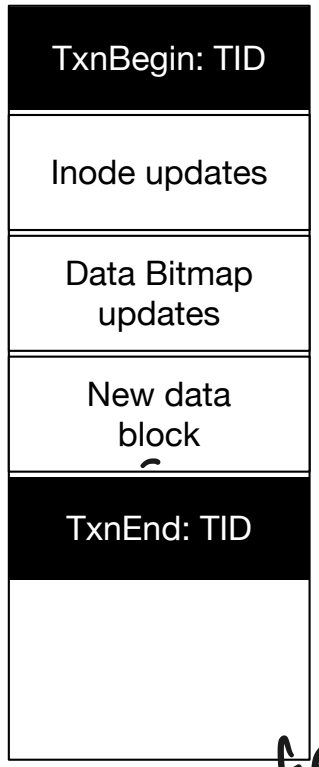


Figure 4: Redo logging in a filesystem

redo logging