- [x] 1. Last time
- [x] 2. Finish directories
- [x] 3. Performance
- [ ] 4. Crash recovery
  - intro
  - ad hoc
  - copy on write
  - journaling

---

2. Finish directories

Example: /a/foo.c
/b/c/essay.txt

inode #7 (by assumption)

We can give a file another name; for example
/a/hello.txt

This is because the file itself does not know its own name. Looking only at the file, the "name" is just a i-node number.

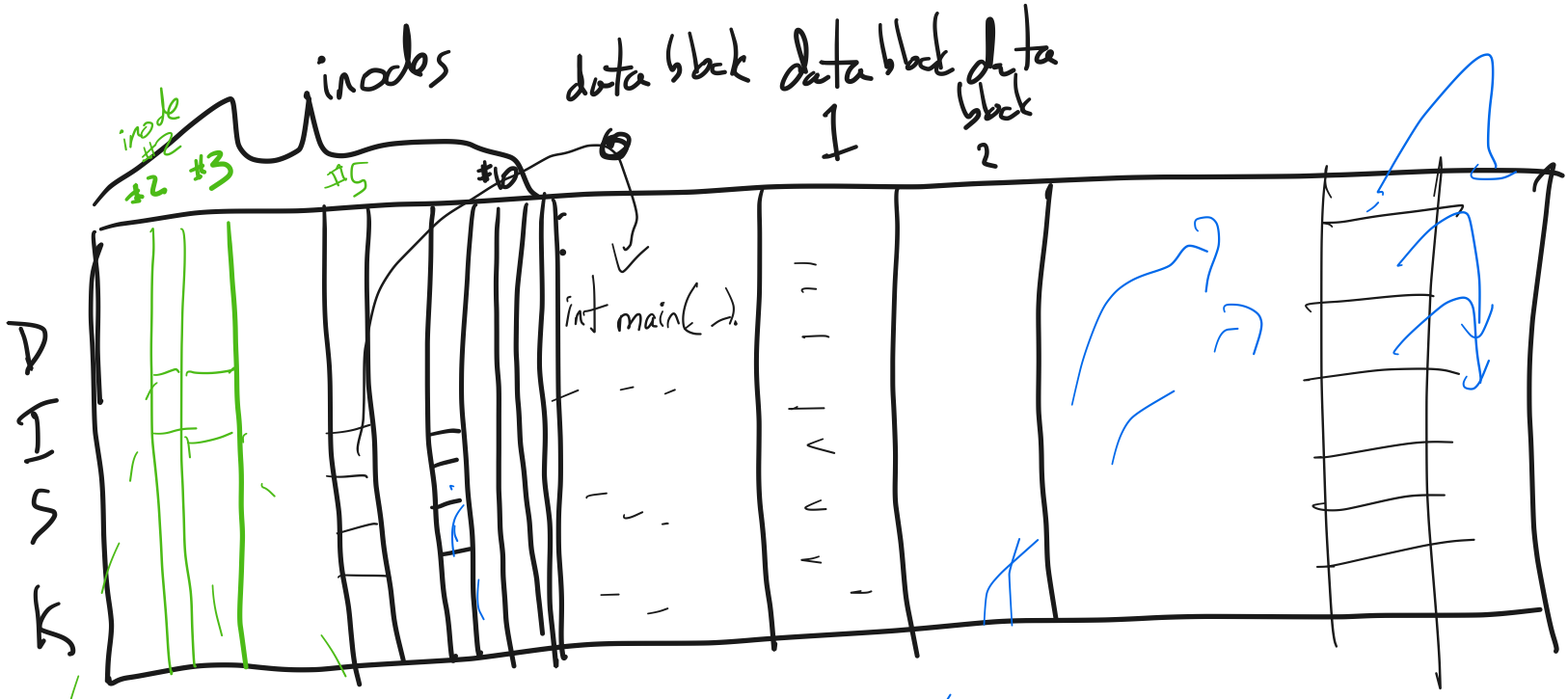
Human-usable names, such as /a/foo.c
are in directory entries, for example:

come from directory

<a, 3>
<b, 4>

dirent ← readdir()
↳ lab2 d_name
d_no

which means "let us all call inode 3 by the segment "/a", and similarly for inode 4 and "/b".



<a, 3>
<b, 4>
<mylink, 10>

<foo.c, 5>
<hello.txt, 7>

$ ln /b/c/essay.txt /a/hello.txt

"link"

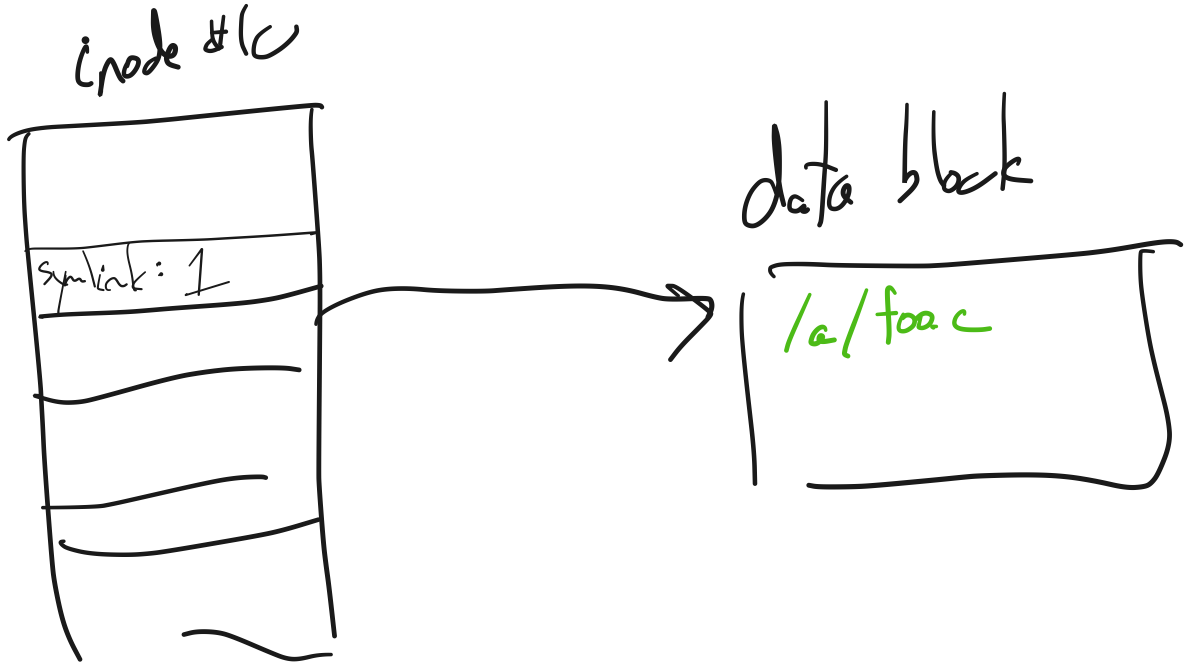
syscall: "unlink();"

---

soft links vs. hard links

$ ln -s /a/foo.c /mylink

creates a new inode (#10)



symlink bit set 1.

/a/b/c

dst name

$ ln ____ ____

mount points

$ cat /a/foo.c

---

3. Performance

case study: FFS (1984)

problems w/ the original:

- blocks too small (512)
- inode array was at beginning of disk
- free blocks were stored in a linked list on the disk
- poor clustering of related objects
  - consecutive file blocks
  - inodes relative to disk blocks
  - [illegible] directory

inodes for a given directory

result:

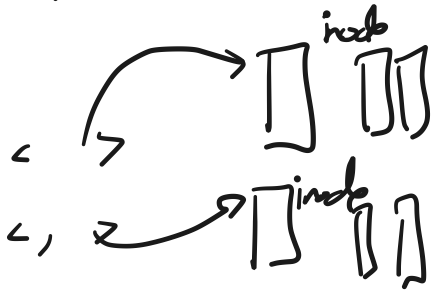
ls

grep <path> *.c

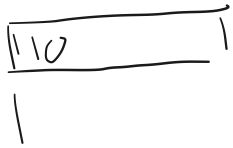
} ⇒ slow

Improvements?

- Make data blocks / inodes close to each other



- Cluster files in the same directory
- Make data blocks bigger (4KB, 8KB, 16KB)
- Free blocks: store separately (bitmap)



$\frac{1\text{ block}}{1\text{ bit}}$ × 500 GB disk / 4KB blocks = 12

$2^{39} / 2^{12} = 2^{27} \approx$ 1 disk: 128,000,000 bits

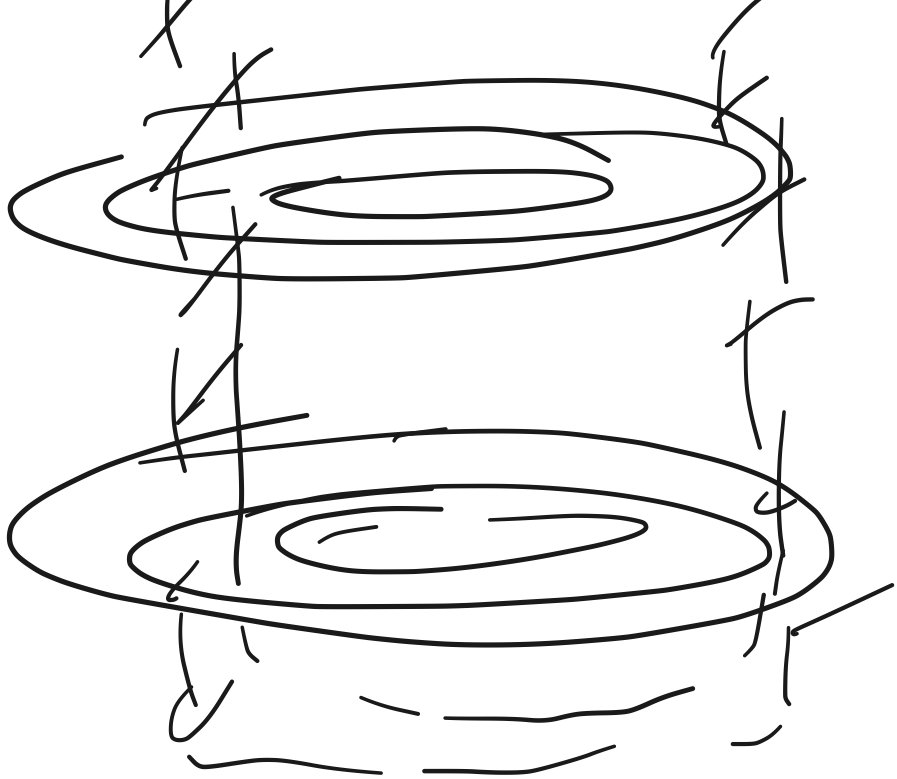
$\approx$ 16 MB ⇐ store in memory

2^{27} bits $\times \frac{1 \text{ byte}}{2^3 \text{ bits}} = 2^{24}$ bytes

$= 16$ MB

⇒ reserve space (lie to the user abt free space)

- symbolic links
- atomic "rename" ($ mv abc.txt def.txt)
- cylinder groups (for clustering)



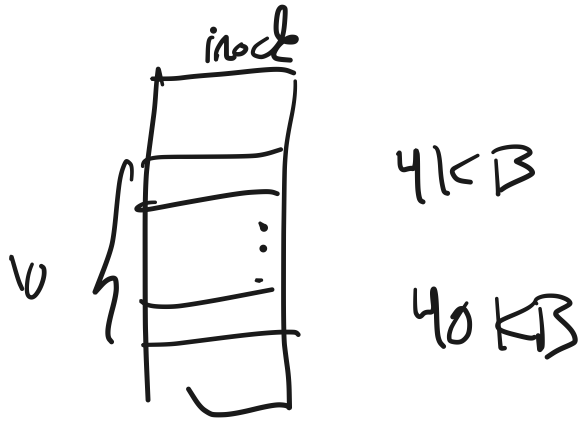
[superblock | bookkeeping | inodes | bitmap | data blocks (512 bytes each)]

attempt: put inodes + their data blocks in the same cyl. grp.

attempt: put inodes of ~~the~~ files in the same dir. in the same cyl. group.

new directory: choose a cyl. group w/ higher than avg. # of free inodes.

as a file grows, after it crosses 48 KB, spill to next cyl. group, and do likewise for every 1 MB thereafter.



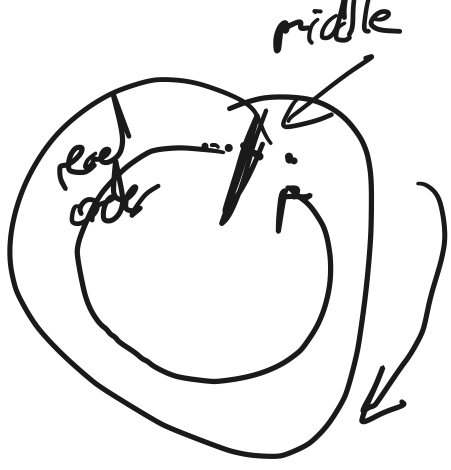
total perf:

20-40% of disk B/w for large files

10-20x improvement on the predecessor.

Other things they did:

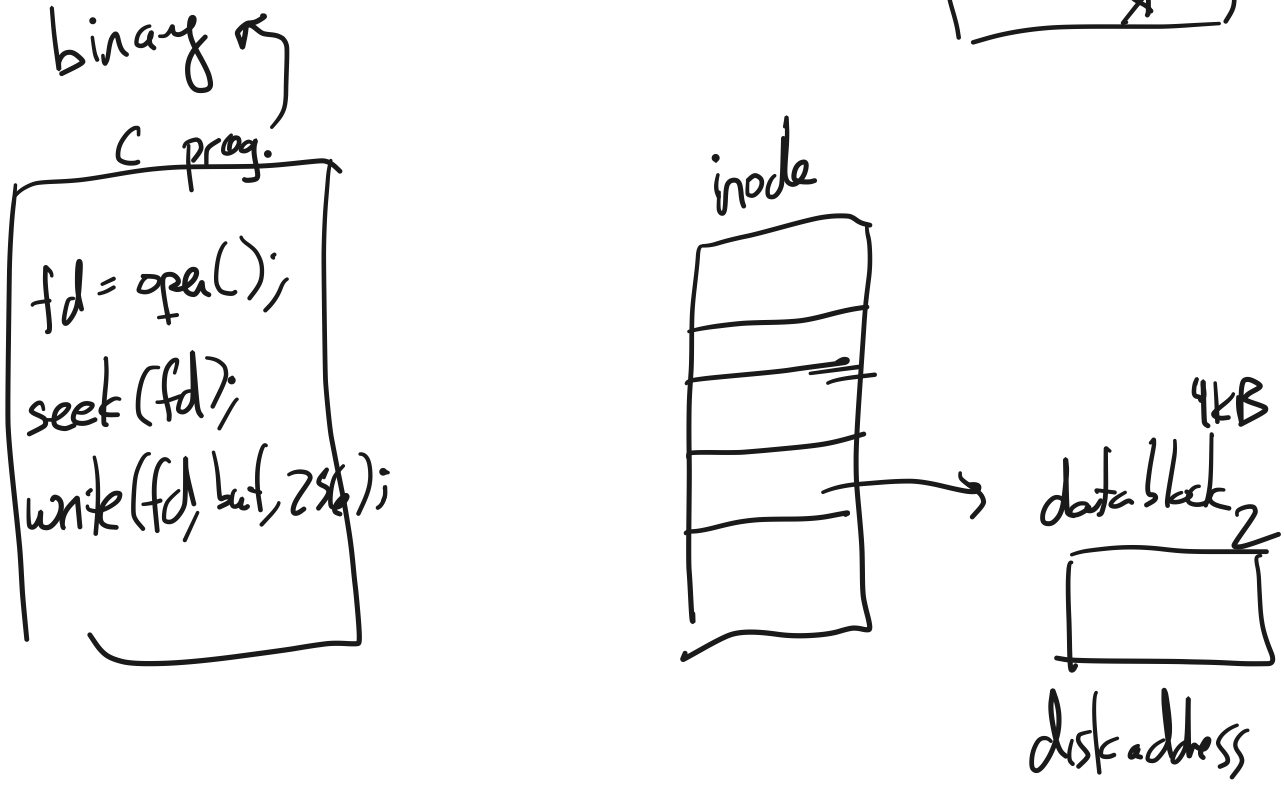
- buffer cache
- read entire track



4. Crash recovery

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

Ex.

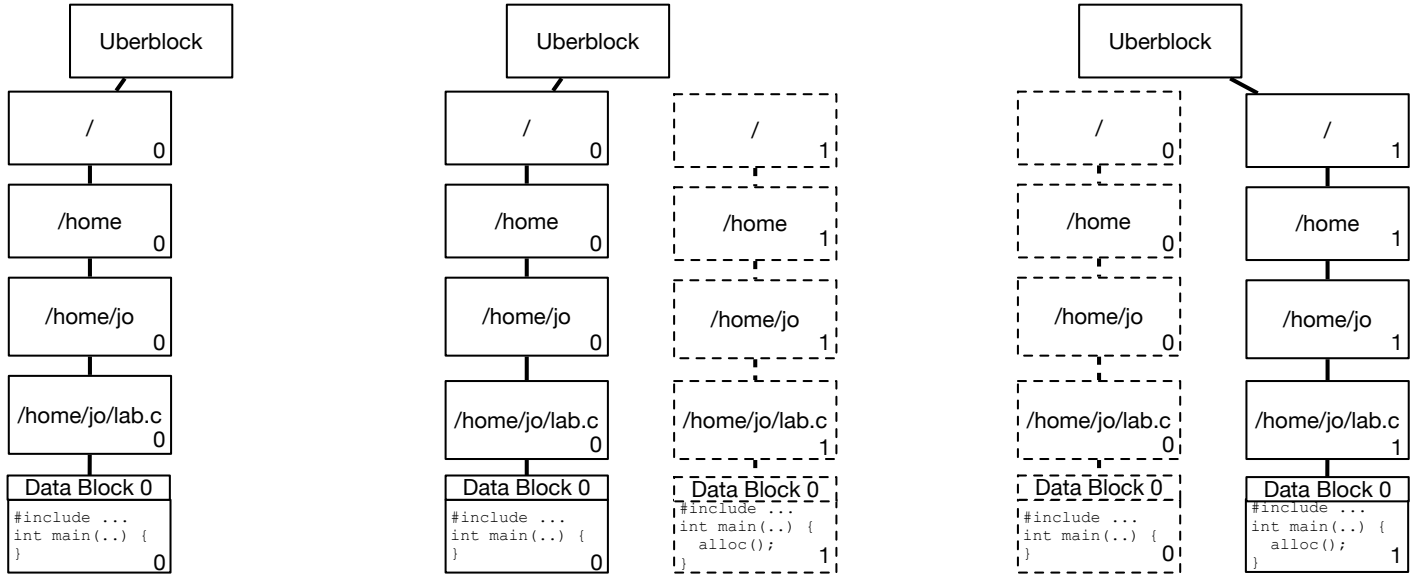


- add to inode
- write data to the block

- update the bitmap

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

CS202 Handout 13

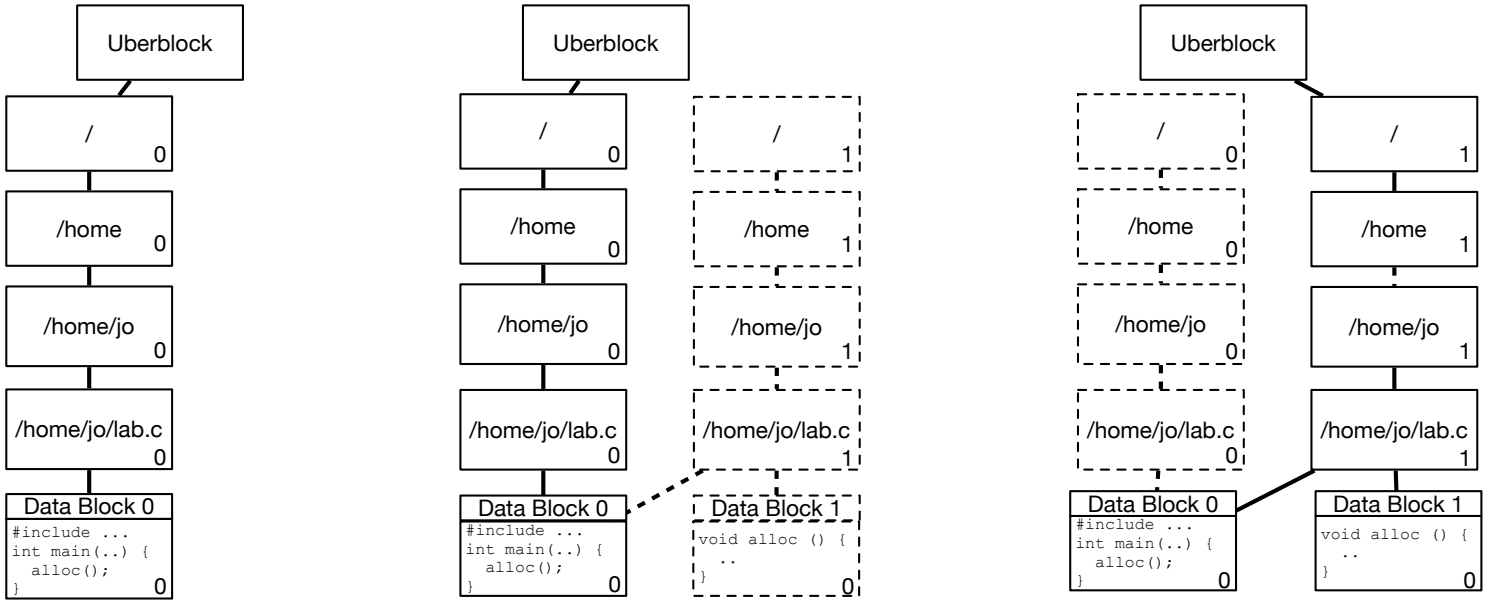


(a) Initial State

(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

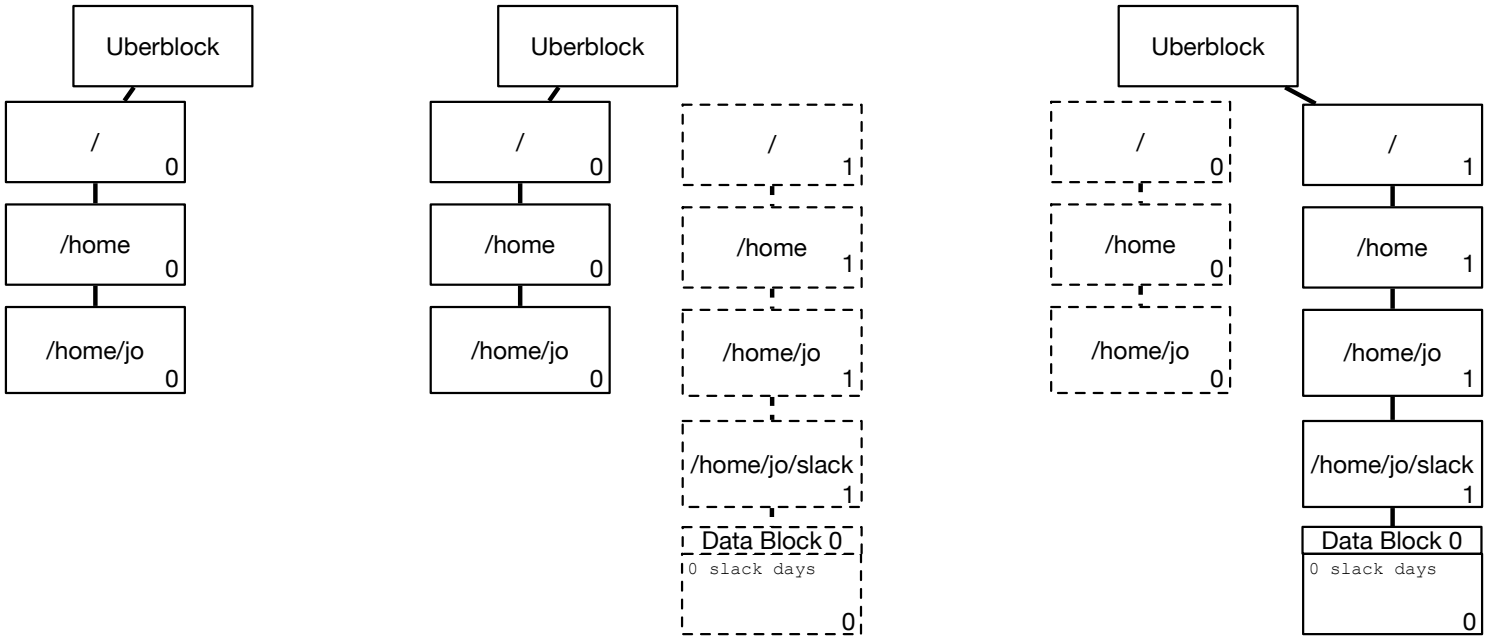


(a) Initial State

(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

Figure 4: Redo logging in a filesystem