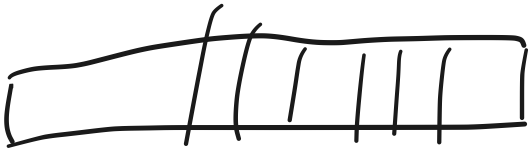
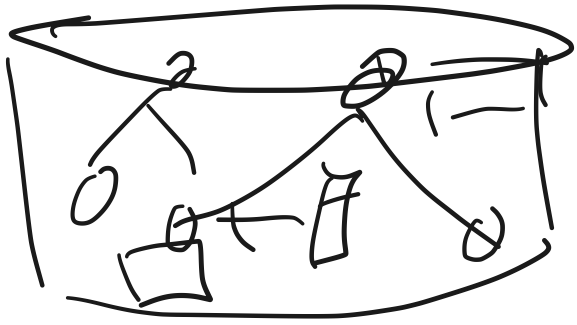
- [x] 1. Last time
- [x] 2. Crash recovery, contd.
- [x] 3. RPC, client/server systems
- [ ] 4. NFS
  - [x] Intro + background
  - [ ] How it works
  - [x] lab5 detour
  - [ ] Statelessness of protocol
  - [ ] Transparency

---

2. Crash recovery

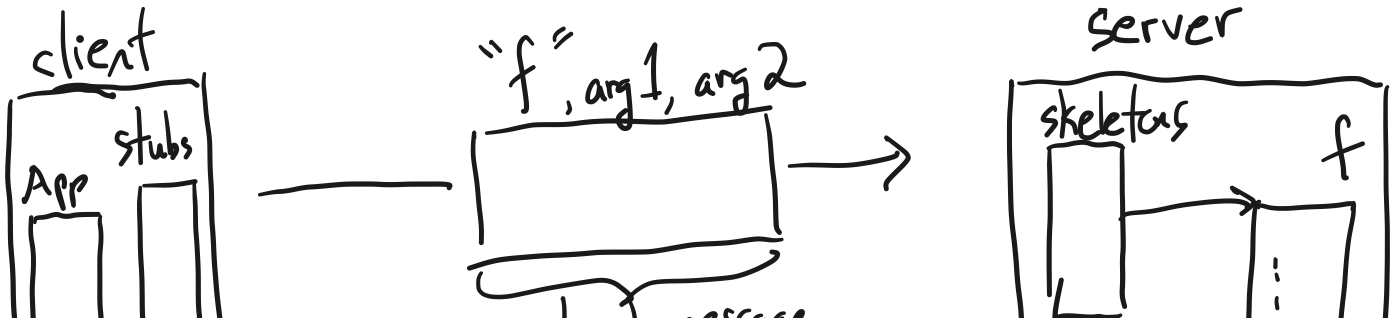
See last time's scribble

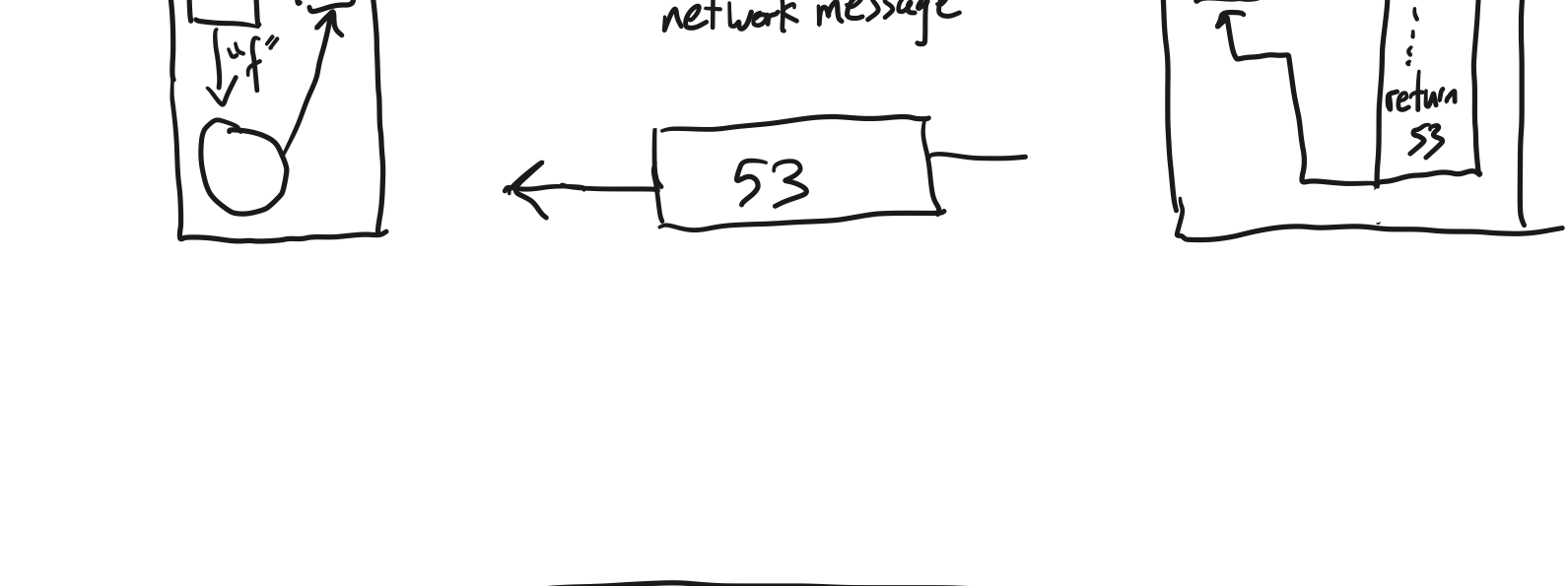




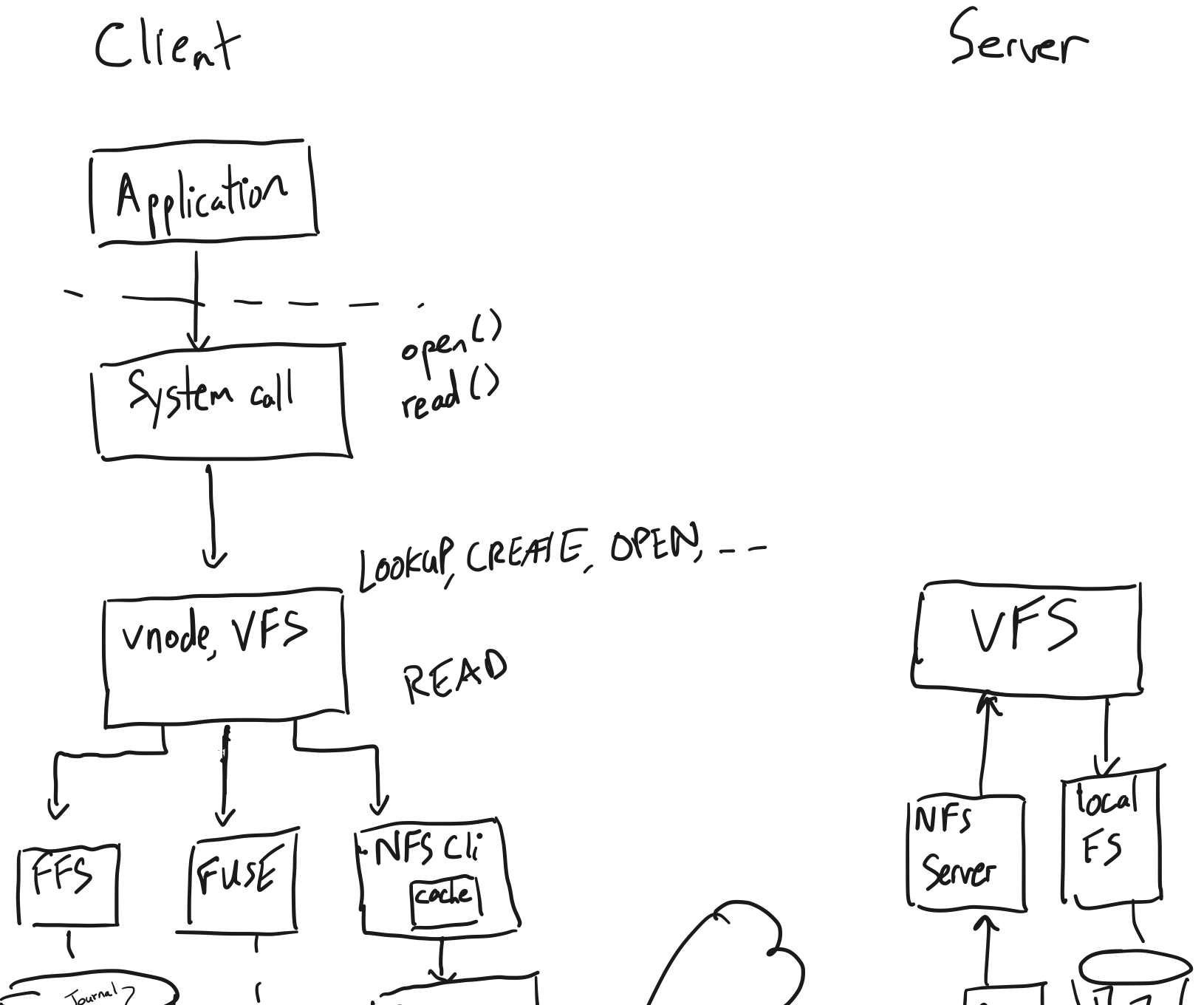
---

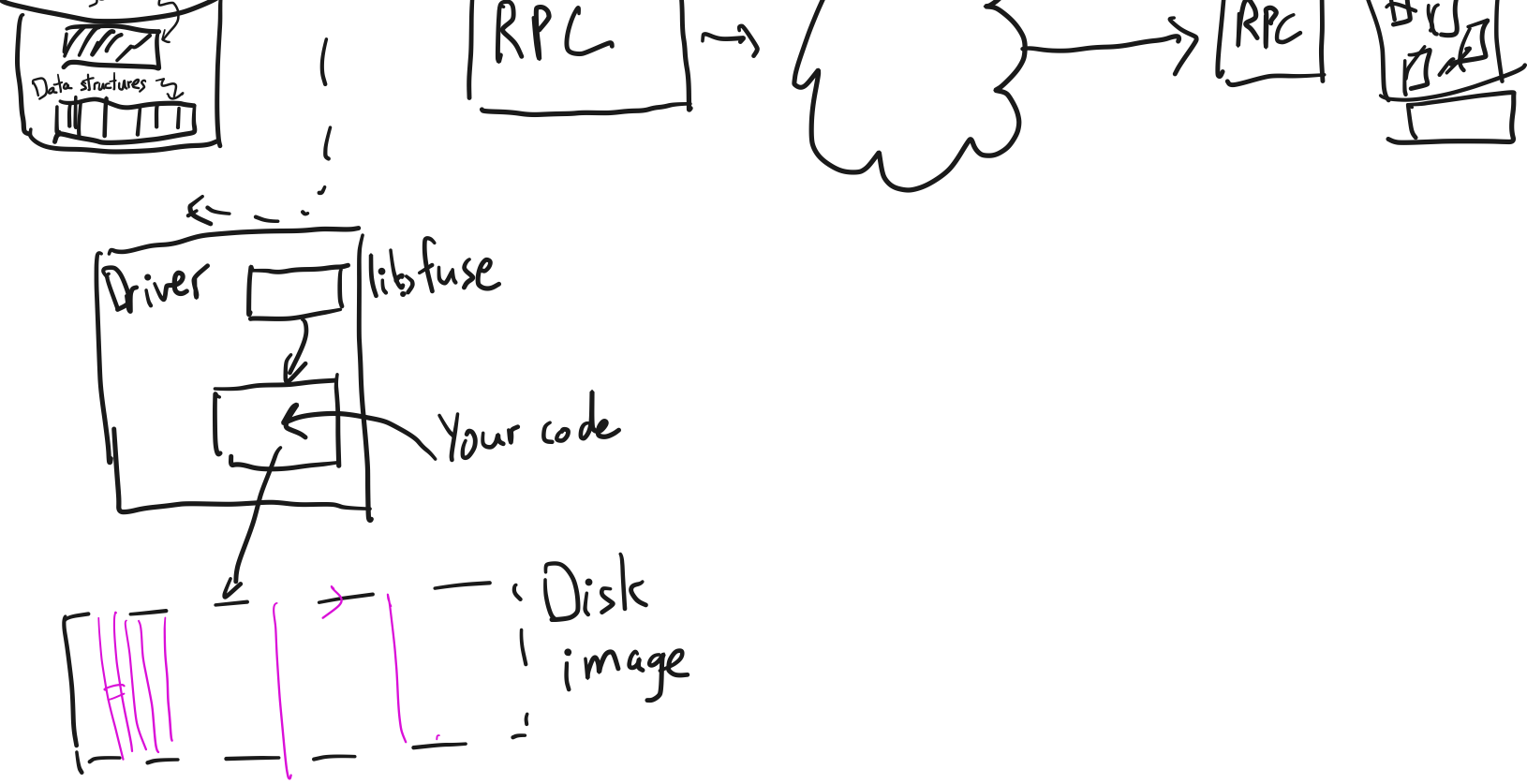
3. RPC, client server




"f"
network message
53
return
53

4. NFS

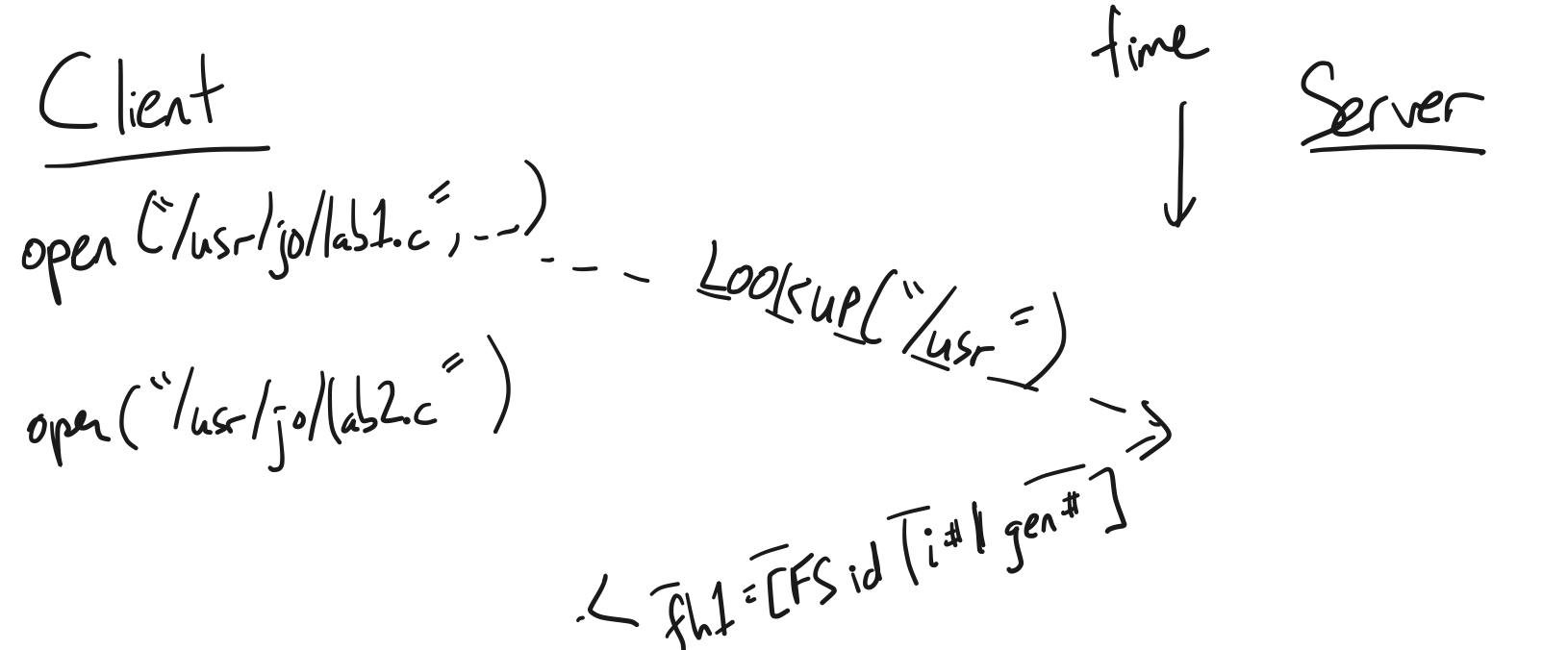

Client
Server
Application
System call
open()
read()
vnode, VFS
LOOKUP, CREATE, OPEN, --
READ
FFS
FUSE
NFS Cli
cache
VFS
NFS Server
local FS



lab notes:

→ each inode gets its own disk block

→ pointers to pointers ——



LOOKUP(fh1, "jo") →

← fh2

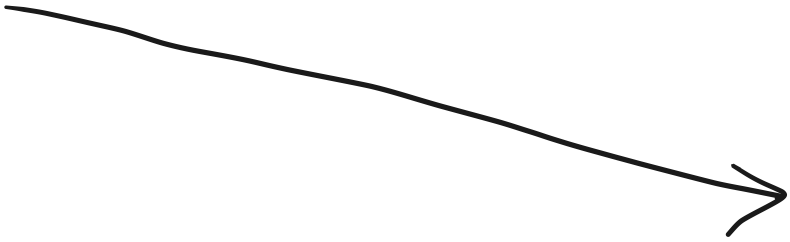
LOOKUP(fh2, "lab1.c") →

← fh3 = [FS id | i# | gen#]

write(fd, buf, sz);

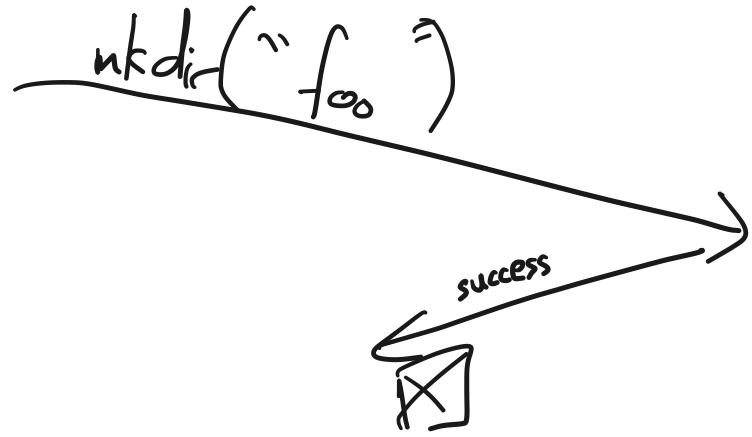
WRITE(fh3, offset, data, size) →

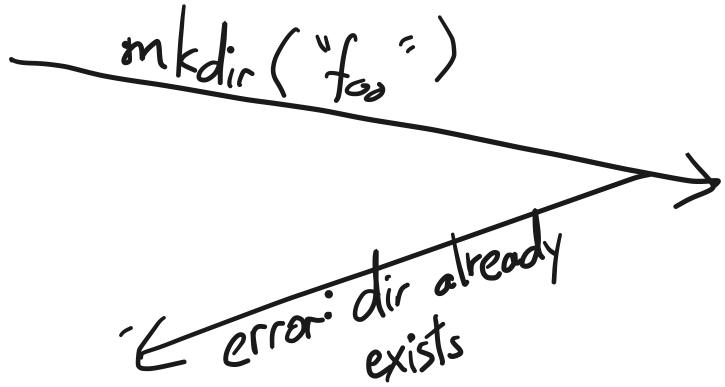
← return code



write: idempotent

mkdir()



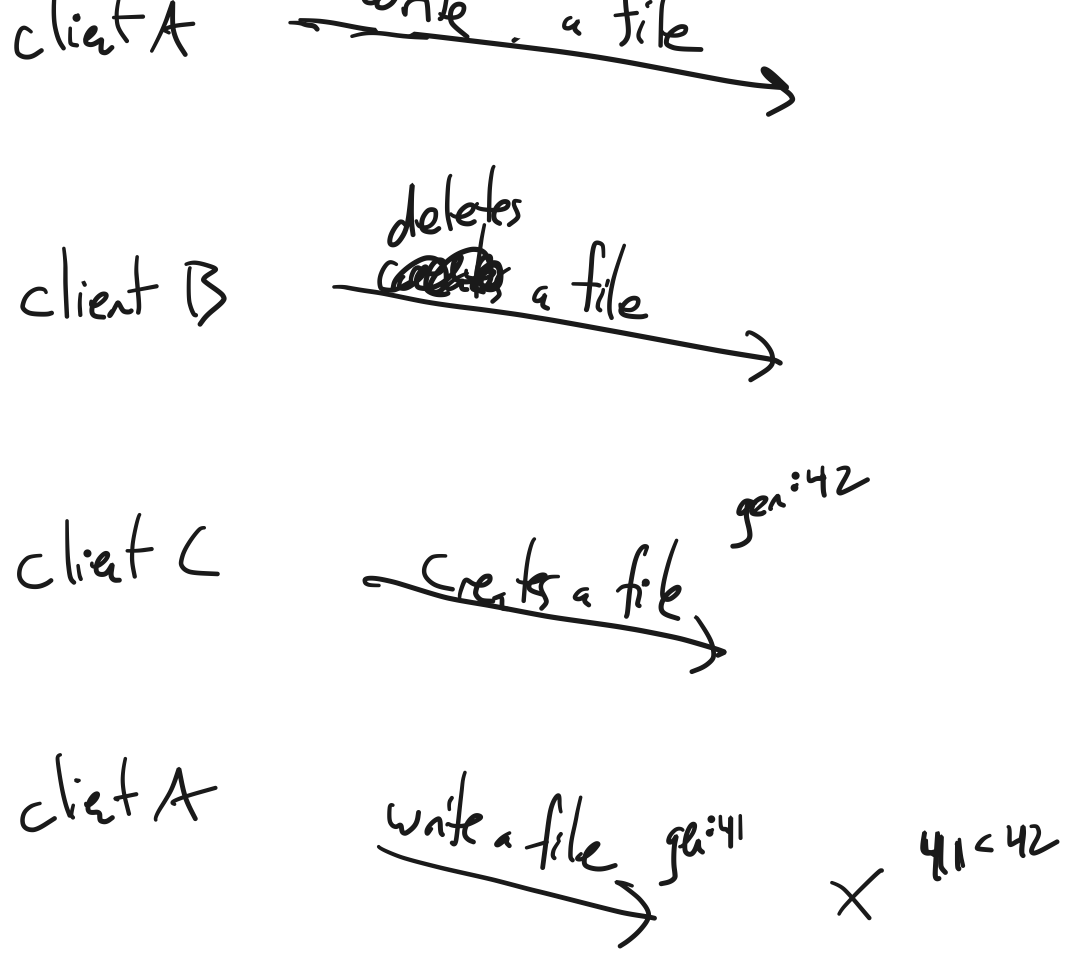


---

④ Trasparency (or lack thereof)

- generation # recall: FH = [FS ID | i# | gen #]

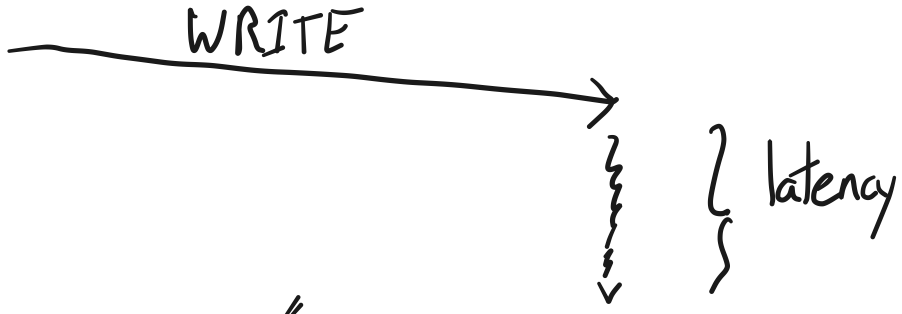
gen: 41



successful operations can return errors

Caching in NFS

Writes must put their data on disk (or persistently) before the server responds to the client.



So clients cache their writes

A

```
write();
/* caching */
close();
```

B

```
open();
read();
```

no guarantee that the data is seen.