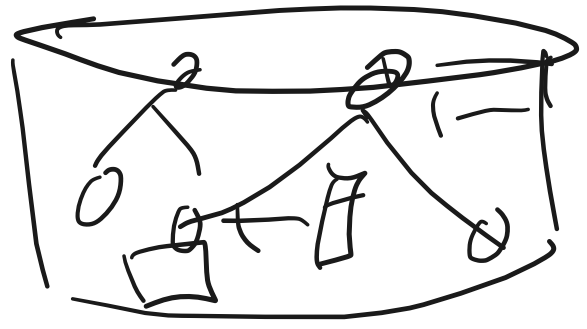
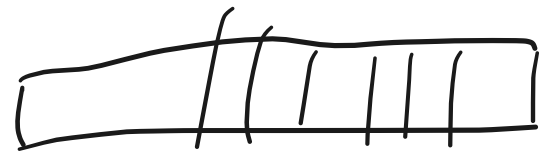


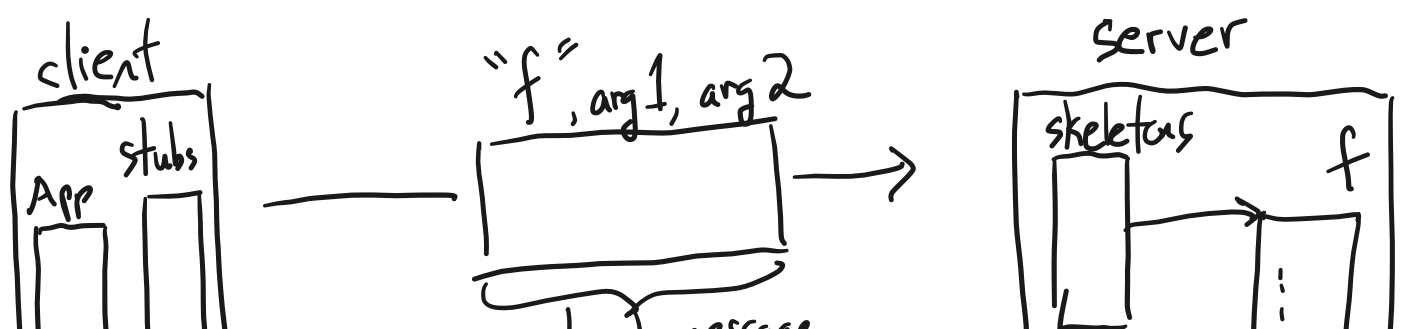
- 1. Last time
- 2. Crash recovery, contd.
- 3. RPC, client/server systems
- 4. NFS
 - Intro + background
 - How it works
 - lab5 detour
 - Statelessness of protocol
 - Transparency

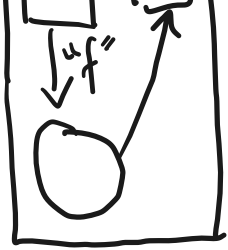
2. Crash recovery

See last time's scribble



3. RPC, client server





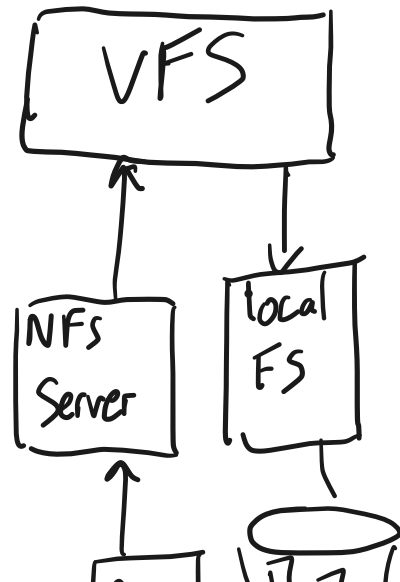
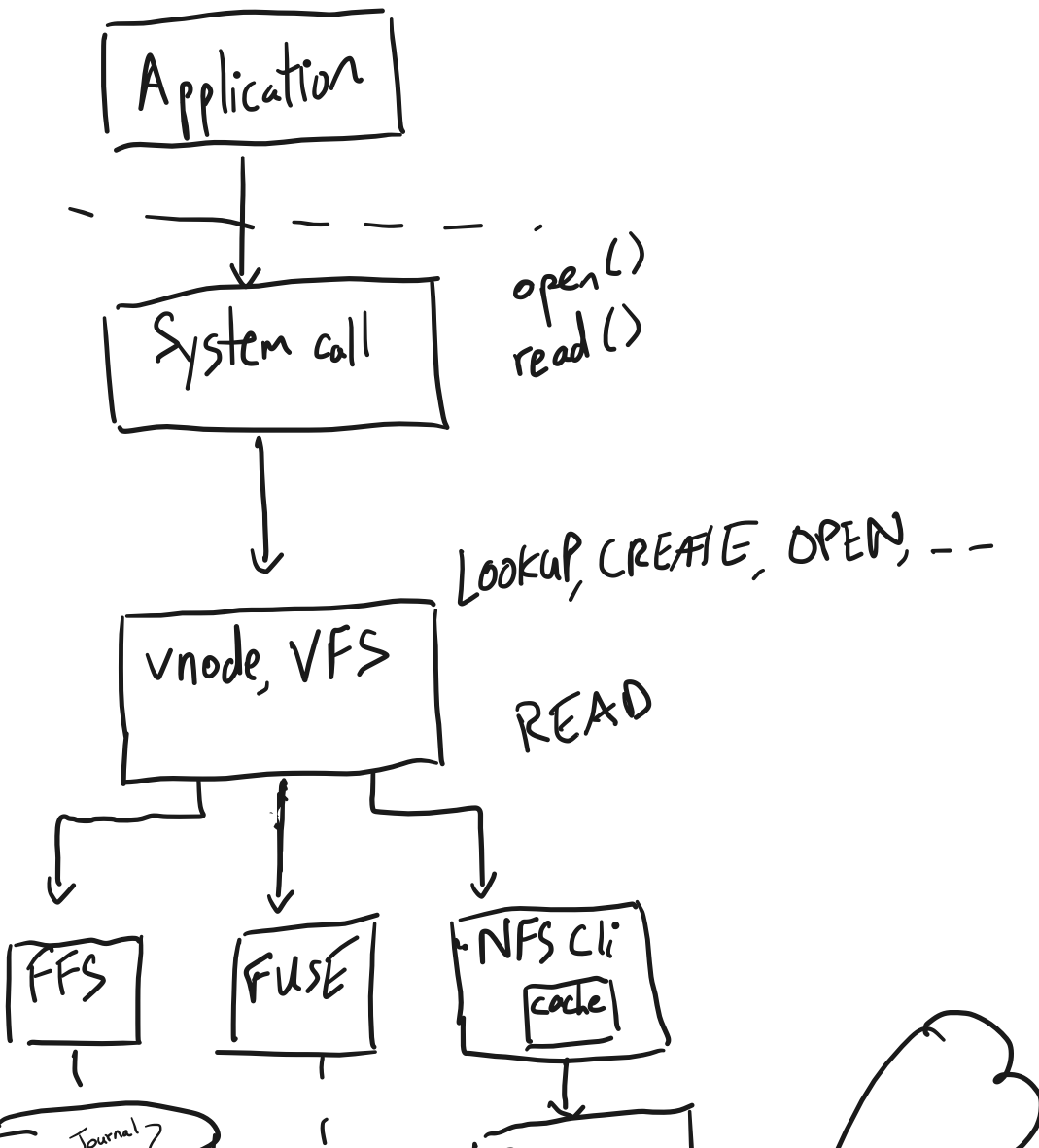
network message

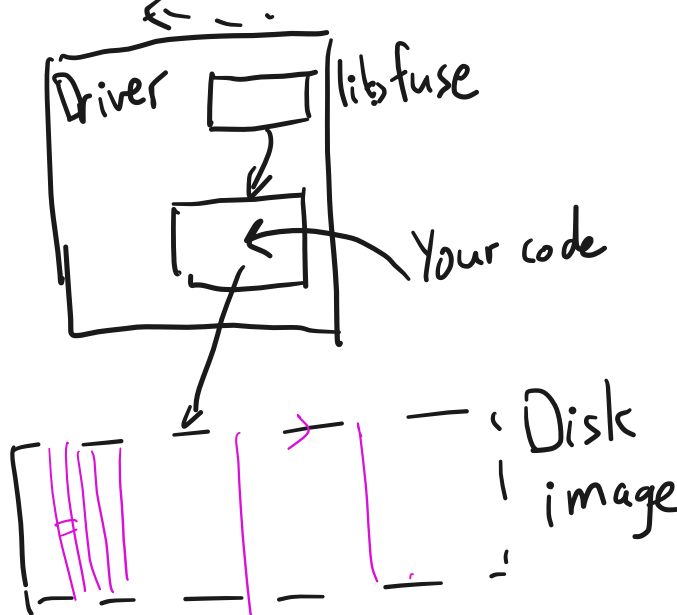


4. NFS

Client

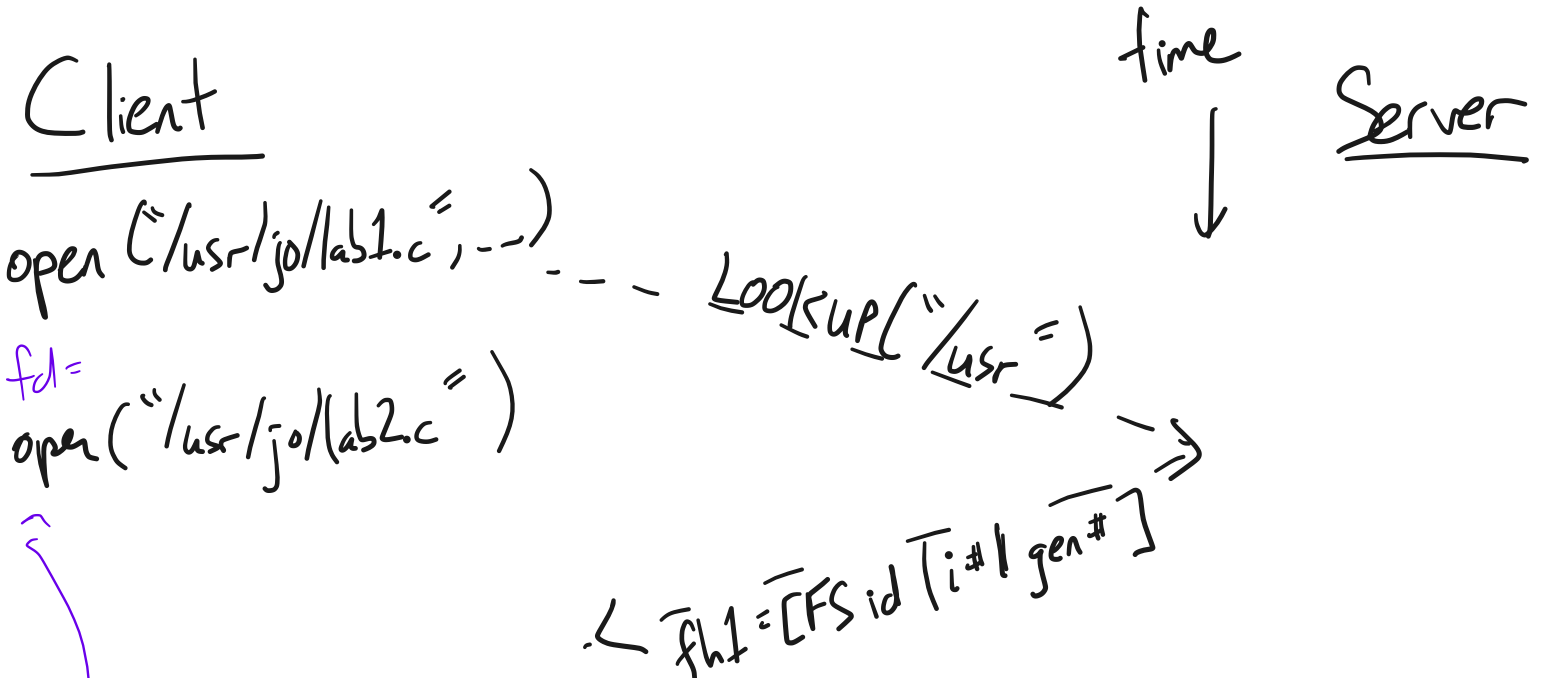
Server





lab notes:

- each inode gets its own disk block
- pointers to pointers —



LOOKUP(fh1, "joe") →

← fh2 = [FS id | i# | gen#]

LOOKUP(fh2, "lab1.c") →

← fh3 = [FS id | i# | gen#]

<fh3, fd>

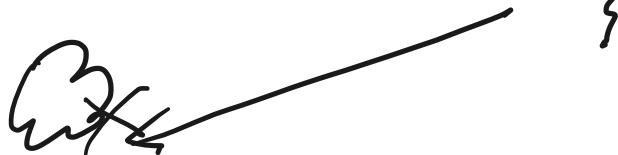
write(fd, buf, sz);

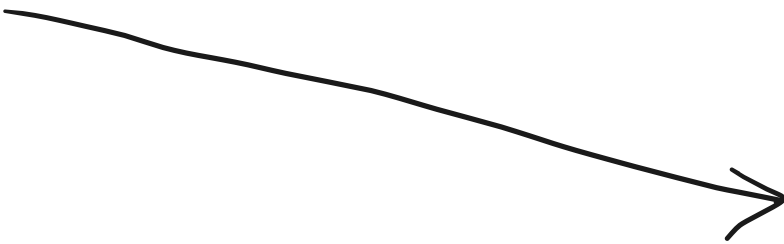
WRITE(fh3, offset, data, size)

← return code

cli

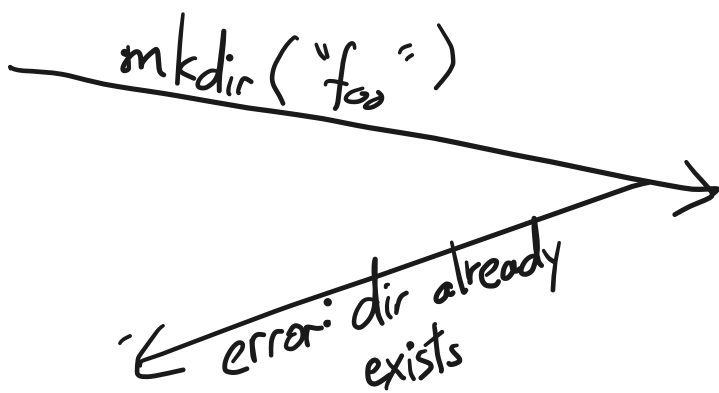
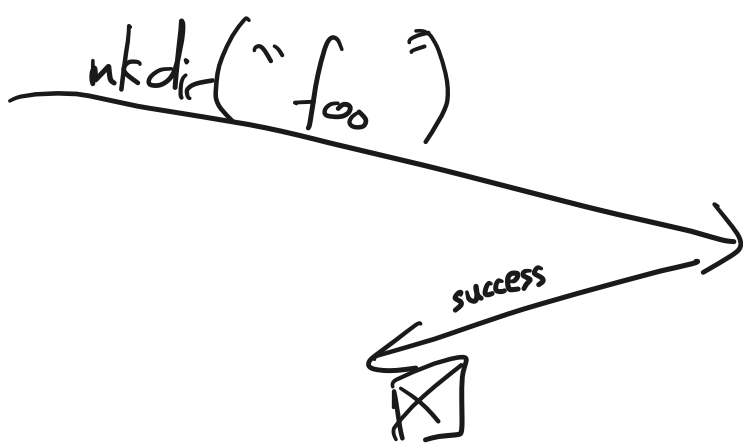
serv.





write: idempotent

mkdir()



④ Transparency (or lack thereof)

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

gen: 41

client A → write a file

client B → ~~creates~~ deletes a file

client C → creates a file ^{gen:42}

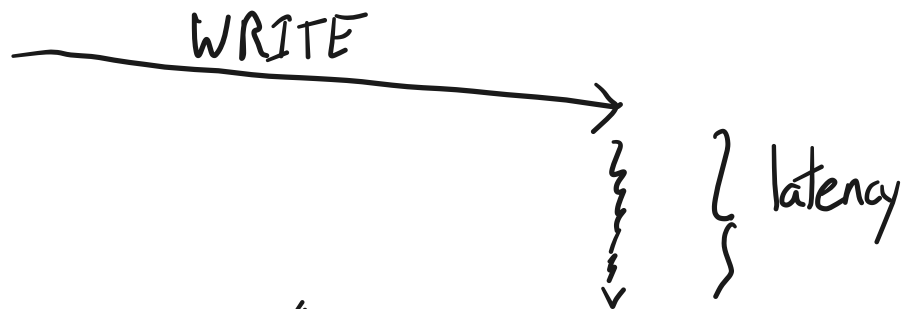
client A → write a file ^{gen:41} × 41 < 42

successful operations can return errors

Caching in NFS

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

cache



← "OK"

So clients cache their writes

A

write();

/* caching */

close();

B

open();

read();

no guarantee
that the data is
seen.