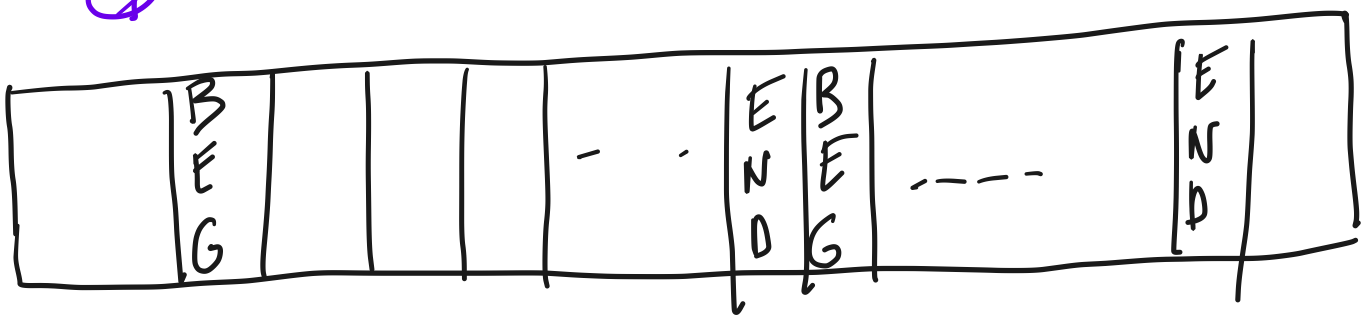
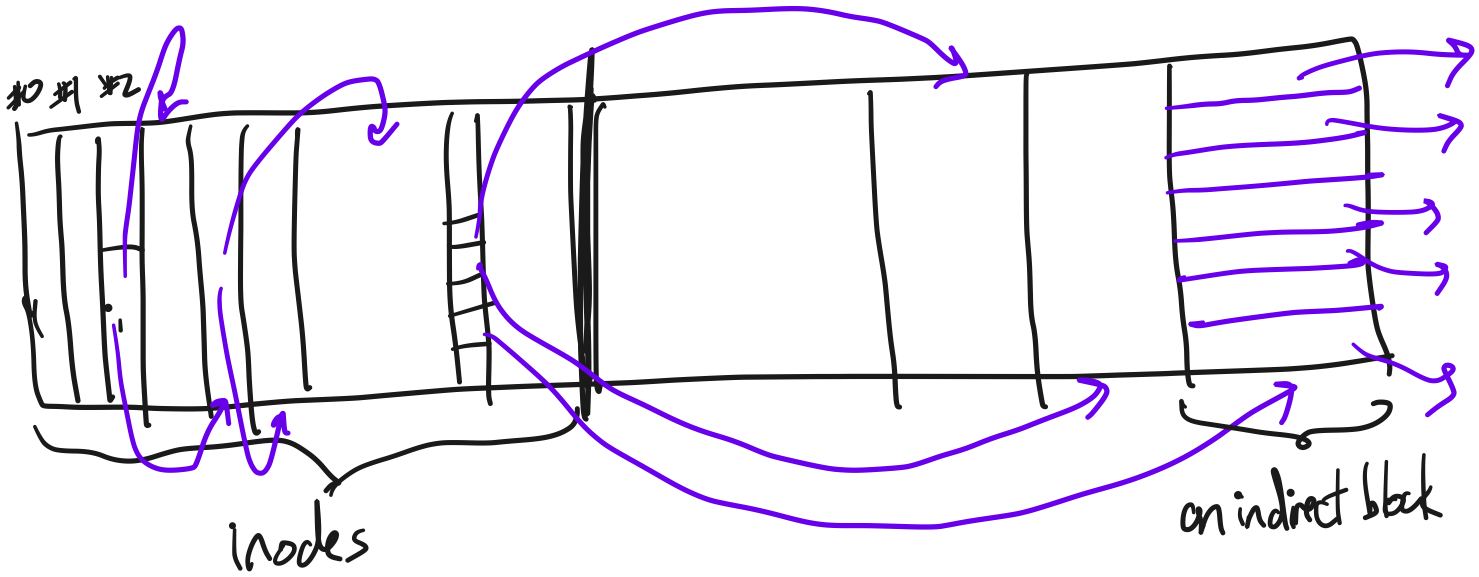
- [x] 1. Last time
- [x] 2. Crash recovery, continued
- [x] 3. RPC, client/server systems
- [ ] 4. NFS (case study of RPC + client/server)
  - [x] Intro + background
  - [x] How it works
  - [x] lab5 detour
  - [x] Statelessness (protocol statelessness)
  - [ ] Transparency

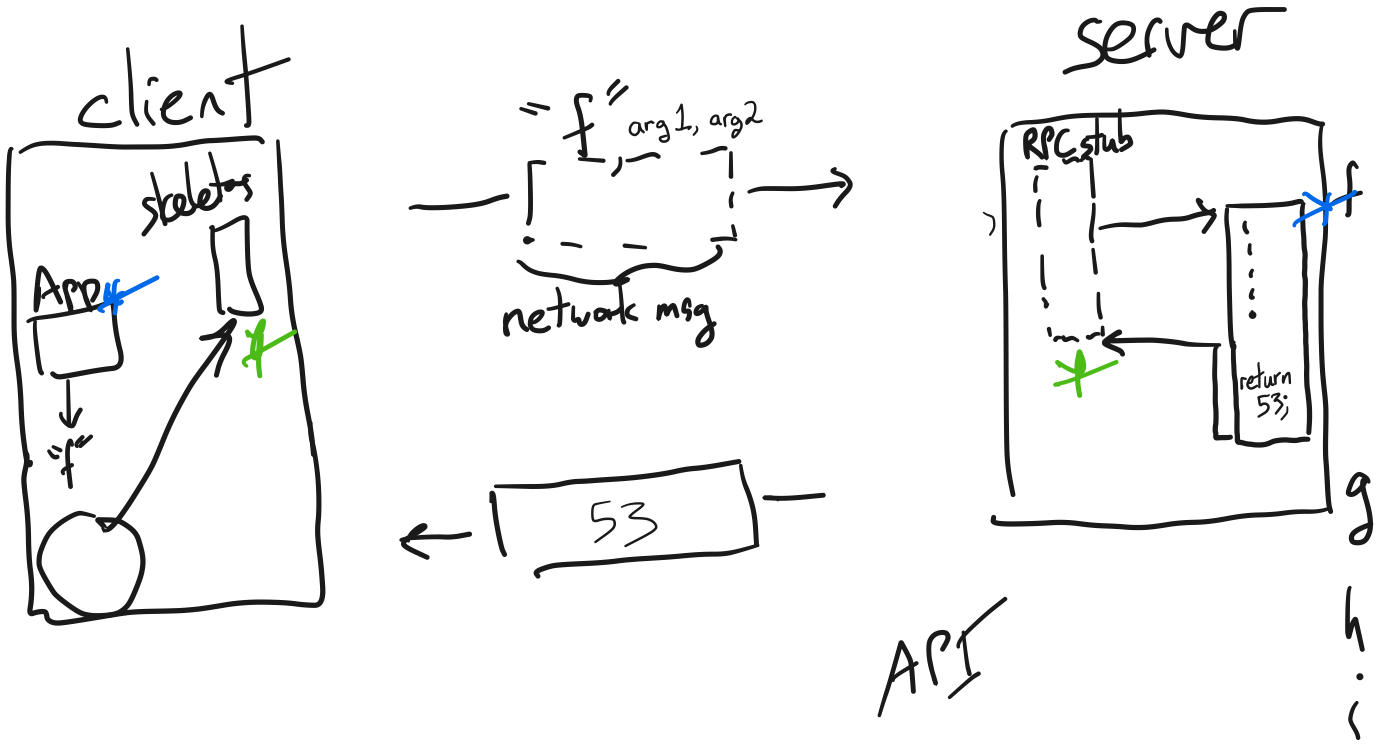
2.





LFS (redo+undo logging)

3. RPC, client server



4. NFS

$ ls

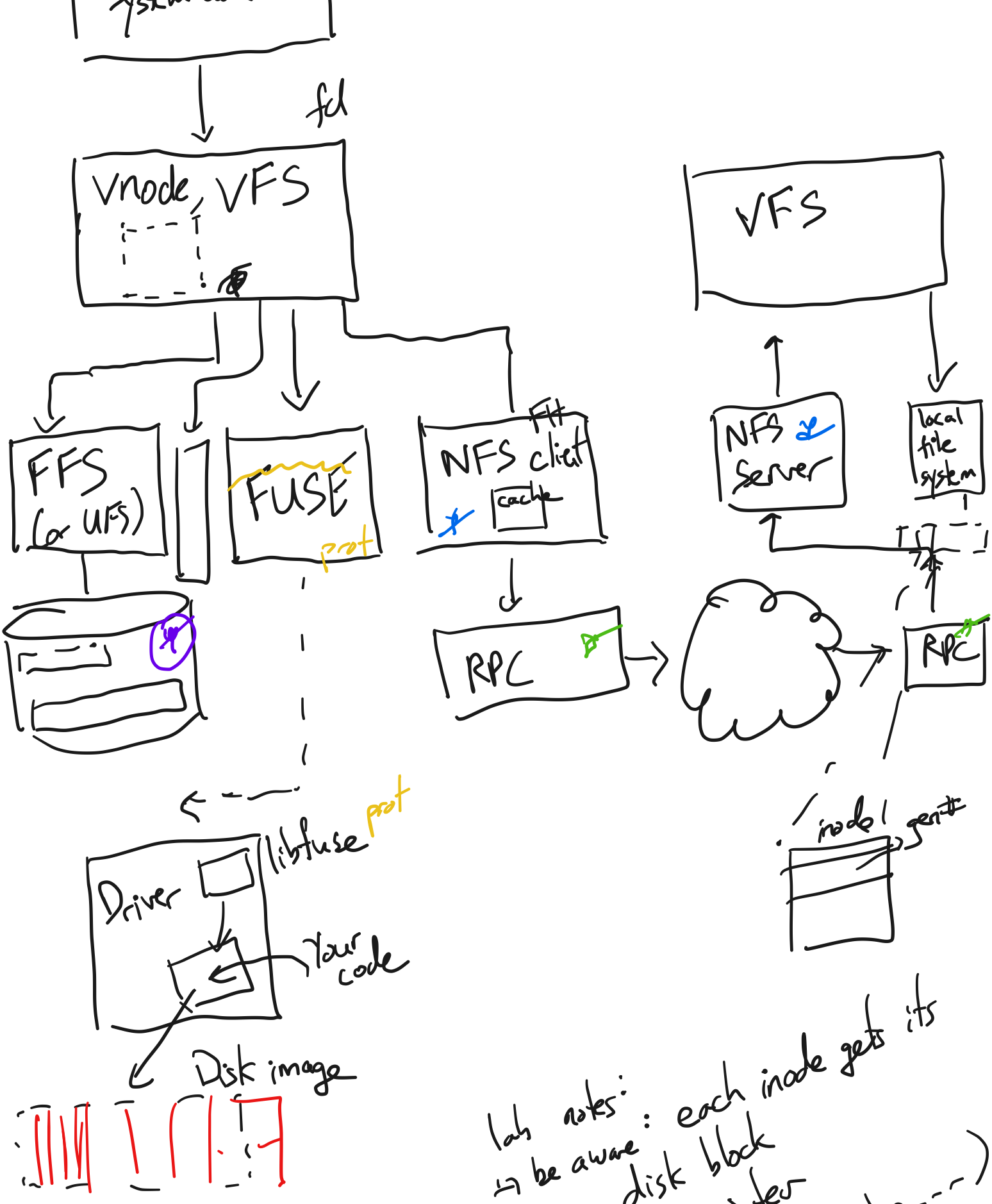
Server

Client

Application

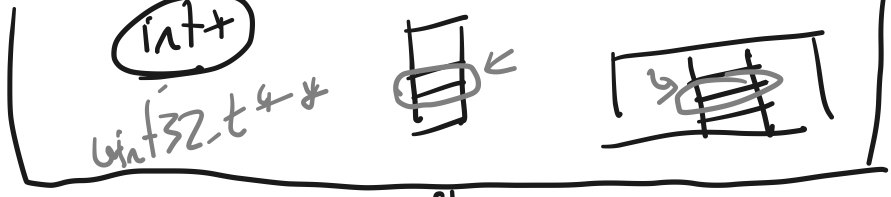
fd = open(___)
write(fd, buf, sz)
close(fd)

System call



lab notes:
→ be aware: each inode gets its own disk block
→ pointer-to-pointer
func(..., **ppblockno, ...)

f(char* ...)



open("/usr/jo/lab1.c",..)

/usr/jo/lab1.c, WR_ONLY

LOOKUP("/usr", fh0)

time

<- fh1: [FS id | inode# | generation #]

LOOKUP(fh1, "/usr/jo", fh2) "jo"

<- fh2

LOOKUP(fh2, "/usr/jo/lab1.c") "lab1.c"

<- fh: [FS id | inode# | generation #]

could be as a result of close() ... WRITE(fh, position (offset), buf

FS id | inode# |

data, size)

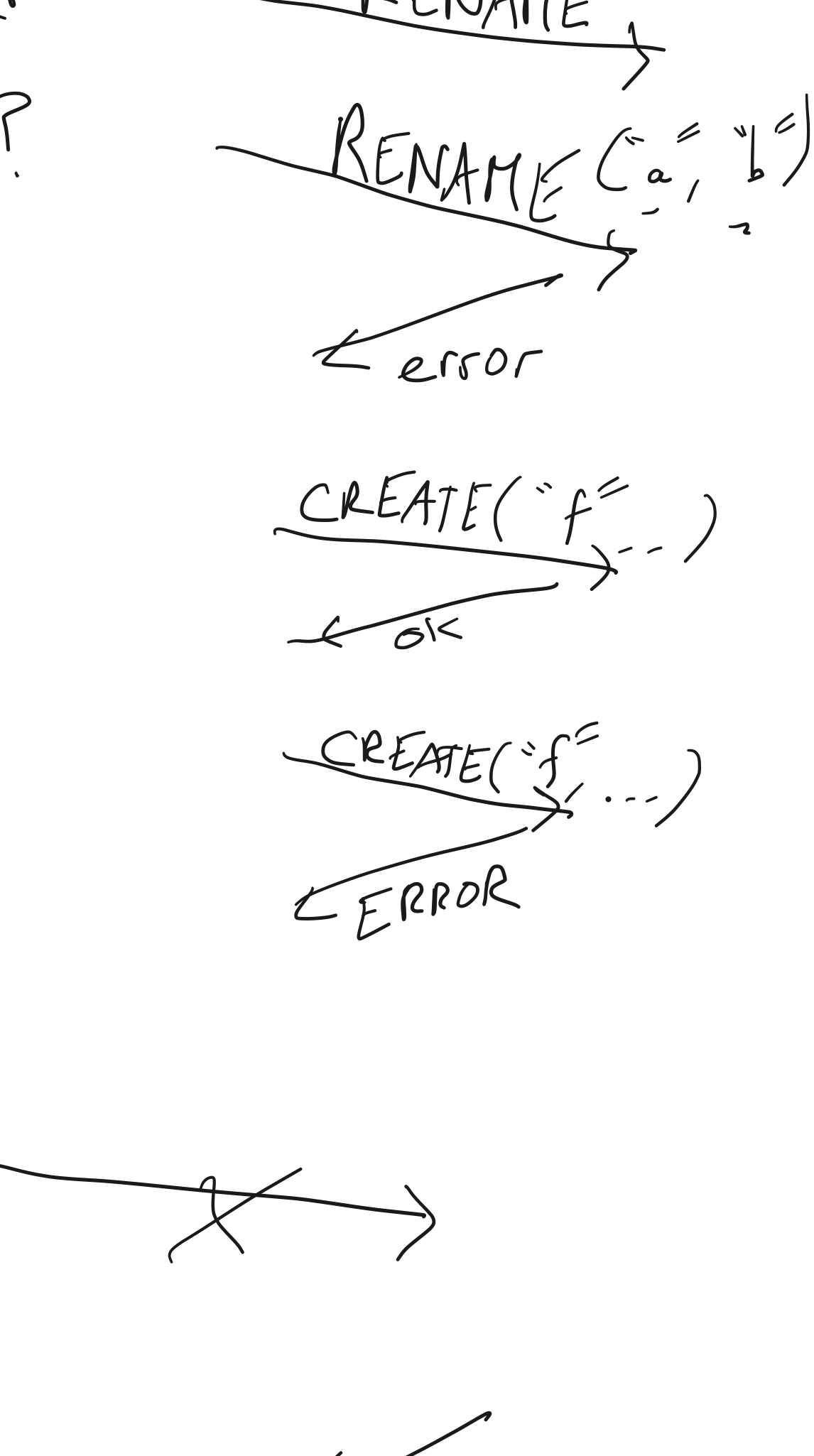
<- return code

WRITE: (idempotent

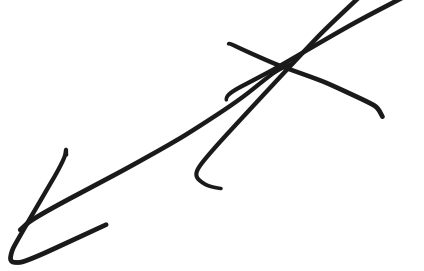
RENAME(a, b)

READ :

rename : ?

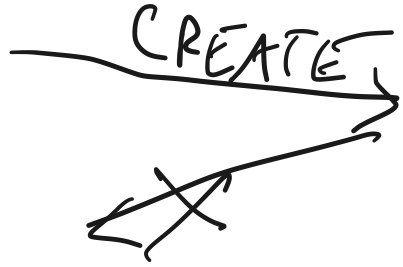


?



4. Transparency (we don't have it always!)

- One issue: gen #s added to legacy FS.
- Successful ops can return errors



caching

read-caching

write-caching

A: write();
/* caching */
close();

B: open();
read();

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

B
open();
read()

no guarantee the data is seen.

if ((rc = close(fd)) < 0)
error---