Persistence - I/O device (disk)

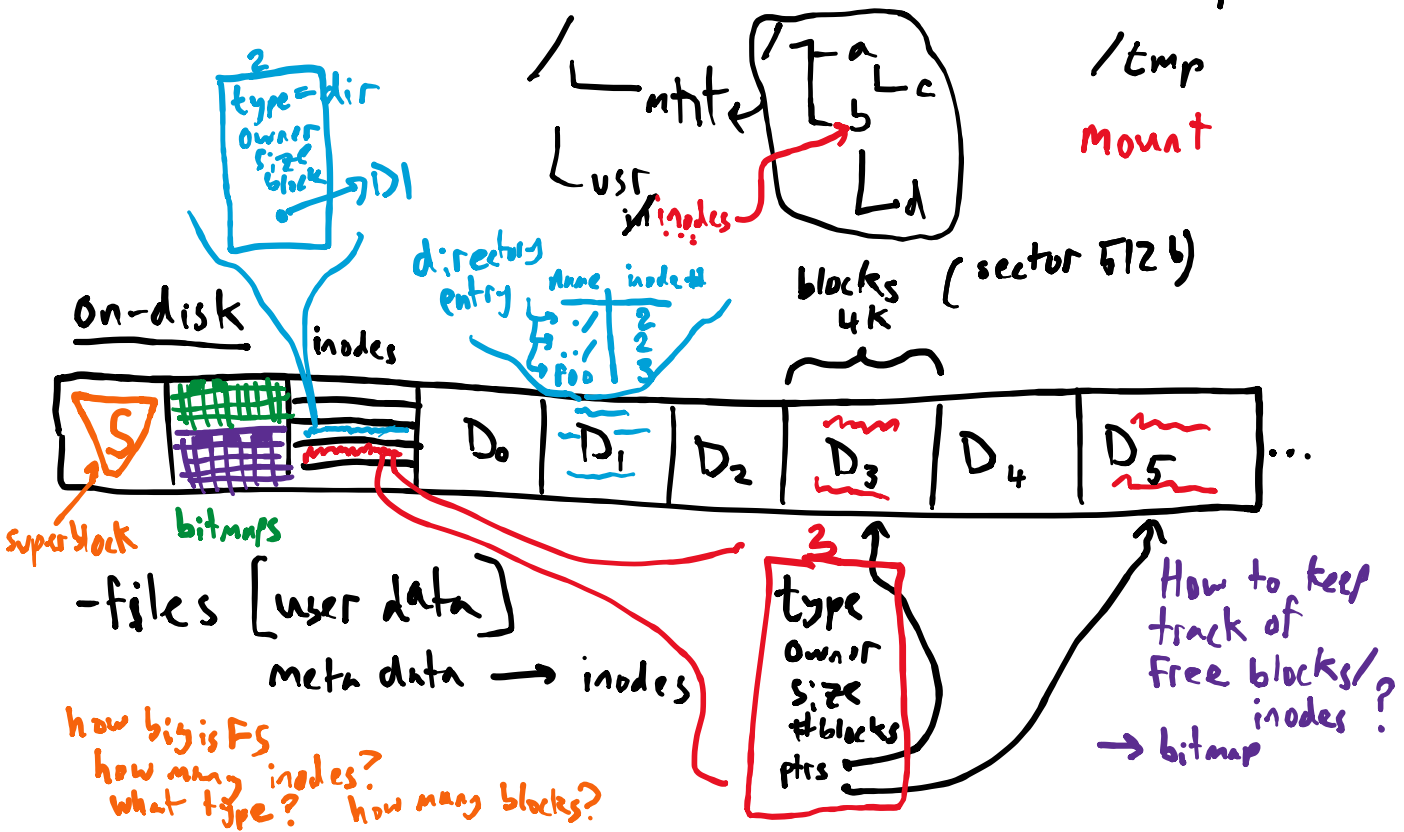
FS API files / directory

File
- open → fd → open file table (in-memory)
- read/write fd
- lseek offset
- fork/dup
- stat - info about file — size, perms, last accessed, "metadata" — on-disk: inode # / inode / "index node" → pointers to data blocks
- rm - unlink

Directories
- just a special type — ./ ../ /
- maps human-readable → inode # — /a/b/c/foo
- readdir (ls) → path traversal
- hard links (ln)
- soft links (ln -s) (symbolic) — special type of file → dangling pointer — contains a path

Mounting different FS

each filesystem is a "volume" → disk → partition



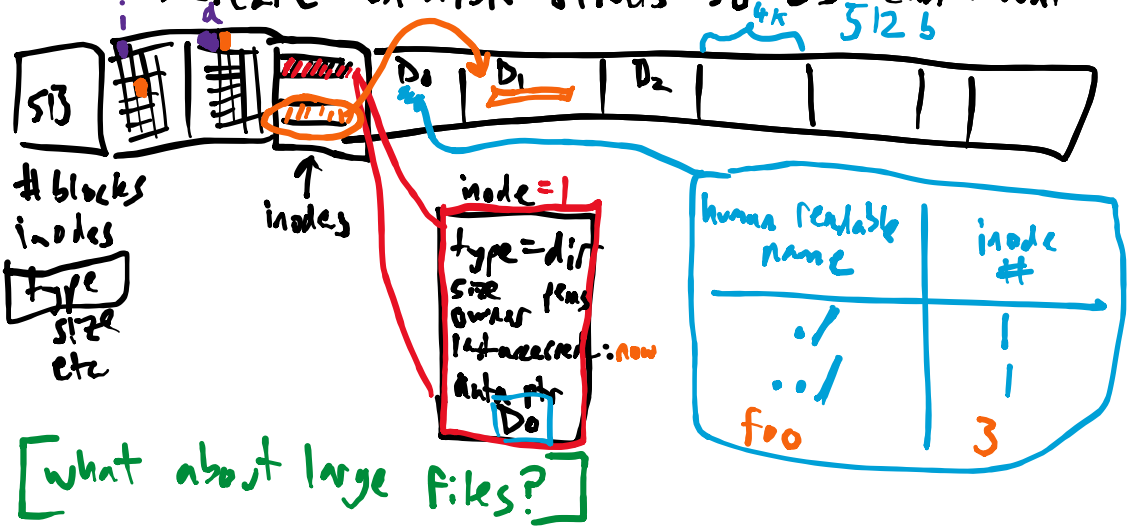
dd

metadata about FS
SUPERBLOCK

mkfs: create an empty filesystem

mkfs <disk partition>

↳ create on-disk structs so OS can mount




[what about large files?]

Create a file fd = open("/foo", O_WRONLY, O_CREAT);

1. read / 1. check permissions — read inode
   2. check uniqueness — read D0
2. create empty file "foo"
   allocate inode { 1. read inode bitmap
   2. write inode bitmap
   3. write inode #3 [type=file, owner=..., size=0]
3. link this to directory
   1. update dentry
      write to data block of /
   2. update inode of /
      access time, ...
      write inode of /
4. create in-memory structures
   fd → open file table "in-memory inode"
   return fd

write to file  write (fd, buffer, 4k);

1. allocate a data block
   - read data bitmap
   - write data bitmap
2. update inode  change - size
   - blk ptr
   - modify time, etc...
   - read inode
   - write inode
3. write data block

## Close file

free in-memory descriptor

free file struct from open file table

} no disk activity

what about long pathnames

"/a/b/c/d/x.txt"

open: what is read from disk

root inode (perms, loc of data)

root data (inode # of a)

a's inode

a's data (look for b's inode #)

b's inode

path traversal

b's data (look for c's inode #)

c's i

c's d

d's i

d's data

xtxt inode (final perm checks)

Next time

efficiency : page cache

: what about large files?