File System Implementation.

Last class: Data Structures (on-disk) | Access Methods.

FS APIs

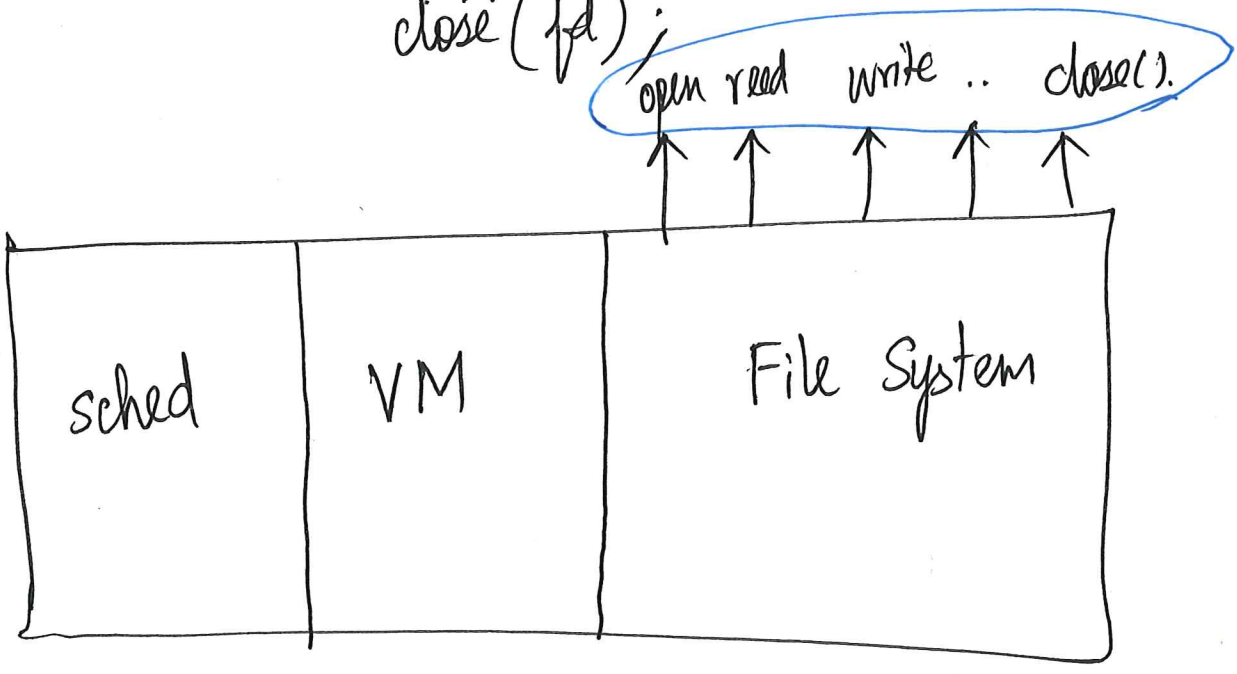
types: files, directories, symbolic links

files → low-level name (number eg. 160171)

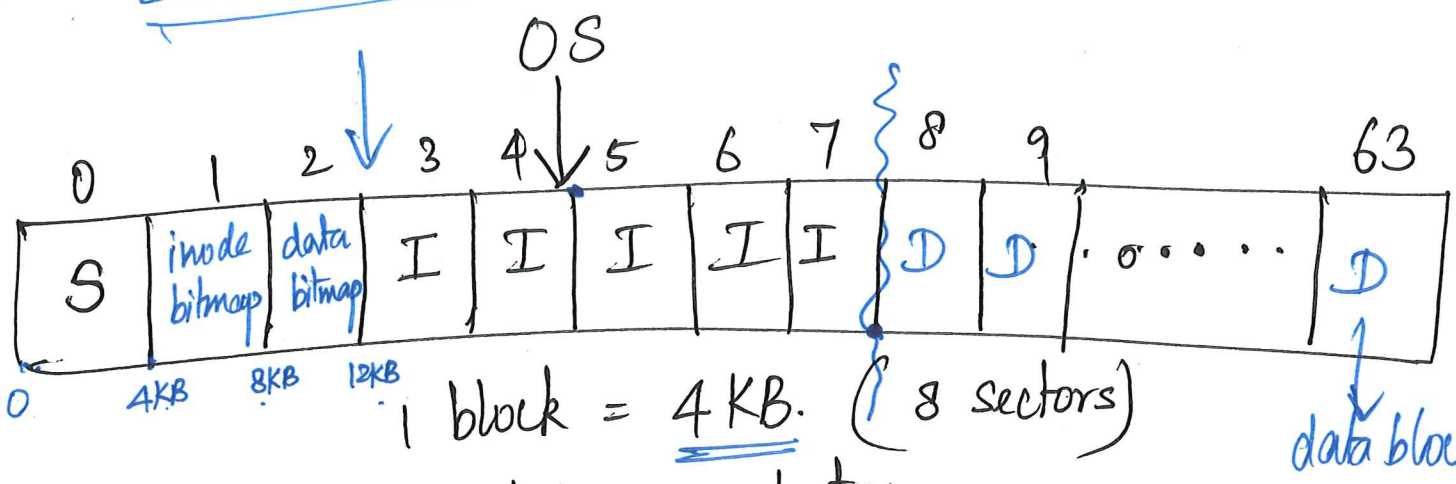
directories → human name → low-level name

APIs:

```
fd = open("file", O_RDONLY)
read(fd, buffer, size)
write(  "  "  ).
...
fsync(fd);
lseek(fd, ...);
...
close(fd);
```



First: store user data. — DATA blocks



1 block = 4 KB. (8 sectors)

1 sector = 512 bytes

Size of HDD = 64 × 4 KB

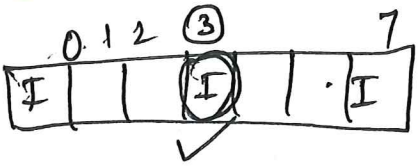
$= 2^6 \times 2^2$ KB = ~~1MB~~

$= 256$ KB

Second: per-file metadata ⇒ inode = index node → direct, ...

#blocks, #links, size, type, owner,

permissions, inode number, access time.

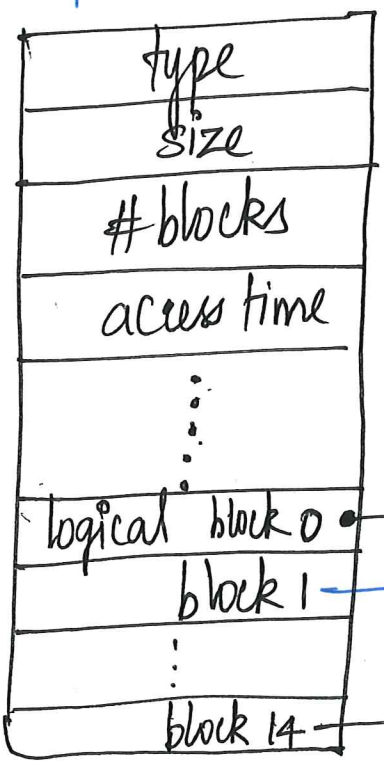


VSFS

Very Simple File System

size of (inode) = 128 bytes to 256 bytes

inode



→ data block 8

→ data block 21

→ data block 51

5 blocks → inodes

1 block → $\frac{4KB}{256} = \frac{2^{12}}{2^8} = 2^4 = 16$

⇒ 5 block ⇒ 16 x 5 = 80 inodes

⇒ 80 files

OS/FS → read block 8?

block #

$8 * 4KB = 32KB = \frac{2^{15}}{2^9}$

$= 2^6 = 64$

sector #

OS/FS → read inode # 32?

offset from inode start = $32 * 256 = 2^5 * 2^8 = 8KB$

inode addr = $12KB + 8KB = 20KB$

sector # = $\frac{20 \times 2^{10}}{512} = \frac{20 \times 2}{1} = 40$

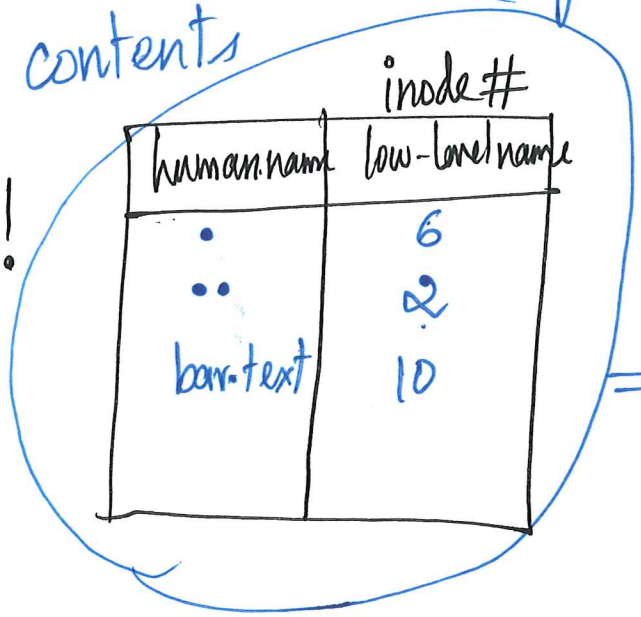
sector #

Third: need space for directories

→ just use inodes (type = directory)

→ contents

※ directory is just a file!



/foo

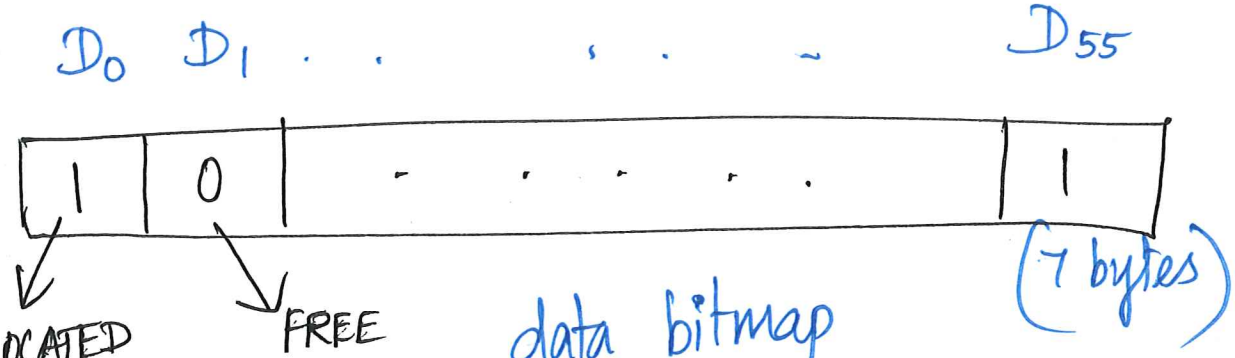
⟹ Data block for a directory.

Fourth: need way to track FREE / ALLOCATED

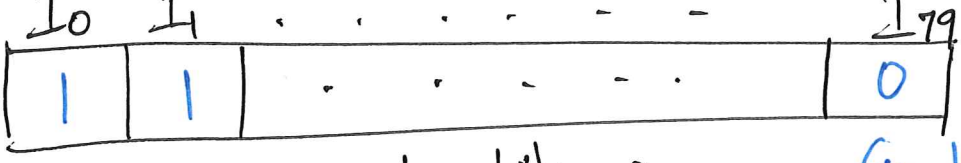
blocks

inodes data blocks.

bitmap



data bitmap



inode bitmap. (10 bytes)

Fifth: How big is the FS?
where in the disk ← bitmaps?
inodes? start addr.
data blocks? ✗

→ type of FS.
ext2, ext3, ext4

→ How many inodes are in inode table?

SUPER BLOCK.

Access Methods

/foo/bar → file

dir

contents of foo

H.N.	inode #
.	4
..	2
bar	10

data block of dir. foo

1. Read

read(fd, buf, size)

fd → bar.

read

→ ~~access~~ the "data" blocks of bar.

→ read inode of bar.

→ ~~data bitmap~~

→ inode of ~~bar~~ foo

→ read data block of foo.

→ read inode ~~dir~~ of root

→ read "data" block of root

→ read "super block" to get the inode # of the root dir.

content of dir / (root)

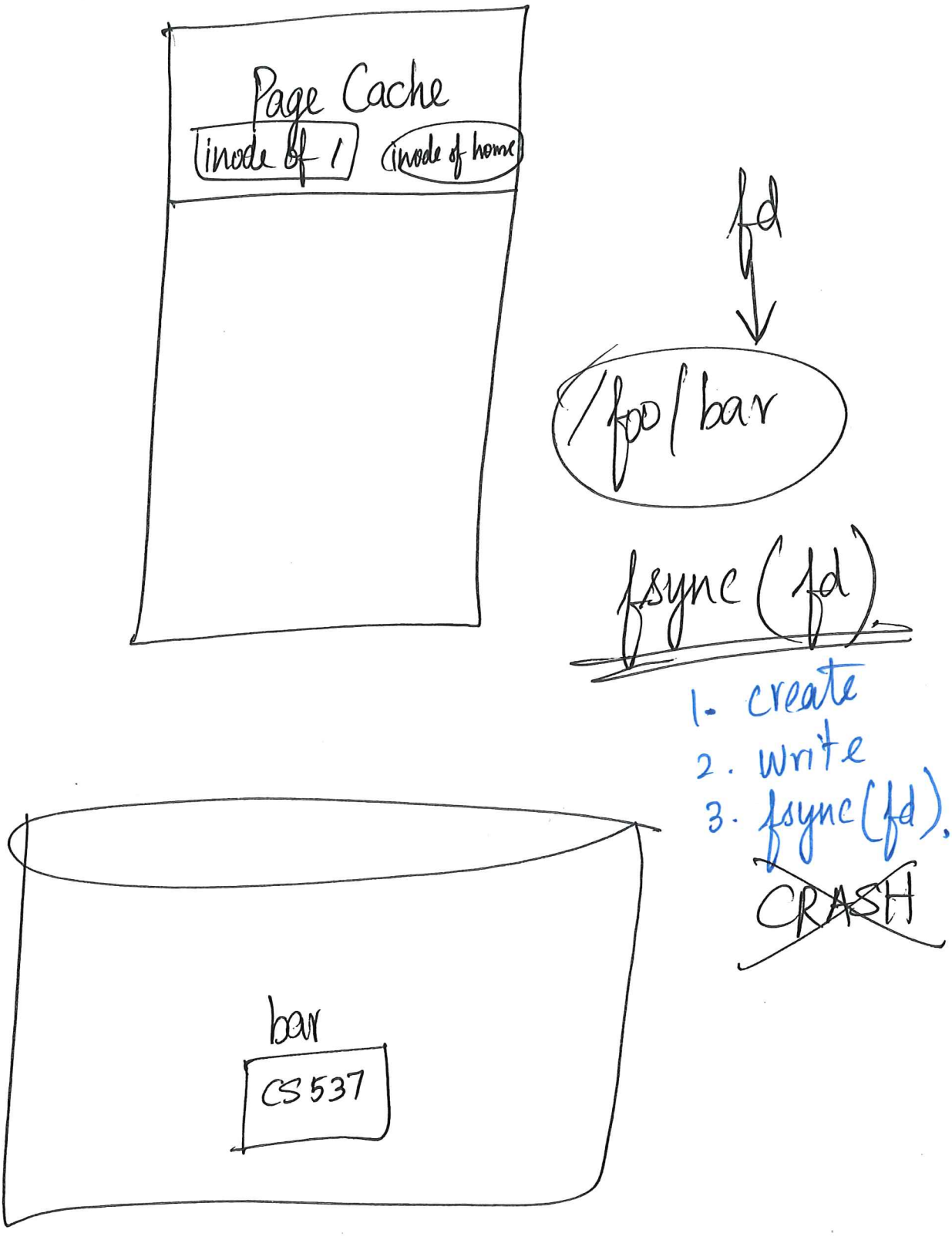
H.N	inode #
.	2
..	0
foo	4

2.

2. Write /foo/bar

rc = write(fd, buf, size);

1. read inode of root(/)
2. read data of root(/)
3. read inode of foo.
4. read data of foo.
5. read inode of ~~foo.~~ bar
6. read data bitmap to find a free block.
7. Write to data bitmap. for the corres. data block.
8. Write to inode of bar. (addr of the new data block).
9. Write to the data block of bar
10. Write to inode of bar.


Page Cache
inode of /
inode of home
fd
/foo/bar
fsync(fd).
1. create
2. write
3. fsync(fd).
CRASH
bar
CS537