CS202: File Systems 1

- Last Time:

  Disks

  Interface: Read or write sector

  block ← In this lec., notes, etc.

  Performance characteristics

  - Sequential access is much faster than random access

  Why?

File System

Primary Use: Read & Write Data Stored In A Device That Provides Persistent Storage

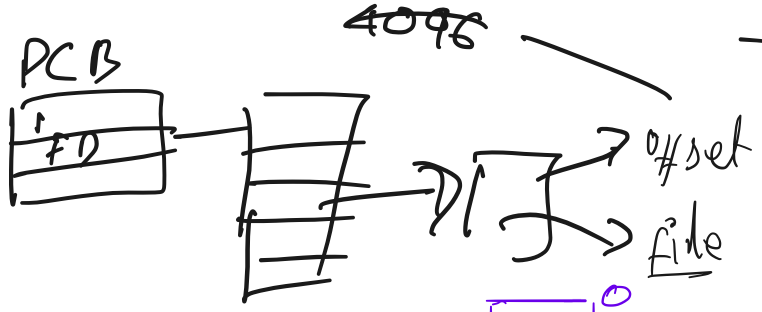
(Persistent ≡ Survives Process & Computer Restart)

Needs To Provide A Way To

- Manage Space On Device
- Name Data Stored On Device.

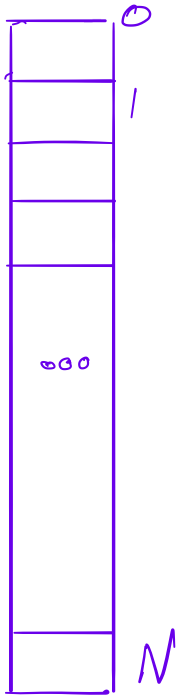
File Systems have targetted many devices, we will focus on disks.

Files & File Systems



fd = open("/tmp/out.txt", ...);

lseek(fd, _)



File System

```
read(fd, buf, 4096);
...
read(fd, buf, 4096);
...
```

Map <file, offset> → disk block

Interface

Open: Open a file

↳ O_CREAT: Create a new fil

lseek: set offset

read: Read from current offset

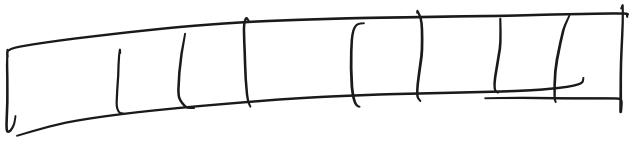
write: Write starting from current offset (might need to allocate blocks)

GOALS

- Minimize disk access time
  - ↳ Sequential is better
- Minimize wasted space

EMPIRICAL OBSERVATIONS: Most files are small; most of the disk is used by large files

Have both sequential & random access.



Q: How to Layout files

- Dictates how disk blocks are allocated
- Read and write performance.

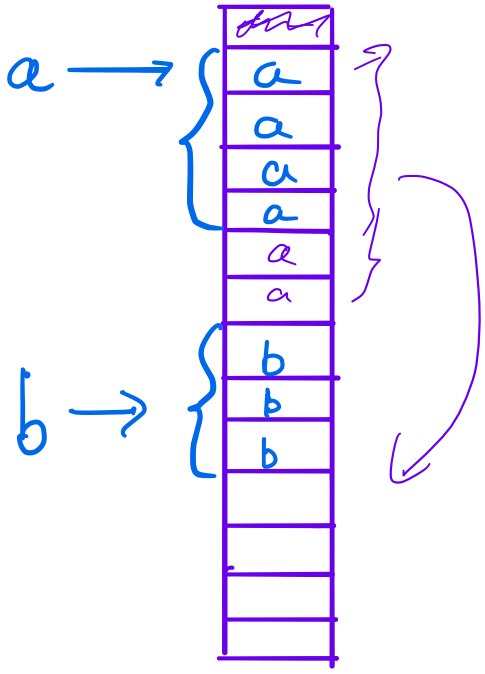
(a) Contiguous Allocation

Put all of a file's blocks next to each other

+ Great for seq & random access

- Adding a new block to a file is challenging

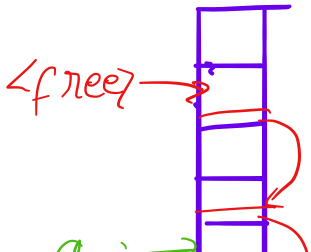
- FRAGMENTATION



(b) Linked Files

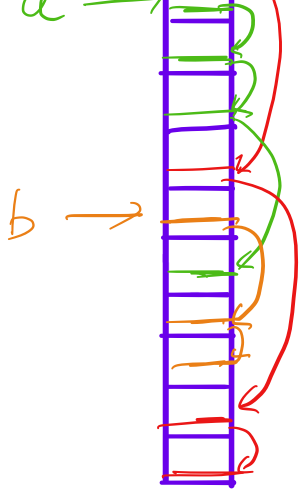
+ Easy to extend file

- Only reasonable for sequential access



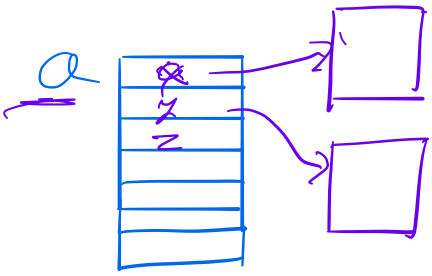
+ fast sequential access

- BAD RANDOM ACCESS PERFORMANCE

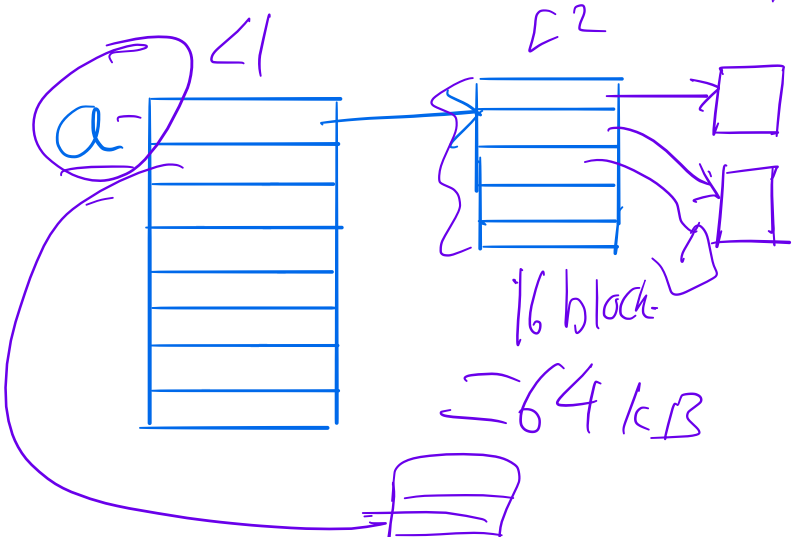


© INDEXED

↳ STORE THEM IN A TREE



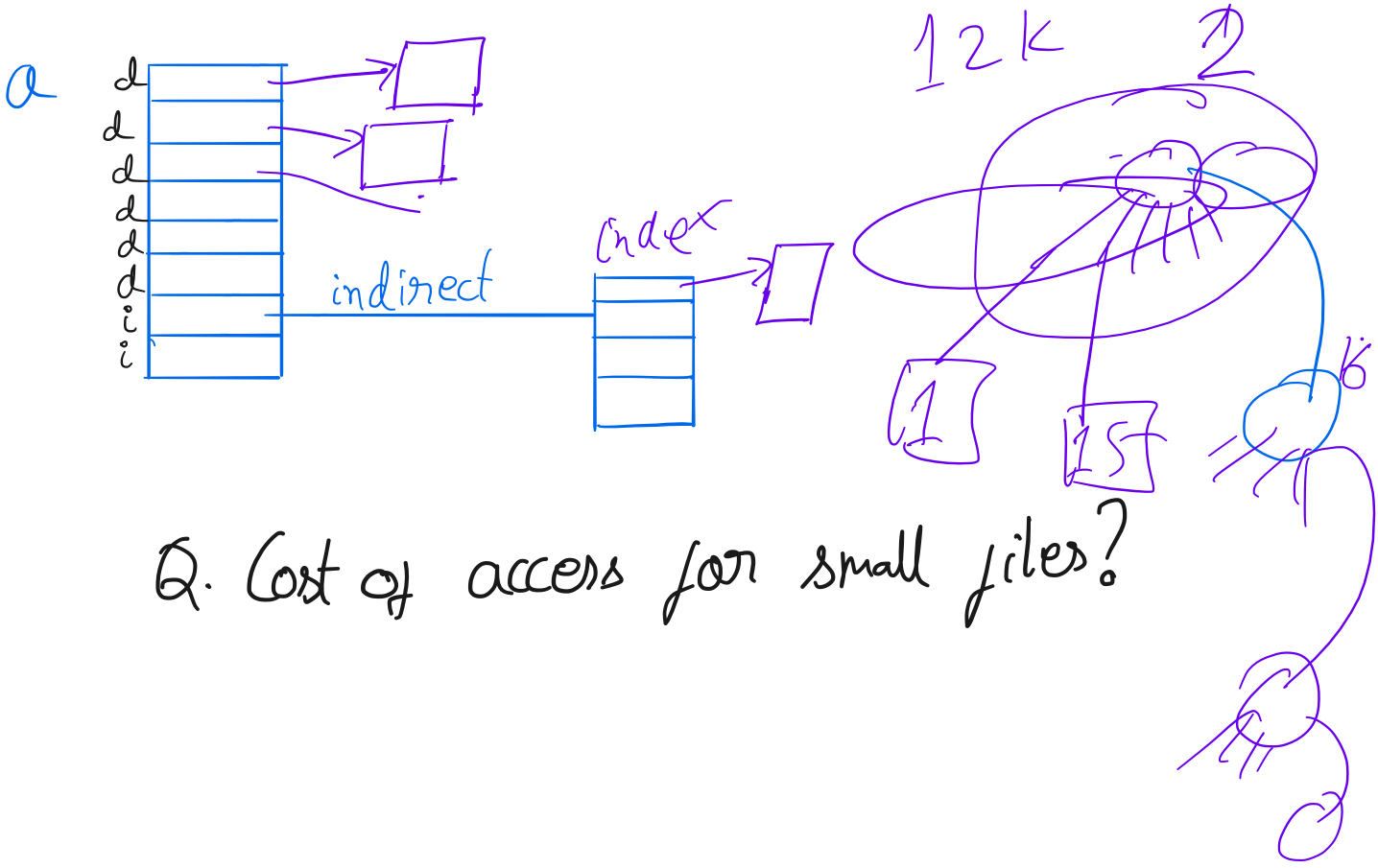
Q. How many entries?



fd = open(a)
lseek(fd, 3.4.2^10)
read(fd, c, 1)

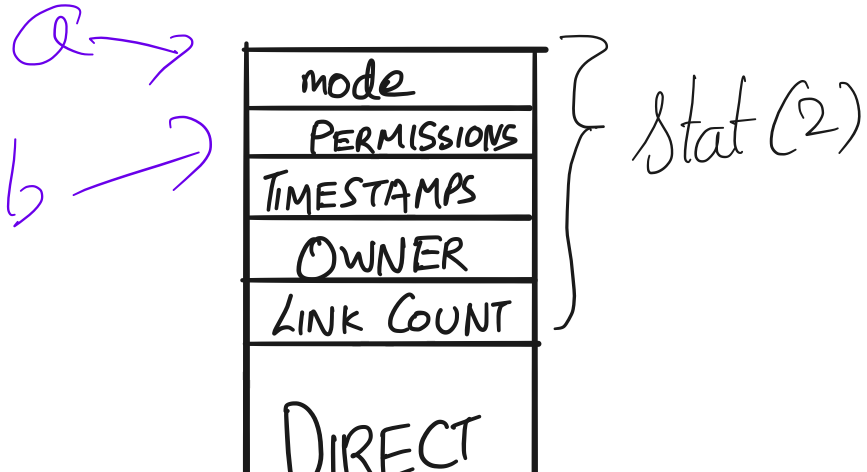
Q: Cost of reading 1 byte?

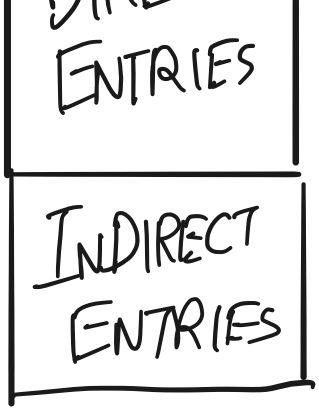
Q: Cost of reading file with 1 block?



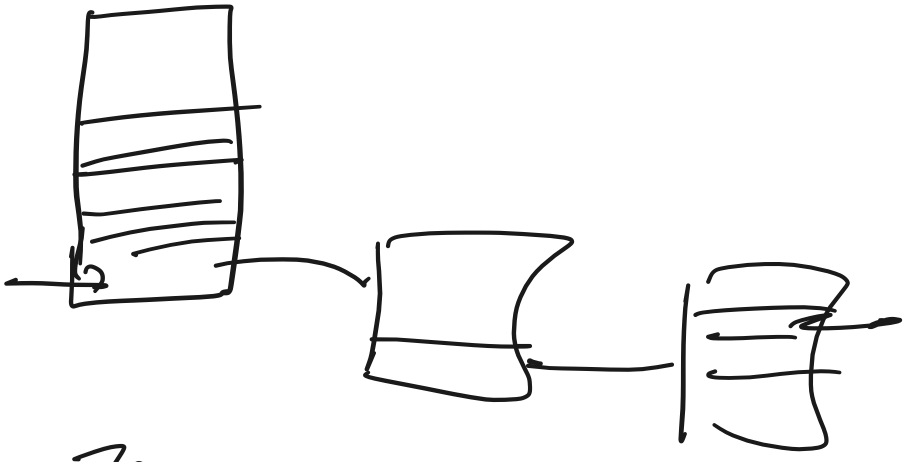
Q. Cost of access for small files?

Index Nodes – inode

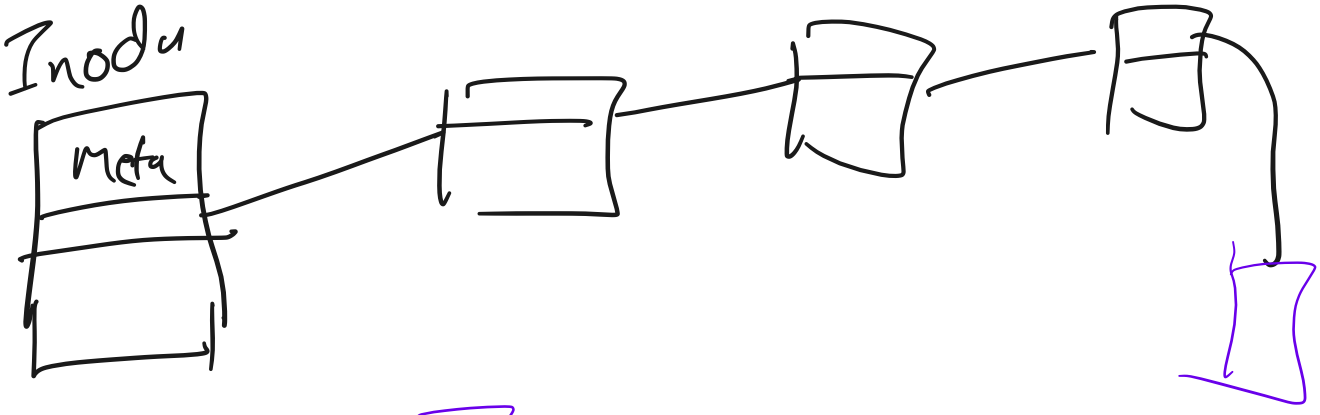


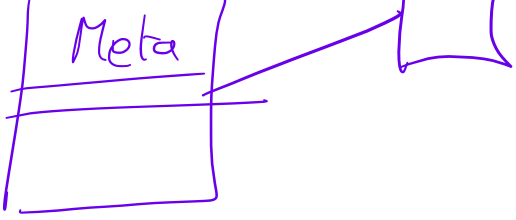


DIRECTORIES



70,000




Meta