Last Time:
- Disks

Interface: Read on write sector block. In this less, note, 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
Files & File Systems

Files & File Systems

```
fd = open("/tmp/out.txt", ...);
read(fd, buf, 4096);
...
read(fd, buf, 4096);
...
```

Interface

```
open: Open a file
    \rightarrow O_CREAT: Create a new file

lseek: set offset
    for read from current 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
  - Avg reasonable
- Fast sequential access

- BAD RANDOM ACCESS PERFORMANCE

© Indexed

⇒ Store them in a tree

Q: How many entries?

```c
fd = open(a)
libc (fd, 3, 2)
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

mode
PERMISSIONS
TIMESTAMPS
OWNER
LINK COUNT

\{ Stat (2) \}

Jim a.