

File Management

Note Title

2/6/2007

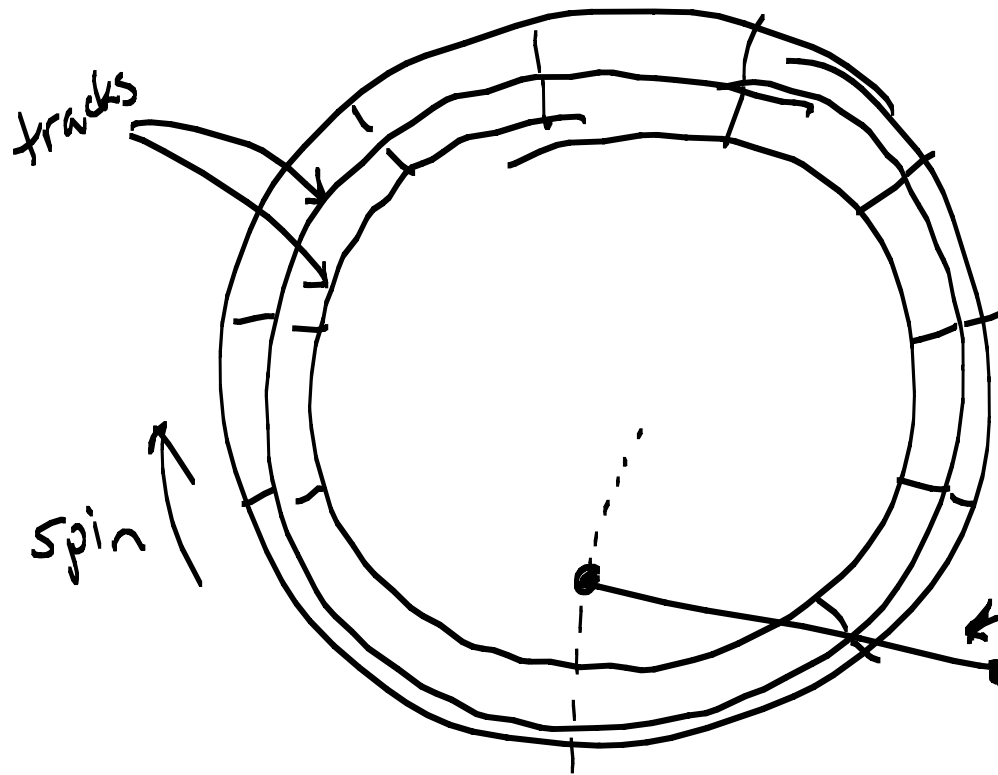
Problem: Want persistent data, but

- RAM is volatile.
- RAM is a scarce resource.

Possible Solutions:

- Non-volatile RAM (Flash memory, carbon nanotubes)
- External storage (disk)

- Physical device
- Operating System File Abstractions
- Programming Language Support for Files
- File Formats
- Other abstractions



(track #,
sector #)

⇒ disk block/
disk page

Driver
Buffer Memory → O.S.

Driver abstraction
Disk blocks accessed by track/sector

Driver abstraction

Disk blocks accessed by track/sector

Too low level of an abstraction!

- sharing — agree on which apps use which space
- protection
- Users can't be expected to remember where the data is

⇒ abstraction for external storage management

- Operating System provides a File System
variation among O.S.
- Language can provide higher level abstraction on top
mask variation
provide richer set of tools

Disk blocks

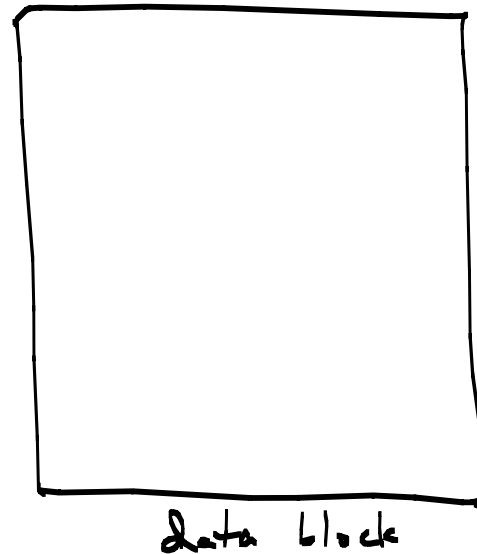
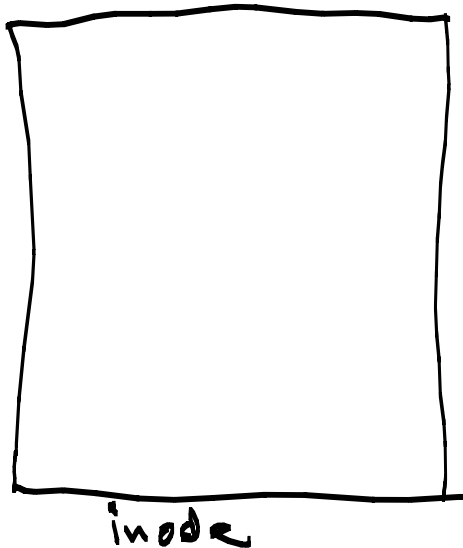
File System \rightarrow uses disk blocks as a primitive for building an on-disk data structure

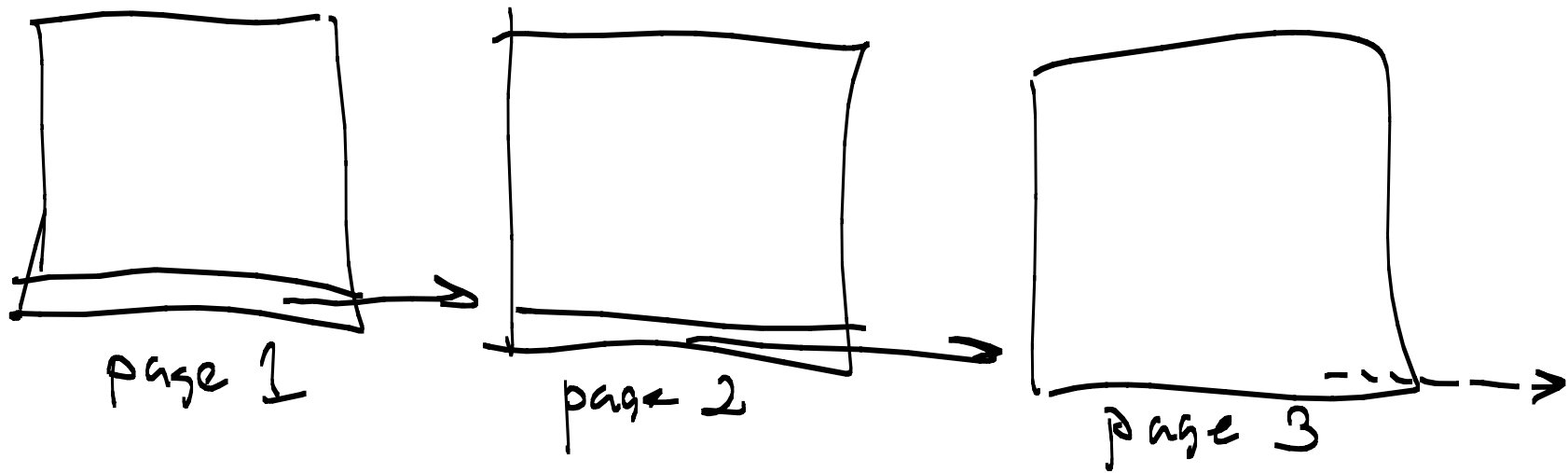
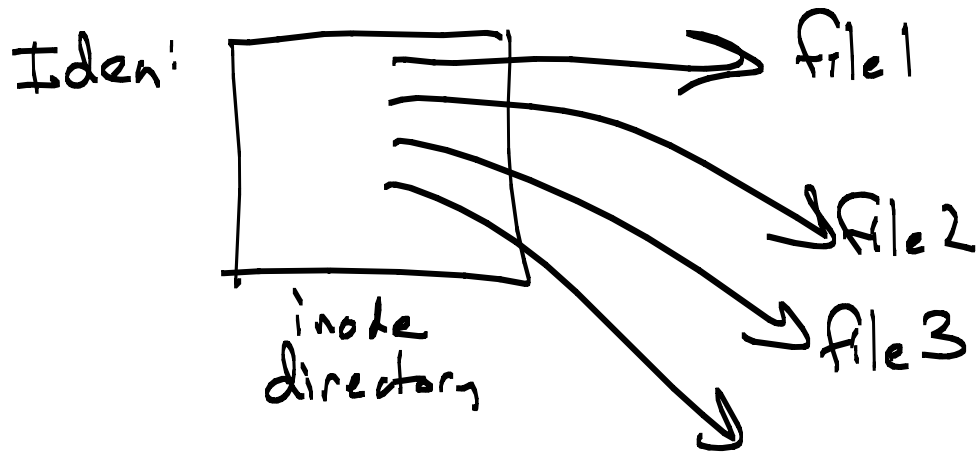
Latency is slower than RAM

Fixed "object" size — one block

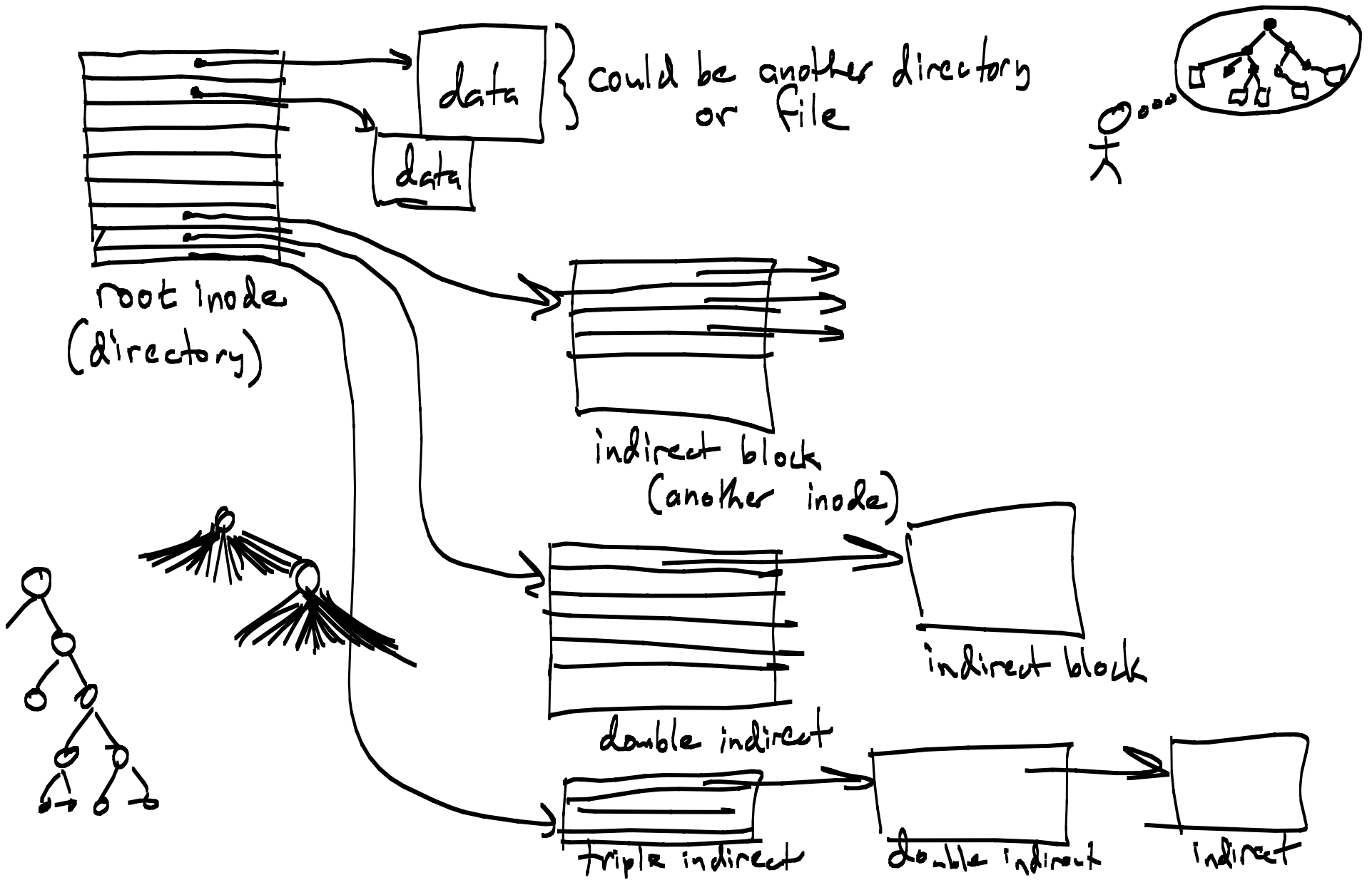
\Rightarrow pack data in — lots of locality
minimize # of reads to find what you want

Each block in Unix File system





way too slow!!



File System Abstraction

- User View (Files + Folders + string names)
- API w/ User interface

name a file
open a file
read/write
close

name a directory
list its contents (add a file, ...)

(O.S. basic features)

At this point, discussed (on the chalkboard):

- streams
- orthogonality of source/destination, stream management, & data format
- example of wrapping streams with various filter types