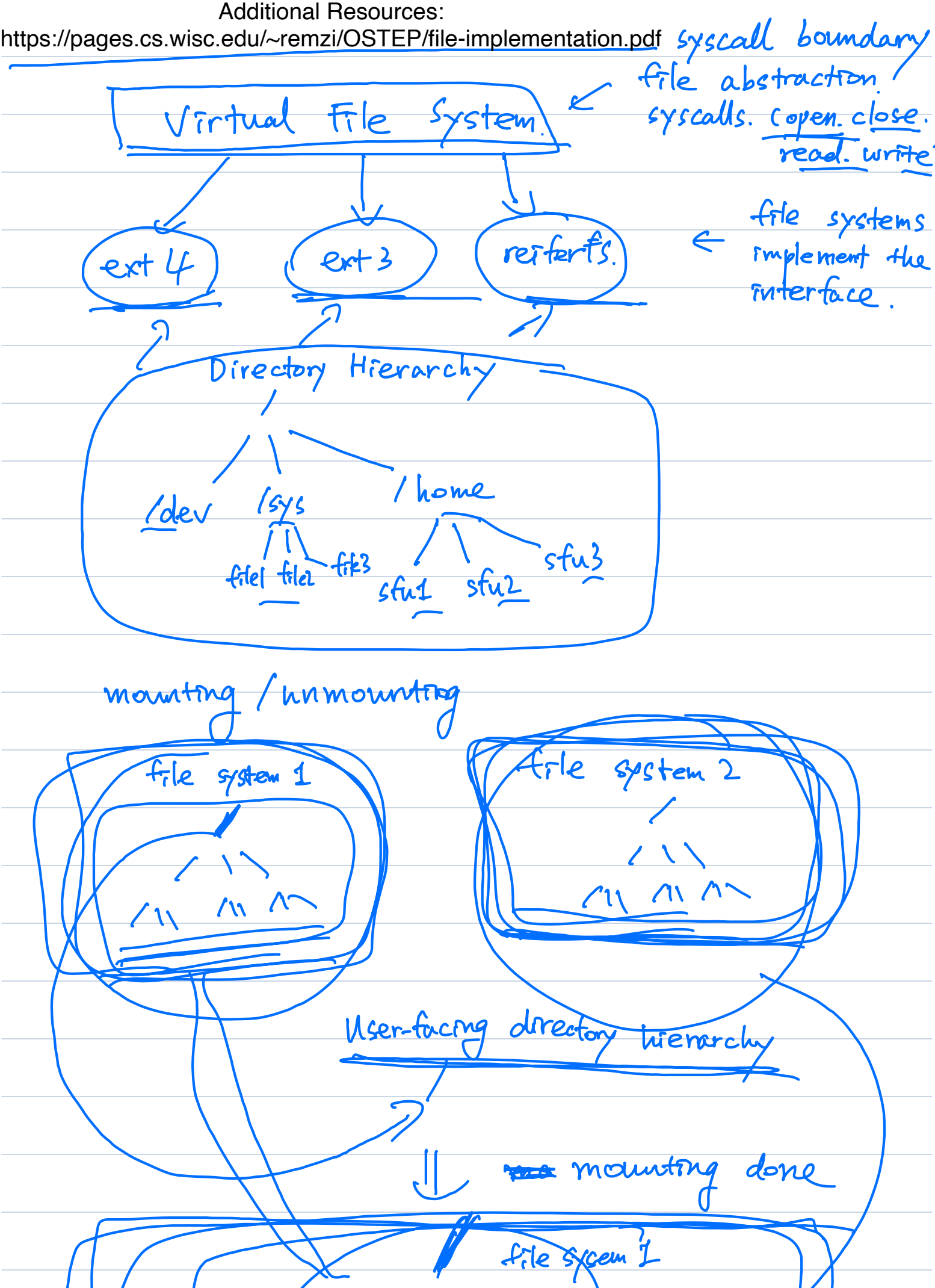
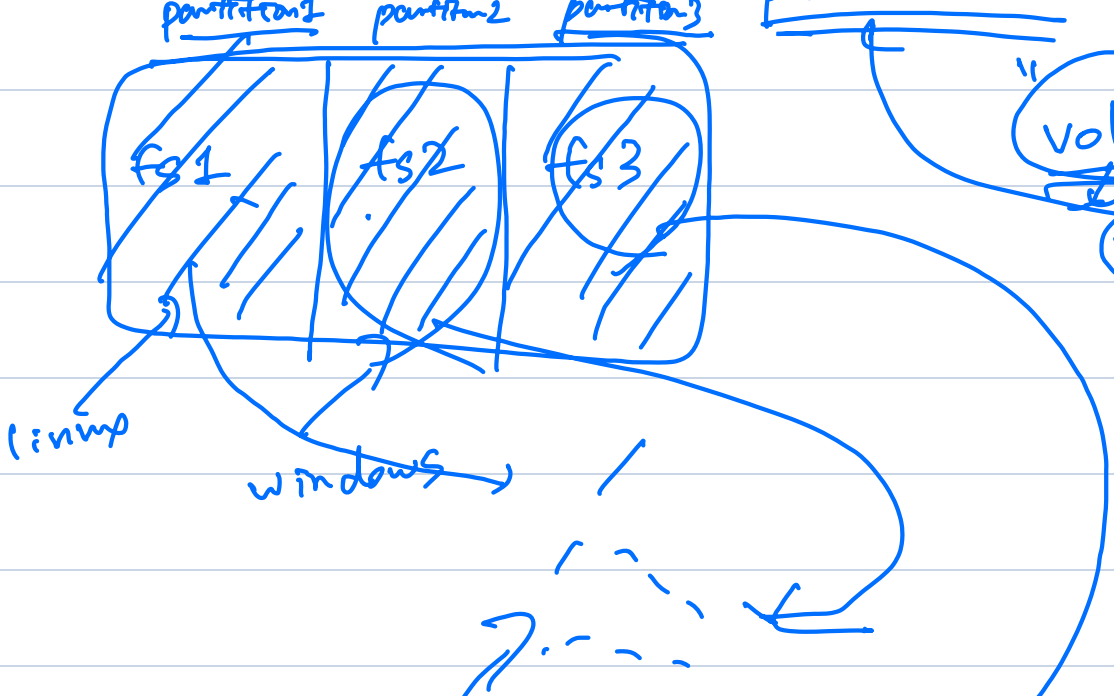
Additional Resources:
https://pages.cs.wisc.edu/~remzi/OSTEP/file-implementation.pdf



"mount" command

/dev



partition for a disk.

"volumes.

fs'

MBR

disk block : 4KB. (matches the page size)

* files. : data content
* Directories : considered files.
  content is the files that they contain.

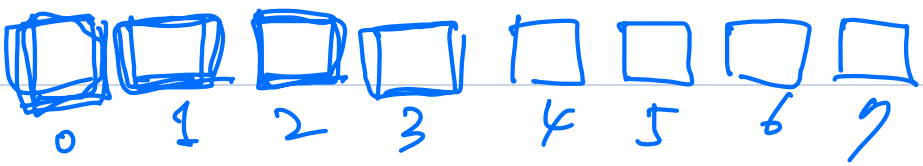
metadata : i-nodes

data : file contents

data blocks

metadata blocks (i-nodes)

* Example

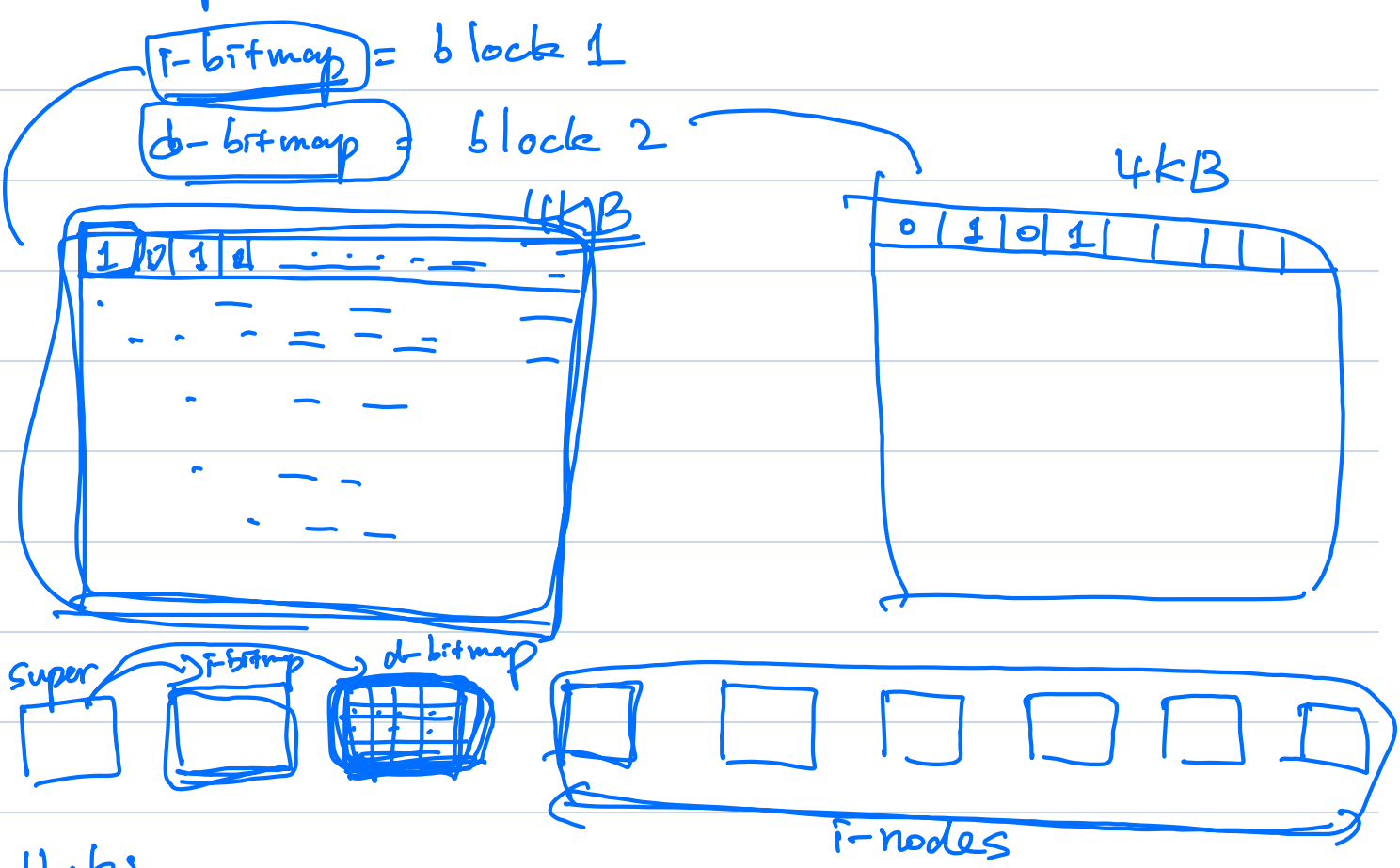


: meta data.
* i-node block
* i-bitmap. d-bitmap
* super block

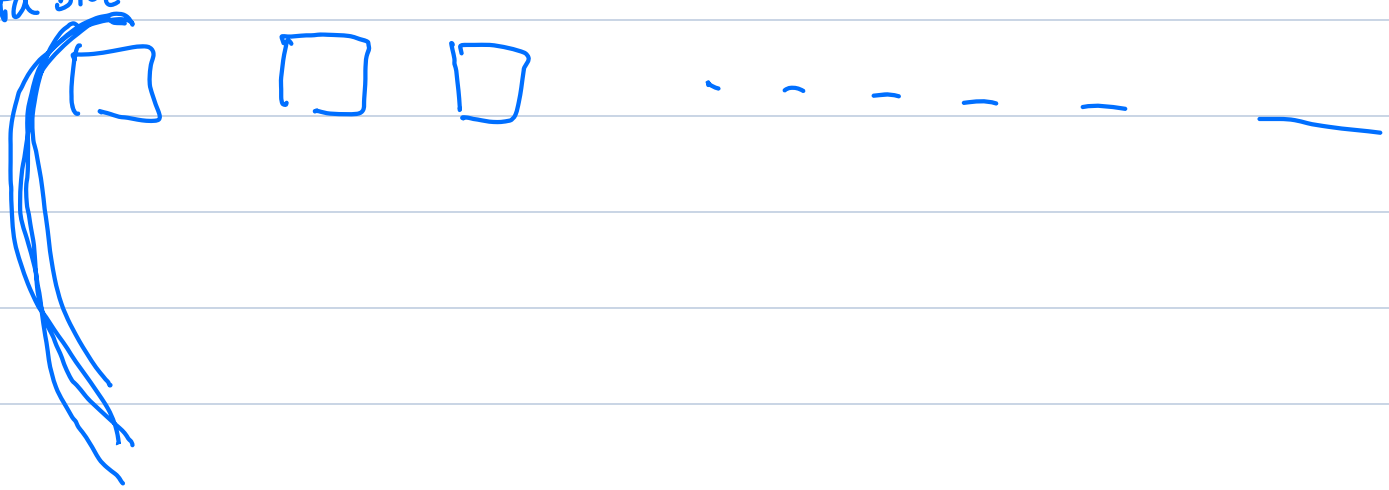
: data blocks

* meta data blocks
  * i-node blocks
  * i-bitmap. d-bitmap
  * Superblock (first block) block 0

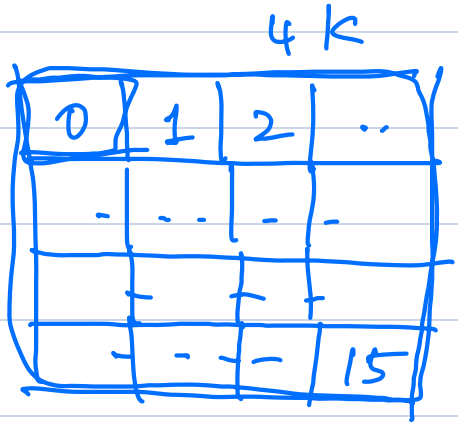
* Example.



data blocks



* i-node block

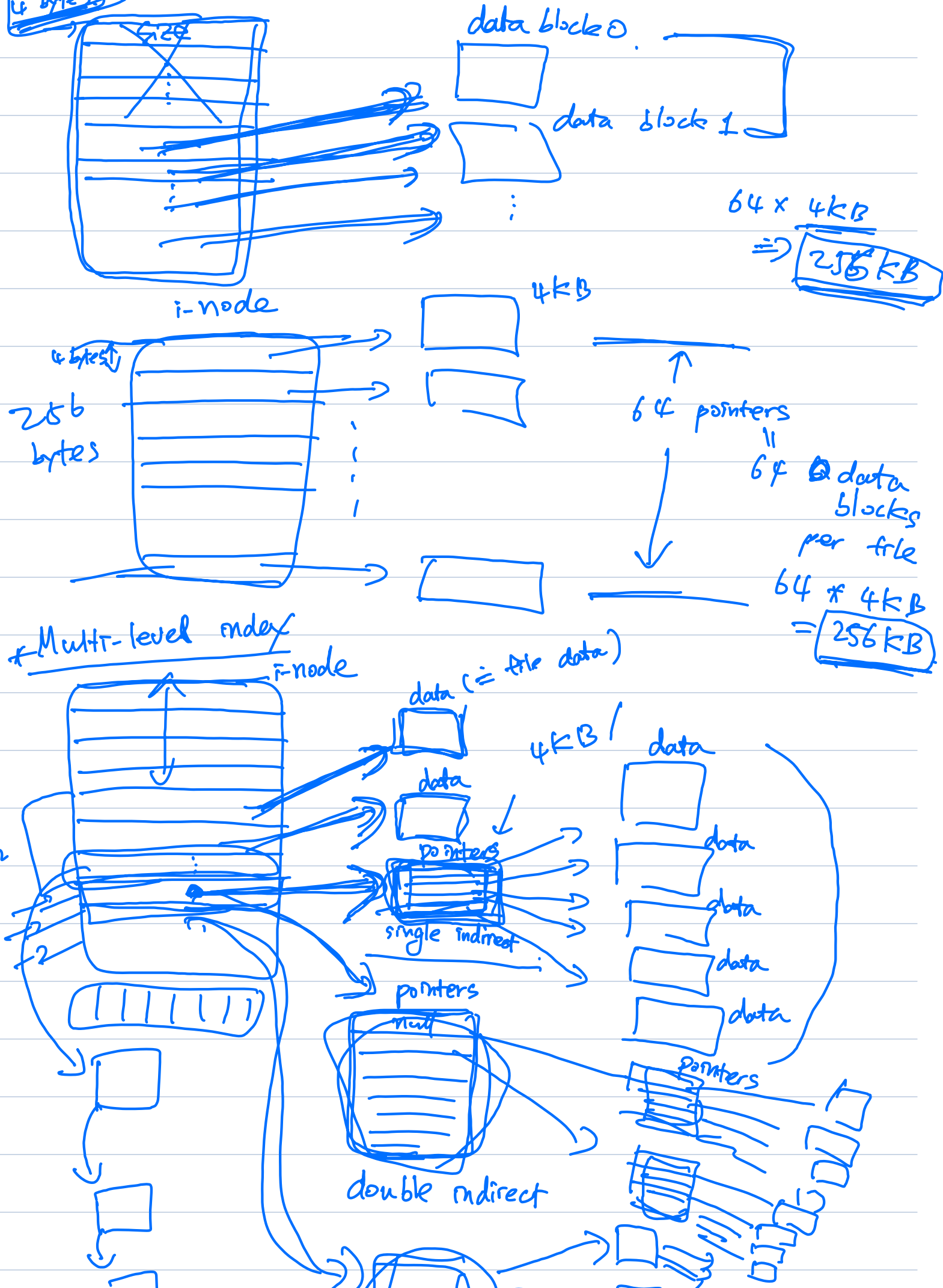


256 (4)

* size
* ~~time~~ last accessed time
* time created
* permission
⋮

* file name is not stored. (stored in the directory file that contains this file)
* where the data blocks are

i-node 256 bytes


4 bytes
size
data block 0
data block 1
64 x 4kB
=> 256kB
i-node
4kB
4 bytes
256 bytes
64 pointers
=
64 data blocks per file
64 * 4kB
= 256kB
* Multi-level index
i-node
data (= file data)
4kB
data
pointers
single indirect
pointers
double indirect
12
1
2
2

* Q: Assume block size: 4KB
pointer size: 4 bytes

→ 12 direct entries (pointers)
→ 1 single indirect entry (pointer)
→ 1 double indirect entry (pointer)
→ 1 triple indirect entry

what's the maximum file size?

A: 12 * 4KB

1024 * 4KB

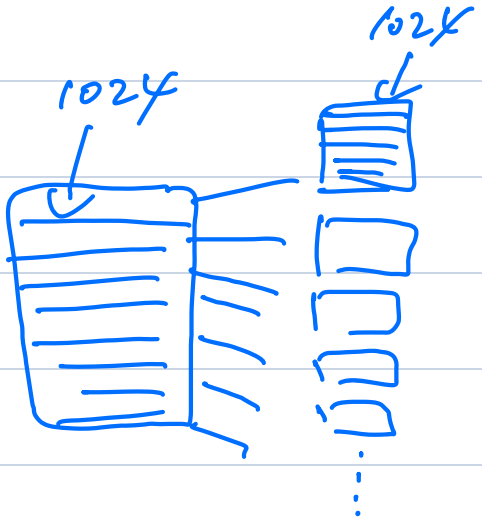
1024 * 1024 * 4KB

1024 * 1024 * 1024 * 4KB

= 4TB

4KB block / 4 = 1K.
1024 pointers.

4KB/4B
= 1K
= 1024



= 1024 * 1024

* open: /foo/bar

i-node #2

read root i-node
read root data block →

file0 → i-node
file1 → i-node
foo → i-node
⋮

read foo i-node
read foo data block → file2 → i-node
file3 → i-node
bar → i-node
⋮

read bar i-node

fd = open ("/foo/bar");

→ bar i-node
read (fd, buf, ...)
write (fd, buf, ...)

* read (fd)
read bar i-node
read bar data
write bar i-node

* write (fd)
read bar i-node
read data bitmap
write data bitmap
write data block
write bar i-node

* create "/foo/bar"

• read root i-node
read root data → foo → i-node
read foo i-node
read foo data → ~~bar~~

— read r-node bitmap
— write r-node bitmap ←

— write foo data { bar → r-node

— read bar r-node
— write bar r-node (initialization)
— write foo r-node

* caching. (write-back
write-through