6.5081 : File systems

User-friendly names / pathnames

Share files between users/processes

Persistence / durability

Why interesting?

two lab

Abstraction is useful

Crash safety ← Wed.

Disk layout

Performance ← storage devices are slow

buffer cache

concurrency

API example / file systems calls

```
fd = open("x/y", ...);
write(fd, "abc", 3)
link("x/y", "x/z")
unlink("x/y")
write(fd, "def", 3)
```

pathnames

human readable

object

offset is implicit

multiple names

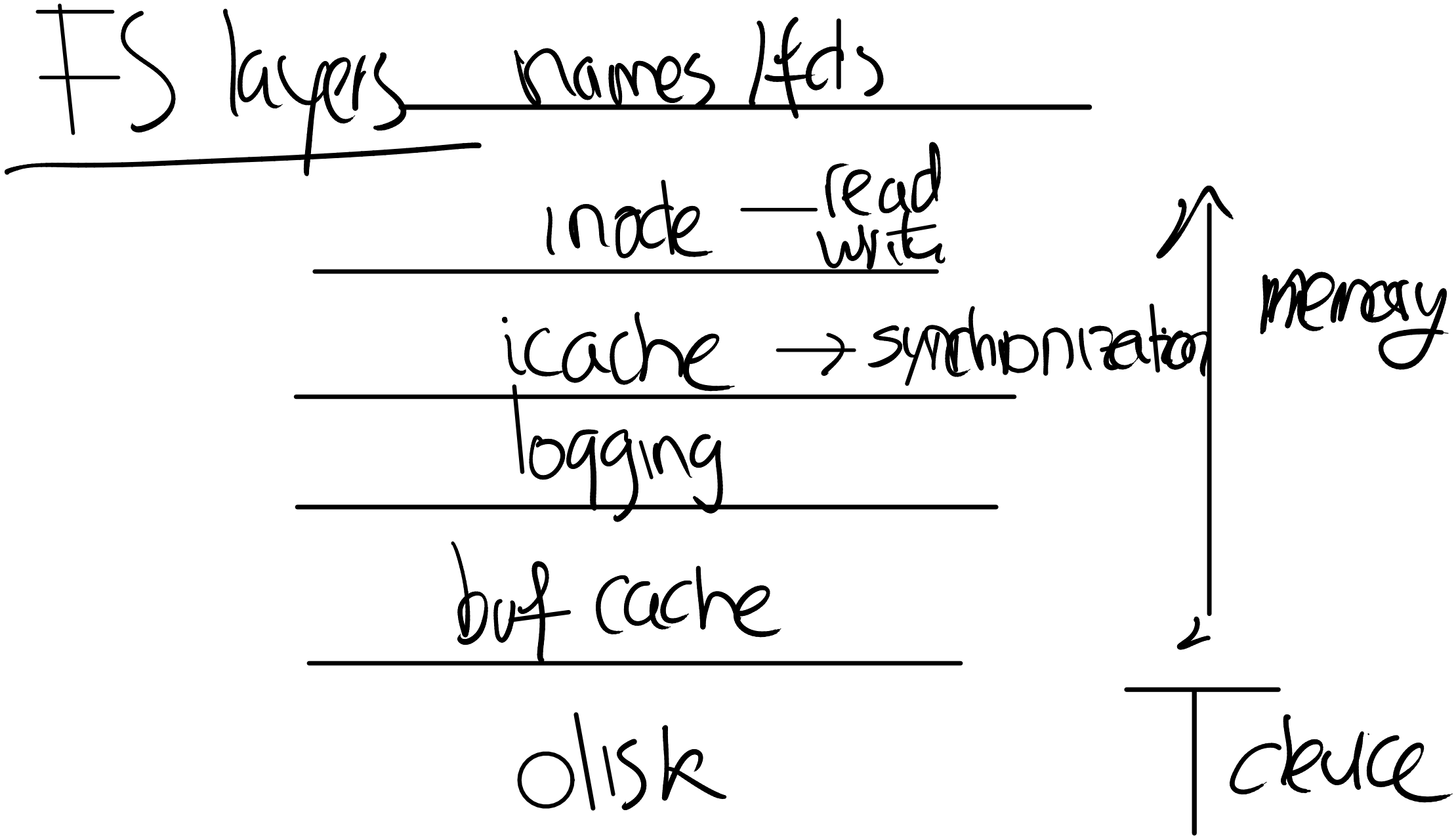
File system structures

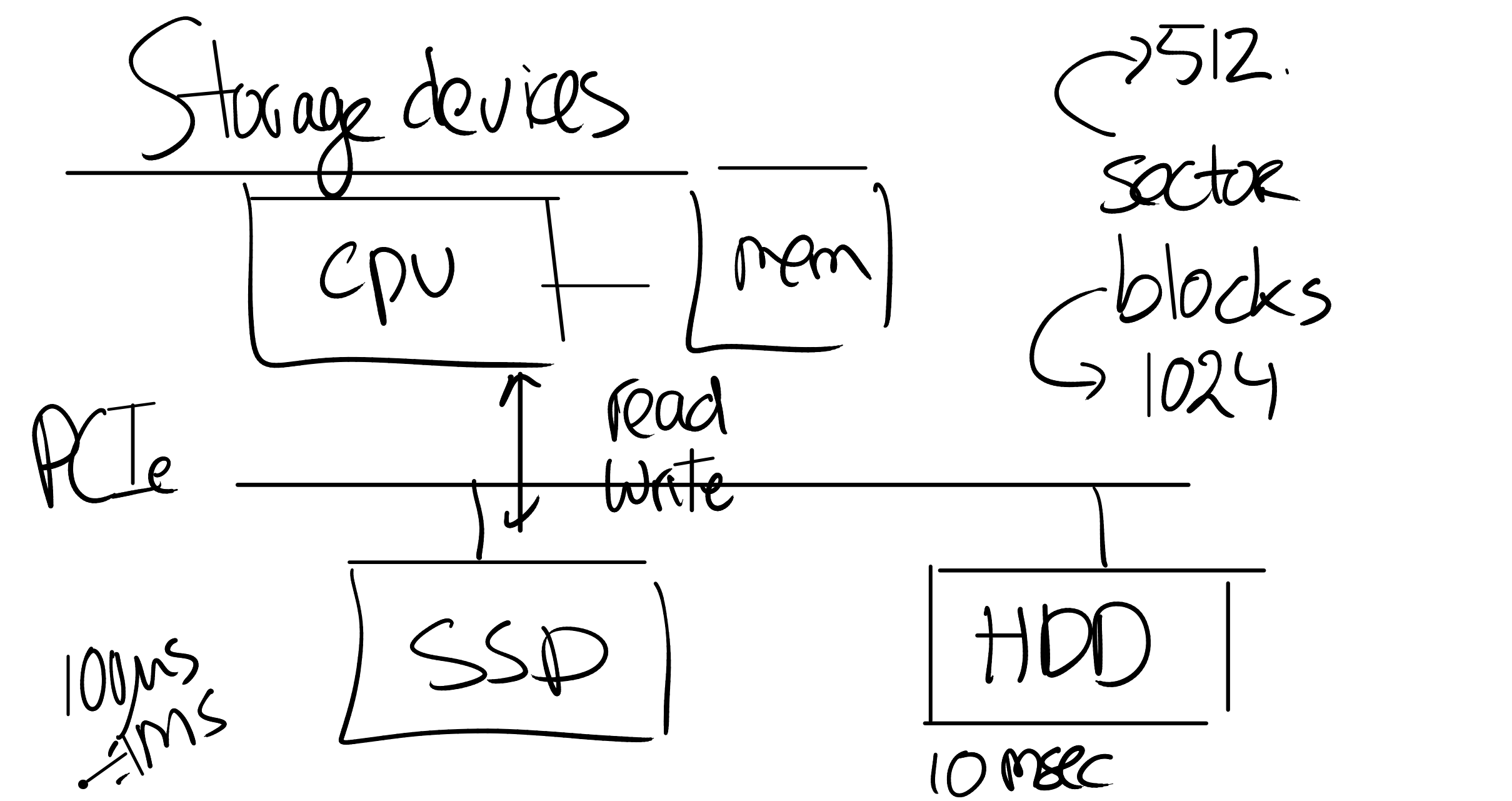
← file info, independent of name

inode #

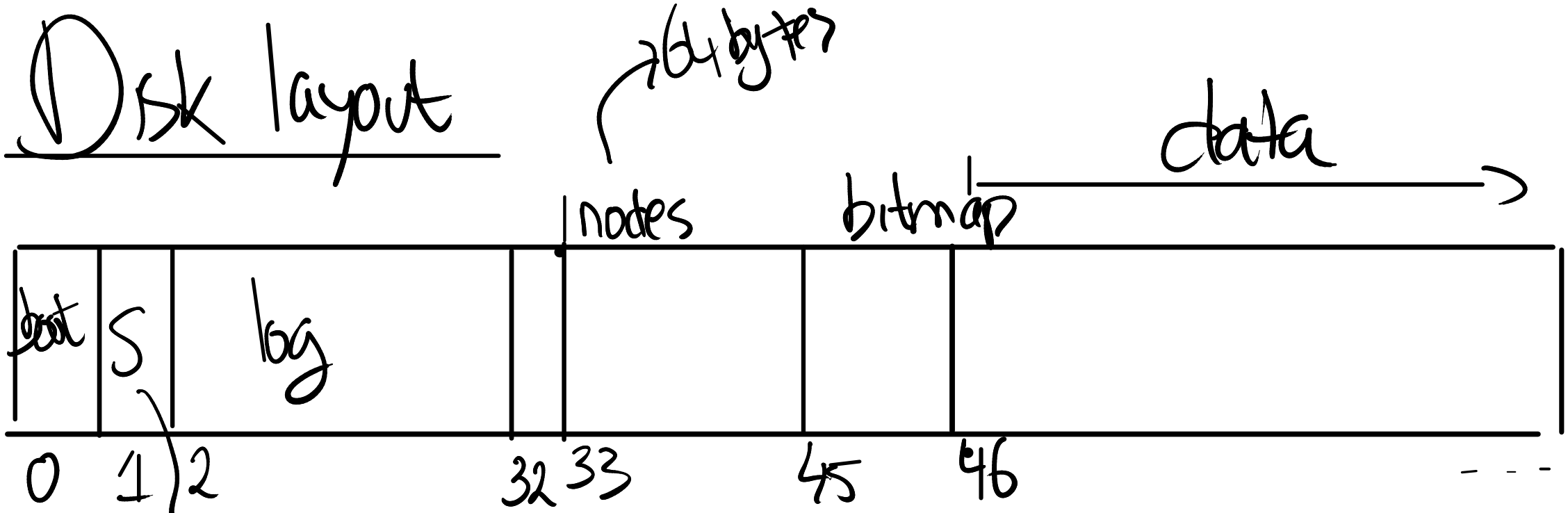
→ link count
open fd count
} both are zero

fd must maintain an offset


FS layers
names / fds
inode
read
write
icache
synchronization
memory
logging
buf cache
disk
Tkuce


Storage devices
cpu
mem
PCIe
read
write
SSD
HDD
100μs
1ms
10 msec
512.
sector
blocks
1024

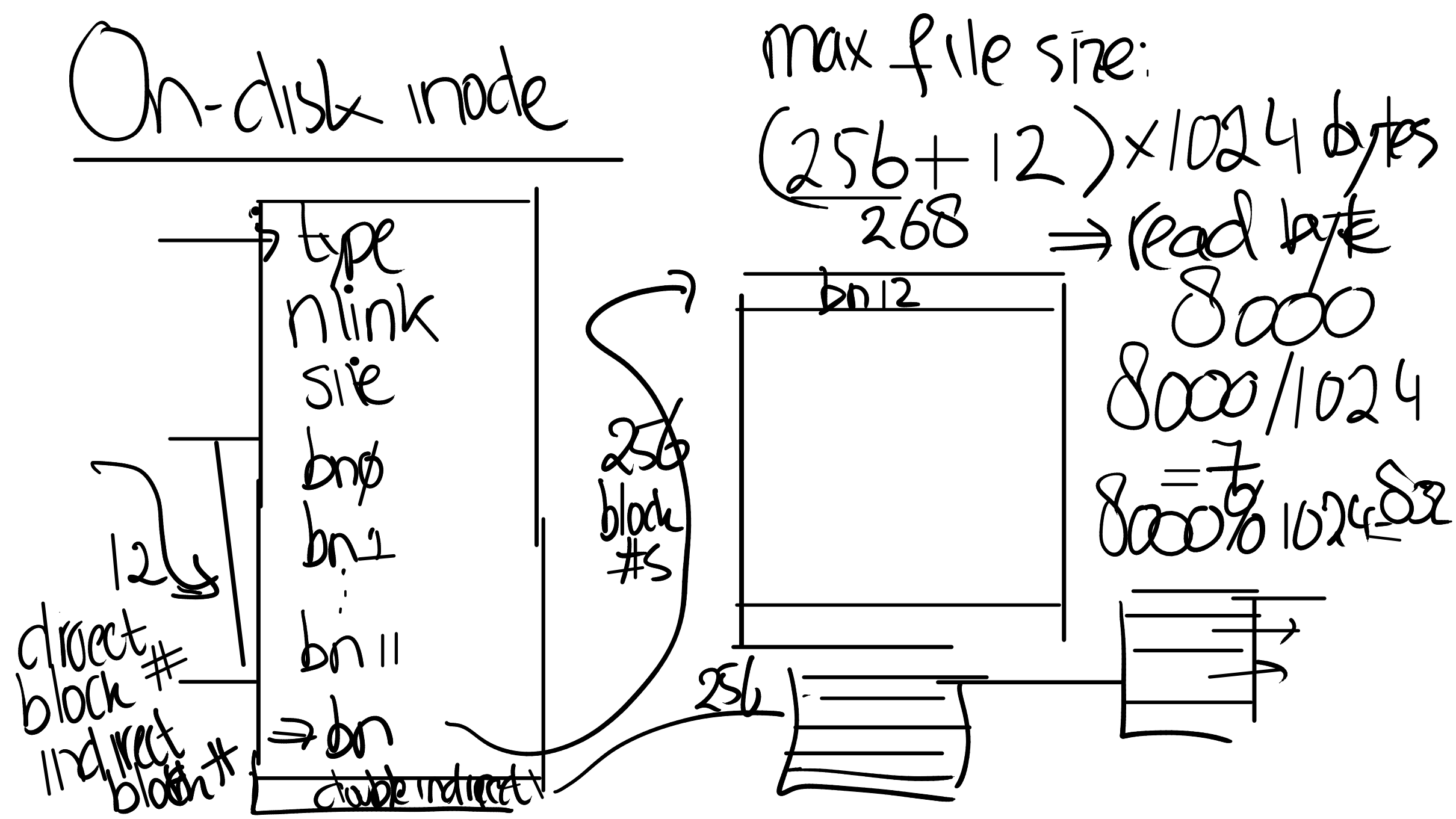
Disk layout



metadata

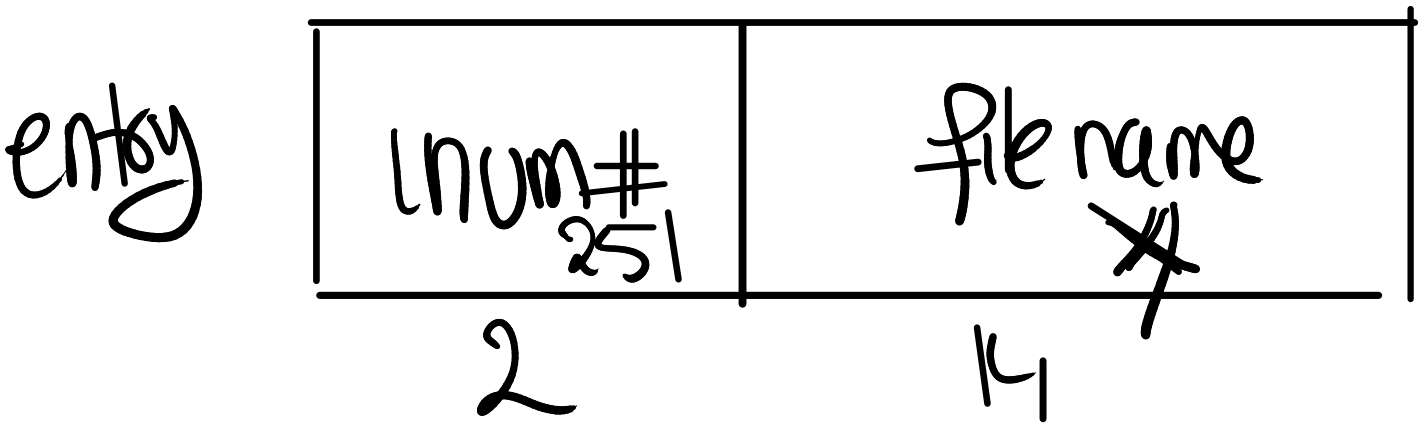
read
inode 10 $\longrightarrow$ $32 + \frac{\text{inode\#} \times 64}{1024}$

On-disk inode



Directories (xv6)

dir = file w. some structure



pathname lookup

/x/x

↳ root inode (7)
scans block for name "x"

↳ read inode 251
scan its block for "x"

Bcache block cache

One copy of block in mem

Sleep locks

LRU

two levels of bdinfo

Summary

fs = on-disk data structure

↳ xv6 very simple one

block cache

Wed: crash safety