Hello !

PERSISTENCE: FILE SYSTEMS

Shivaram Venkataraman

CS 537, Spring 2023

ADMINISTRIVIA

Midterm 2 → early next week

Project 6 → extra slip days

April 14th → Check Canvas

AGENDA / LEARNING OUTCOMES

What are the API to create/modify directories?

How does file system represent files, directories?

What steps must reads/writes take?

RECAP

FILE API WITH FILE DESCRIPTORS

file descriptor

traversal

```
int fd = open(char *path, int flag, mode_t mode)
read(int fd, void *buf, size_t nbyte)
write(int fd, void *buf, size_t nbyte)
close(int fd)
```

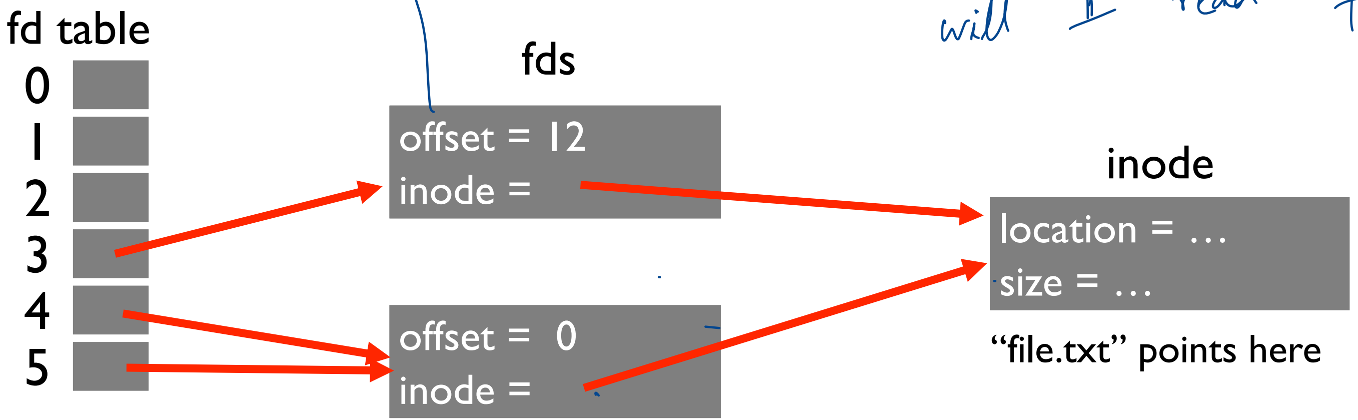
→ done doing reads/writes on this file

advantages:

- string names
- hierarchical
- traverse once
- offsets precisely defined

DUP

read which position in the file will I read from



```
int fd1 = open(“file.txt”); // returns 3
read(fd1, buf, 12);
int fd2 = open(“file.txt”); // returns 4
int fd3 = dup(fd2);         // returns 5
```

automatically increments
offset by 12

COMMUNICATING REQUIREMENTS: FSYNC

File system keeps newly written data in memory for awhile

mem buffers are full → flush

Write buffering improves performance (why?)

↳ memory faster than disk

timer (every 1 min) ↗ flush

But what if system crashes before buffers are flushed?

fsync(int fd) forces buffers to flush to disk, tells disk to flush its write cache

↓ If the disk has some internal cache

Makes data durable

↓

Database →

Editor → Save

DELETING FILES

There is no system call for deleting files!

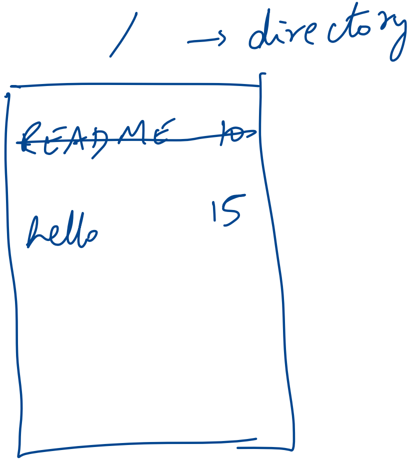
When no directory points to a file it can be garbage collected.

Inode (and associated file) is **garbage collected** when there are no references

Paths are deleted when: `unlink()` is called

FDs are deleted when: `close()` or process quits

removes entry from the directory



/ → directory

~~README~~	~~10~~
hello	15

RENAME

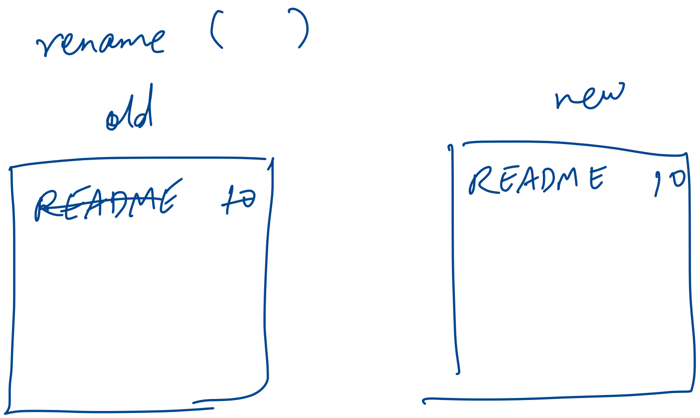
rename(char *old, char *new): → move a file from one path to another

- deletes an old link to a file
- creates a new link to a file

→ very large file 1GB of data

Just changes name of file, does not move data

(Even when renaming to new directory)



→ Property provided

Renames are atomic!

Either file exists in old path or new one

vi → open README

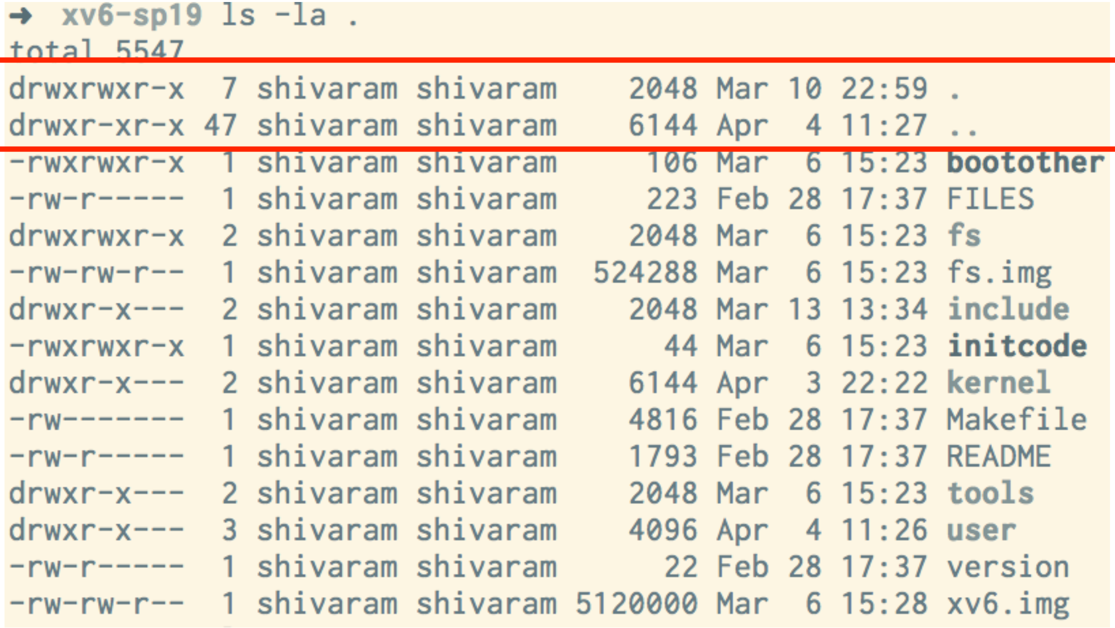
.... edits .... → .README.swp

save → rename .README.swp README

DIRECTORY CALLS

mkdir: create new directory

readdir: read/parse directory entries → ls → read the xv6 implementation



```
➜ xv6-sp19 ls -la .
total 5547
drwxrwxr-x  7 shivaram shivaram    2048 Mar 10 22:59 .
drwxr-xr-x 47 shivaram shivaram    6144 Apr  4 11:27 ..
-rwxrwxr-x  1 shivaram shivaram     106 Mar  6 15:23 bootother
-rw-r-----  1 shivaram shivaram     223 Feb 28 17:37 FILES
drwxrwxr-x  2 shivaram shivaram    2048 Mar  6 15:23 fs
-rw-rw-r--  1 shivaram shivaram  524288 Mar  6 15:23 fs.img
drwxr-x---  2 shivaram shivaram    2048 Mar 13 13:34 include
-rwxrwxr-x  1 shivaram shivaram      44 Mar  6 15:23 initcode
drwxr-x---  2 shivaram shivaram    6144 Apr  3 22:22 kernel
-rw-------  1 shivaram shivaram    4816 Feb 28 17:37 Makefile
-rw-r-----  1 shivaram shivaram    1793 Feb 28 17:37 README
drwxr-x---  2 shivaram shivaram    2048 Mar  6 15:23 tools
drwxr-x---  3 shivaram shivaram    4096 Apr  4 11:26 user
-rw-r-----  1 shivaram shivaram      22 Feb 28 17:37 version
-rw-rw-r--  1 shivaram shivaram 5120000 Mar  6 15:28 xv6.img
```

this dir

parent directory

LINKS

Hard links: Both path names use same inode number

File does not disappear until all hard links removed; cannot link directories

```
echo “Beginning…” > file1
ln file1 link
cat link
# “Beginning…”

ls -li
# 18 -rw-rw-r-- 2 shivaram shivaram 10 Apr  6 21:32 file1
# 18 -rw-rw-r-- 2 shivaram shivaram 10 Apr  6 21:32 link
```

→ creates a hard link named "link"

unlink file1

~~file1 18~~

→ link 18

SOFT LINKS

Soft or symbolic links: Point to second path name; can softlink to dirs

```
ln -s oldfile softlink
```

src

↳ new linked file

Soft link

Go to ./old file

any application that opens file will be "redirected"

Confusing behavior: "file does not exist"!

Confusing behavior: "cd linked_dir; cd ..; in different parent!

If i delete oldfile then the link is broken

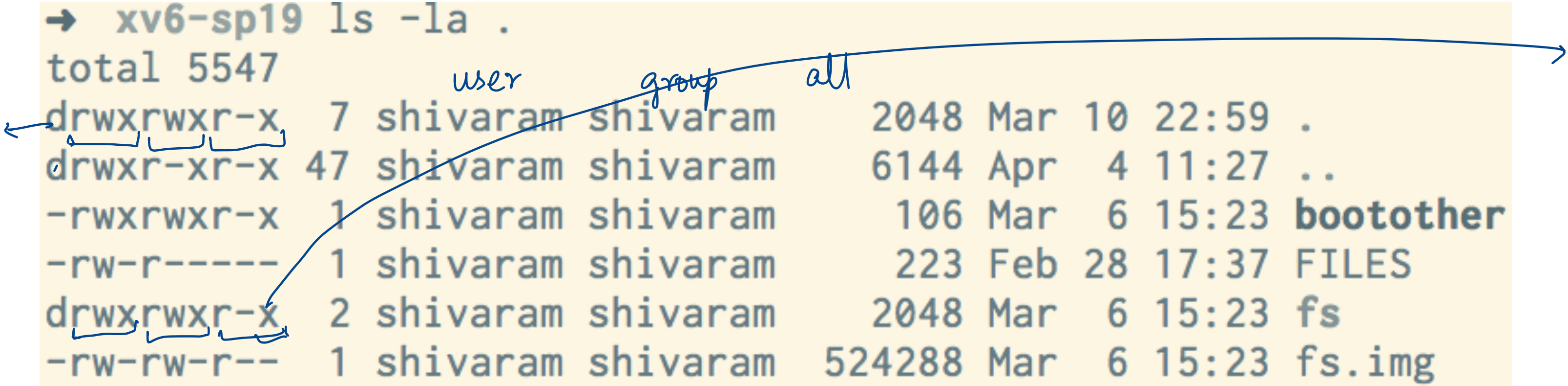
ls output = python → python 3.5

ls /usr/lib/python

↓

python 3.5

PERMISSIONS, ACCESS CONTROL



```
➜  xv6-sp19 ls -la .
total 5547
drwxrwxr-x  7 shivaram shivaram    2048 Mar 10 22:59 .
drwxr-xr-x 47 shivaram shivaram    6144 Apr  4 11:27 ..
-rwxrwxr-x  1 shivaram shivaram     106 Mar  6 15:23 bootother
-rw-r-----  1 shivaram shivaram     223 Feb 28 17:37 FILES
drwxrwxr-x  2 shivaram shivaram    2048 Mar  6 15:23 fs
-rw-rw-r--  1 shivaram shivaram  524288 Mar  6 15:23 fs.img
```

user group all

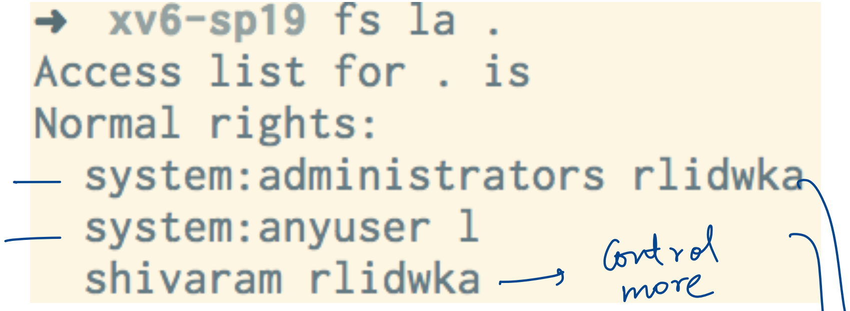
need x bit to access a directory

d → if its a directory

user group all

rwx

read write execute



```
➜  xv6-sp19 fs la .
Access list for . is
Normal rights:
  system:administrators rlidwka
  system:anyuser l
  shivaram rlidwka
```

Control more

multiple groups

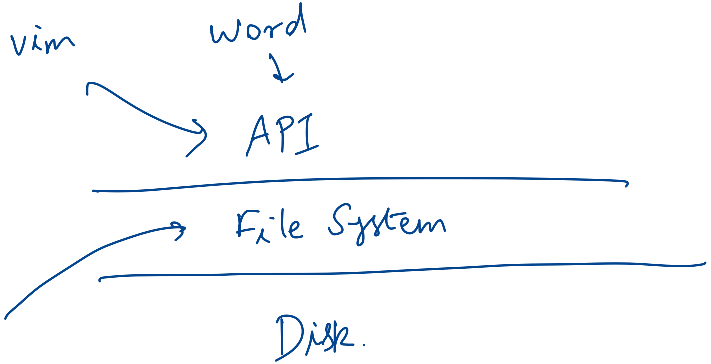
→ AFS

FILE API SUMMARY

Using multiple types of name provides convenience and efficiency

Hard and soft link features provide flexibility.

Special calls (fsync, rename) let developers communicate requirements to file system

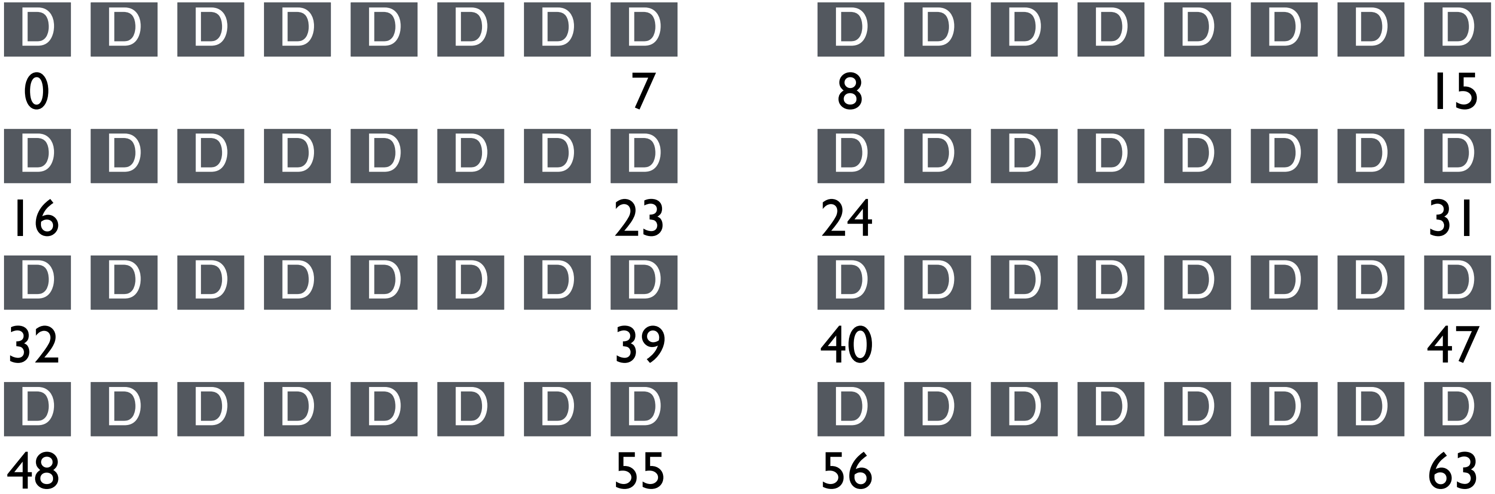


FILESYSTEM DISK STRUCTURES

FS STRUCTS: EMPTY DISK

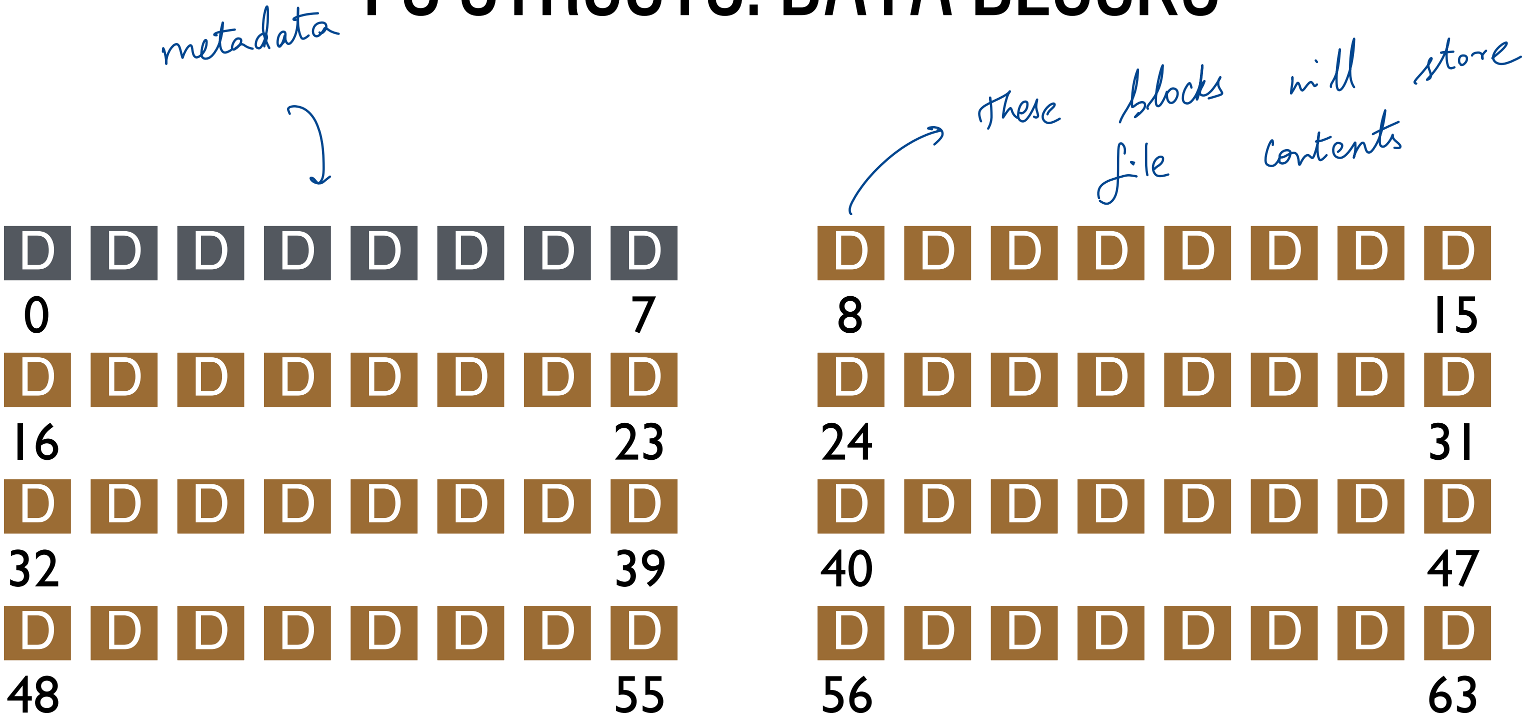
Disk

= 256 KB



Assume each block is 4KB

FS STRUCTS: DATA BLOCKS



metadata

these blocks will store file contents

D D D D D D D D
0 7

D D D D D D D D
8 15

D D D D D D D D
16 23

D D D D D D D D
24 31

D D D D D D D D
32 39

D D D D D D D D
40 47

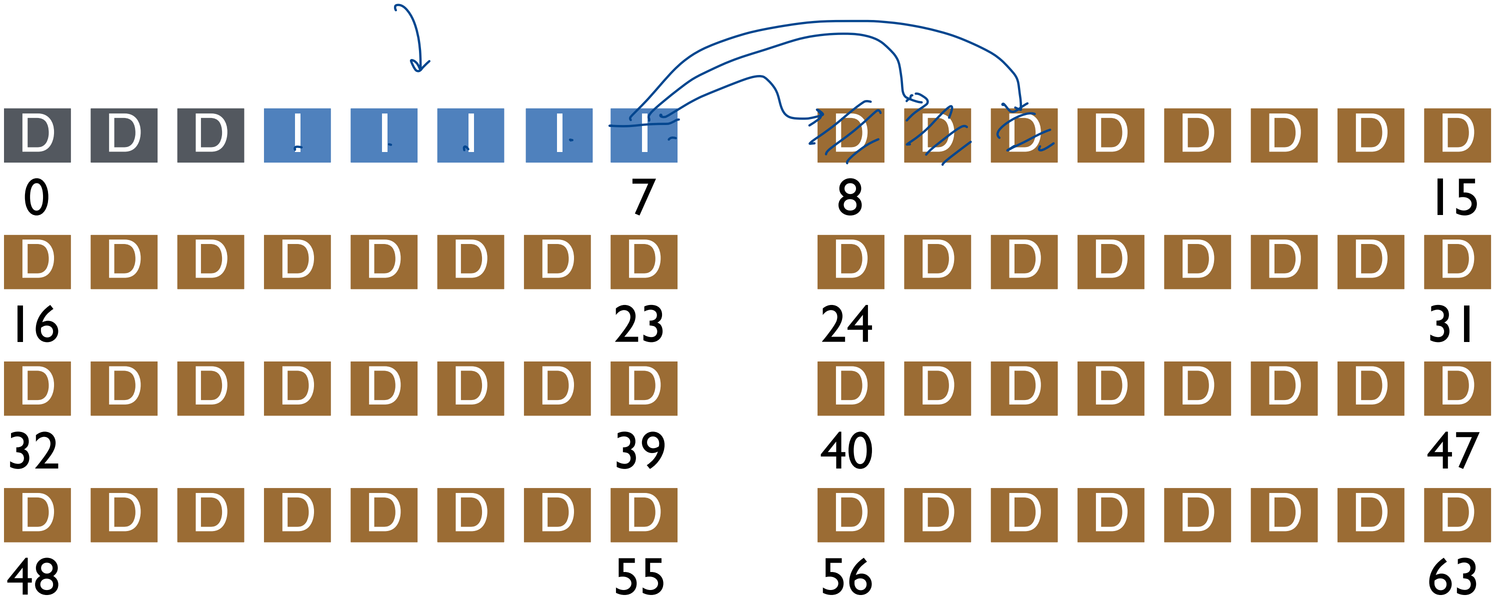
D D D D D D D D
48 55

D D D D D D D D
56 63

Simple layout → Very Simple File System

INODE POINTERS

these blocks
store inodes

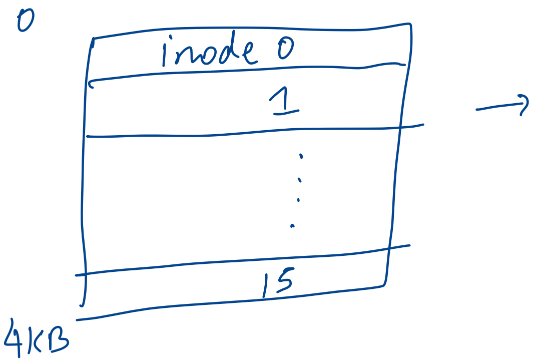


ONE INODE BLOCK

Each inode is typically 256 bytes (depends on the FS, maybe 128 bytes)

4KB disk block

16 inodes per inode block.



16 × 5 = 80 inodes with 5 inode blocks

inode 16	inode 17	inode 18	inode 19
inode 20	inode 21	inode 22	inode 23
inode 24	inode 25	inode 26	inode 27
inode 28	inode 29	inode 30	inode 31

INODE

type (file or dir?) →

uid (owner) → who owns

rwx (permissions)

size (in bytes)

Blocks

time (access) → timestamps

ctime (create) →

links_count (# paths) → hard links

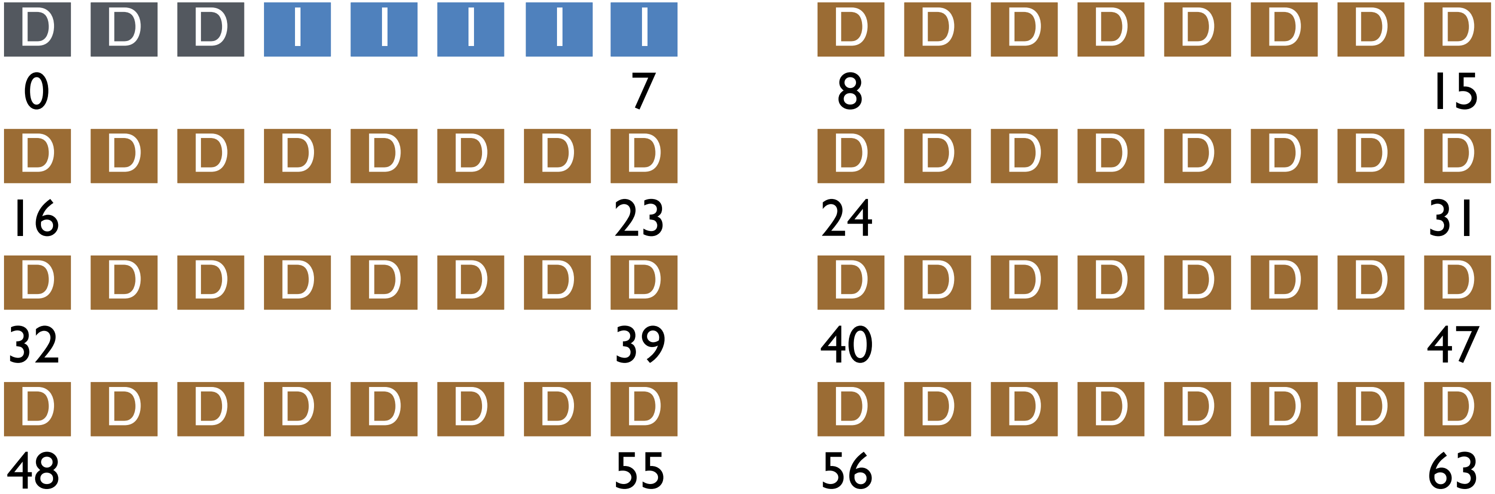
addrs[N] (N data blocks)

Inode also contains pointers to the data blocks

addrs 8, 9, 10

addrs 8, 24, 37

FS STRUCTS: INODE DATA POINTERS



D D D I I I I I

0 7

D D D D D D D D

8 15

D D D D D D D D

16 23

D D D D D D D D

24 31

D D D D D D D D

32 39

D D D D D D D D

40 47

D D D D D D D D

48 55

D D D D D D D D

56 63

INODE

type (file or dir?)
uid (owner)
rwx (permissions)
size (in bytes)
Blocks
time (access)
ctime (create)
links_count (# paths)
addrs[N] (N data blocks)

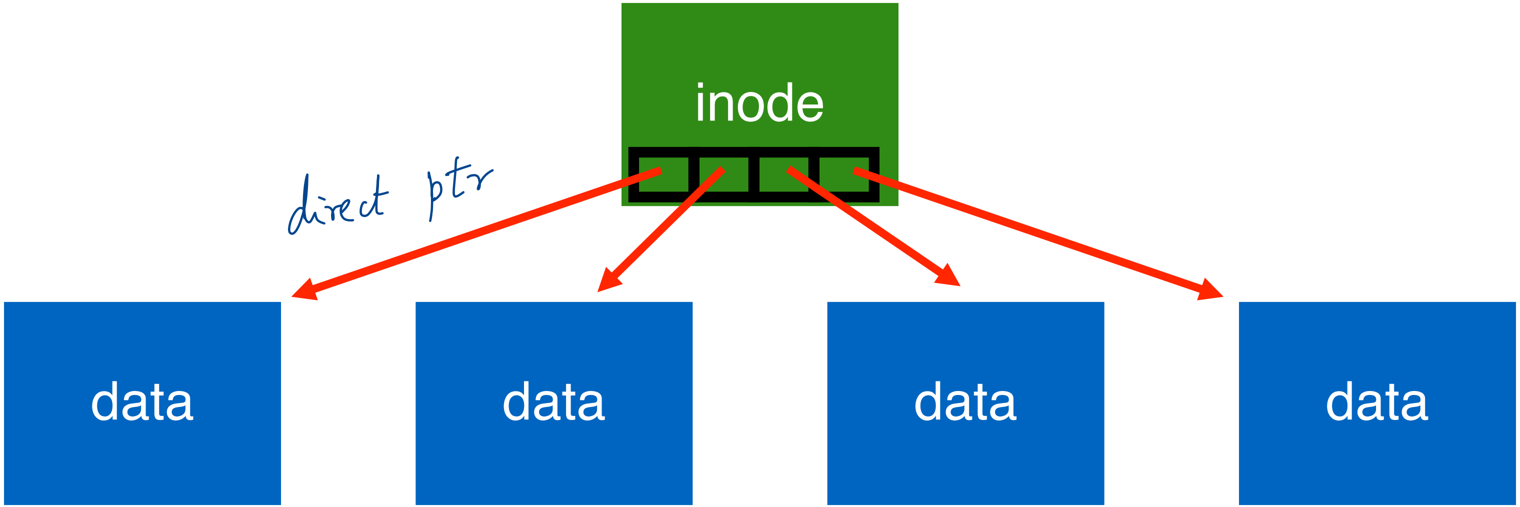
Assume single level (just pointers to data blocks)

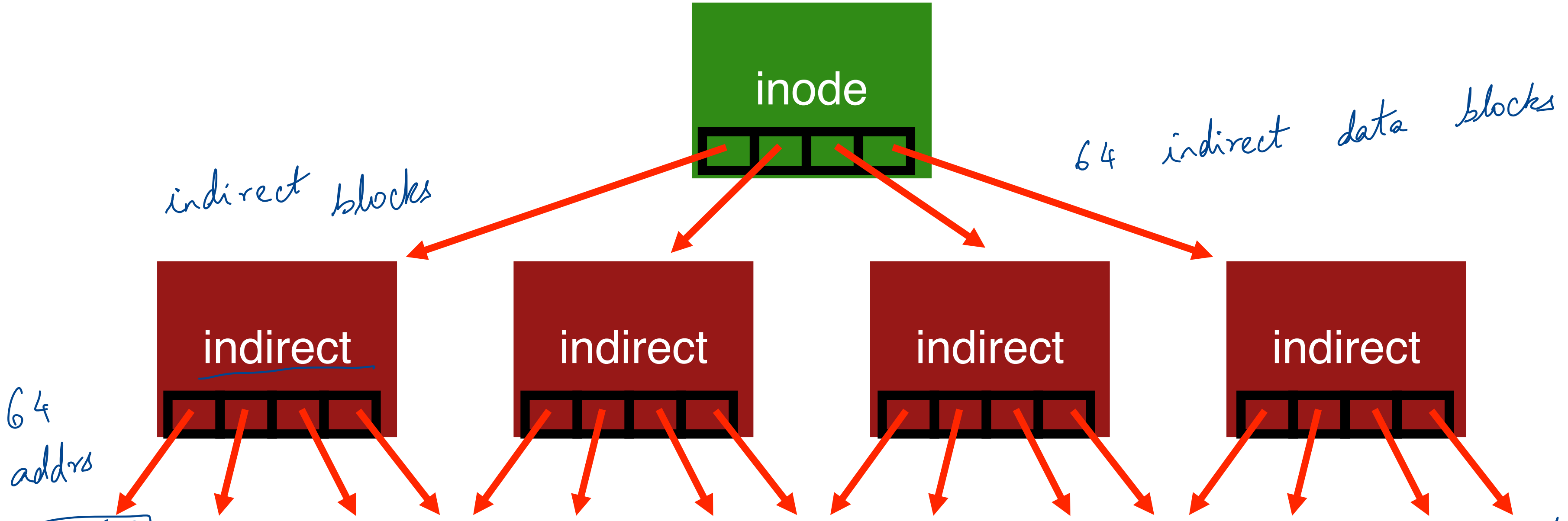
What is max file size?

Assume 256-byte inodes
(all can be used for pointers)
Assume 4-byte addrs → block address
block size = 4KB

How to get larger files?

Each 4 byte is an addr
→ 64 pointers x 4 KB = 256 KB


inode
direct ptr
data
data
data
data



4MB

Indirect blocks are stored in regular data blocks

→ Space indirect blocks

Largest file size with 64 indirect blocks?

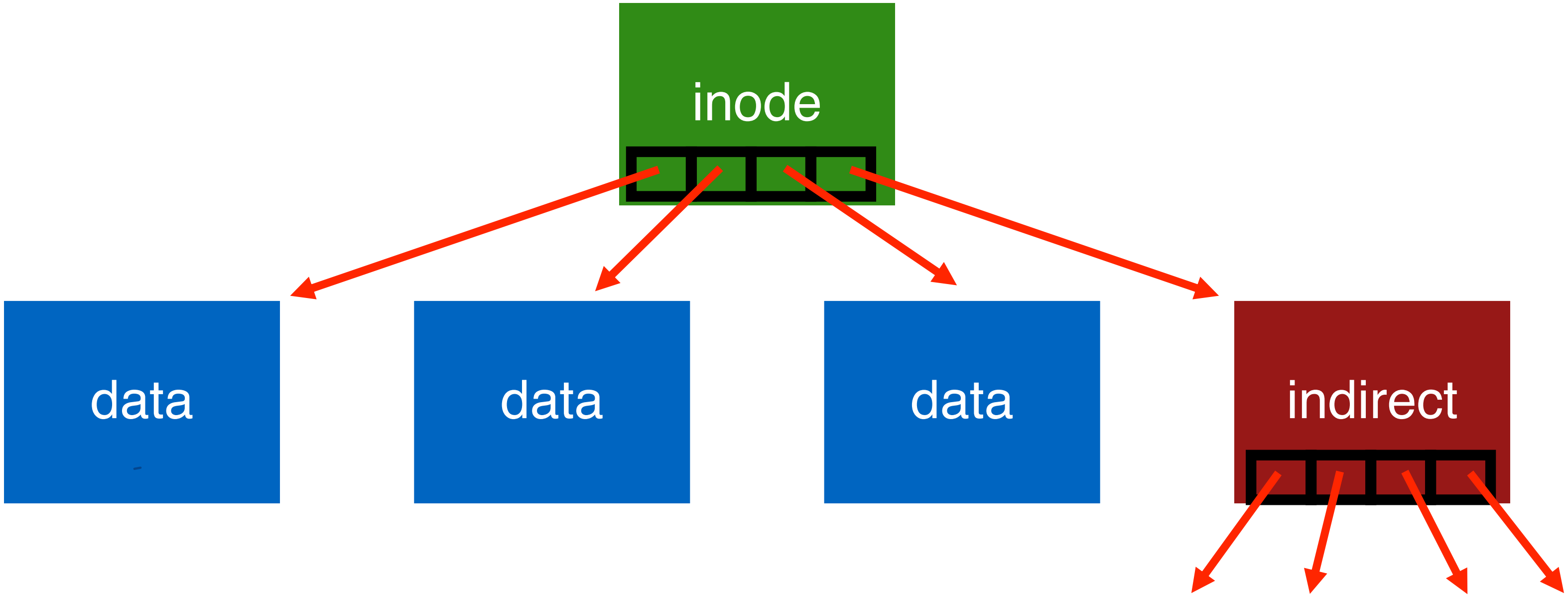
Any Cons? → multiple hops to get data → small files

Each indirect block = 4KB

each add = 4 bytes

⇒ Each indirect block = 1024 data blocks × 4KB = 4MB × 64

= 256 MB



Better for small files!
How to handle even larger files?

↳ double indirect

triple indirect

} more levels in tree

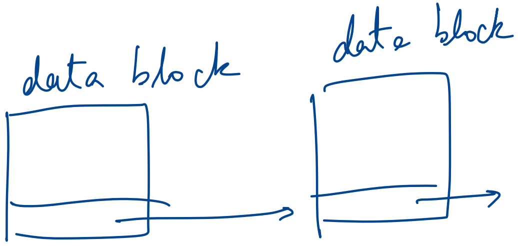
for small files → only access the data blocks directly

OTHER APPROACHES

ext 3 or ext 4

Extent-based

Linked lists (File-allocation Tables) → FAT 32



Multi-level Indexed

Questions → Trade-offs

- Amount of fragmentation (internal and external)
- Ability to grow file over time?
- Performance of sequential accesses (contiguous layout)?
- Speed to find data blocks for random accesses? → linked lists might be bad
- Wasted space for meta-data overhead (everything that isn't data)?

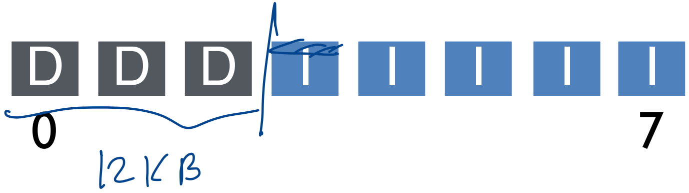
  Meta-data must be stored persistently too!

QUIZ 25

https://tinyurl.com/cs537-sp23-quiz25

Assume 256 byte inodes (16 inodes/block), block size = 4KB.
What is the offset for inode with number 0?

12 KB



What is the offset for inode with number 4?

(0, 1, 2, 3)

12 KB + 4 × 256

= 13 KB

What is the offset for inode with number 40?

12 KB + 40 × 256

12 + 10 = 22 KB

DIRECTORIES

mkdir test

File systems vary

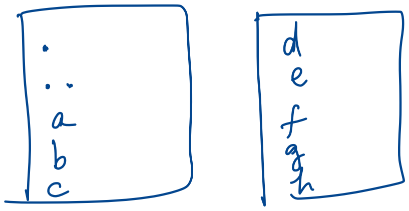
Common design:

Store directory entries in data blocks

Large directories just use multiple data blocks

Use bit in inode to distinguish directories from files

inode for test (bit for dir set)



Various formats could be used

- lists
- b-trees

Within a block how do I represent contents

SIMPLE DIRECTORY LIST EXAMPLE

Created something called foo → inode 80

valid	name	inode
1	.	134
1	..	35
~~1~~ 0	foo//	80
1	bar	23

} all directories have . & ..

unlink("foo")

ALLOCATION

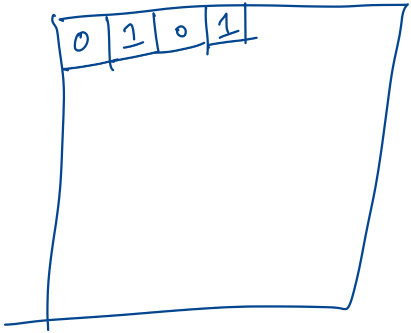
How do we find free data blocks or free inodes? → Meta data about FS

Which data blocks are allocated and which are not?

Free list → list of data blocks unused

Bitmaps

Bitmaps



0	1	0	1

0 → block is free

1 → block is used

Tradeoffs in next lecture…

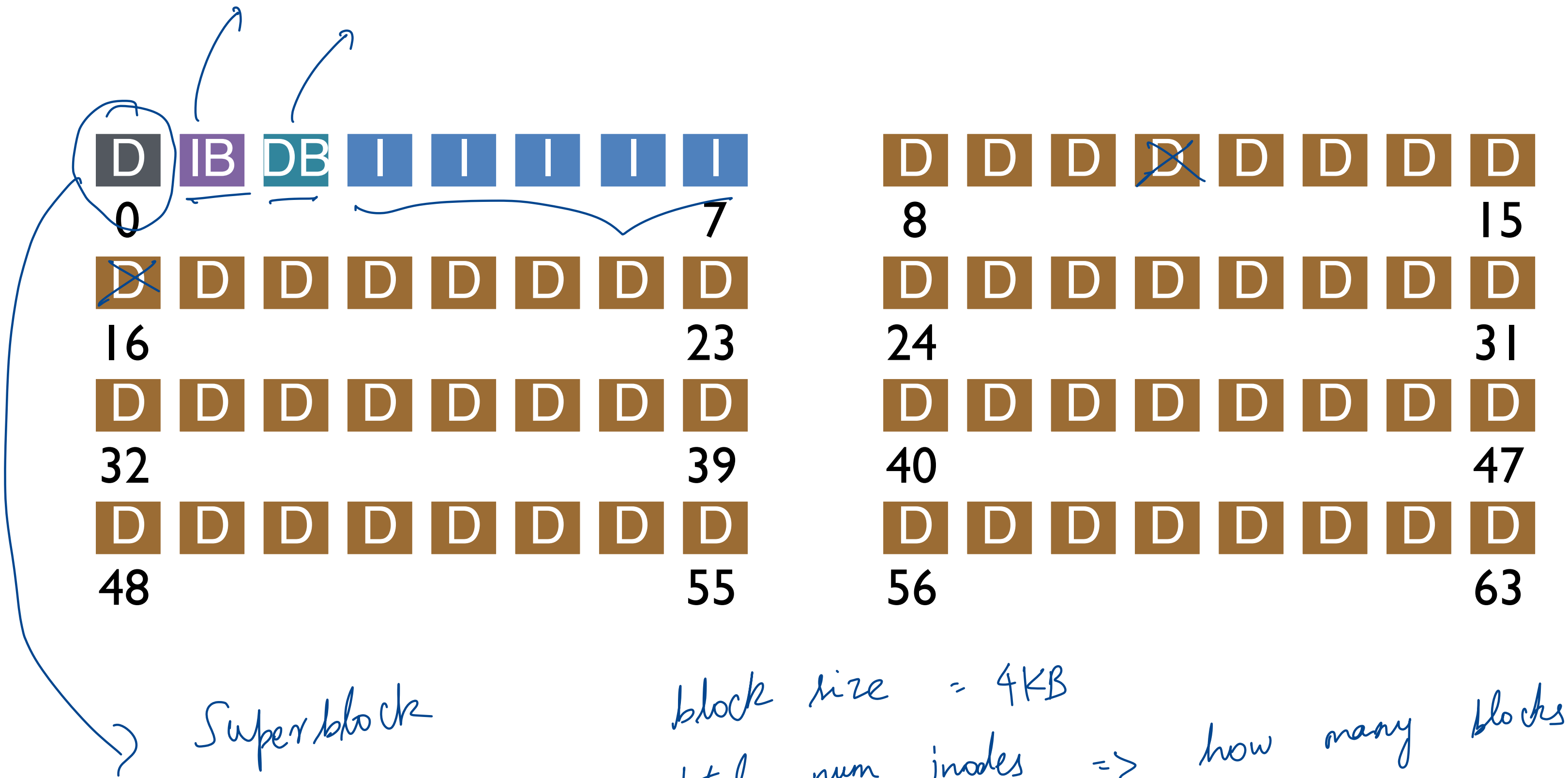
4096 blocks in my FS

4096 bits = 512 bytes

FS STRUCTS: BITMAPS

inode bitmap

data bitmap



block size = 4KB

total num inodes => how many blocks

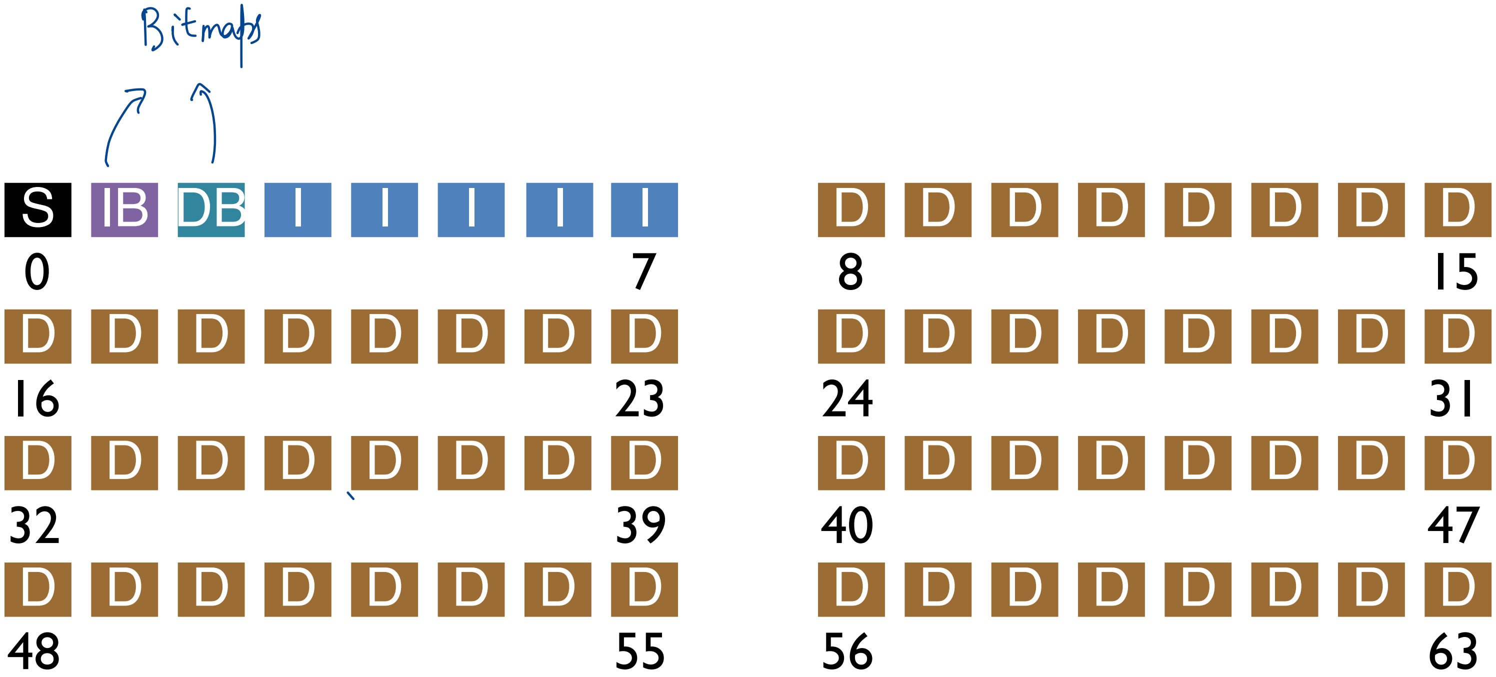
SUPERBLOCK

Need to know basic FS configuration metadata, like:

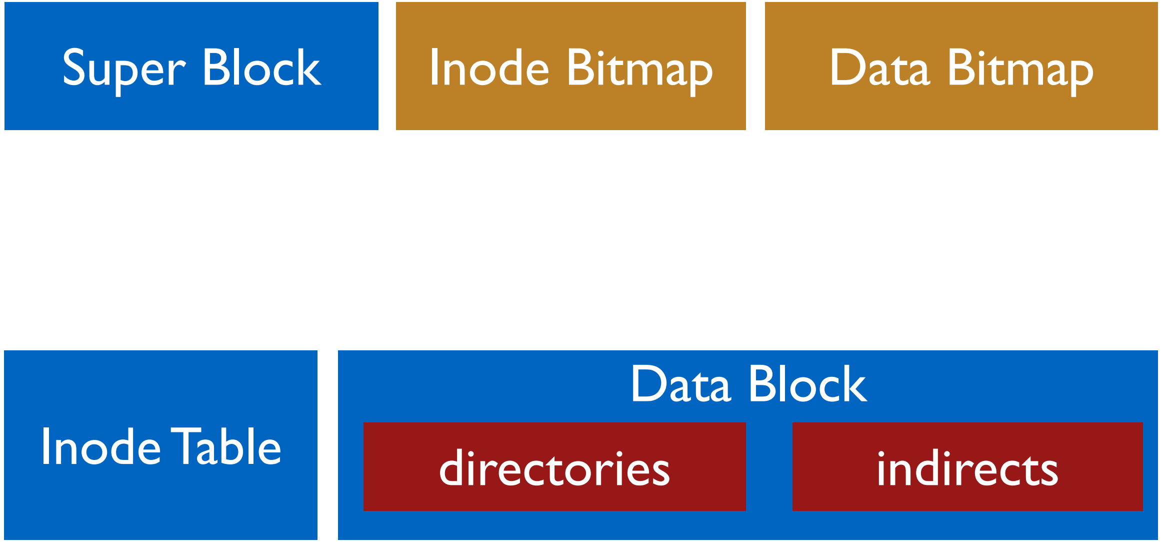
- block size
- # of inodes

Store this in superblock

FS STRUCTS: SUPERBLOCK



SUMMARY



Super Block

Inode Bitmap

Data Bitmap

Inode Table

Data Block

directories

indirects

NEXT STEPS

Next class: Filesystem operations, FFS!