Operating Systems

Lecture 32: Filesystems Implementation

Nipun Batra
Nov 18, 2018

File Names

Three types of names:
- inode number
- path
- file descriptor

Why?

File Names

inode

- unique name
- remember file size, permissions, etc

path

- easy to remember
- hierarchical

file descriptor

- avoid frequent traversal
- remember multiple offsets

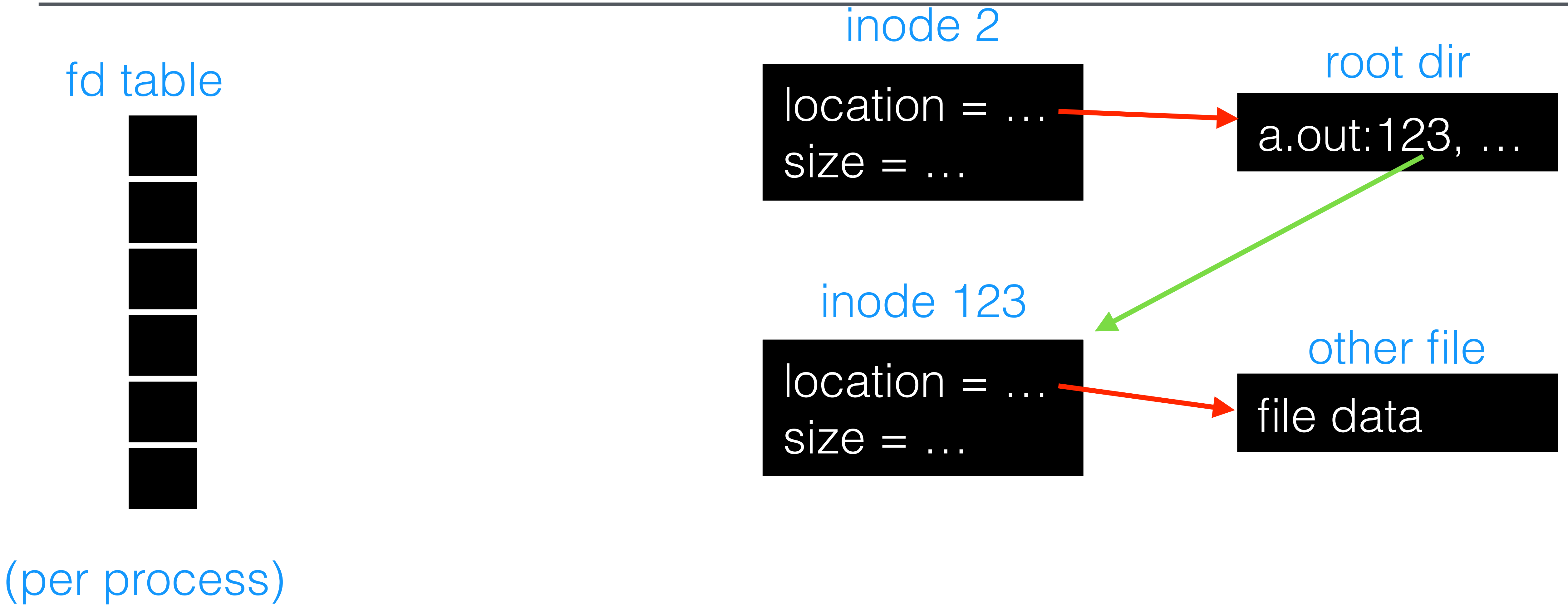
File API

```
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)
```

Special Calls

```
fsync(int fd)

rename(char *oldpath, char *newpath)
```

Inodes, Paths, FDs



fd table

(per process)

inode 2

location = …
size = …

root dir

a.out:123, …

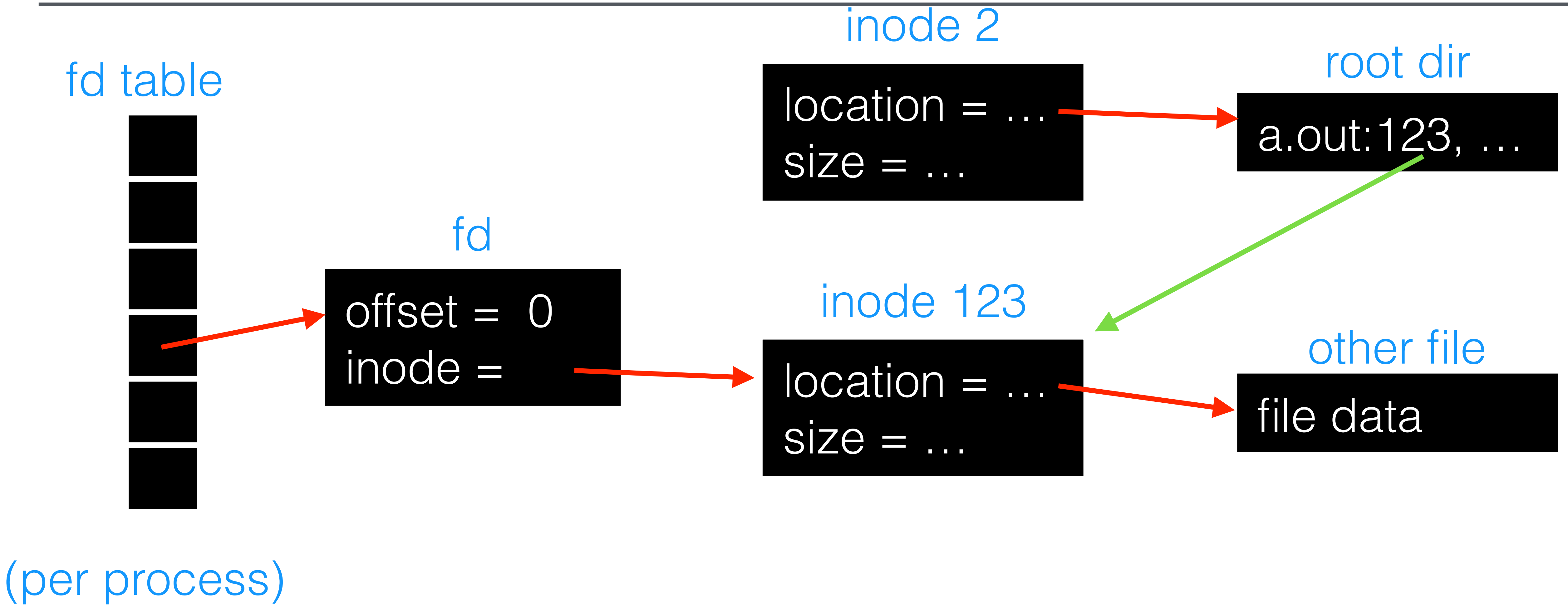
inode 123

location = …
size = …

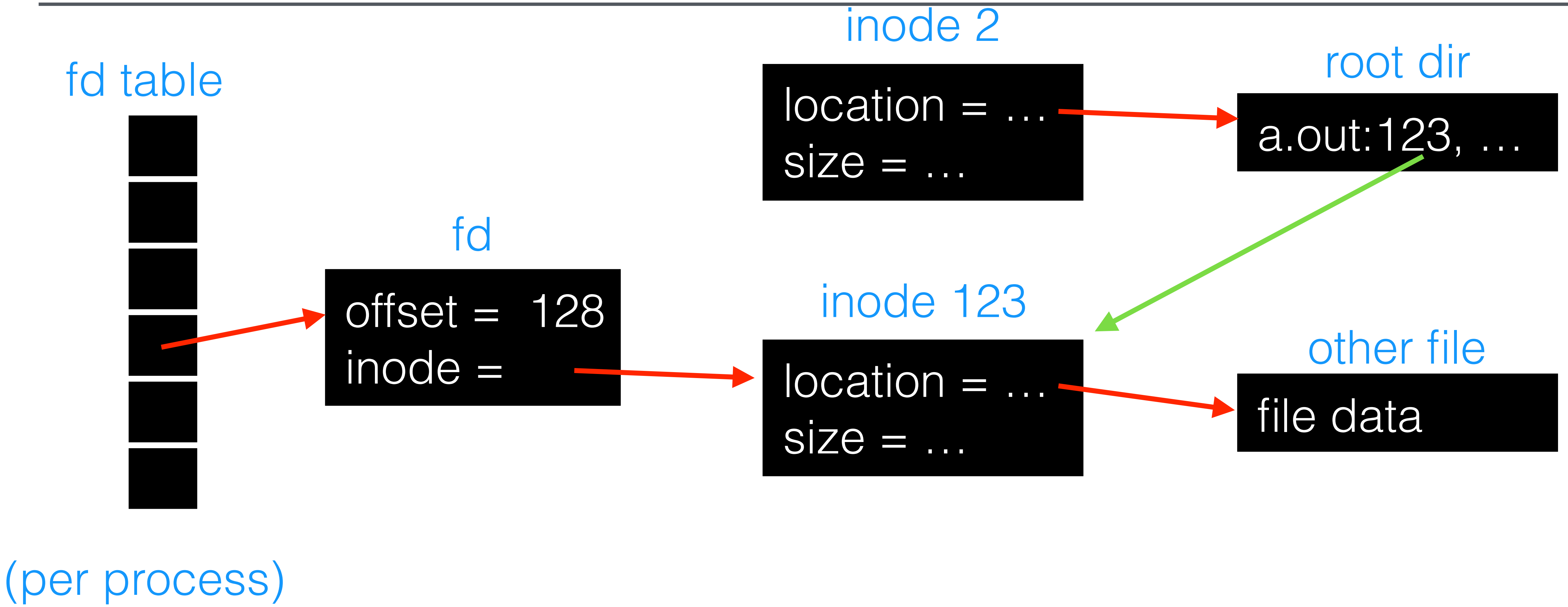
other file

file data

Inodes, Paths, FDs



Inodes, Paths, FDs



fd table

(per process)

fd

offset = 128
inode =

inode 2

location = …
size = …

root dir

a.out:123, …

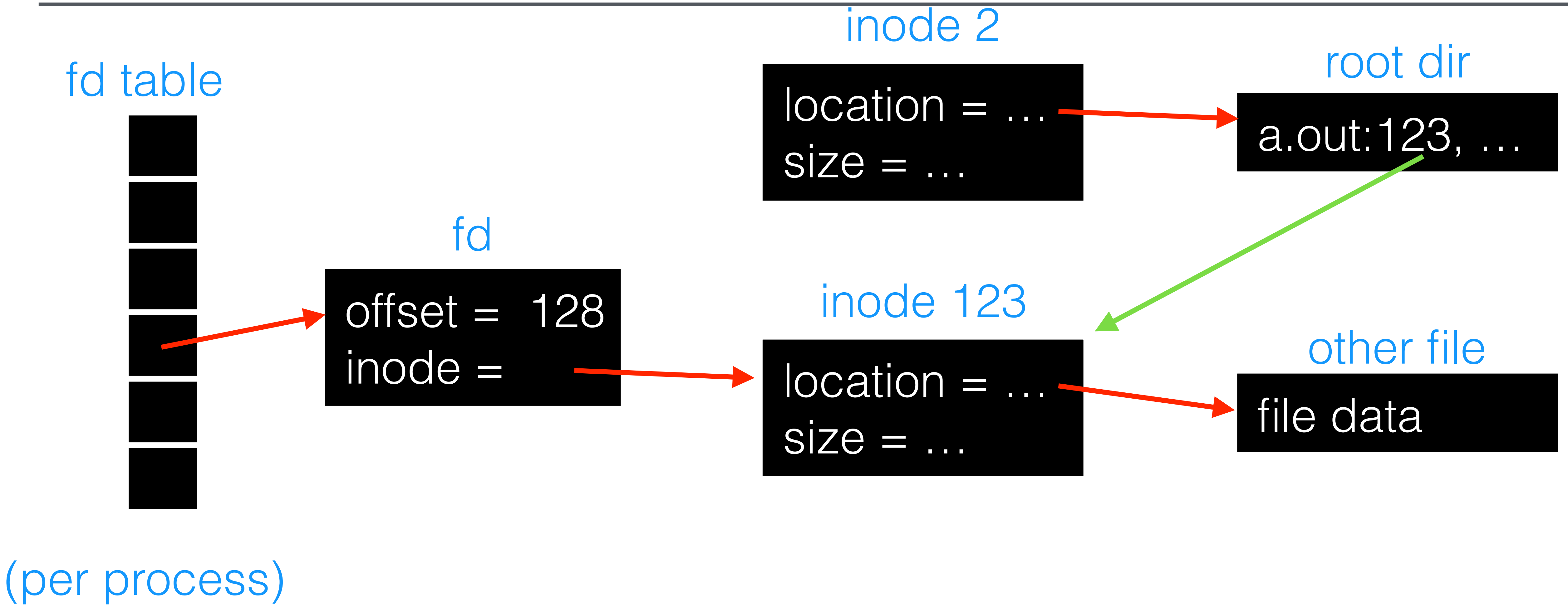
inode 123

location = …
size = …

other file

file data

Inodes, Paths, FDs



opened /a.out, read 128 bytes

Implementation

Implementation

1. On-disk structures

Implementation

1. On-disk structures
   - how do we represent files, directories?

Implementation

1. On-disk structures
    - how do we represent files, directories?

Implementation

1. On-disk structures
   - how do we represent files, directories?

2. Access methods

Implementation

1. On-disk structures
   - how do we represent files, directories?

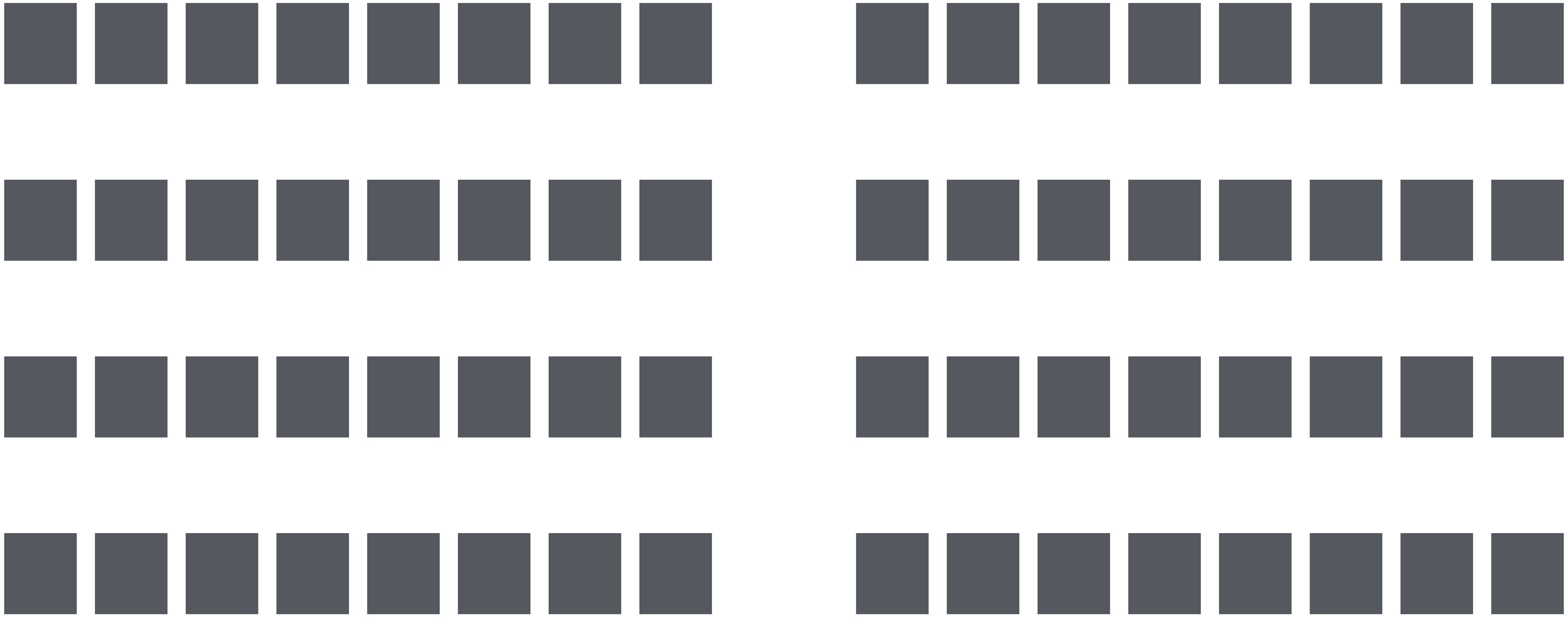
2. Access methods
   - what steps must reads/writes take?

Structures

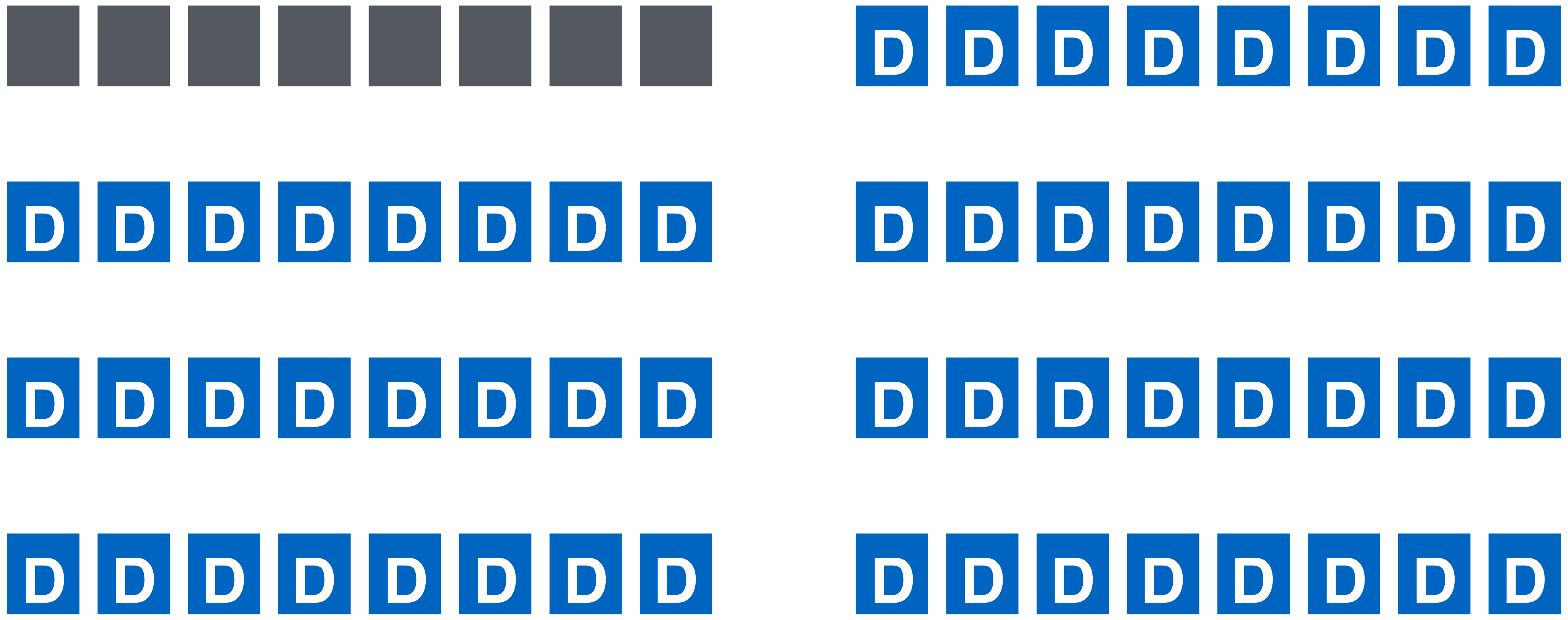
Common file-system structures

- data block
- inode table
- indirect block
- directories
- data bitmap
- inode bitmap
- superblock

FS Structs: Empty Disk



Data Blocks


D D D D D D D D
D D D D D D D D D D D D D D D D
D D D D D D D D D D D D D D D D
D D D D D D D D D D D D D D D D

Structures

Common file-system structures

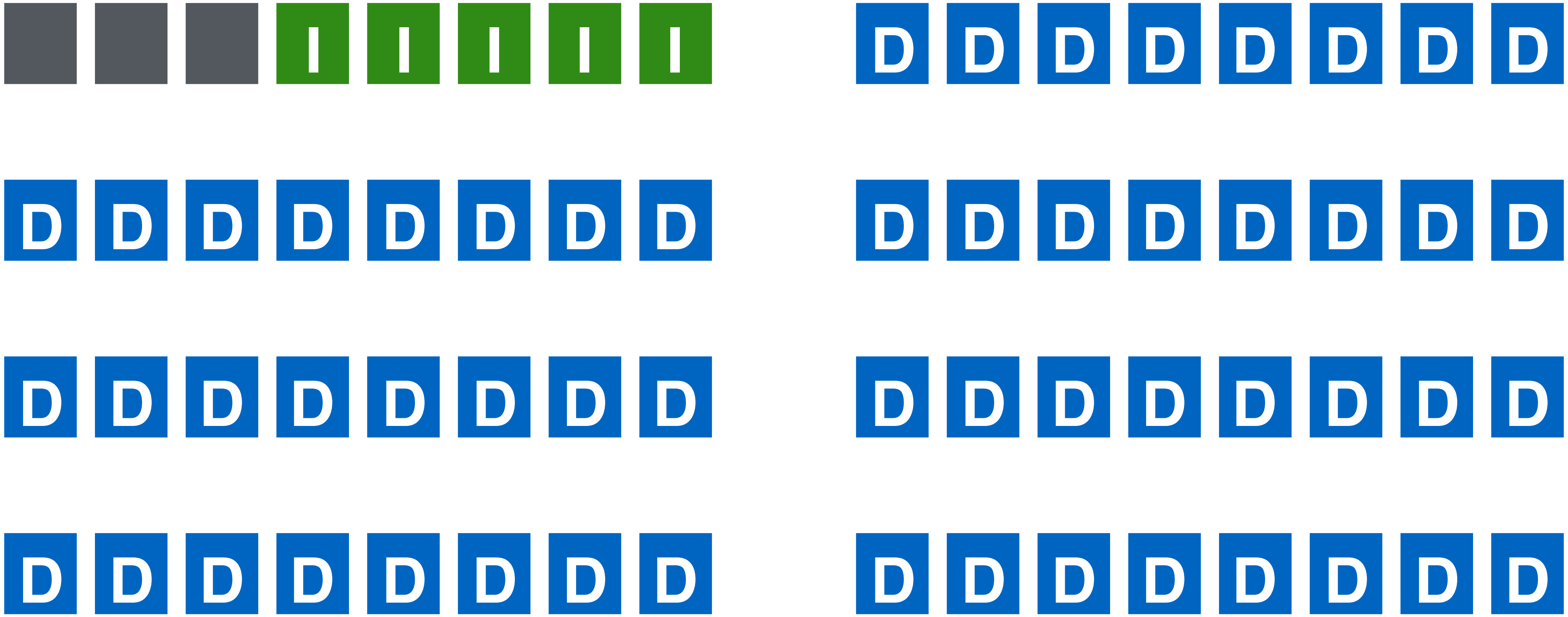
- data block
- inode table
- indirect block
- directories
- data bitmap
- inode bitmap
- superblock

Structures

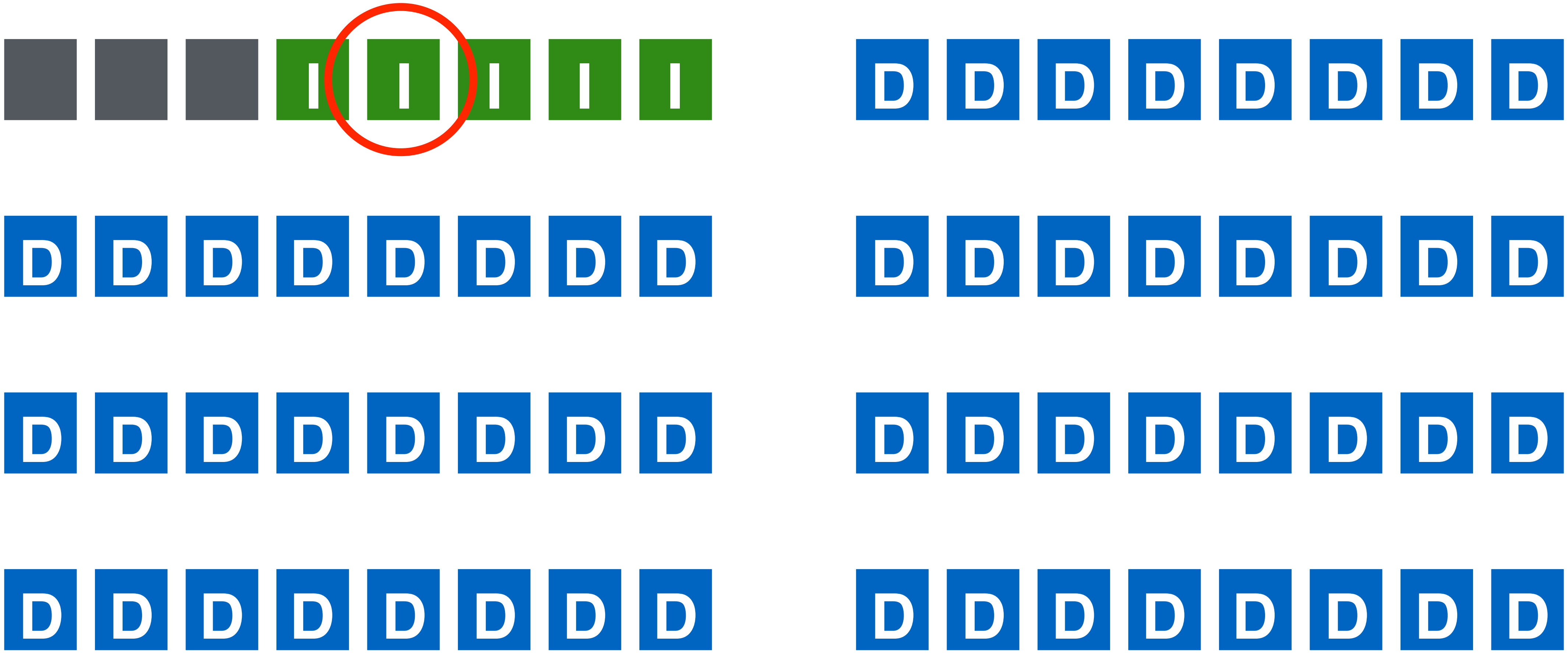
Common file-system structures

- data block
- inode table
- indirect block
- directories
- data bitmap
- inode bitmap
- superblock

Inodes


I I I I I
D D D D D D D D
D D D D D D D D
D D D D D D D D
D D D D D D D D
D D D D D D D D
D D D D D D D D
D D D D D D D D

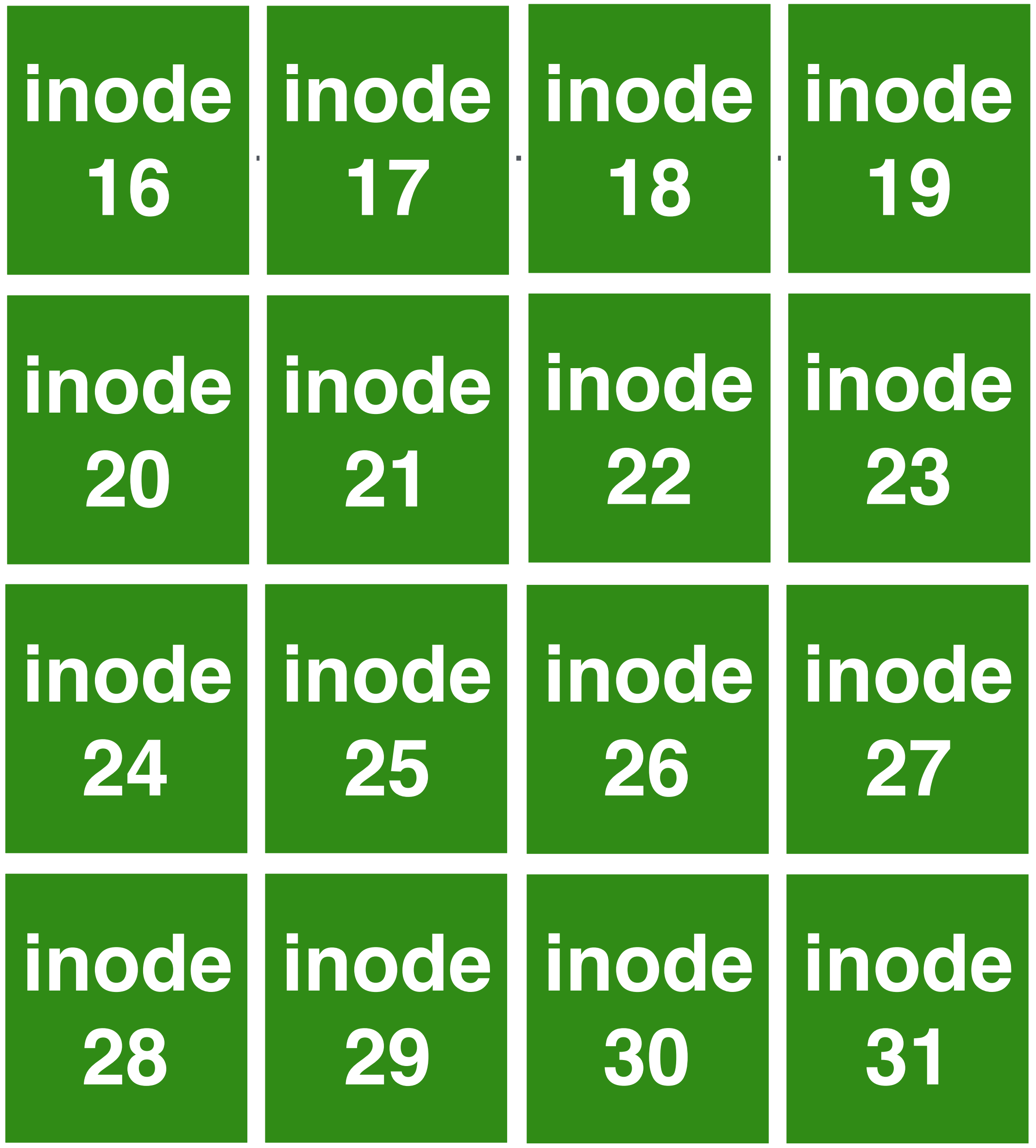
Inodes


I I I I I
D D D D D D D D
D D D D D D D D
D D D D D D D D
D D D D D D D D
D D D D D D D D
D D D D D D D D
D D D D D D D D

Inode Block

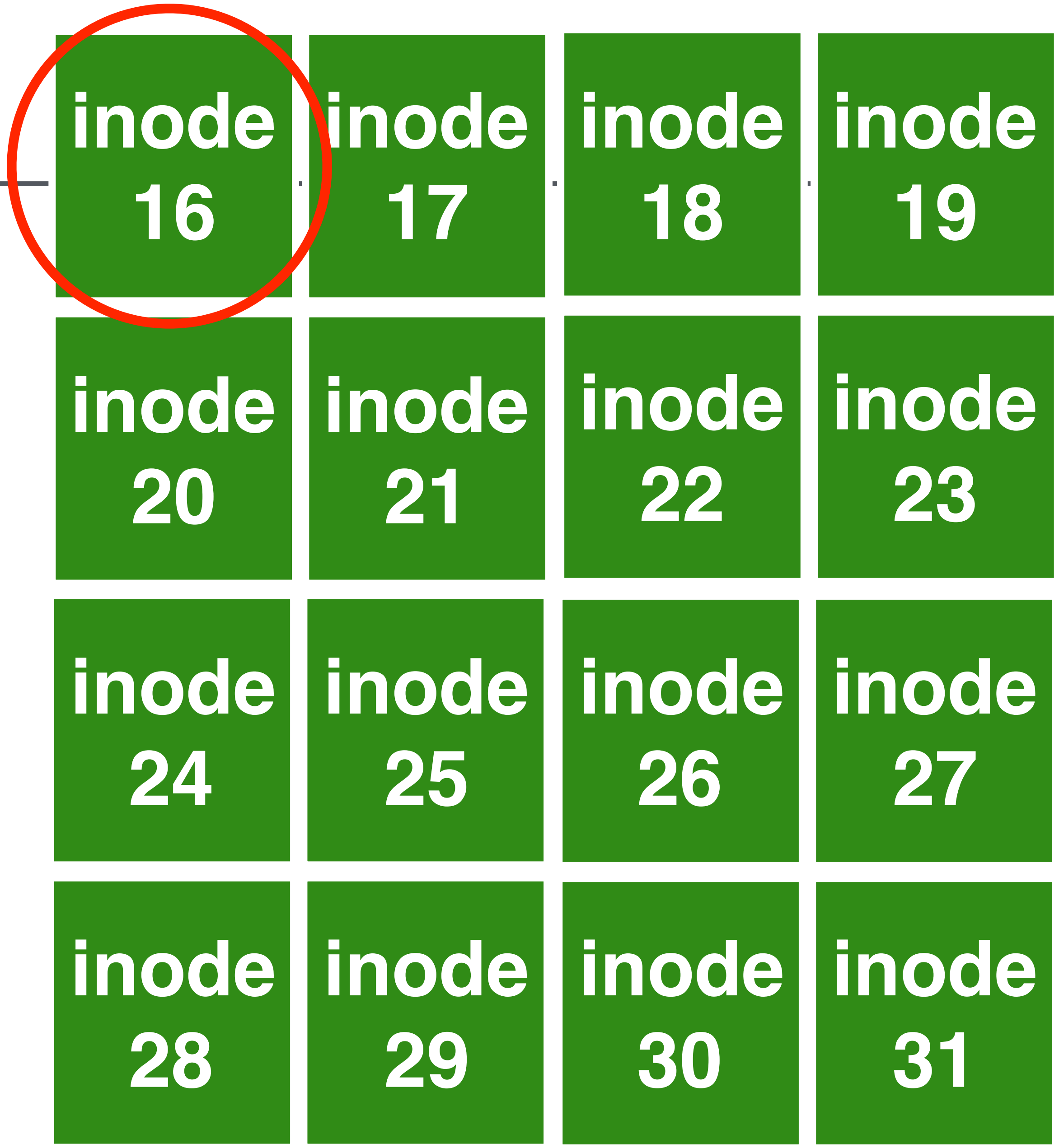
Inodes are typically 128 or 256 bytes (depends on the FS).

So 16 - 32 inodes per inode block.



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 Block



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

Inodes are typically 128 or 256 bytes (depends on the FS).

So 16 - 32 inodes per inode block.

Inode

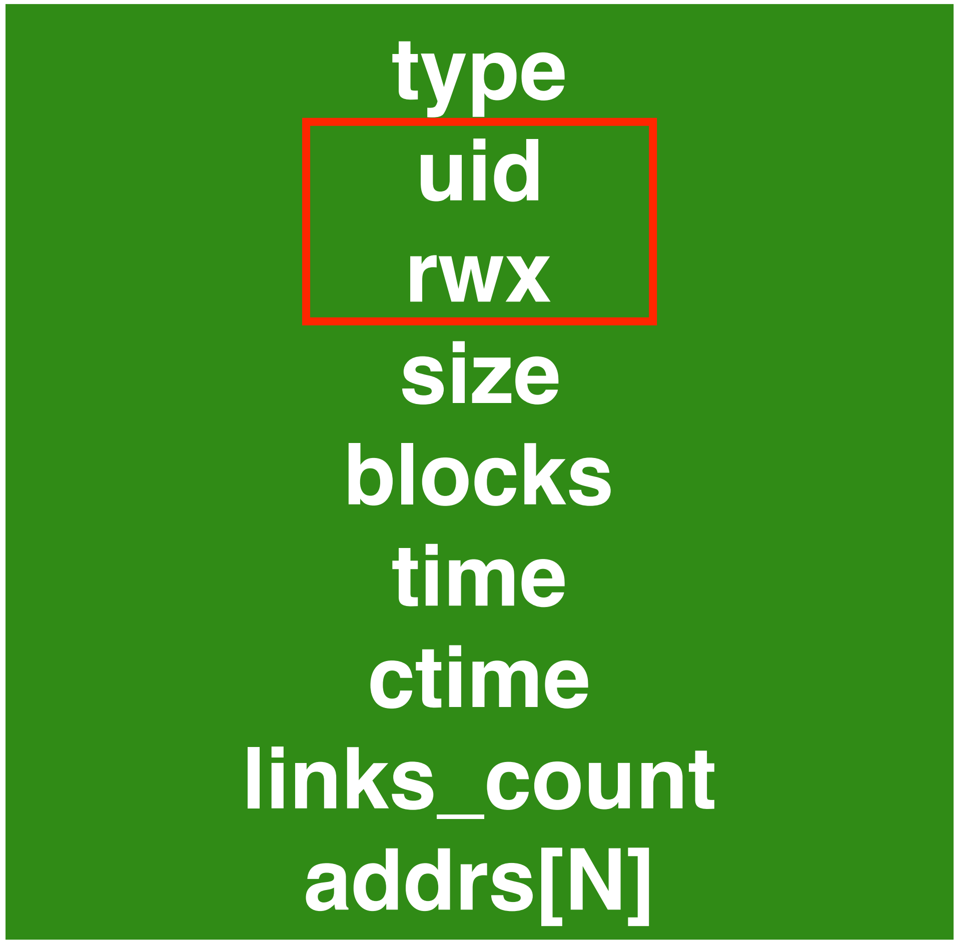
type
uid
rwx
size
blocks
time
ctime
links_count
addrs[N]

Inode

type
uid
rwx
size
blocks
time
ctime
links_count
addrs[N]

file or directory?

Inode



type

uid

rwx

size

blocks

time

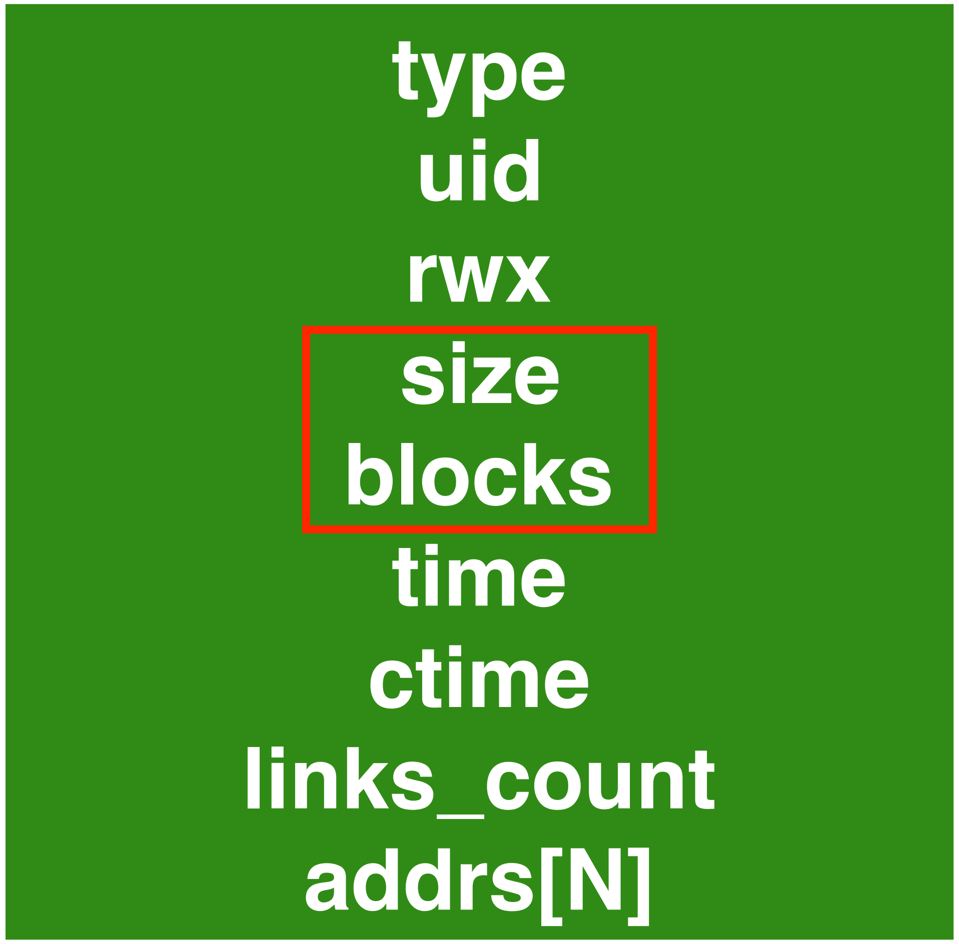
ctime

links_count

addrs[N]

user and permissions

Inode



type

uid

rwx

size

blocks

time

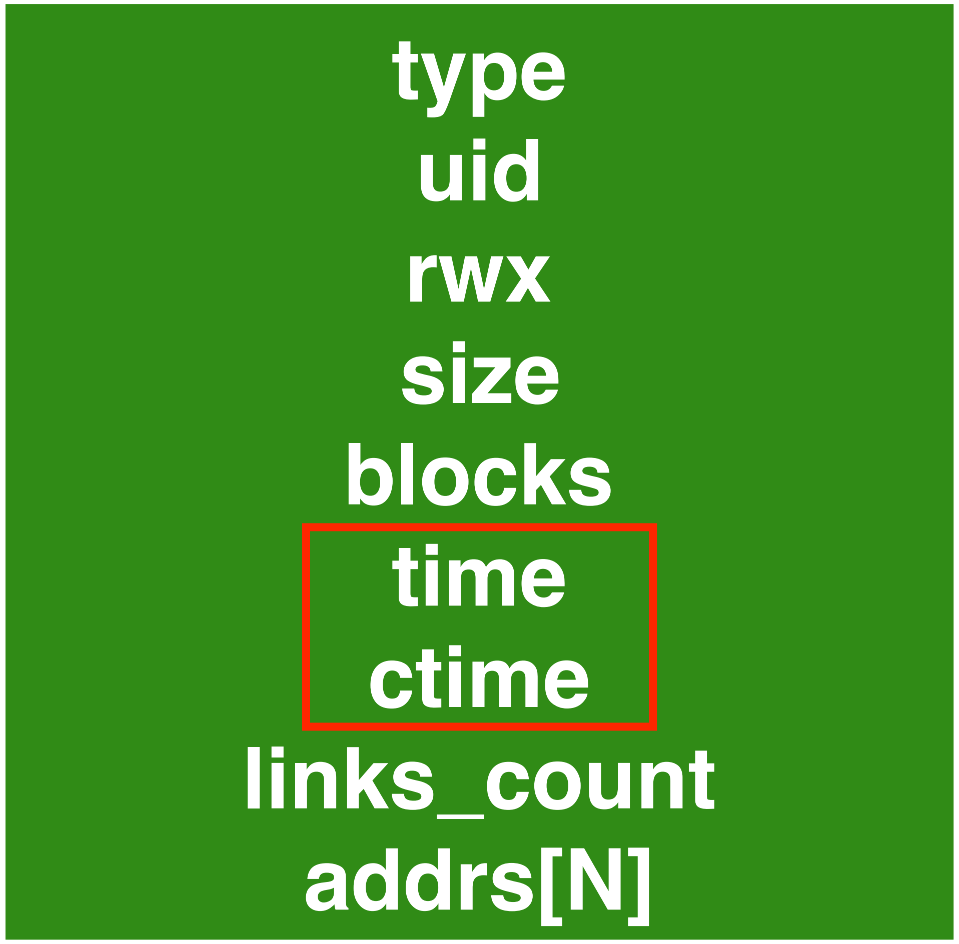
ctime

links_count

addrs[N]

size in bytes and blocks

Inode



type
uid
rwx
size
blocks
time
ctime
links_count
addrs[N]

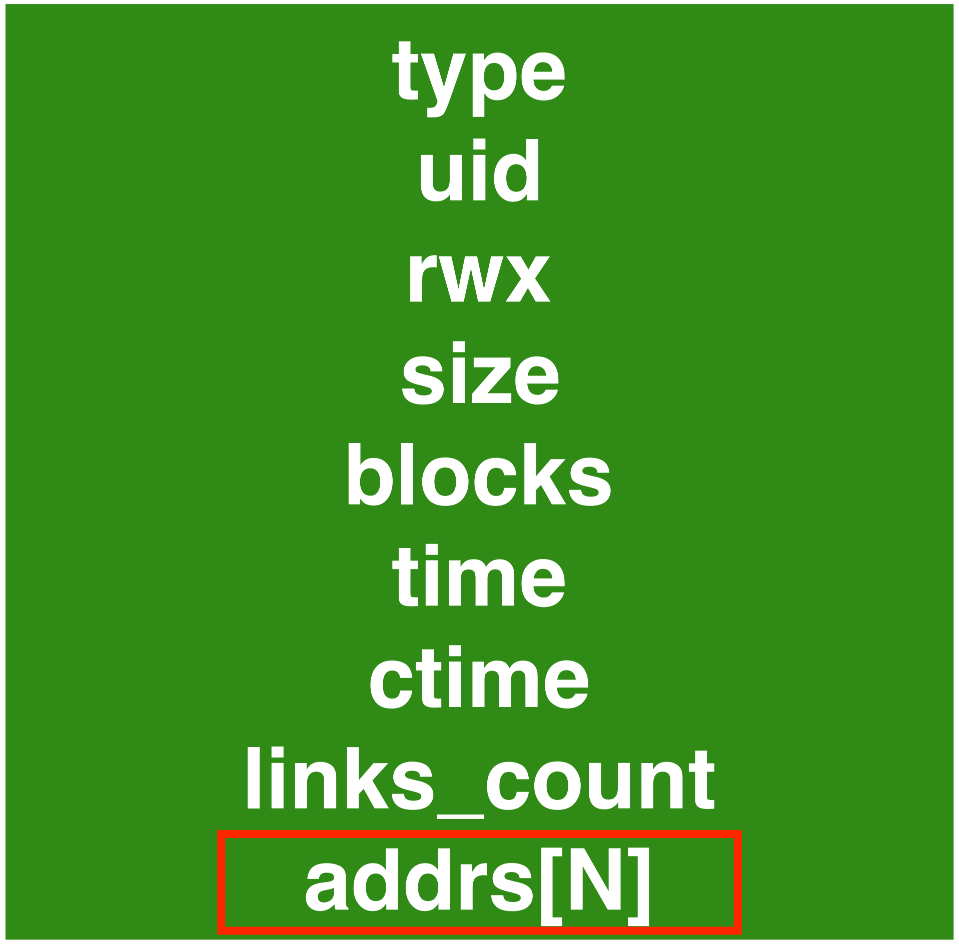
access time, create time

Inode

type
uid
rwx
size
blocks
time
ctime
links_count
addrs[N]

how many paths

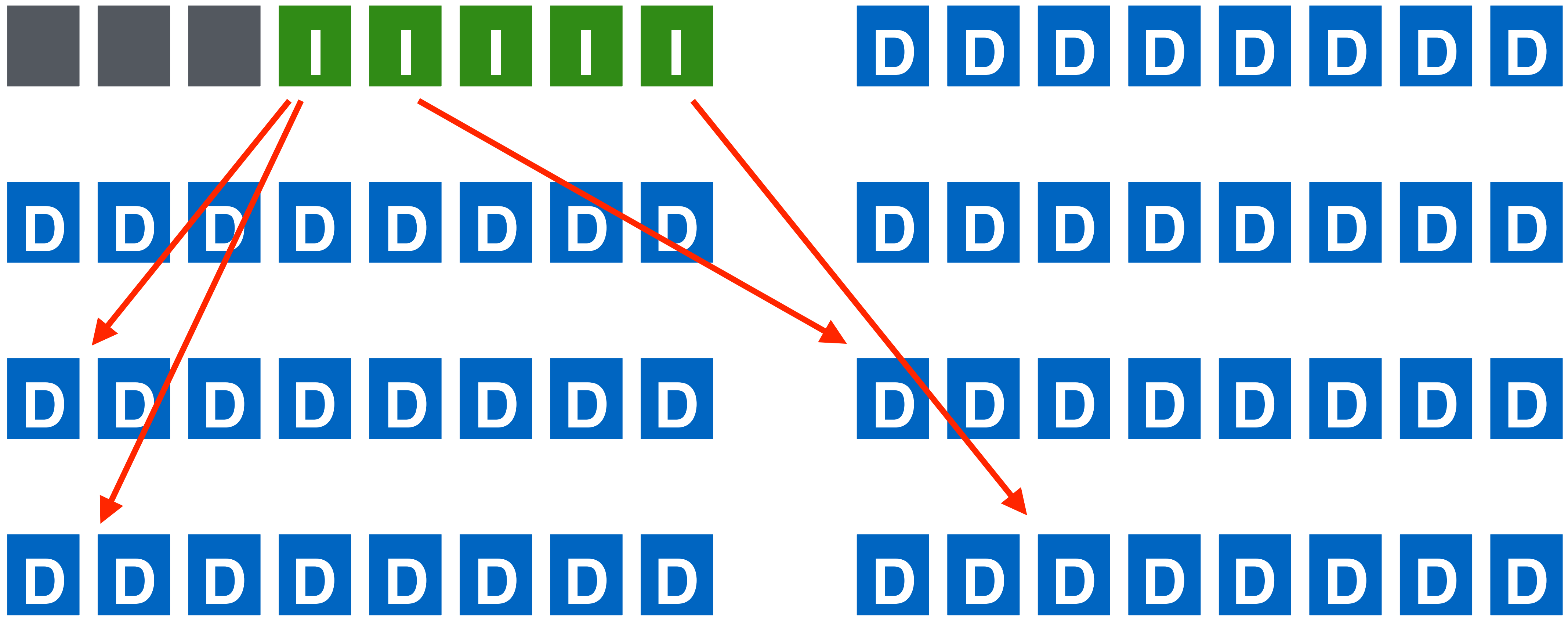
Inode



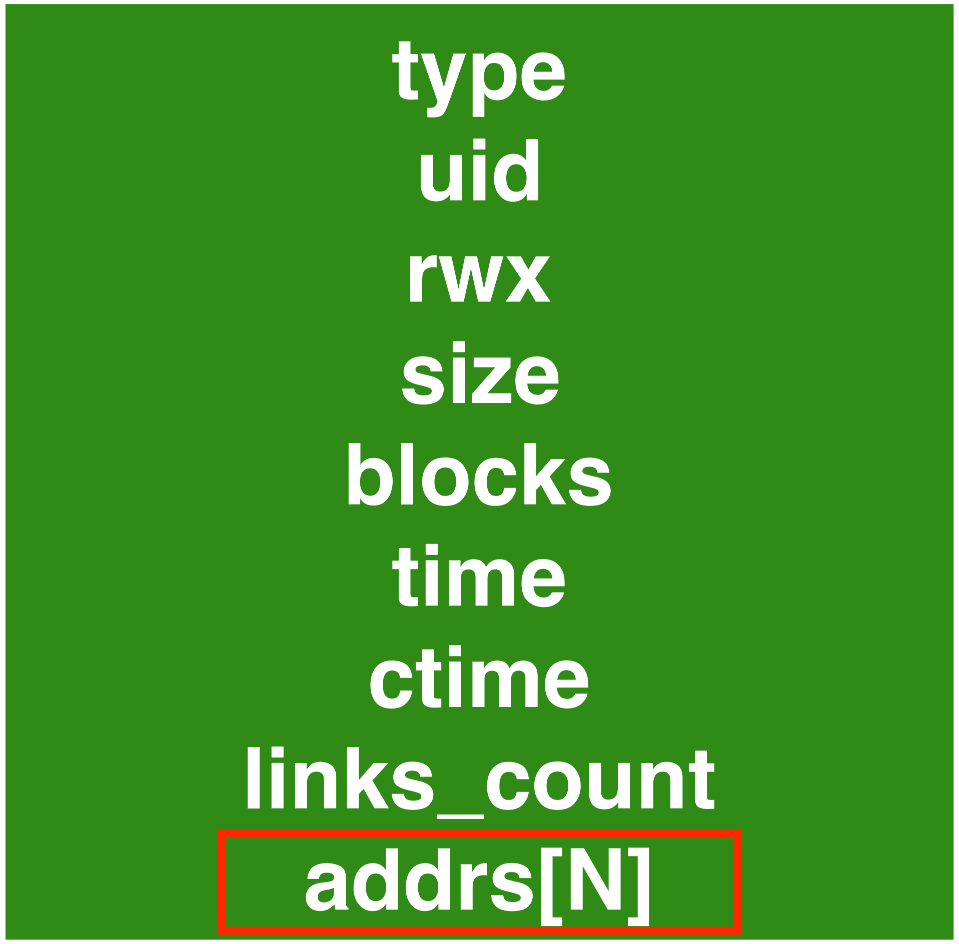
type
uid
rwx
size
blocks
time
ctime
links_count
addrs[N]

N data blocks

Inode

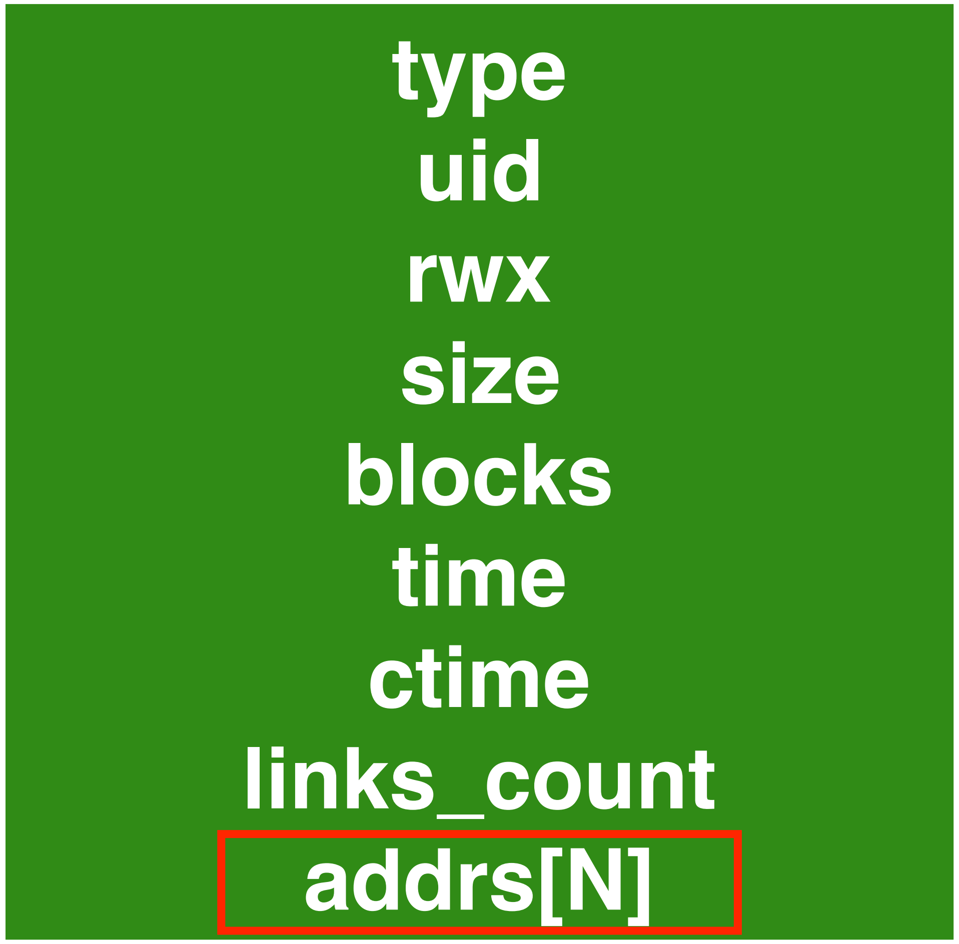


Inode


type
uid
rwx
size
blocks
time
ctime
links_count
addrs[N]

Assume 4-byte addrs.
What is an upper bound
on the file size?
(assume 256-byte inodes)

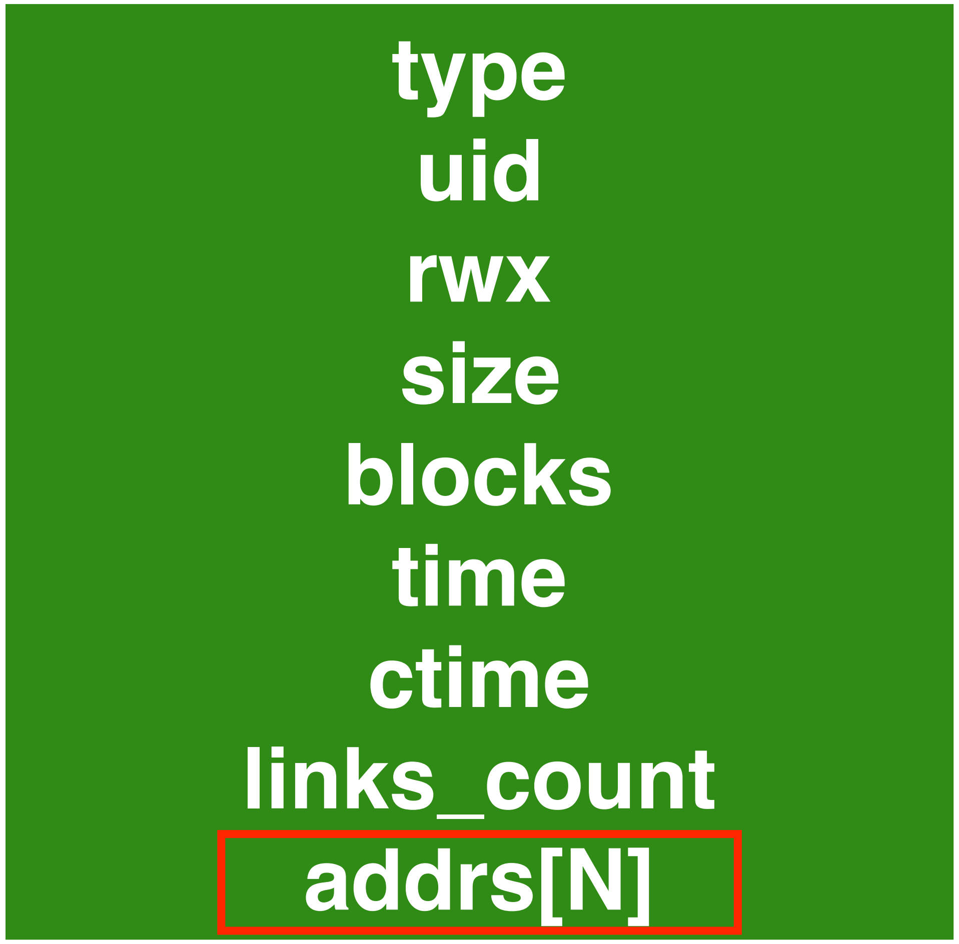
Inode



type
uid
rwx
size
blocks
time
ctime
links_count
addrs[N]

Assume 4-byte addrs.
What is an upper bound on the file size?
(assume 256-byte inodes)

Inode

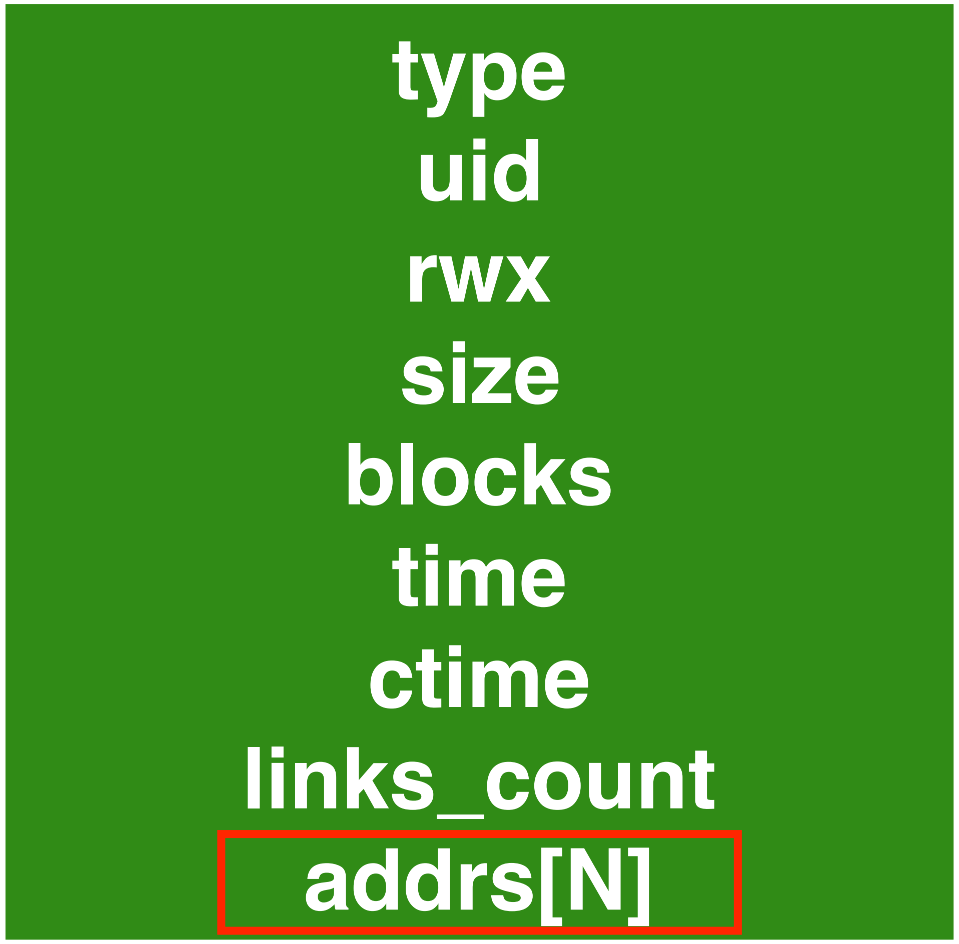


type
uid
rwx
size
blocks
time
ctime
links_count
addrs[N]

Assume 4-byte addrs.
What is an upper bound
on the file size?
(assume 256-byte inodes)

- Upper bound on #addresses = 256/4 = 64
- Upper bound file size = 4K*64 = 256K
- Typical # addresses = 12, and thus typical max size = 48K

Inode



type
uid
rwx
size
blocks
time
ctime
links_count
addrs[N]

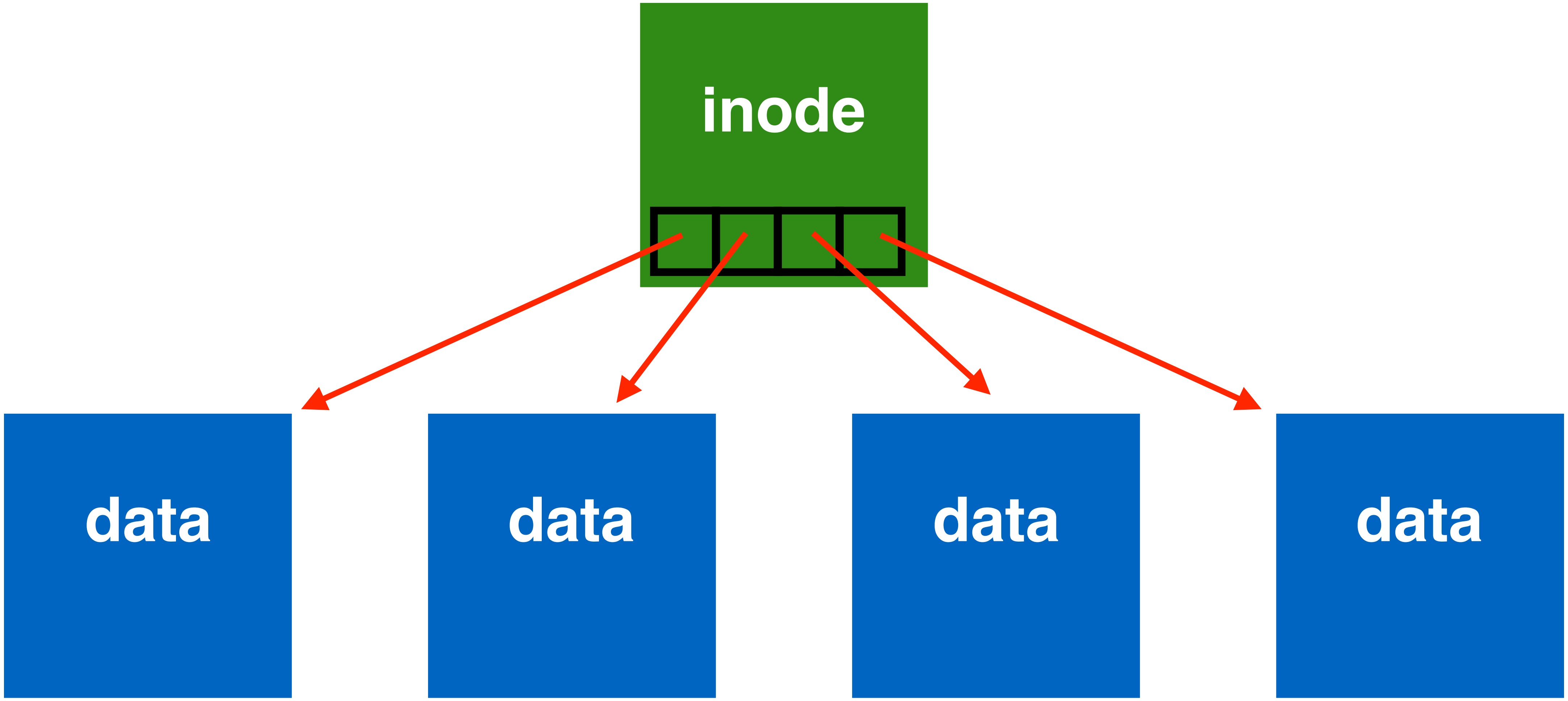
Assume 4-byte addrs.
What is an upper bound on the file size?
(assume 256-byte inodes)

How to get larger files?

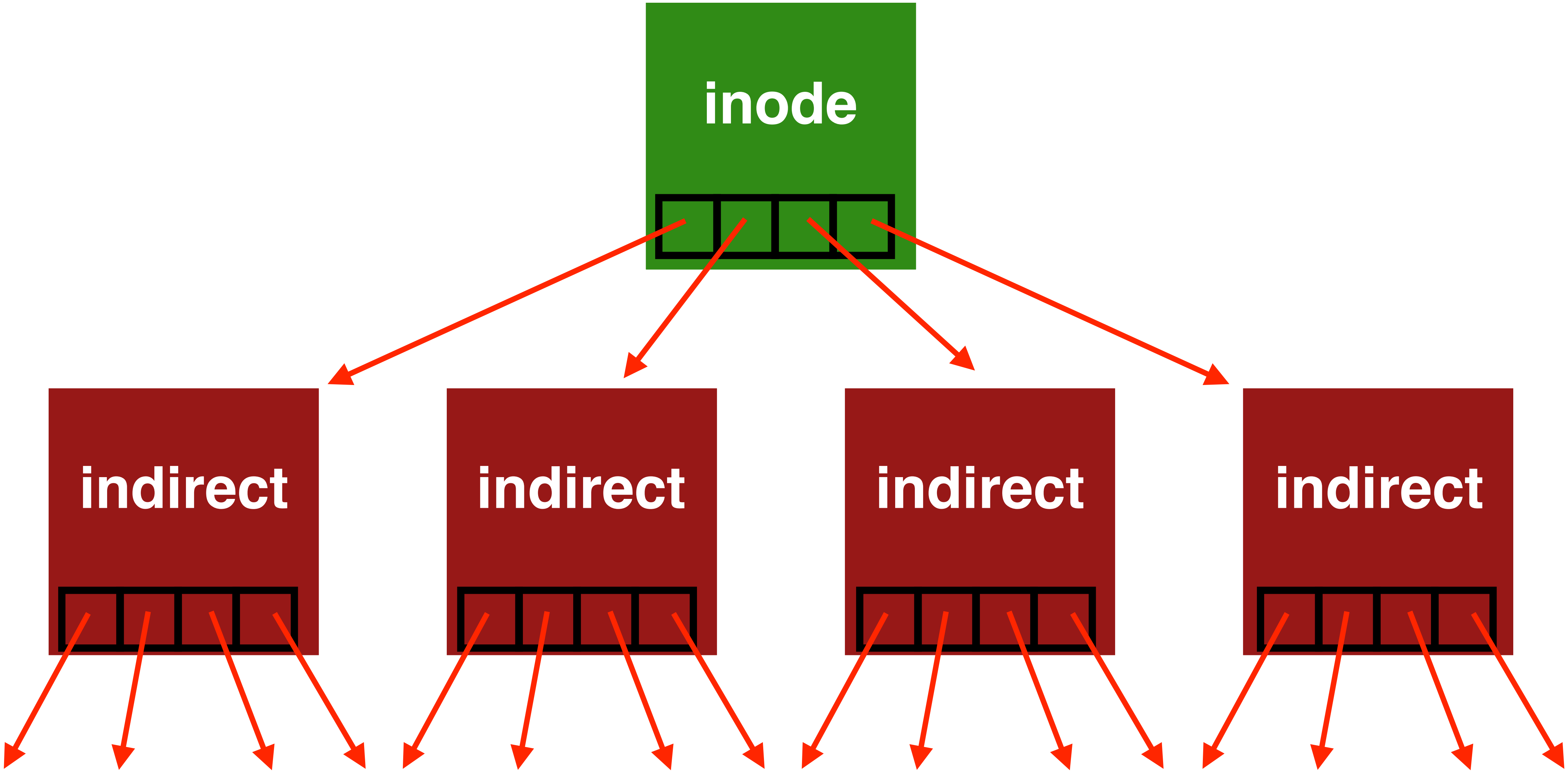
Structures

Common file-system structures

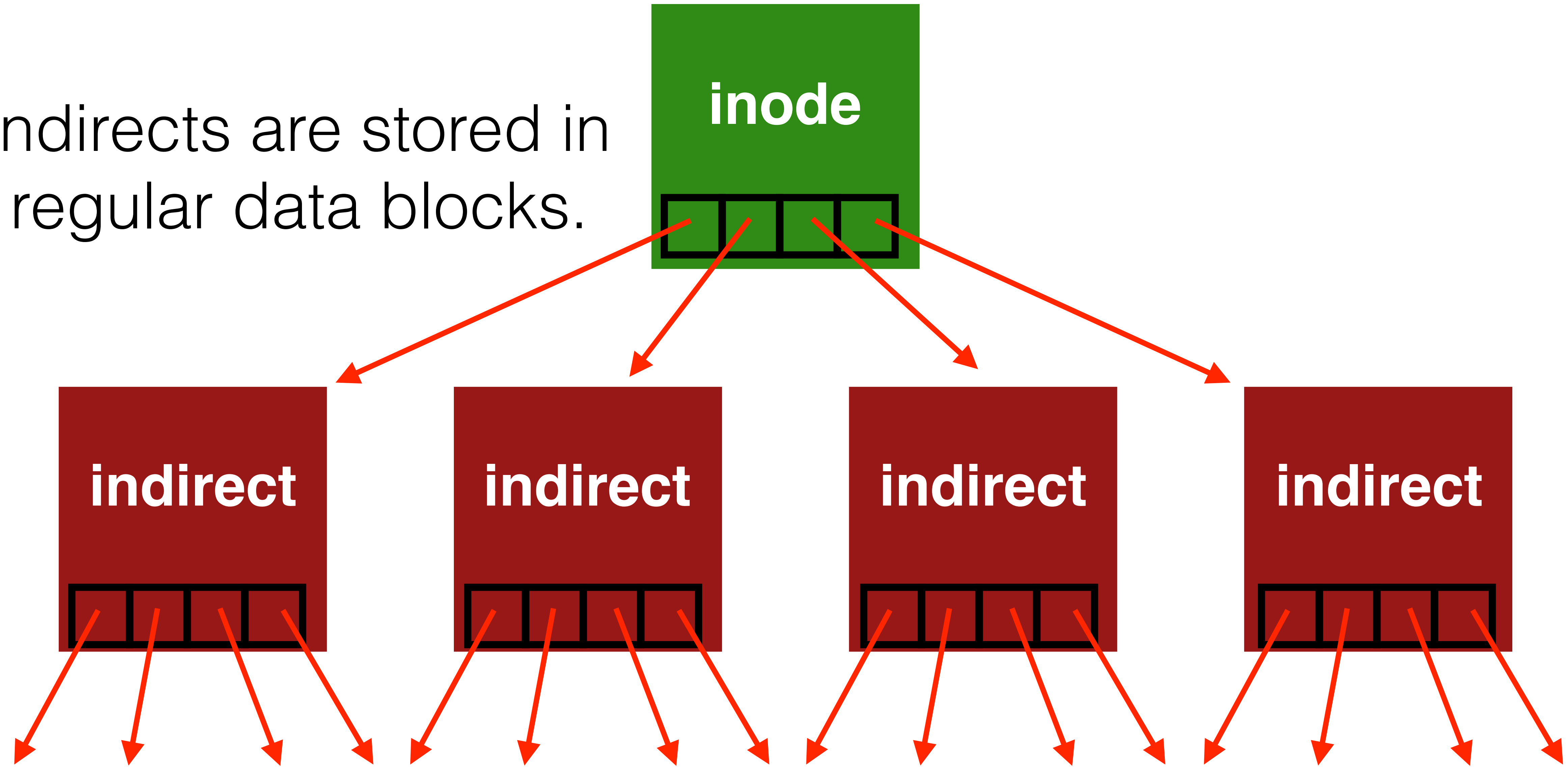
- data block
- inode table
- indirect block
- directories
- data bitmap
- inode bitmap
- superblock

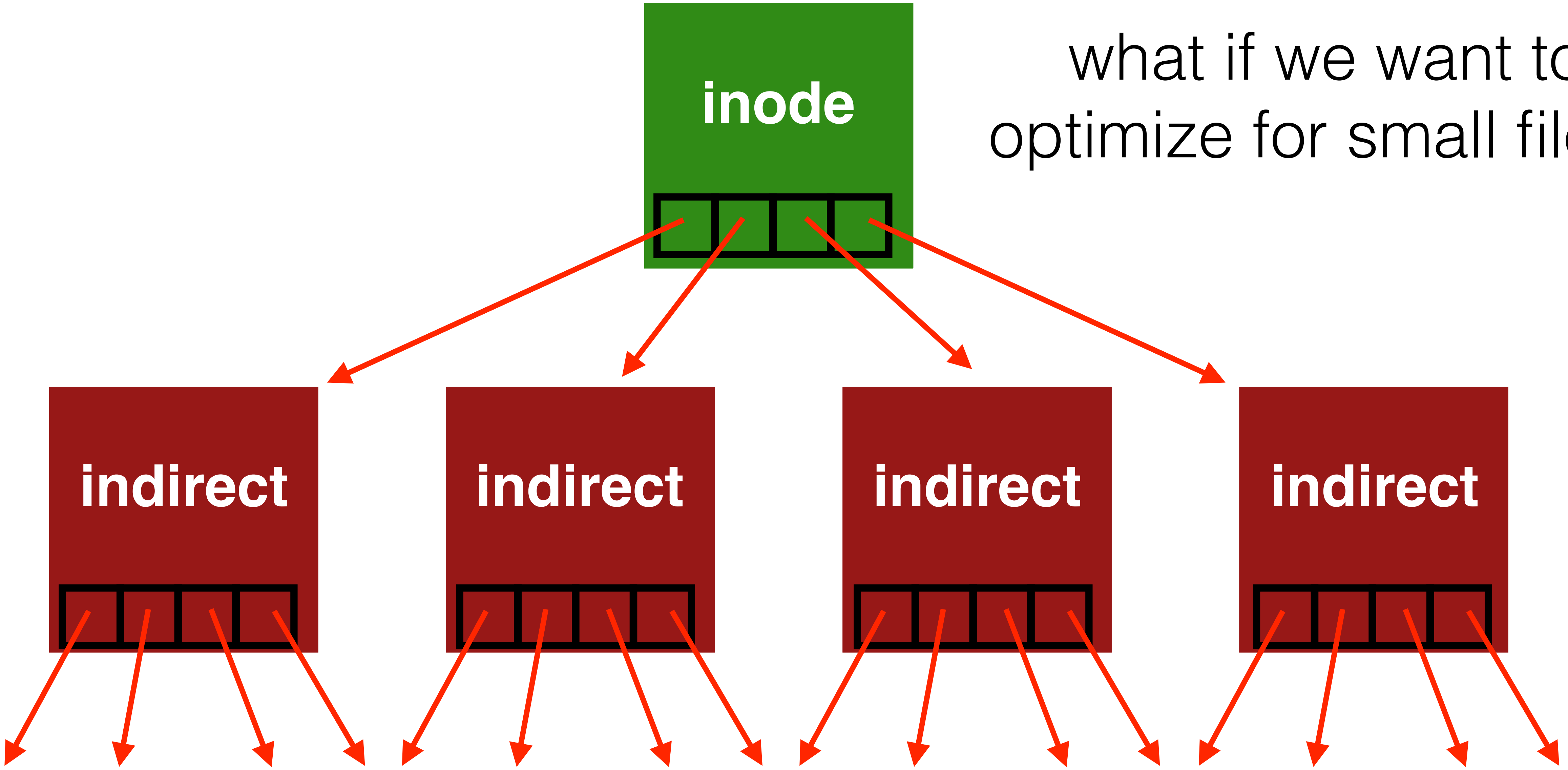


inode
data
data
data
data

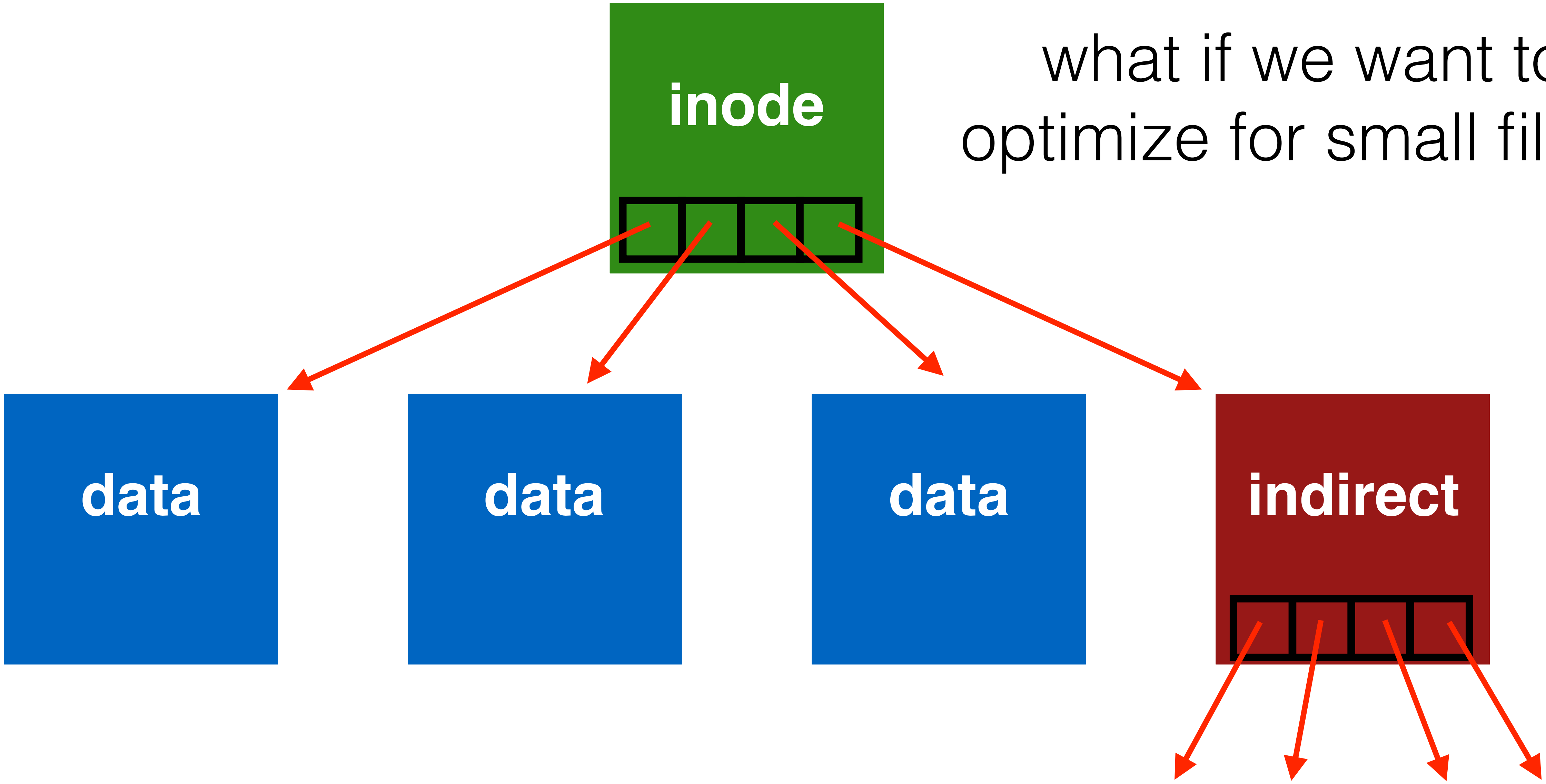

inode
indirect
indirect
indirect
indirect

indirects are stored in regular data blocks.





what if we want to optimize for small files?



what if we want to optimize for small files?

Structures

Common file-system structures

- data block
- inode table
- indirect block
- directories
- data bitmap
- inode bitmap
- superblock

Directories

File systems vary.

Common design: just store directory entries in files.

Various formats could be used
- lists
- b-trees

Simple List Example

valid	name	inode
1	.	134
1	..	35
1	foo	80
1	bar	23

Simple List Example

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

unlink(“foo”)

Structures

Common file-system structures

- data block
- inode table
- indirect block
- directories
- data bitmap
- inode bitmap
- superblock

Allocation

How do we find free data blocks or free inodes?

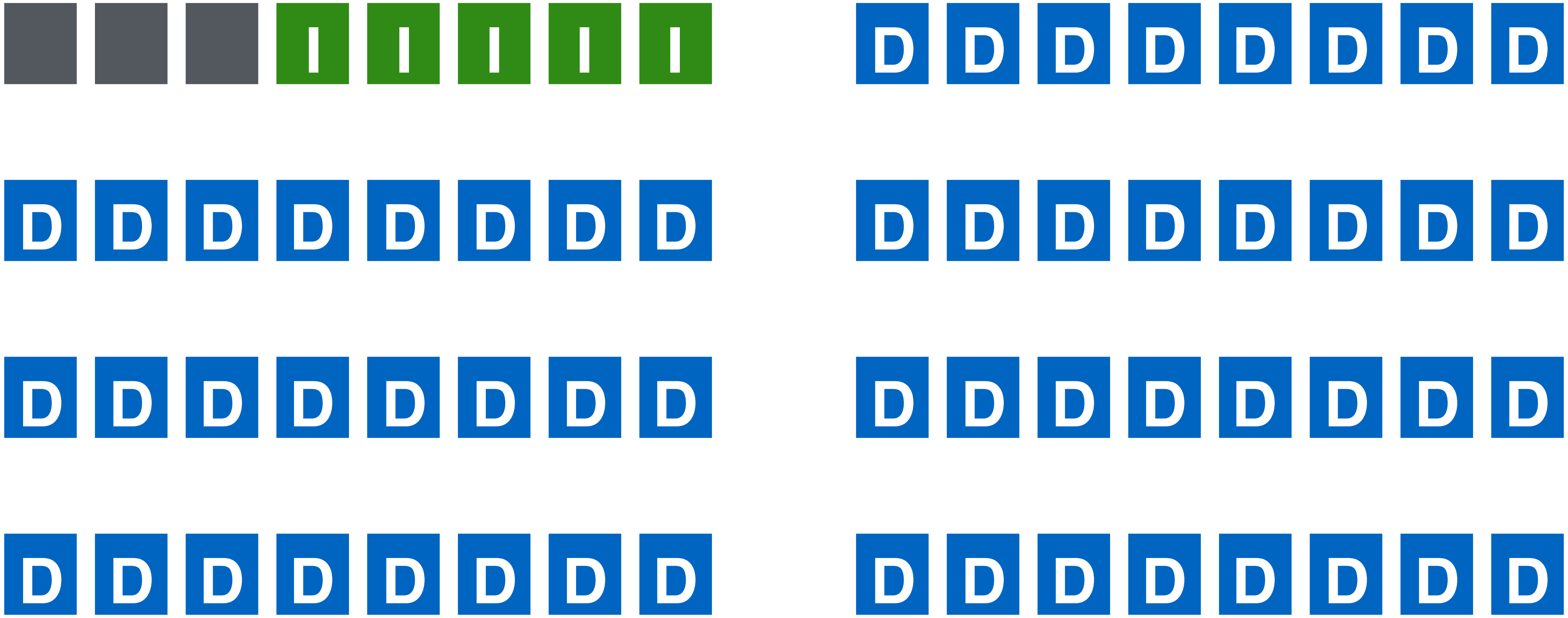
Allocation

How do we find free data blocks or free inodes

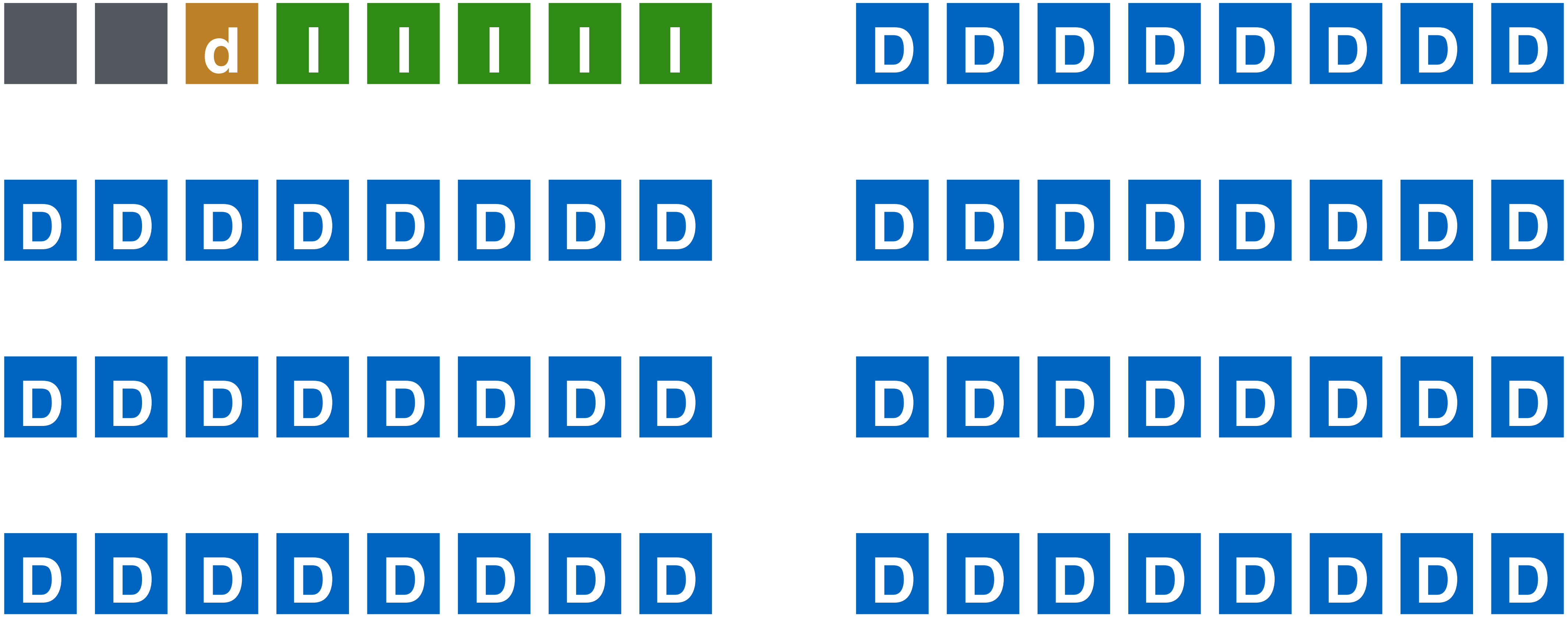
- Free list
- Bitmaps

Tradeoffs?

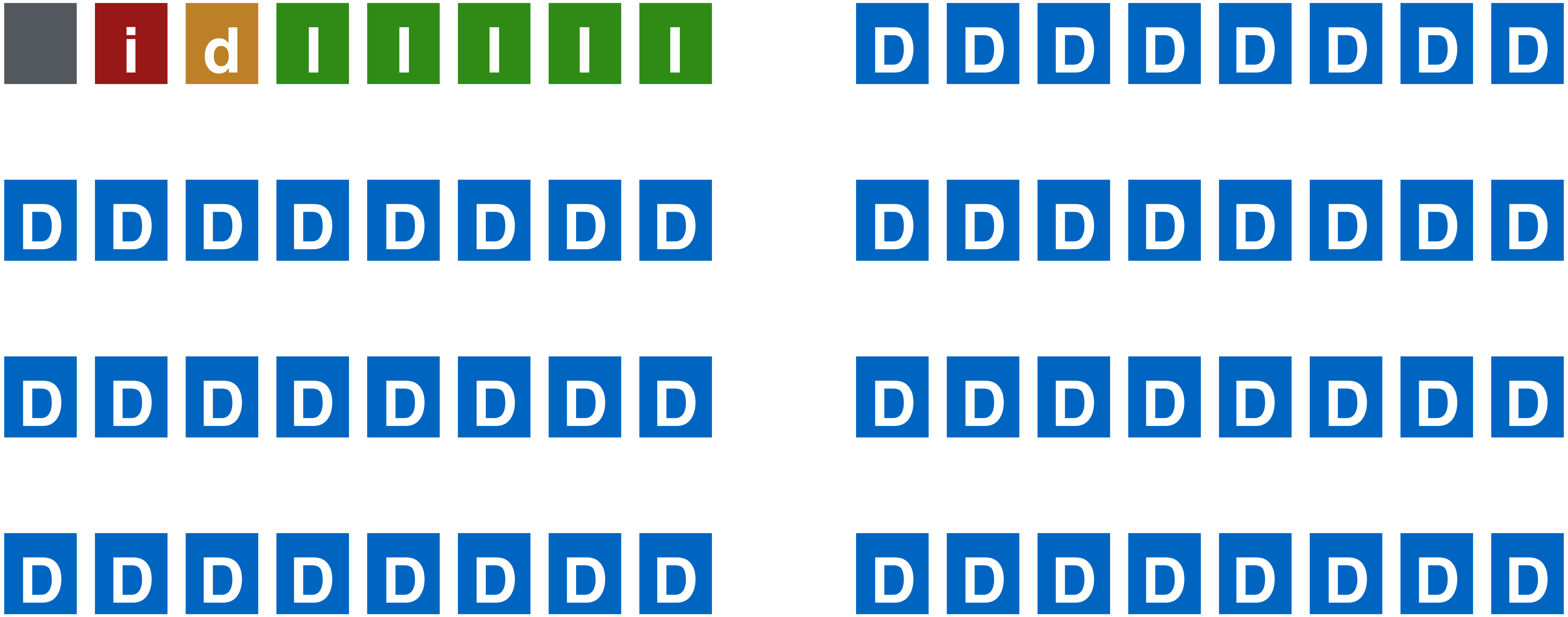
Bitmaps


I I I I I
D D D D D D D D
D D D D D D D D
D D D D D D D D
D D D D D D D D
D D D D D D D D
D D D D D D D D
D D D D D D D D

Data Bitmap


d I I I I I D D D D D D D D
D D D D D D D D D D D D D D D D
D D D D D D D D D D D D D D D D
D D D D D D D D D D D D D D D D

Inode Bitmap



i d I I I I I D D D D D D D D
D D D D D D D D D D D D D D D D
D D D D D D D D D D D D D D D D
D D D D D D D D D D D D D D D D

Structures

Common file-system structures

- data block
- inode table
- indirect block
- directories
- data bitmap
- inode bitmap
- superblock

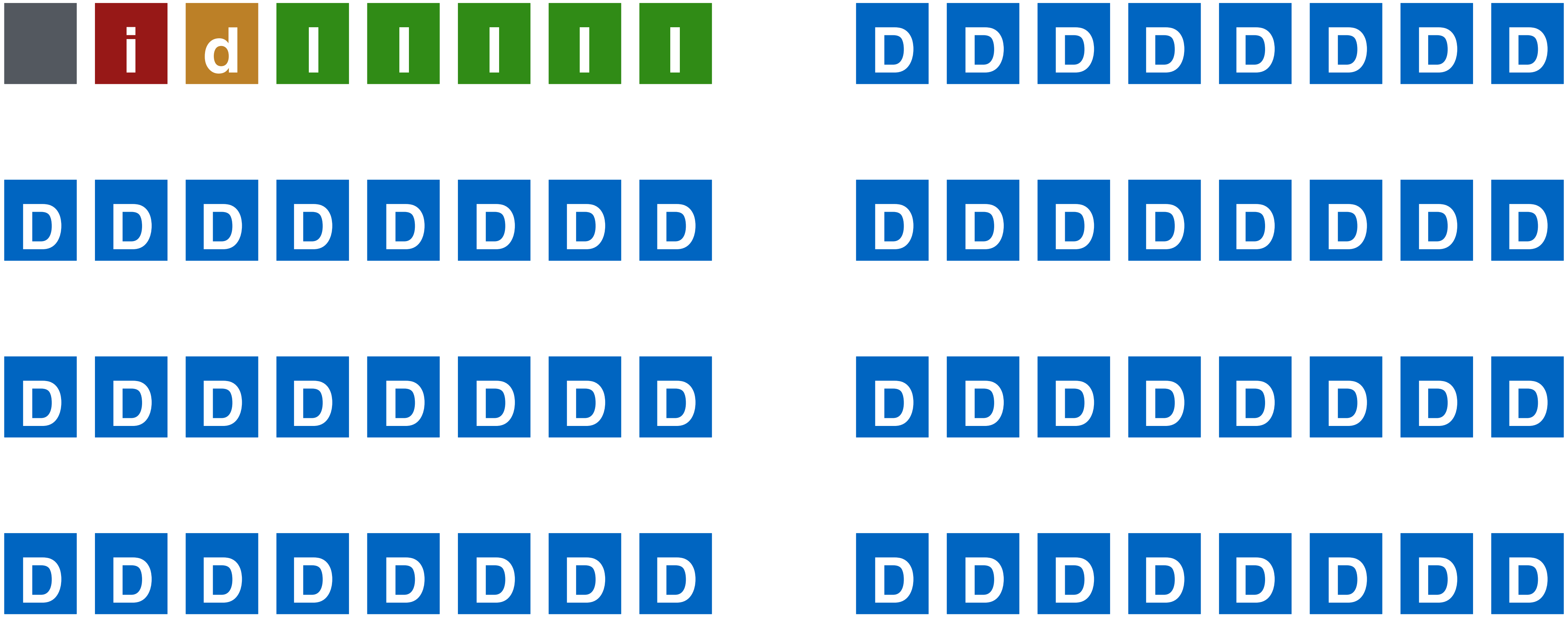
Superblock

Need to know basic FS metadata, like:

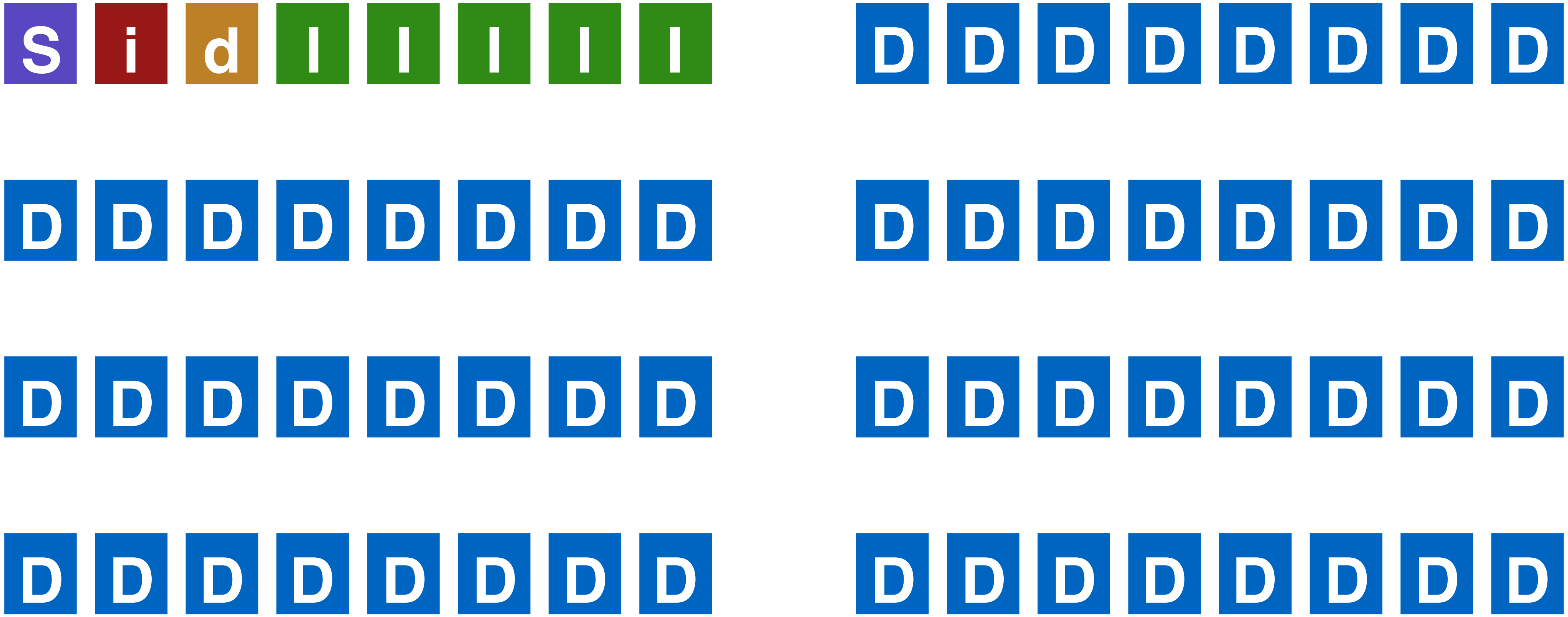
- block size
- how many inodes are there
- how much free data

Store this in a superblock

Super Block


i d I I I I I
D D D D D D D D
D D D D D D D D D D D D D D D D
D D D D D D D D D D D D D D D D
D D D D D D D D D D D D D D D D

Super Block


S i d I I I I I
D D D D D D D D
D D D D D D D D
D D D D D D D D
D D D D D D D D
D D D D D D D D
D D D D D D D D
D D D D D D D D

Structure Overview

Structures:

- superblock
- data block
- data bitmap
- inode table
- inode bitmap
- indirect block
- directories

Operations

FS
- mkfs
- mount

File
- create
- write
- open
- read
- close

Operations

FS

- mkfs
- mount

File

- create
- write
- open
- read
- close

mkfs

mkfs

- Different version for each file system (e.g., mkfs.ext4, mkfs.xfs, mkfs.btrfs, etc)

mkfs

- Different version for each file system (e.g., mkfs.ext4, mkfs.xfs, mkfs.btrfs, etc)

mkfs

- Different version for each file system (e.g., mkfs.ext4, mkfs.xfs, mkfs.btrfs, etc)

- Initialize metadata (bitmaps, inode table).

mkfs

- Different version for each file system (e.g., mkfs.ext4, mkfs.xfs, mkfs.btrfs, etc)
- Initialize metadata (bitmaps, inode table).

mkfs

- Different version for each file system (e.g., mkfs.ext4, mkfs.xfs, mkfs.btrfs, etc)
- Initialize metadata (bitmaps, inode table).
- Create empty root directory.

Operations

FS

- mkfs
- mount

File

- create
- write
- open
- read
- close

mount

mount

- Add the file system to the FS tree.

mount

- Add the file system to the FS tree.

mount

- Add the file system to the FS tree.
- Minimally requires reading super-block

Operations

FS

- mkfs
- mount

File

- create
- write
- open
- read
- close
- lseek

open /foo/bar

data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data

open /foo/bar

data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data
		read					

open /foo/bar

data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data
		read					
					read		

open /foo/bar

data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data
		read					
					read		
			read				

open /foo/bar

data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data
		read					
					read		
			read				
						read	

open /foo/bar

data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data
		read					
					read		
			read				
						read	
				read			

Operations

FS

- mkfs
- mount

File

- create
- write
- open
- read
- close

read /foo/bar

data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data
		read					
					read		
			read				
						read	
				read			

First few operations same as: open /foo/bar

read /foo/bar

	data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data
OPEN			read					
						read		
				read				
							read	
					read			
					read			

read /foo/bar

	data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data
OPEN			read					
						read		
				read				
							read	
					read			
					read			
								read

read /foo/bar

	data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data
OPEN			read					
						read		
				read				
							read	
					read			
					read			
								read
					write			

Operations

FS
- mkfs
- mount

File
- create
- write
- open
- read
- close

create /foo/bar

data	inode	root	foo	bar	root	foo
bitmap	bitmap	inode	inode	inode	data	data

create /foo/bar

data	inode	root	foo	bar	root	foo
bitmap	bitmap	inode	inode	inode	data	data
		read				
					read	

create /foo/bar

data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data
		read				
					read	
			read			
						read

create /foo/bar

data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data
		read				
					read	
			read			
						read
	read					
	write					
						write
				read?		
				write		
			write			

Operations

FS

- mkfs
- mount

File

- create
- write
- open
- read
- close

write to /foo/bar (assuming OPENED)

data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data

write to /foo/bar (assuming OPENED)

data	inode	root	foo	bar	root	foo	bar
bitmap	bitmap	inode	inode	inode	data	data	data
				read			

write to /foo/bar (assuming OPENED)

data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data
				read			
read							

write to /foo/bar (assuming OPENED)

data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data
				read			
read							
write							

write to /foo/bar (assuming OPENED)

data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data
				read			
read							
write							
							write

write to /foo/bar (assuming OPENED)

data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data
				read			
read							
write							
							write
				write			

Operations

FS
- mkfs
- mount

File
- create
- write
- open
- read
- close

close /foo/bar

data	inode	root	foo	bar	root	foo	bar
bitmap	bitmap	inode	inode	inode	data	data	data

close /foo/bar

data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data

nothing to do on disk!

Efficiency

How can we avoid this excessive I/O for basic ops?

Efficiency

How can we avoid this excessive I/O for basic ops?

Cache for:

- reads
- write buffering

Structures

Common file-system structures

- superblock
- data block
- data bitmap
- inode table
- inode bitmap
- indirect block
- directories

Write Buffering

Why does procrastination help?

Write Buffering

Why does procrastination help?

Overwrites, deletes, scheduling.

Shared structs (e.g., bitmaps+dirs) often overwritten.

Write Buffering

Why does procrastination help?

Overwrites, deletes, scheduling.

Shared structs (e.g., bitmaps+dirs) often overwritten.

We decide: how much to buffer, how long to buffer…
 - tradeoffs?