File system implementation: data

Glenn Bruns
CSUMB

Lecture Objectives

After this lecture, you should be able to:

- describe the on-disk data structures for a simple file system

Problem statement

We need to build a file system out of a bunch of disk blocks.

Design goals:

- high performance
- low storage waste
- ability to handle very large files
- support for inode and directory concepts

Lecture Objectives

After this lecture, you should be able to:

- describe the on-disk data structures for a simple file system

How to store file system on disk?

We have a bunch of 4 KB block disks.

Need to track:

- file contents
- file metadata
- directory info
- which blocks are in use

Remember:

- file has inode #; a bunch of data
- directory has inode #; a list of (name, inode) pairs

We'll focus on the "very simple file system" of OSTEP

Main files systems on Linux are ext3 and ext4

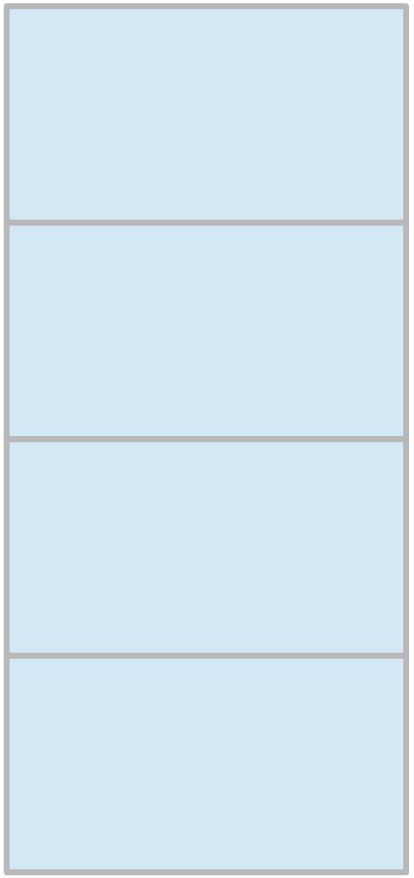
Now you design it

All you have to work with is a bunch of 4 KB data blocks

files have:
- inode #
- metadata
- content

directories have:
- inode #
- (name, inode #) pairs

You need to be able to find root directory



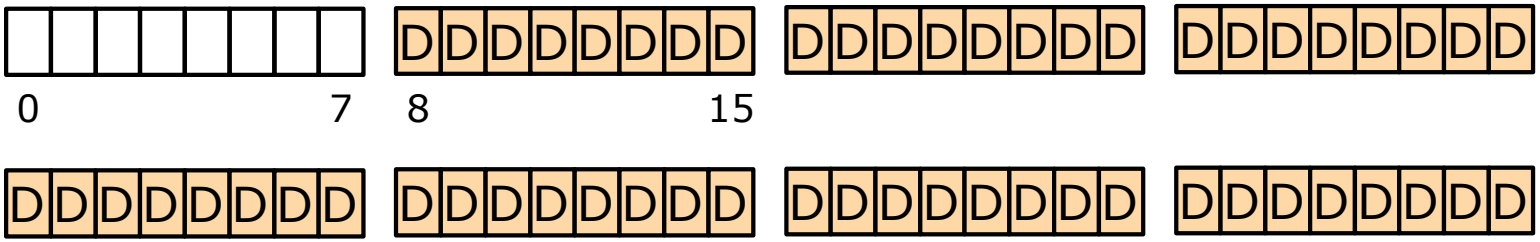
...

Very simple file system: file contents

We'll build the file system on a disk partition of only 64 blocks.

Each block is 4 KB in size.

Let's allocate 56 blocks for file contents (data region).



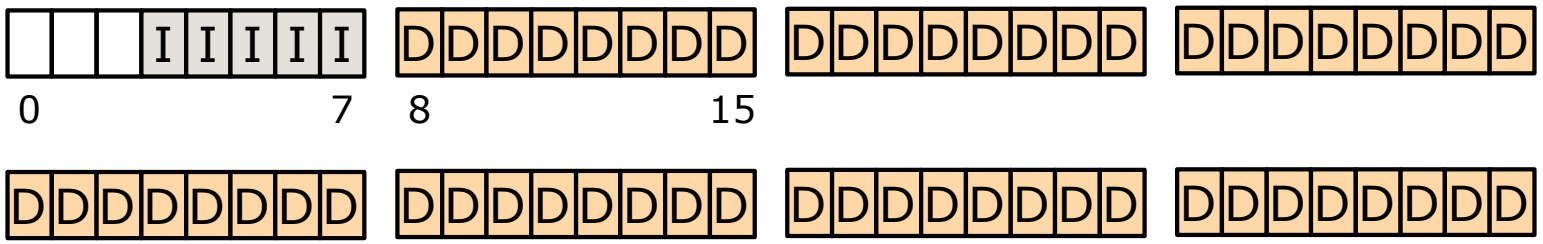
inode data

inode will capture file metadata and location of the file's data blocks

Let’s use 5 blocks for inodes (the inode table)

Each inode will be 256 bytes (so 16 inodes/block)

This gives us space for 5 * 16 = 80 inodes



Allocation data

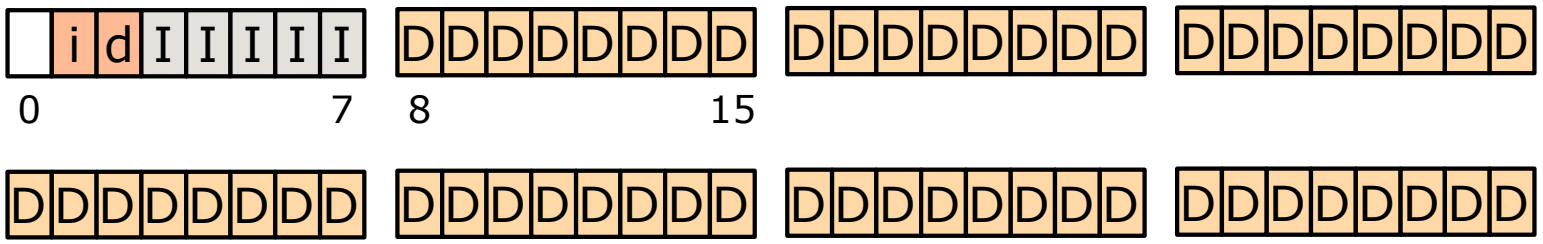
We need to track which data blocks and inodes are being used

We can use bitmaps

- each bit tracks 1 item
- 0 = free, 1 = used

With 1 block we can track 32K items.

We'll use 1 block for inode bitmap, 1 for data bitmap

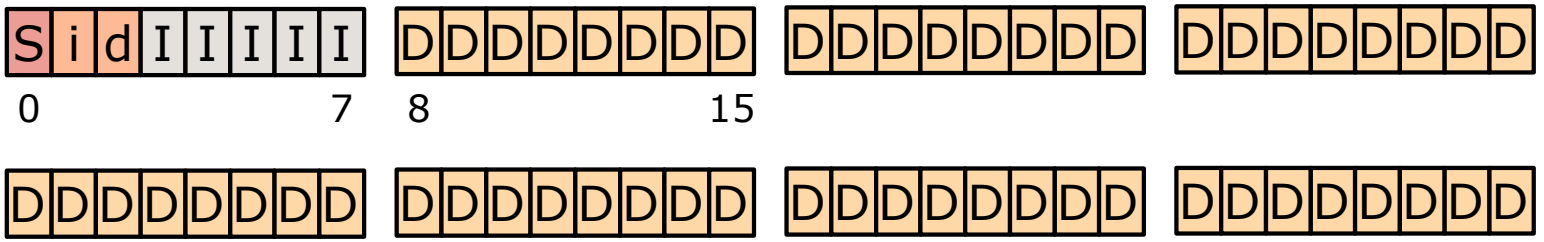


SUPERBLOCK!

We also need some top-level info:

- how many inodes and data blocks in the file system?
- where does the inode region begin? (block 3)
- etc.

This will go in the superblock, stored in 1 block.



Details on inodes

The inode for a file contains file metadata

- the type of the file
- size (in bytes), permissions, time of last access, …
- where to find data for the file

Question: does the inode contain the file name?

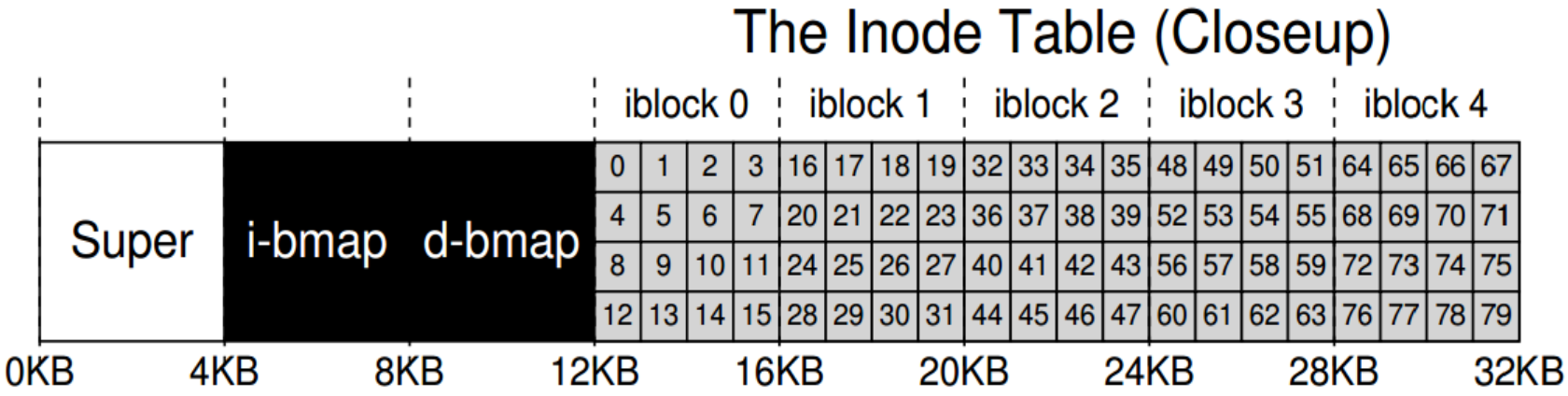
```
$ stat pscp.txt
  File: `pscp.txt'
  Size: 720             Blocks: 8          IO Block: 4096   regular file
Device: fc31h/64561d    Inode: 1445224     Links: 1
Access: (0644/-rw-r--r--)  Uid: (11481/brun1992)   Gid: ( 8041/shell_faculty)
Access: 2017-04-28 21:42:39.509812078 -0700
Modify: 2016-10-05 13:40:40.000000000 -0700
Change: 2017-04-25 05:37:40.735723000 -0700
```

Finding inode data

The inode number (i-number) tells us where the inode is located.

To get inode from i-number:

- get start address of inode region from superblock
- use i-number to get offset into inode region

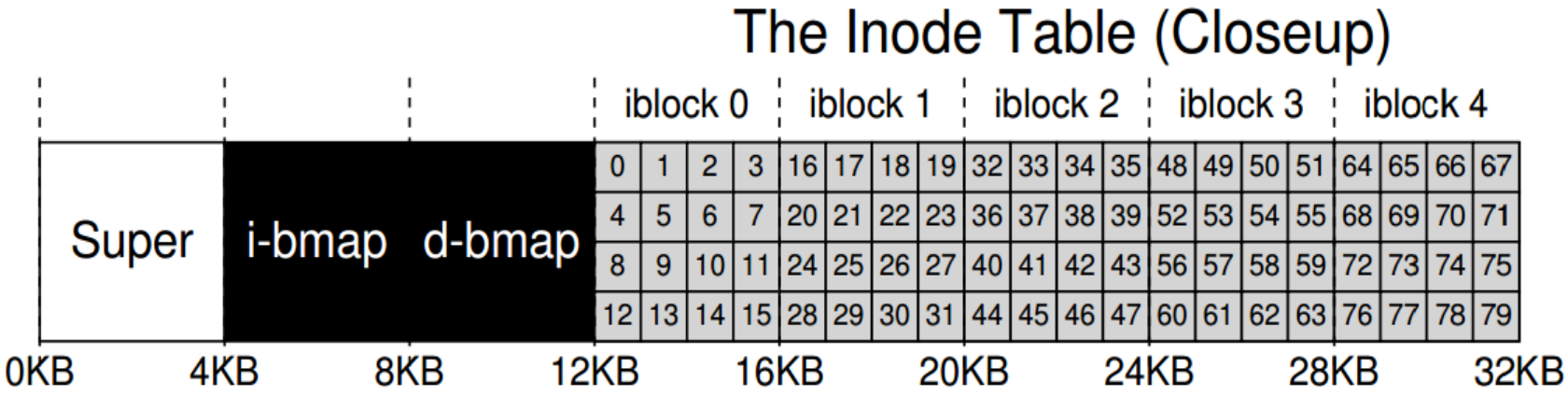


Exercise

The superblock shows that the start address of the inode region is 12KB. What is the disk address of the inode given i-number 23?

To get inode from i-number:

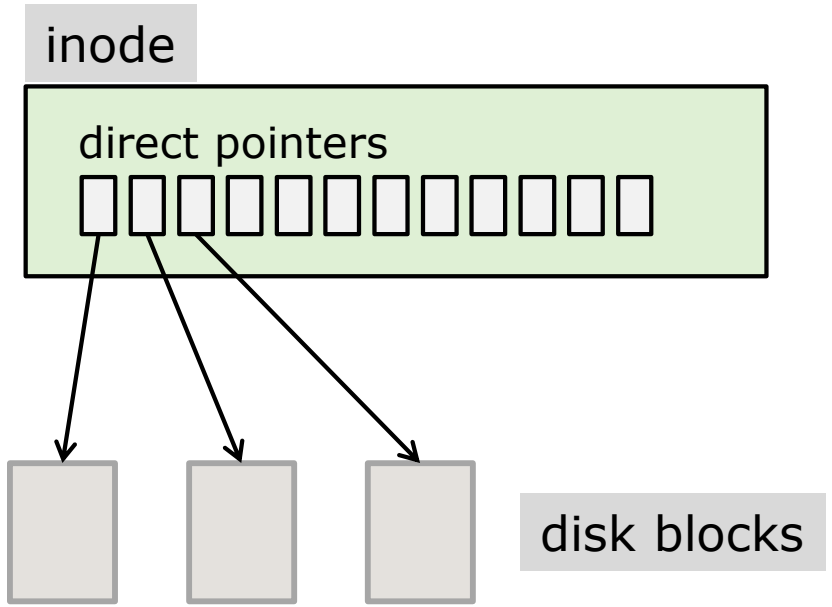
- get start address of inode region from superblock
- use i-number to get offset into inode region



Getting a file's data

Design 1:

- in inode, keep 12 direct pointers to disk blocks (use 4 bytes for each ptr)
- to get content for file:
  - get inode for file
  - get the first direct pointer
  - look at the disk block

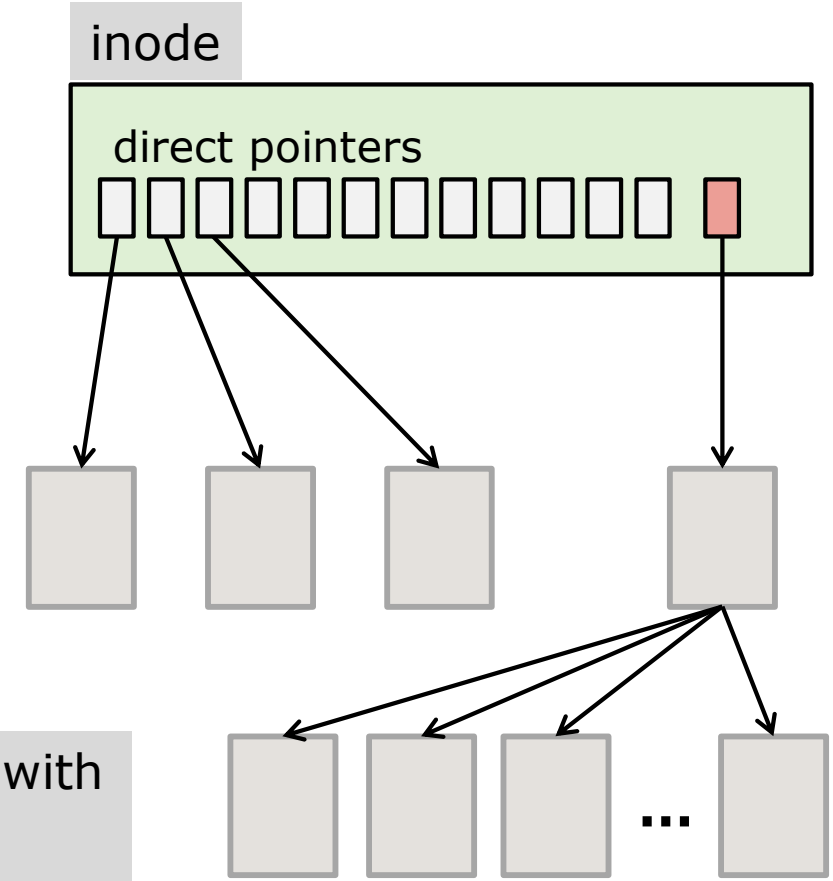


What is the problem with this approach?

What is the largest file you can have with this approach?

Getting a file's data: design 2

- in inode, keep 12 direct pointers to disk blocks
- also, keep one indirect pointer
- the indirect pointer points to a block of direct pointers



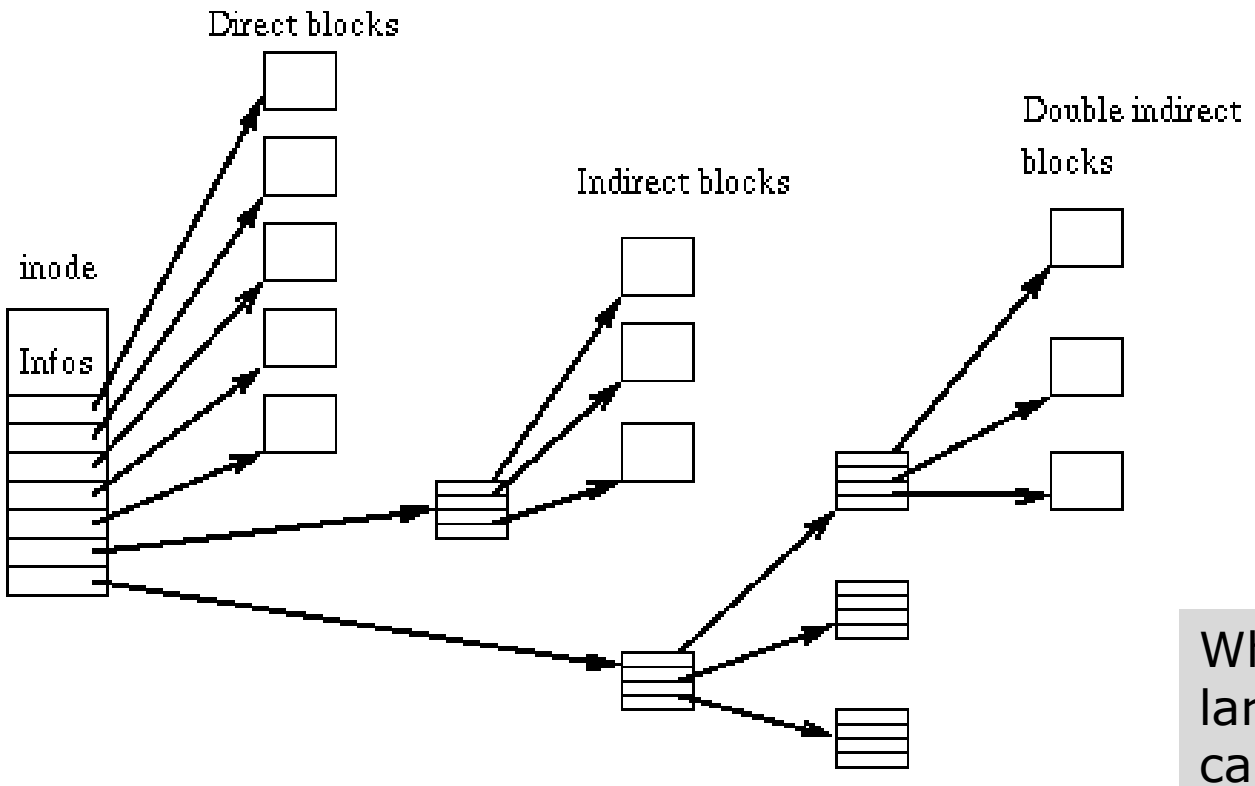
What is the largest file you can have with this approach?

total # pointers: 12 + (4096 / 4) = 1036

biggest file: 1036 * 4 KB ~ 4 MB

Getting a file's data: design 3

keep some direct pointers, one indirect pointer, and one double indirect pointer



What is the largest file you can have with this approach?

Size of files in a file system

from our text:

Most files are small	Roughly 2K is the most common size
Average file size is growing	Almost 200K is the average
Most bytes are stored in large files	A few big files use most of the space
File systems contains lots of files	Almost 100K on average
File systems are roughly half full	Even as disks grow, file systems remain ~50% full
Directories are typically small	Many have few entries; most have 20 or fewer

from hosting.csumb.edu:

min	0
max	650 MB
median	2.5 KB
mean	110 KB

Data structure for directories

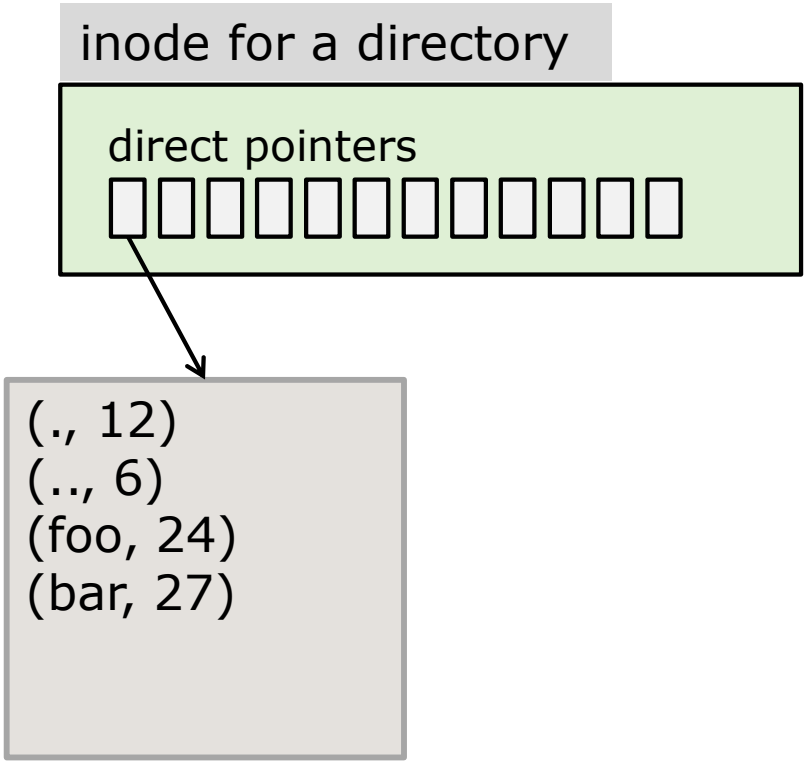
Each inode will include file type: “regular”, or “directory”

The data blocks for a directory contain a list of:

(name, inode number)

entries.

A special inode number (like 0) can be used when a file is deleted.



What is max number of directory entries? Assume 64 bytes per (name, i-num).

Multiple blocks are used for a large directory

Summary

We examined the structure of a simple file system.

- Raw material: a bunch of disk blocks
- The files system used these blocks for:
  - content (data blocks)
  - inodes
  - allocation information (bit maps)
  - the superblock