CS202-001 OS

Review Session 5 TA: Jinli Xiao

- [x] 0. Record + Attendence
- [x] 1. Background knowledge
- [x] 2. Lab 5 Overview
- [x] 3. Q & A

Note: in this whiteboard we are using 1-based index for inode. In lab 5 code we are using 0-based index for `filebno`.

---

1. Filesystems

$ cat ~/foo/bar.txt

Files:

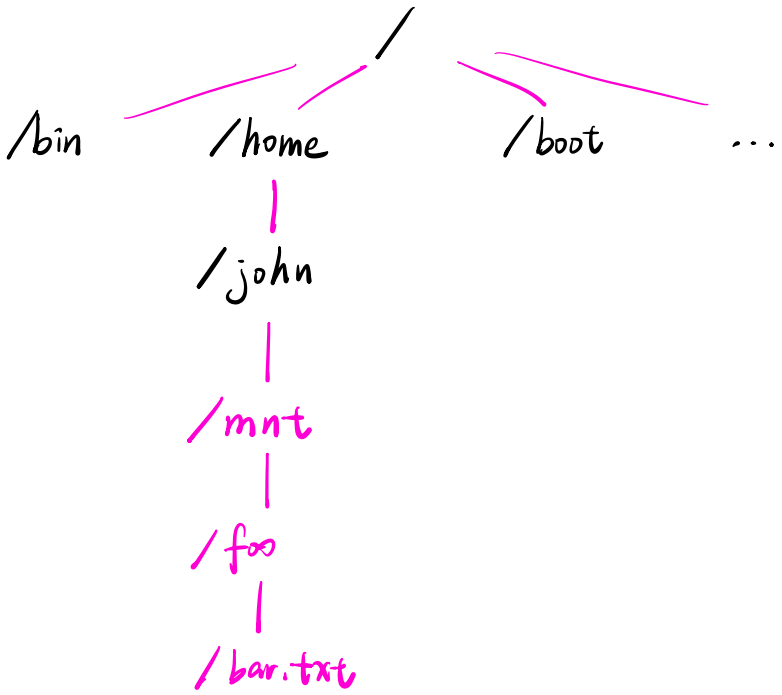
user's perspective: named bytes/data on hardware storage
filesystem's perspective: group of disk blocks

What is filesystem?

- a way of organizing & storing data on storage devices
- provides hierarchical structure for files & directories

Examples of storage devices?

USB/Flashdrive, Disk Images



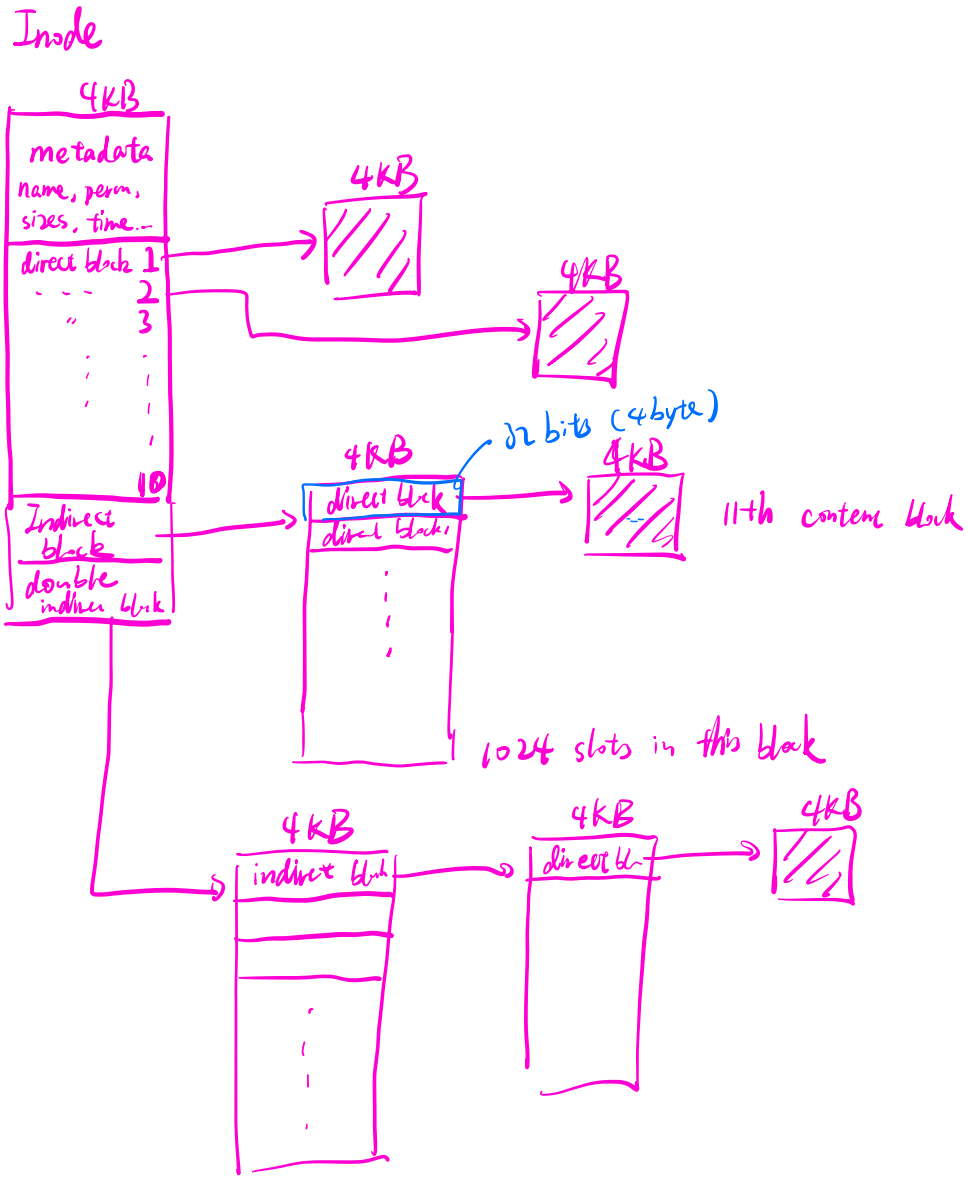
Key Abstractions

File: collection of data on the storage device
can contain text, images, videos, etc.

Filename: a string of characters used to identify & locate files

Directory: container of files (and directories) to help organize

Q: How do we implement this?



Q: What is a similar data structure we have seen?

pagetable

This sparse / imbalanced tree allows us to handle both small & large files.

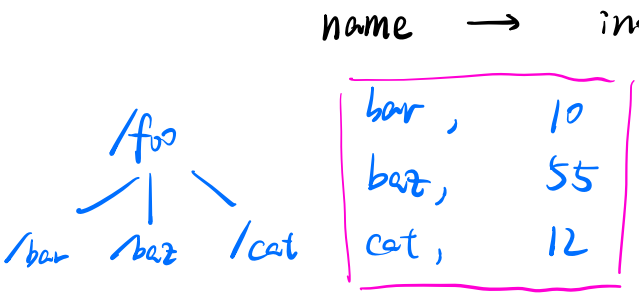
FS contains a fixed-size array of inodes. Each inode is indexed by a number, known as i-number.

Mental model:

<i-number> ⟶ inode

How about directories?

- It is implemented as a special type of file. The content contains a list of entries for the files and subdirectories within the directory.



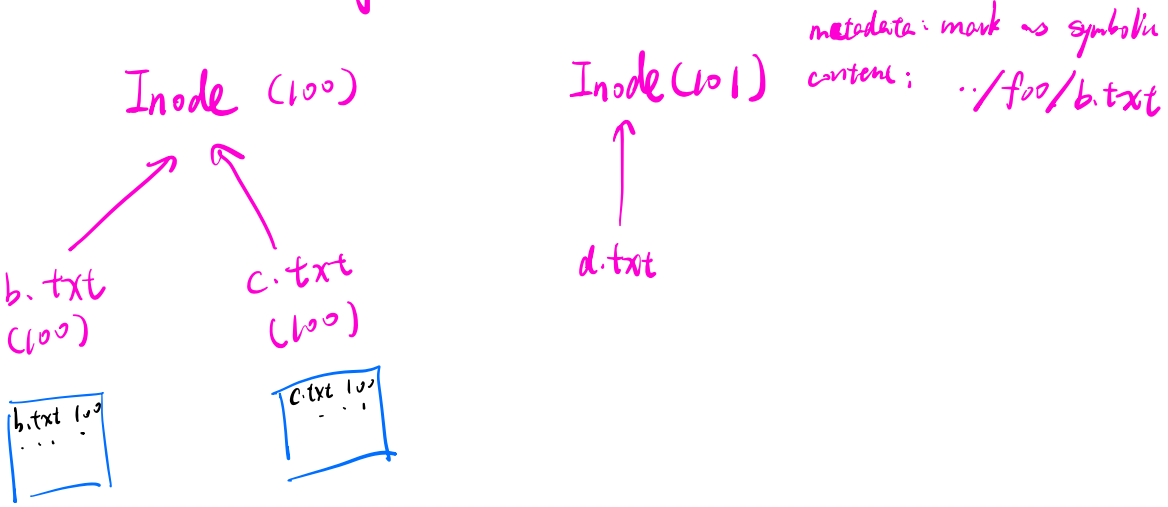
Hard Link v.s. Soft (symbolic) Link

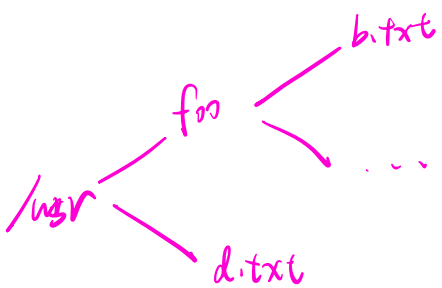
Hard Link:

- have the same inode # as the file
- cannot have hard links across filesystems
- cannot refer to directories

Soft Link:

- allocated its own inode. The content contains the path to the file it refers to.
- can refer to directories

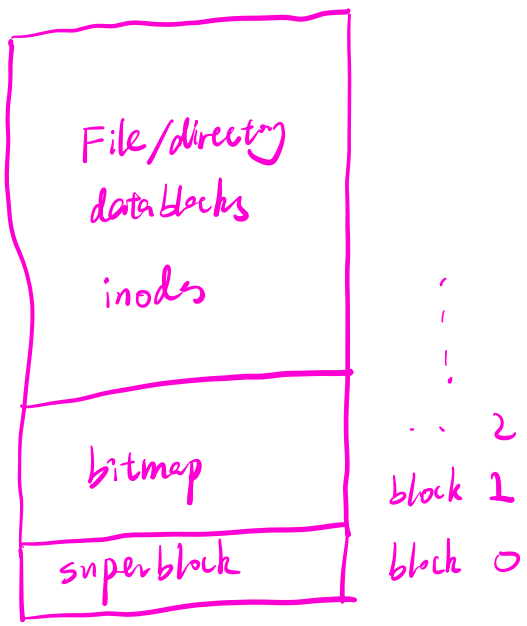




2. Lab 5 Overview

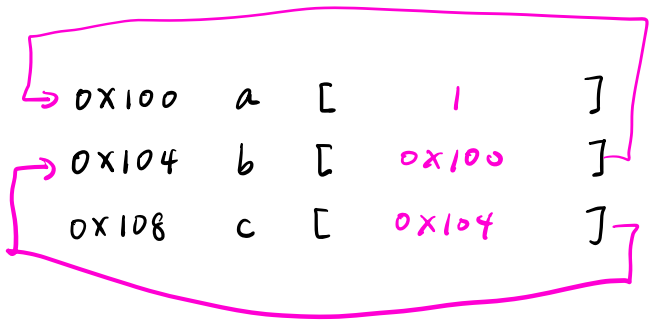
2.1 FS in Lab 5

- 1 region in which both inode & data block nodes resides
- Each inode is allocated its own disk block
- Each disk block is 4KB
- Superblock is block 0, holding metadata about the FS & ptr to the root directory.
- Bitmap: an array of bits 001...01111...0
  - 1: free to use
  - 0: already allocated
- Each inode includes
  - 10 direct ptrs
  - 1 indirect ptrs ⟶ 1024 direct blocks
  - 1 double indirect ptrs. ⟶ 1024 indirect blocks



2.2 Ptr-to-Ptr mental model

```
line 1   int a = 1;
line 2   int *b = &a;
line 3   int **c = &b;
```



2.3 Essential Functions

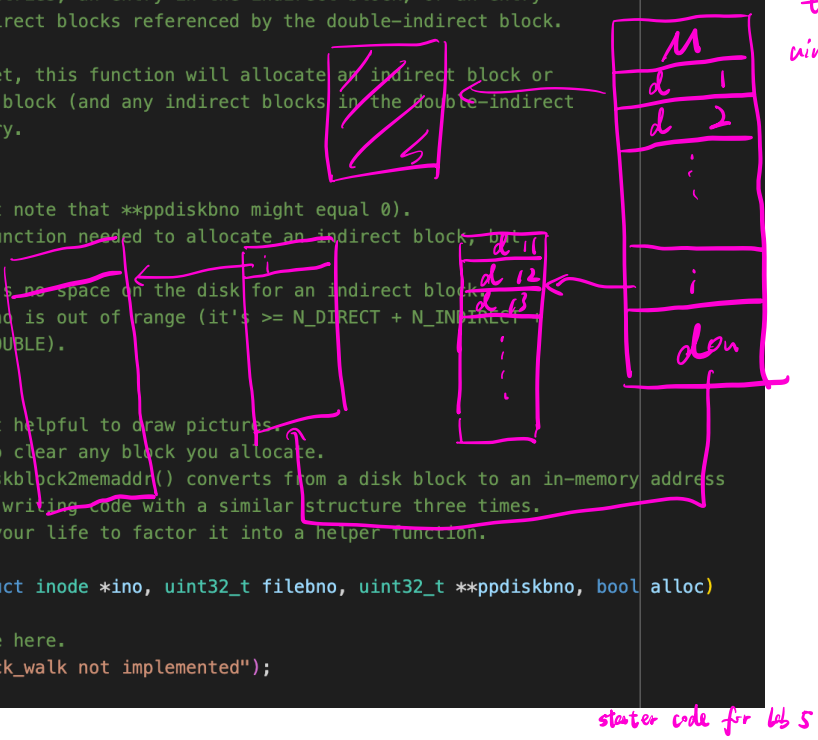
inode_block_walk

find the slot for the 'filebno'-th content block in the inode 'ino'. Allocate (double) indirect block(s) if necessary.

uint32_t **ppdiskbno; *ppdiskbno = addr;

type uint32_t *

```
13  // Find the disk block number slot for the 'filebno'th block in inode 'ino'.
14  // Set '*ppdiskbno' to point to that slot.  The slot will be one of the
15  // ino->i_direct[] entries, an entry in the indirect block, or an entry
16  // in one of the indirect blocks referenced by the double-indirect block.
17  //
18  // When 'alloc' is set, this function will allocate an indirect block or
19  // a double-indirect block (and any indirect blocks in the double-indirect
20  // block) if necessary.
21  //
22  // Returns:
23  //  0 on success (but note that **ppdiskbno might equal 0).
24  //  -ENOENT if the function needed to allocate an indirect block, but
25  //      alloc was 0.
26  //  -ENOSPC if there's no space on the disk for an indirect block.
27  //  -EINVAL if filebno is out of range (it's >= N_DIRECT + N_INDIRECT +
28  //              N_DOUBLE).
29  //
30  // Hints:
31  //  - You may find it helpful to draw pictures.
32  //  - Don't forget to clear any block you allocate.
33  //  - Recall that diskblock2memaddr() converts from a disk block to an in-memory address
34  //  - You may end up writing code with a similar structure three times.
35  //  It may simplify your life to factor it into a helper function.
36  int
37  inode_block_walk(struct inode *ino, uint32_t filebno, uint32_t **ppdiskbno, bool alloc)
38  {
39      // LAB: Your code here.
40      panic("inode_block_walk not implemented");
41  }
```



starter code for lab 5

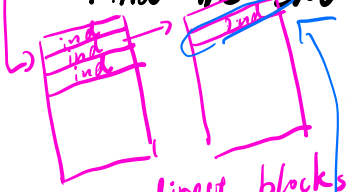
inode_get_block

find the 'filebno'-th content block in the inode 'ino'. Allocate content block if necessary.

Exercise:

- Find the slot for 5-th content block in inode

- Find the slot for 1032-nd content block in inode



1032 − 10 = 1022

direct blocks: 10

indirect blocks: 1024

2060
− 10
2050
− 1024
1026
− 1024
2

- Find the slot for 2060-th content block in inode

FUSE

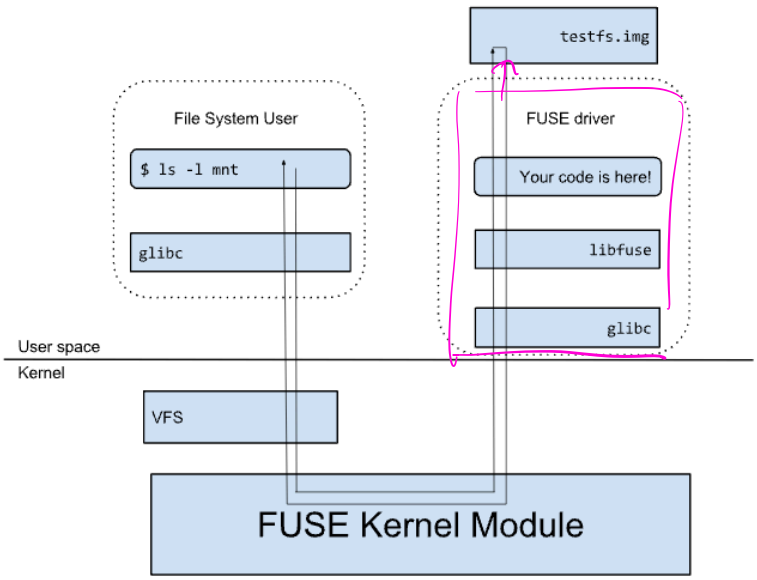


figure from lab 5 writeup

/cs202/lab5/mnt