Project-6: Unix-like File System Layer

Adem Efe Gencer

November 20, 2015

Announcements

- Project-5 was due November 18, 11:59 PM
- Prelim-2 is on November 24.
- Project-6 is due December 2, 11:59PM.
- Check Piazza for updates.

Project Scope

1. Implement a Unix-like file system layer: *ufsdisk*.
2. Use *free space bitmaps* to keep track of free and used blocks.
3. (Optional) Implement *a file system checker* (i.e. fsck) to check the integrity of your file system.

Recap: Intro

- File systems are built on one or more *block stores.*
- The block store abstraction provides:
  - a disk-like interface: read / write blocks
  - a sequence of blocks -- each typically a few kilobytes
- The block store abstraction doesn’t deal with:
  - file naming,
  - user permissions,
  - distinguishing files from directories,
  - …

Recap: Block Store Abstraction

- Simple interface:
  - block_t block
    - block of size BLOCK_SIZE
  - nblocks() -> integer
    - returns size of the block store in #blocks
  - read(block number) -> block
    - returns the contents of the given block number
  - write(block number, block)
    - writes the block contents at the given block number
  - setsize(nblocks)
    - sets the size of the block store

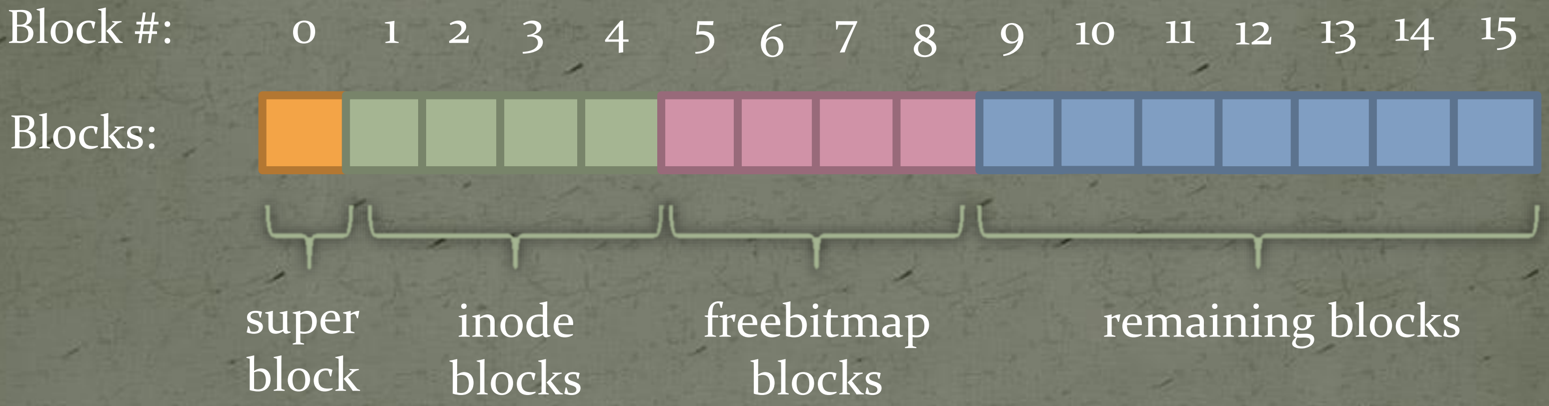
Recap: block_if.h

```
#define BLOCK_SIZE      512         // # bytes in a block
typedef unsigned int block_no;      // index of a block

struct block { char bytes[BLOCK_SIZE]; };
typedef struct block block_t;

typedef struct block_if *block_if;
struct block_if {
    void *state;
    int (*nblocks)(block_if bif);
    int (*read)(block_if bif, block_no offset, block_t *block);
    int (*write)(block_if bif, block_no offset, block_t *block);
    int (*setsize)(block_if bif, block_no size);
    void (*destroy)(block_if bif);
};
```

A Unix-like file system layer: ufsdisk

- The underlying block store is partitioned into:
  - A superblock:
    - At block #0.
  - A fixed number of inodeblocks:
    - From block #1 to #inodeblocks.
    - The #inodeblocks is given in superblock.
  - A fixed number of freebitmapblocks:
    - From #inodeblocks+1 to (#inodeblocks+#freebitmapblocks).
    - The #freebitmapblocks is stored in superblock.
  - Remaining blocks:
    - Datablocks, free blocks, indirect blocks.

Ufsdisk: layout



Block #: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Blocks:

super block

inode blocks

freebitmap blocks

remaining blocks

Ufsdisk: superblock (1 per underlying blockstore)

```
struct ufs_superblock {
        unsigned int magic_number;        // magic number of ufsdisk
        block_no n_inodeblocks;           // # ufs_inodeblocks
        block_no n_freebitmapblocks;      // # freebitmap blocks
};
```

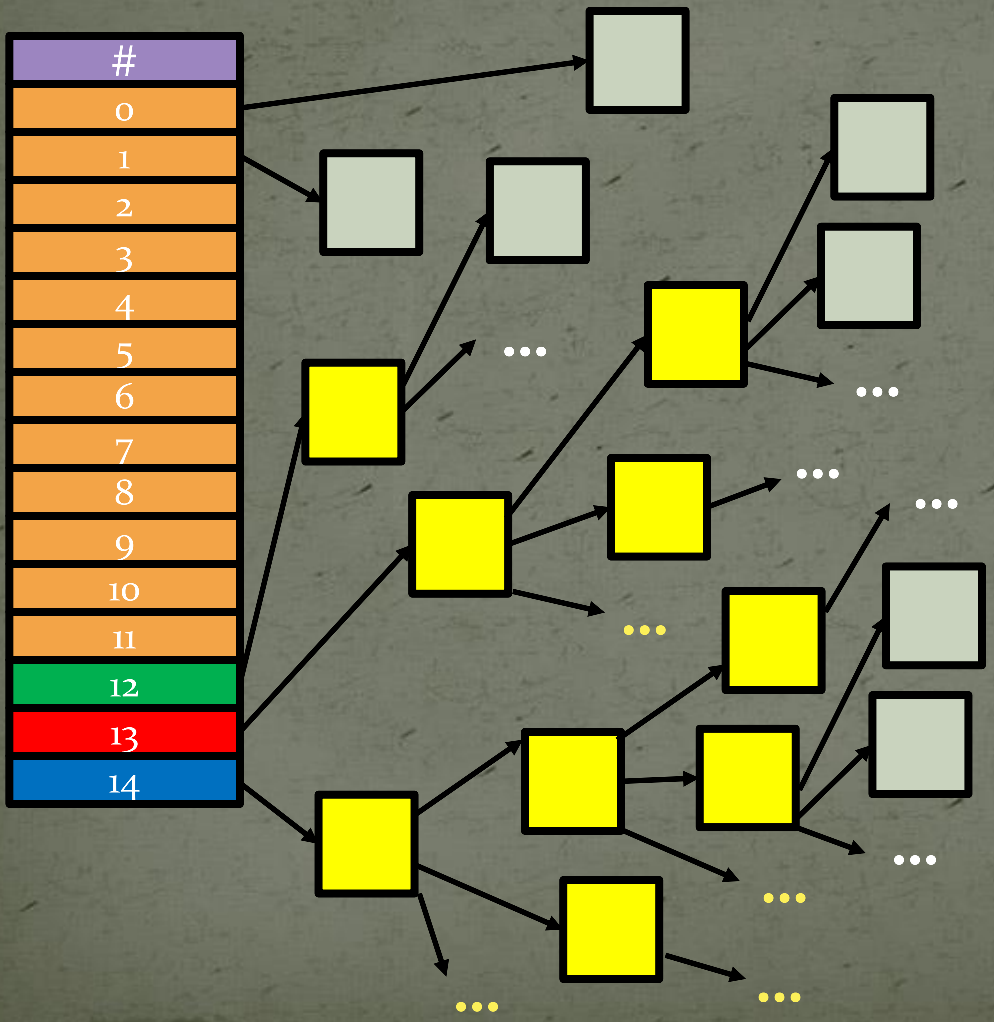
Ufsdisk: inodeblock

```
#define INODES_PER_BLOCK          (BLOCK_SIZE / sizeof(struct ufs_inode))

struct ufs_inodeblock {
        struct ufs_inode inodes[INODES_PER_BLOCK];
};
```

Ufsdisk: inode (1 per virtual blockstore)

```
#define REFS_PER_INODE              15

struct ufs_inode {
    block_no nblocks;    // total size of the file
    block_no refs[REFS_PER_INODE];
};
```

Ufsdisk: inode (1 per virtual blockstore)



KEY

⬜ :Data block

🟨 :Indirect block

🟪 :Number of blocks

🟧 :Direct pointer

🟩 :Indirect pointer

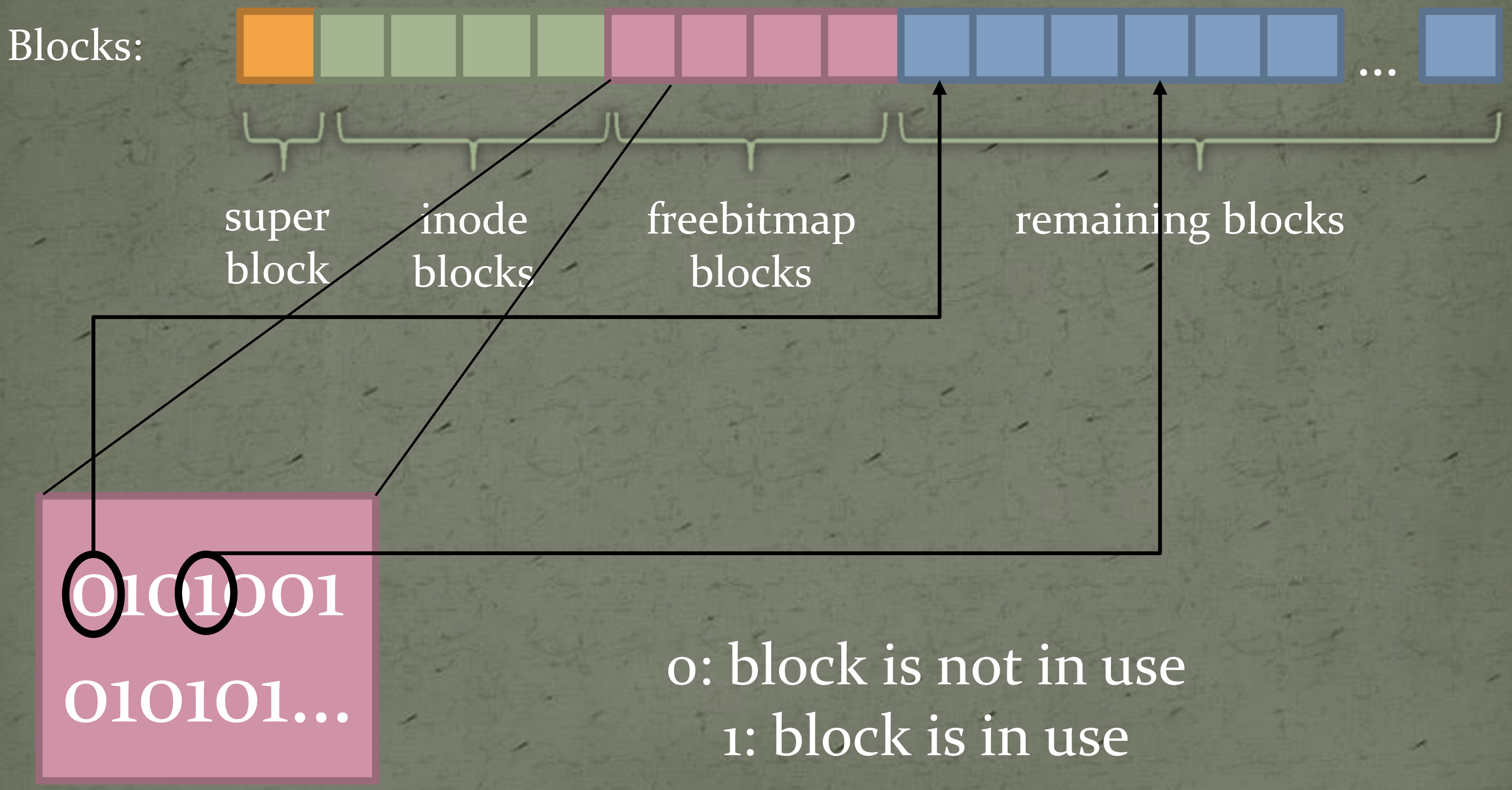
🟥 :Doubly Indirect pointer

🟦 :Triply Indirect pointer

… :Other data blocks

… :Other indirect blocks

Free space bitmaps



0: block is not in use
1: block is in use

Ufsdisk: freebitmap blocks

- Each freebitmap block: a block of bits.
- How many freebitmap blocks, *f*, do I need?

$$\boldsymbol{f} = \left\lceil \frac{nblocks - 1 - \left\lceil \frac{n\_inodes}{INODES\_PER\_BLOCK} \right\rceil}{1 + BLOCK\_SIZE * 2^3} \right\rceil$$

File system checker

- Verifies the consistency of your filesystem – e.g. try fsck (UNIX), chkdsk (Windows).
- If system crashes, filesystem may be corrupted.
- Checks filesystems -- and repairs fixable issues if broken.
  - a datablock is in use but marked as free.
  - a particular block is both an indirblock and datablock.
  - a particular datablock has been used more than once.
  - other issues…

General hints…

- See treedisk_chk.c code for an example file system checker.
- Use the skeleton code:
  - ufsdisk.c
  - ufsdisk.h
- Makefile:
  - Add ufsdisk.o to the list of OBJECTS.

Concluding thoughts

- This is the **last** project!
- Begin early so you have time to study for the finals.
- Come see TAs in office hours.
- Use Piazza: read / post.