Project-6: Unix-like File System Layer

Kai Mast

May 5th 2017

(original slides by Efe Gencer)

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
  - …

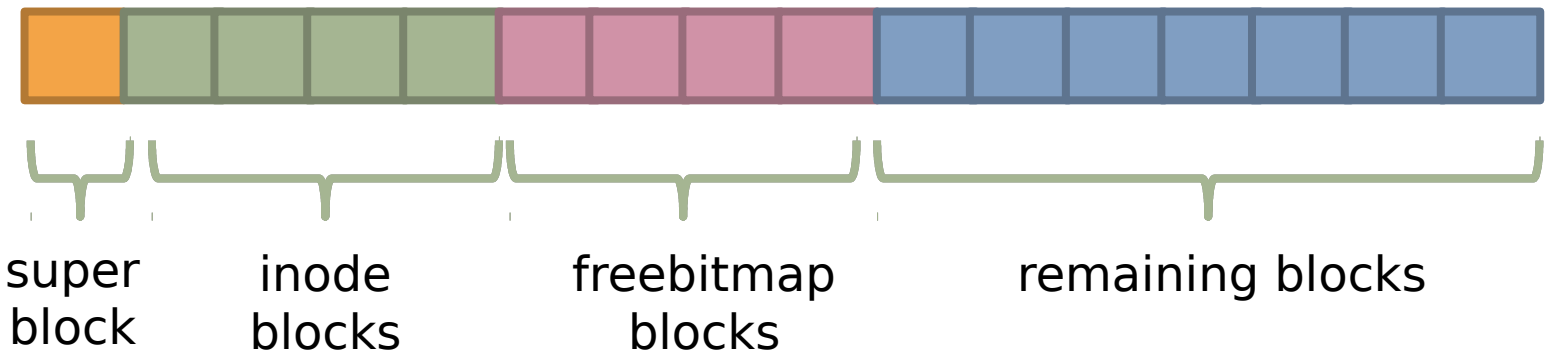
Recap: Block Store Abstraction

- 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);
};
```

Ufsdisk: layout



Ufsdisk: superblock (1 per underlying blockstore)

```
struct ufs_superblock {
    // magic number of ufsdisk
    // identifies the filesystem
   unsigned int magic_number;

   block_no n_inodeblocks;
   block_no n_freebitmapblocks;
};
```

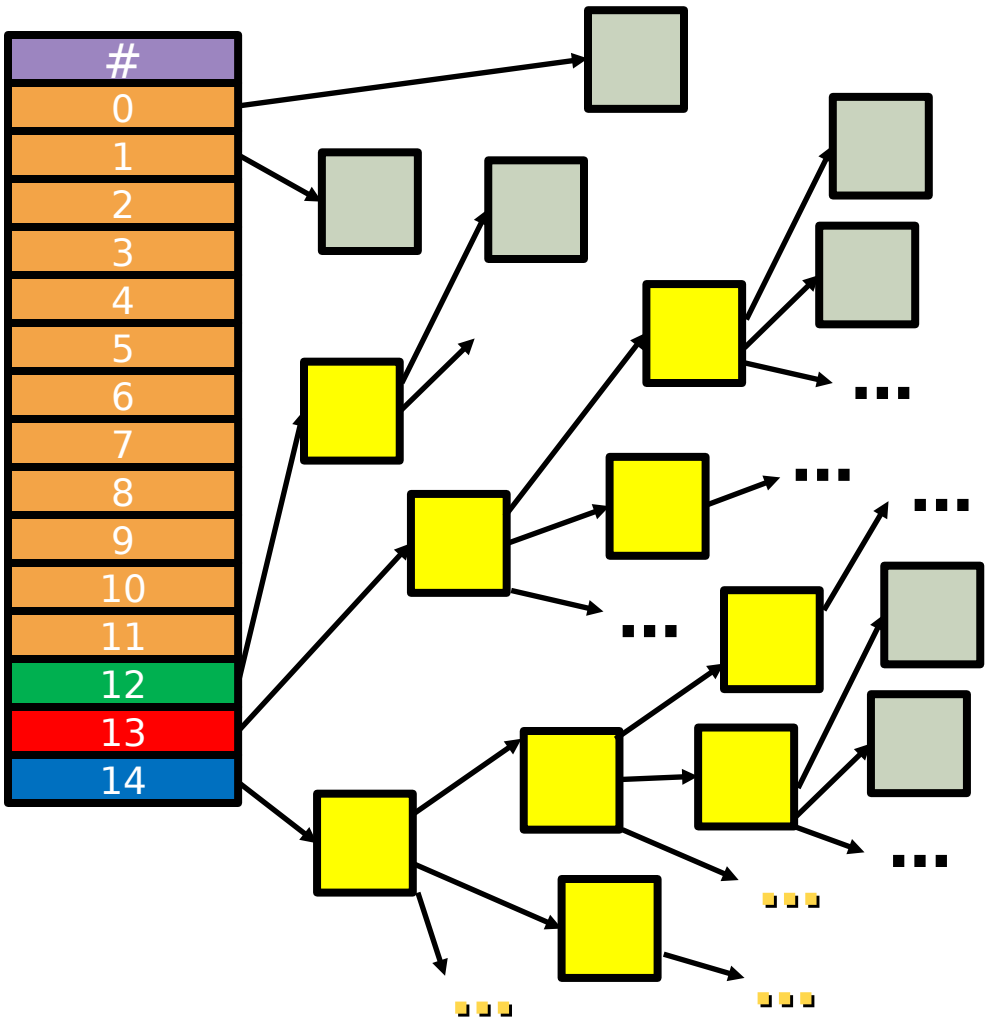
Ufsdisk: inodeblock

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

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

Ufsdisk: inode (1 per virtual blockstore)

```
#define REFS_PER_INODE   15

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

Ufsdisk: inode
(1 per virtual blockstore)



Legend

:Data block

:Indirect block

:Number of blocks

:Direct pointer

:Indirect pointer

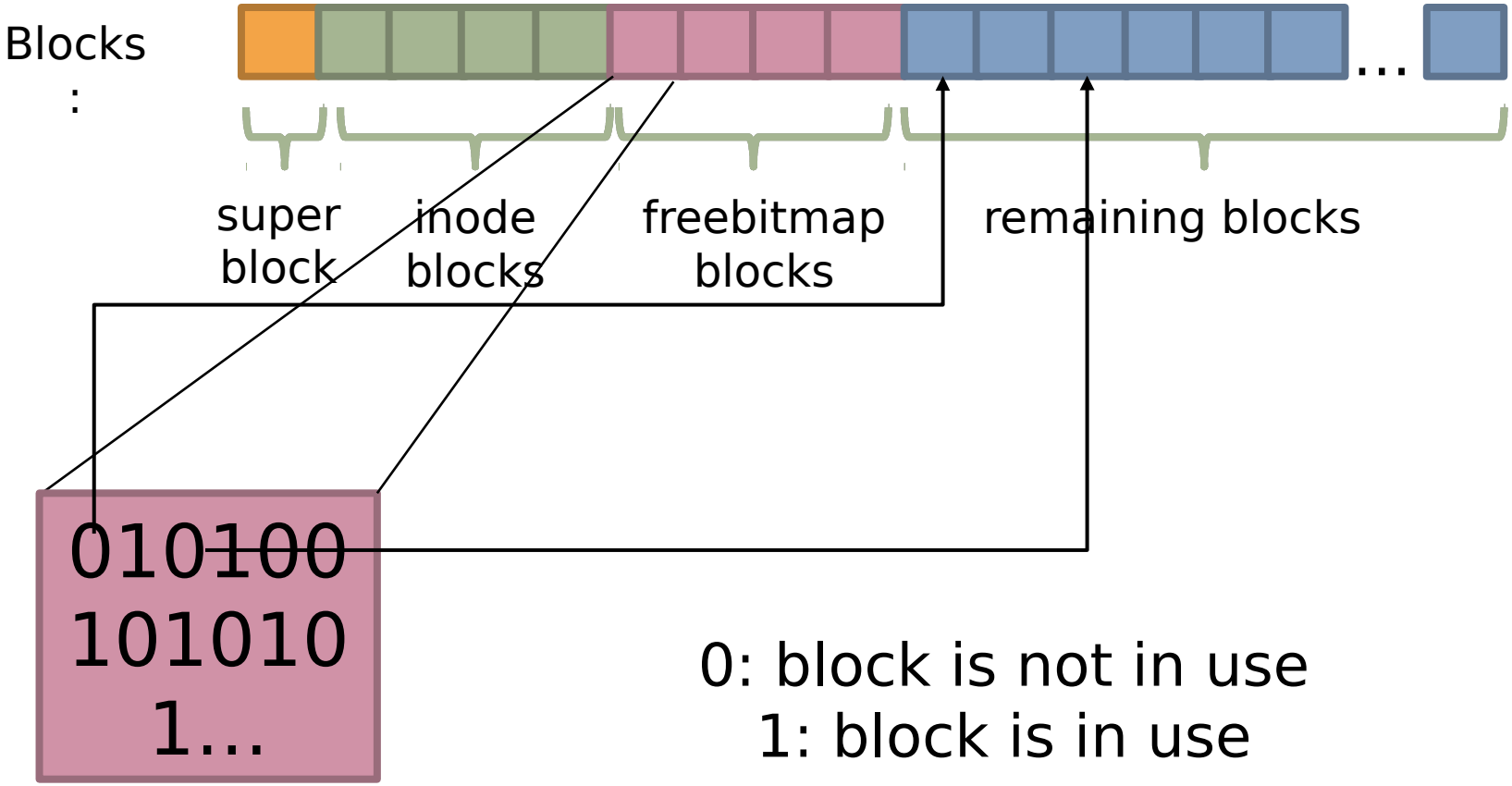
:Double Indirect pointer

:Triple Indirect pointer

:Other data blocks

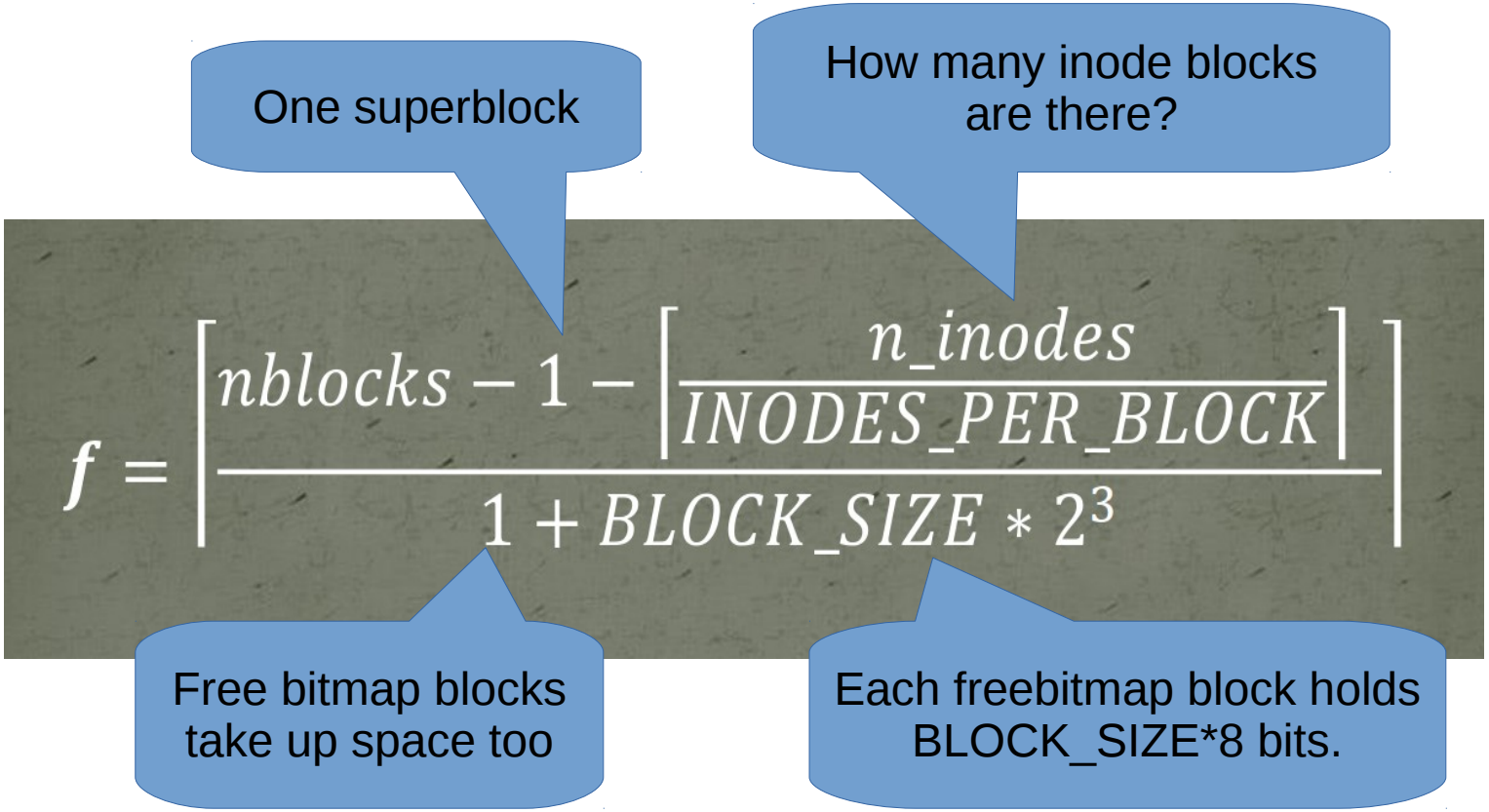
:Other indirect blocks

Free space bitmaps



Ufsdisk: freebitmap blocks

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



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…

Questions?

- This is the **last** project!
- Begin early so you have time to study for the finals.