Project 6

Unix File System

Administrative

- No Design Review
  - A design document instead
    - 2 pages max
- No collaboration with peers
  - Piazza is for clarifications
- Due on January 13 (Dean's Date) @ 5 pm
- All virtual office hours

Overview

- Implement a UNIX-like file system
  - In a 1MB file!
    - Determined by FS_SIZE
    - Pages are 512 bytes, so 2,048 sectors

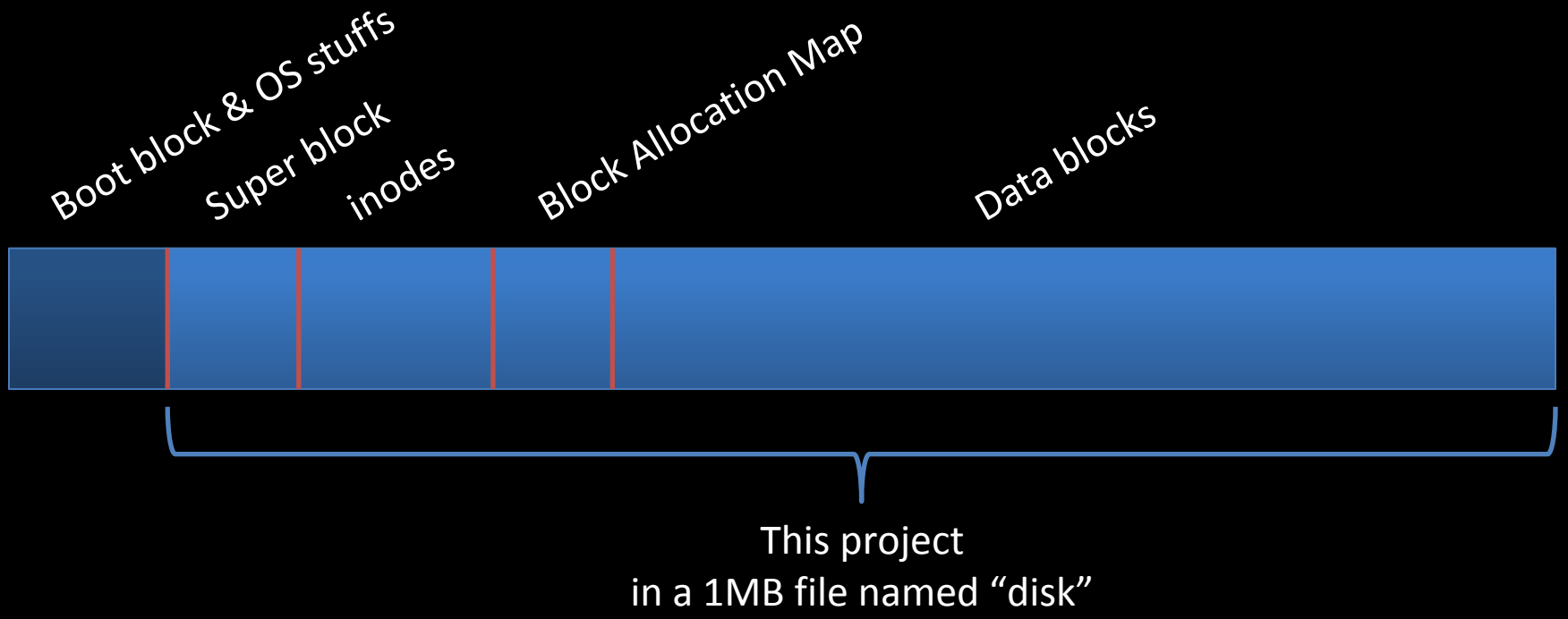
API

- Disk format
- File
  - open, close, read, write, seek
  - link & unlink (which is how you delete a file)
  - stat
- Directory operations
  - make, remove
- Shell
  - ls & chdir (cd)

Don’t Worry About

- Permissions
  - Access Control Lists (ACLs)
- Concurrency

Disk Layout



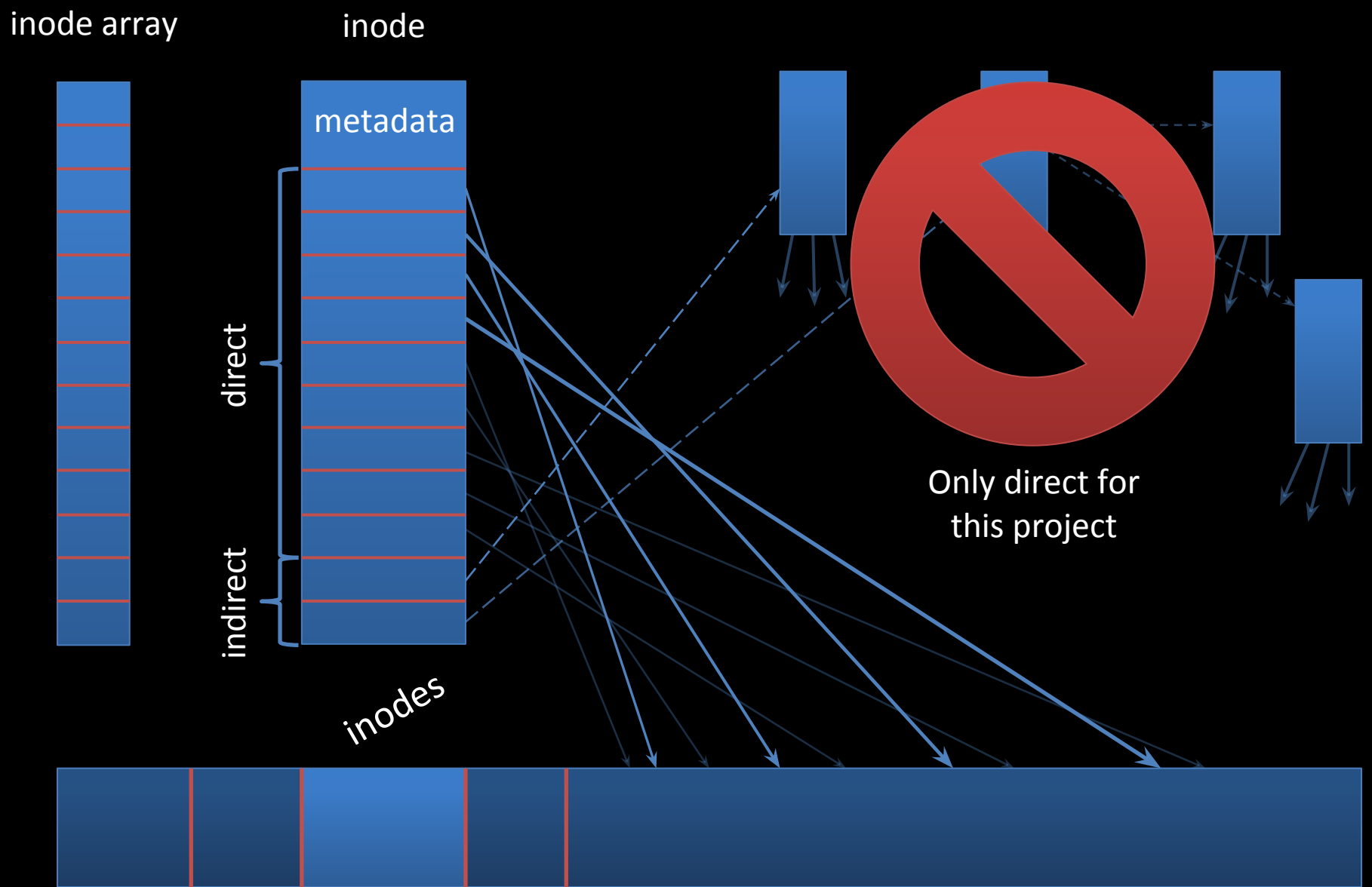
Space between divisions is not representative of actual size

Contains metadata about the disk

Examples:

- Size
- # of inodes
- # of data blocks
- Where inodes start
- Where data blocks start

Where do I go to find …?



inode Metadata

Examples

- File or directory?
- Link count
- Size
- Etc..

fs_init

- A “constructor” for the FS code
- block_init

fs_mkfs

- “Makes” a file system
  - Writes the super block
  - Mark inodes and data blocks to be free
  - Create root directory
  - Initialize file descriptor table

File Creation & Deletion

- fs_open(), fs_link(), fs_unlink()
- open: create a new file if it does not exist
- link: hard link to a file
  - create a link to an existing file
  - hard vs. soft (symbolic) link?
- unlink:
  - delete a file if link count == 0
  - delete directory entry
  - files that are still open

File Access

- open: open existing file (allocate file descriptor)
- read: read bytes from open file
- write: write bytes to open file
- lseek: change position in file
- close: free file descriptor

fs_lseek() Semantics

- Our fs_lseek() only takes two arguments
  - fd, offset
- Unix lseek() takes three
  - fd, offset, whence
- Whence: SEEK_SET, SEEK_CUR, SEEK_END
- Our fs_lseek() assumes SEEK_SET
- What if lseek() wants to seek past the end of the file?

Directories, part 1

- Like a file: List of files and directories
  - Name to inode number mapping
- Can read it like a file
  - Use existing file I/O functions to do directory manipulation
- Always has at least two entries
  - “..” parent directory
  - “.” current directory

Directories, part 2

- mkdir: make a directory
  - create an entry in parent directory
  - create two directories: “..” and “.”
- rmdir: remove directory if empty
- cd: change the current directory
  - For relative path names only

Example fs_mkdir

```
int fs_mkdir(char *file_name)
{
    if (file_name exists) return ERROR;
    /* allocate inode */
    /* allocate data blocks */
    /* set directory entries for “..” & “.” */
    /* set inode entries appropriately */
    /* update parent */
    return SUCCESS
}
```

FAQ

Absolute Path names

- Do we need to support absolute path names?
  - I.e., when I am in “/foo”, do I need to support: chdir /bar/tmp/?
- No
  - You would have to support absolute path names everywhere: open, link, mkdir, etc.

Removing a Directory

- Do I need to support recursive directory removal?
  - I.e., remove all the subdirectories and files contained in a directory I am going to delete?
- No
  - Return an error if directory is not empty

File System Check (fsck)

- Useful for debugging
- Verifies integrity of:
  - Superblock magic number
  - Block allocations
  - Inode allocations
  - Block Allocation Map
  - Directory content
  - Etc.

Implementing the Project

- In Linux
  - Uses a file to simulate a disk
  - Code is provided
  - ./lnxsh
- Shell supports
  - System calls for File System
  - Commands like “ls”, “cat foo”, “create foo200”
- 500-1,000 lines of code

Testing

- A python script to use in testing will be provided
- Multiple tests that each
  - Execute the shell
  - Open an existing file system (or format a new one)
  - Write commands to shell (cat foo)
  - Read output from shell (ABCDEFGHIJKL)
  - Exit
- You should write your own test cases
  - Submit them with your code

Design Document

- Sections on project page
- I like diagrams
- PDF
- Submit together with your implementation
- 2 pages