

CSE 451: Operating Systems

Section 8

Project 2b wrap-up, ext2, and
Project 3

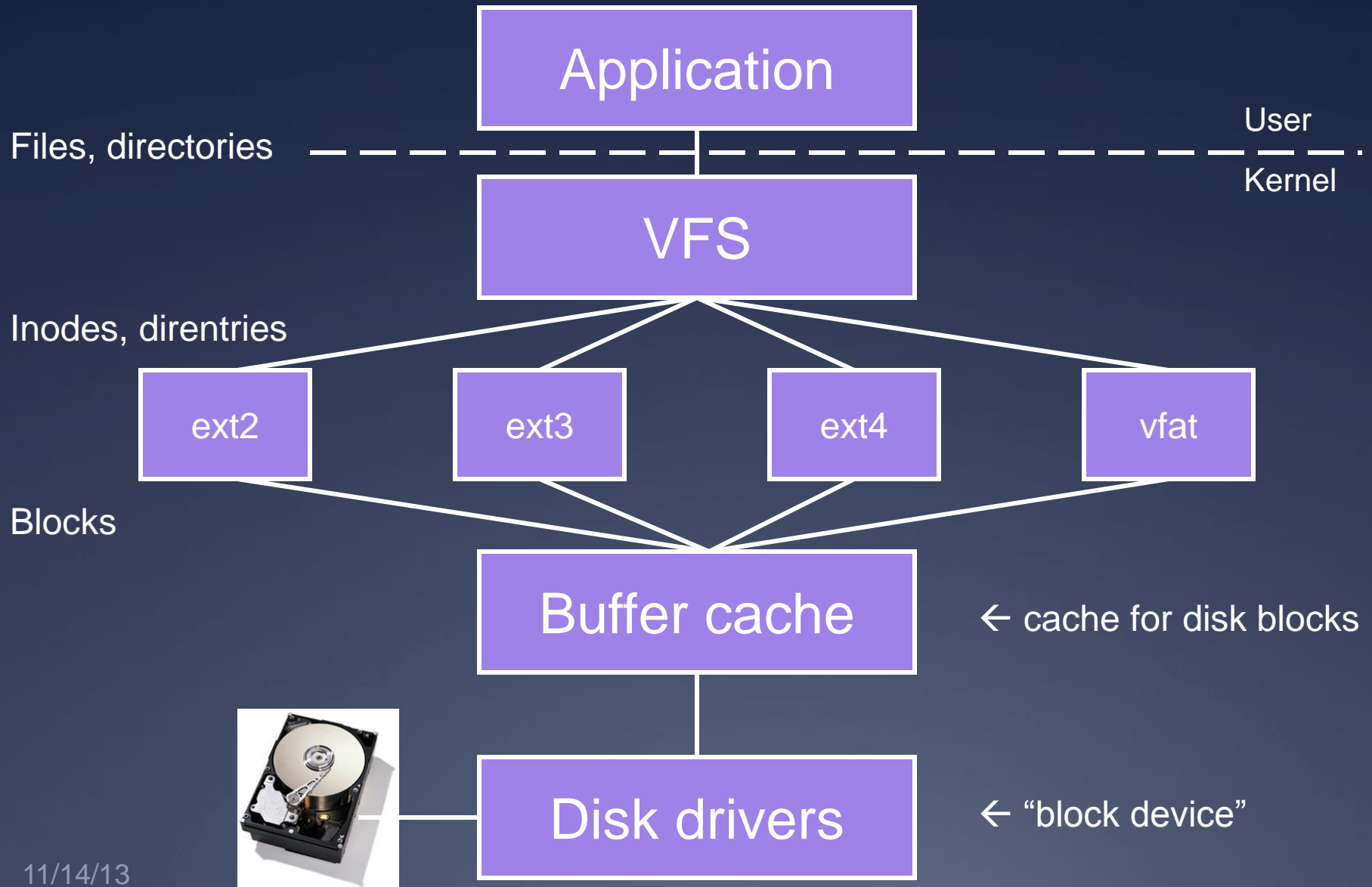
Project 2b

- * Make sure to read thoroughly through the requirements for the writeup in part 6 and answer every question
- * There are multiple ways of measuring throughput that you should discuss
 - * Responses/second
 - * Bytes transferred/second (average throughput per client and total average throughput)
- * Any lingering questions?

Project 3

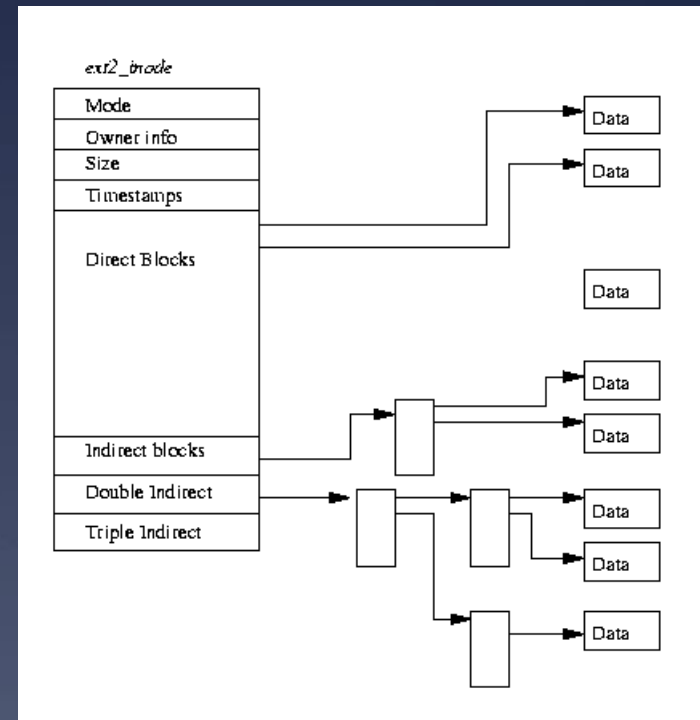
Background FS info

Linux file system layers



Inodes

- * Inode: a structure maintaining all metadata about a file (or directory)
 - * Inode number (unique ID of inode)
 - * Permissions, timestamps
 - * Pointers to *data blocks*
- * Inode does *not* contain: name of file
 - * Where is it actually stored?
 - * One or more file names can point (link) to the same inode. When will this occur?



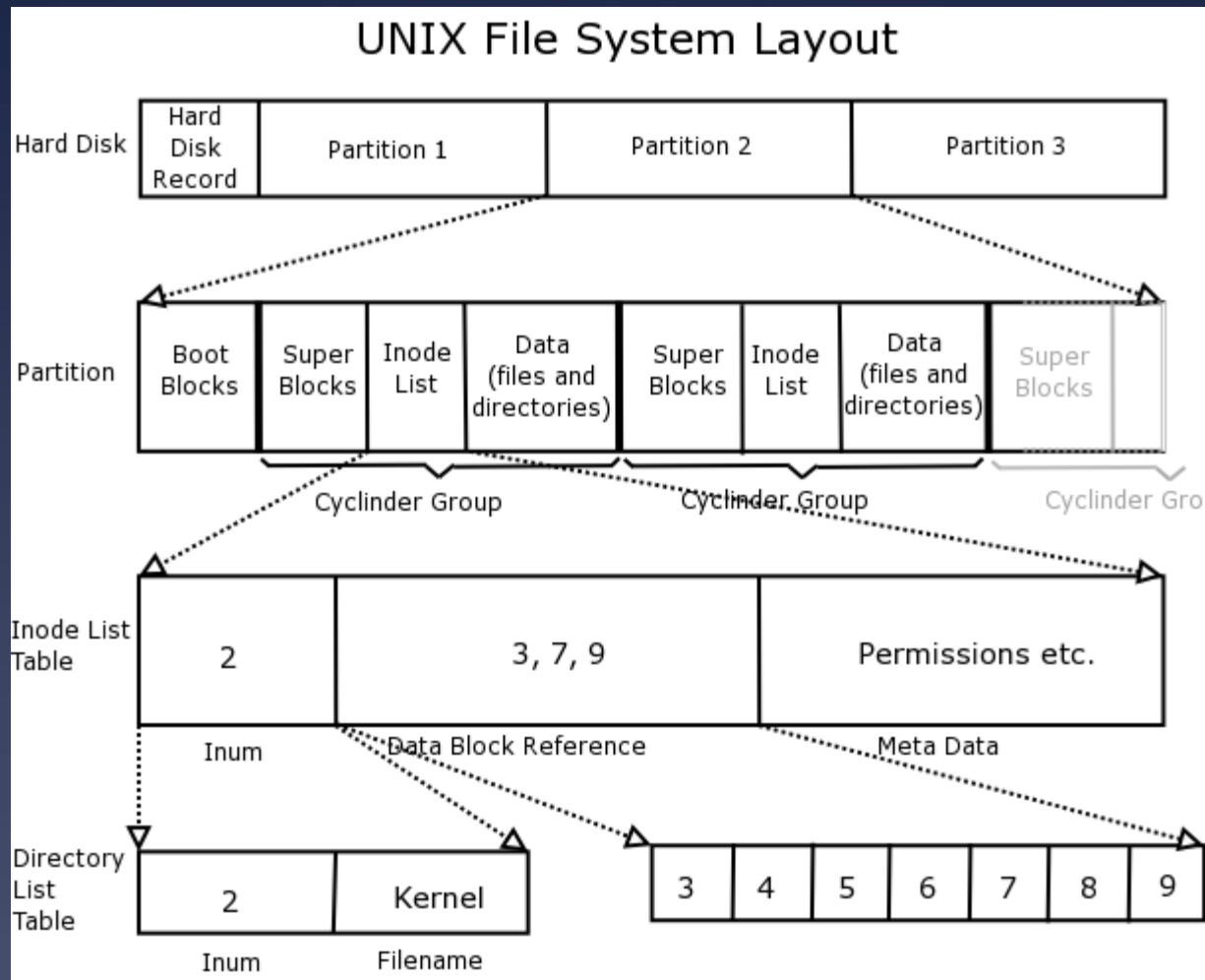
Inode structure

- * Remember, inodes themselves are stored in blocks
 - * What's the size of the inode struct?
 - * So how many inside a 1K block?
- * Max number of inodes (max number of files) usually decided when file system is formatted
 - * mkfs heuristic: create an inode for every three or four data blocks

Directories

- * Directory entry (“dirent”): stores the file inode number, file name, and file type
 - * Directory entries are stored in data blocks
- * Directory: A list of directory entries
 - * An inode with a directory i_mode attribute (check LINUX_S_ISDIR()) stores dirents in its data blocks

ext2 organization



Superblock

- * Superblock always starts at byte 1024
- * Master filesystem structure in ext2
- * Stores global filesystem constants:
 - * Block size
 - * Inode size
 - * Number of blocks
 - * Number of inodes
 - * ...and much more
- * Do not hardcode filesystem constants into your code! Use superblock information instead.

Block groups

- * Block groups store:
 - * A copy of the superblock (why?)
 - * The block group descriptor table
 - * Immediately proceeds the superblock
 - * Contains the block numbers of the block bitmap, inode bitmap, and inode table among other things
 - * A block bitmap (used vs. free blocks)
 - * An inode bitmap (used vs. free inodes)
 - * An inode table (the inodes themselves)
 - * The actual data blocks

Data blocks

- * Blocks for regular files contain file data
- * Blocks for directories contain directory entries:

```
#define EXT2_NAME_LEN 255
struct ext2_dir_entry_2 {
    __u32 inode;    /* Inode number */
    __u16 rec_len; /* Directory entry
                    length */
    __u8  name_len; /* Name length */
    __u8  file_type;
    char  name[EXT2_NAME_LEN]; /* File
                                name */
};
```

Data block for /

Dir. entry	Field	Value
0	Inode	1
	Name	"/"
1	Inode	1
	Name	"/.."
2	Inode	2
	Name	"etc"
3	Inode	3
	Name	"bin"
4	Inode	0
	Name	0

Example data block usage

* For a 4MB file system with 1KB blocks, with hierarchy:

```
/
  etc
    passwd
    fstab
  bin
    sh
    date
```

File/Directory	Size	Data Blocks
/	4 entries + 1 null entry	1
/etc	4 entries + 1 null entry	1
/bin	4 entries + 1 null entry	1
/etc/passwd	1024 bytes	1
/etc/fstab	100 bytes	1
/bin/sh	10,000 bytes	10
/bin/date	5,000 bytes	5
	Total:	20

For more ext2 reading

- * A master reference is available at <http://www.nongnu.org/ext2-doc/ext2.html>
- * Some other helpful resources:
 - * http://homepage.smc.edu/morgan_david/cs40/analyze-ext2.htm
 - * <http://eecs.wsu.edu/~cs460/cs560/ext2fs.html>
 - * Wikipedia also has a decent explanation: http://en.wikipedia.org/wiki/Ext2#ext2_data_structures

Project 3: Undelete

- * Out: Friday 11/17 once we have it ready
- * Due: Wednesday 12/4 at 11:59pm
- * Same groups you've been with previously
- * Some serious understanding is required, so read, discuss with your teammates, read some more, discuss, plan, then execute

Project 3: Undelete

- * Your task: recover deleted files in ext2 file systems
- * Implement this as a loadable kernel module
- * How is this possible?
 - * Even if inode links are removed, inodes and data might still be present
 - * Make a best attempt at recovery of lost files—some are corrupted and beyond hope, so you won't be able to recover them

Project 3: Undelete

- * Tools at your disposal:
 - * Header files in `linux/fs/ext2`
 - * Kernel functions for reading/writing files and exploring file system trees
 - * Skeleton code for the kernel module
 - * A utility for creating and mounting ext2 filesystems of various sizes
 - * A program for printing out block information for an ext2 filesystem file

Tips

- * The filesystem creation tool requires at least 60 1kB blocks or it will fail
- * Think carefully about how to tell whether an inode is deleted. (Hint: you'll need to use the inode bitmap)
- * Do not hardcode any ext2 constants. Use only those provided in headers and those from the superblock
- * You are permitted to keep only a small fixed number of inodes in memory at once (otherwise recovery of large files would be infeasible)

Tips

- * Jeff and I will give out some additional test files, but you should also create your own sample file systems using the provided tool
- * Make sure to restore the accessed and modified times of files as well as their contents
- * Test file systems with indirect data blocks
- * Test your code by restoring file systems with things like large deleted JPGs that are easy to check (visually) for corruption

Tips

- * If your group emails a plan of your approach to the project to Jeff and me by class next Wednesday 11/20, we will review it and give you feedback
 - * Take advantage of this; it will save you a lot of grief leading up to the deadline
 - * Writing a plan is a great way to force yourself to learn the concepts

Questions?