

Section 8:

File Systems

Project 3

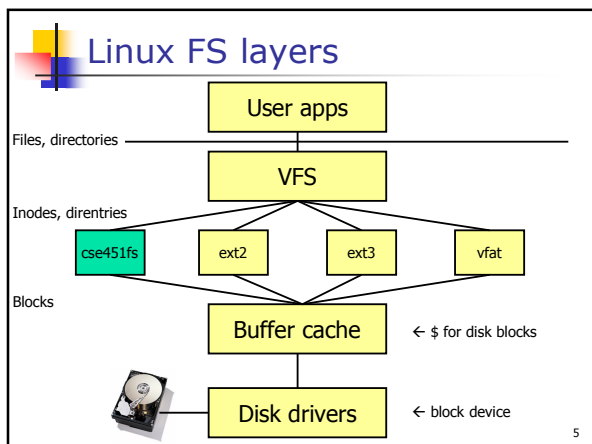
1

Questions?

2

- ## Project 3
- Work with a real file system
 - Given:
 - cse451fs: simplified file system for Linux
 - Goals:
 - Understand how it works
 - Modify implementation to:
 - Increase maximum size of files (currently 13KB)
 - Allow for longer file names (currently 30 chars)
- 3

- ## Project 4 Setup
- Build a kernel module for cse451fs
 - Transfer it to VMware
 - On VMware
 - load the cse451fs module
 - format the file system using (modified) mkfs tool
 - mount your file system
 - Test using tools like ls, cat, etc. (see last slides for gotchas)
 - Step 1: try this procedure with given code
 - Step 2: read cse451fs.h, then dir.c
- 4



- ## File systems in Linux
- Organize blocks in a block device into files and directories
- Core concepts:
- **Inodes and inode numbers**
 - Inode = structure maintaining all metadata about a file (or directory), *except for name*
 - Inode number = unique ID of inode
 - One or more file *names* can point (link) to the same inode
 - Why do we need inode numbers? Can't we use absolute paths as IDs?
- 6

File systems in Linux

- Organize blocks in a block device into files and directories

Core concepts:

- Inodes and inode numbers**
 - Inode = structure maintaining all metadata about a file (or directory), *except for name*
 - Inode number = unique ID of inode
 - One or more file *names* can point (link) to the same inode
 - Inode numbers provide location independence
- So where do we store file names then?

7

File systems in Linux

- Organize blocks in a block device into files and directories

Core concepts:

- Inodes and inode numbers**
 - Inode = structure maintaining all metadata about a file, except for name
 - Inode number = unique ID of inode
 - One or more file *names* can point (link) to the same inode
 - Inode numbers provide location independence
- Directory entry**
 - = pair (name, inode number)
 - A directory is *just a file*, whose contents is a list of directory entries

8

File system disk structure

- What types of things do we need to store in a file system?

9

cse451fs disk structure

boot	superblock	data map	inode blocks	data blocks
------	------------	----------	--------------	-------------

- Superblock:** tells where all other things are
 - Contains **inode map**:
 - Bit array, tracks which inodes are currently in use
 - Contains parameter values (e.g., block size)
- Data map:**
 - Bit array, tracks which data blocks are in use
- Inode blocks:**
 - Contains all inodes (i.e., metadata for files) stored here
- Data blocks:**
 - Contains data of files / directories

10

cse451fs structure

1	1	1	85	4008
boot	superblock	data map	inode blocks	data blocks

```

struct cse451_super_block {
1365  __u16 s_nInodes;           // inode map is tail of superblock
2    __u16 s_nDataMapStart;   // block # of first data map block
1    __u32 s_nDataMapBlocks;  // data map size, in blocks
3    __u32 s_nInodeStart;    // block # of first inode block
85   __u32 s_nNumInodeBlocks; // number of blocks of inodes
88   __u32 s_nDataBlocksStart; // block # of first data block
4008 __u32 s_nDataBlocks;    // number of blocks of data
7    __u32 s_nBusyInodes;    // number of inodes in use
0x451f __u16 s_magic;        // magic number
      unsigned long s_imap;    // name for inode map
};
  
```

Sample values for a 4MB disk with 4 files and 3 dirs using 1K blocks

11

Inode structure

```

#define CSE451_NUMDATAPTRS 10

struct cse451_inode {
  __u16 i_mode;           ← determines if file or dir
  __u16 i_nlinks;        (+ protection)
  __u16 i_uid;
  __u16 i_gid;
  __u32 i_filesize;
  __u32 i_datablocks[CSE451_NUMDATAPTRS];
};
  
```

12

Inode structure

```
#define CSE451_NUMDATAPTRS 10
struct cse451_inode {
    __u16 i_mode;           ← determines if file or dir
    __u16 i_nlinks;        (+ protection)
    __u16 i_uid;
    __u16 i_gid;
    __u32 i_filesize;
    __u32 i_datablocks[CSE451_NUMDATAPTRS];
};
```

- What's the size of the inode struct?
- Multiple inodes per block
 - How many for 1K block?
- mkfs decides how many inodes to create, using heuristic
 - create an inode for every three data blocks
- In general, the max number of inodes (so of files) is decided at FS formatting time

13

Data blocks

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

```
#define CSE451_MAXDIRNAMELENGTH 30
struct cse451_dir_entry {
    __u16 inode;
    char name[CSE451_MAXDIRNAMELENGTH];
};
```

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

- Data block for / directory containing:
 - .. etc bin

14

Sample data block usage

For a 4MB file system with 1KB blocks

```

. /
  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

15

Project 4 requirements

- **Increase maximum sizes of files**
 - Be **efficient for small files** but allow large files
 - Changing constant (=10) is **not enough!**
 - Come up with a better design/structure for locating data blocks
 - E.g., indirect blocks?
 - You **don't** have to support arbitrarily large files
 - Fine to have constant new_max (but new_max >> old_max)
- **Allow for longer file names**
 - Be **efficient for short files** names but allow large file names
 - Again, *don't just change the constant*

16

Approaches for longer file names

- Store long names in a separate data block, and keep a pointer to that in the directory entry.
 - Short names can be stored as they are.
 - Recommended
- Combine multiple fixed-length dir entries into a single long dir entry (win95)
 - It is easier if the entries are adjacent.
- Put a length field in the dir entry and store variable length strings
 - need to make sure that when reading a directory, that you are positioned at the beginning of an entry.

17

Getting started with the code

- Understand the source of the limits in the existing implementation
 - Look at the code that manipulates dir entries
 - mkfs code
 - dir.c in the file system source code
- Longer file names:
 - The code for will largely be in dir.c: add_entry() and find_entry()
 - In mkfs, change how the first two entries (for "/" and "/..") are stored
- Bigger files:
 - super.c:get_block()
 - References to i_datablock[] array in an inode will have to change

18

Linux Buffer Manager Code

- To manipulate disk blocks, you need to go through the buffer cache
- Linux buffer cache fundamentals:
 - blocks are represented by `buffer_heads`
 - Just another data structure
 - Actual data is in `buffer_head->b_data`
 - For a given disk block, buffer manager could be:
 - Completely unaware of it
 - no `buffer_head` exists, block not in memory
 - Aware of block information
 - `buffer_head` exists, but block data (`b_data`) not in memory
 - Aware of block information and data
 - Both the `buffer_head` and its `b_data` are valid ("\$ hit")

19

Accessing blocks

- To read a block, FS uses `bread(...)`:
 - Find the corresponding `buffer_head`
 - Create if doesn't exist
 - Make sure the data is in memory (read from disk if necessary)
- To write a block:
 - `mark_buffer_dirty()` + `brelse()` - mark buffer as changed and release to kernel (which does the writing)

20

Tool limitation warning

- Some stuff in linux kernel is limited to 256 chars
 - e.g. VFS, ls
 - Be careful when testing long filenames!
- `dd` is useful for creating large test files
 - `dd if=/dev/zero of=200k bs=1024 count=200`
- `df` is useful to check you're freeing everything correctly

21

Gcc warning

- gcc might insert extra space into structs
 - How big do you think this is?

```
struct test { char a; int b; }
```
 - Why is this a problem?
 - What if `test` represents something you want on disk?
 - e.g. directory entries
 - Discrepancy between the disk layout and memory layout
 - Fix:

```
struct test2 {
    char a;
    int b;
} __attribute__((packed));
```
 - `sizeof(test2)` is now 5

22