CSE 451
Fall 2003

Section
11/20/2003

Questions from lecture

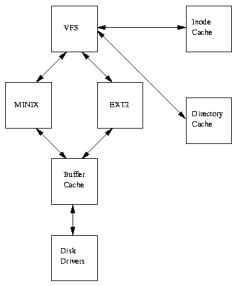
Questions from homework

- Subdirectories as files
- Implementing a referenced bit

Questions from project

Info on next project

- Extending a file system



File systems in linux

- Implement standard interface
  - file_operations
    - read/write/seek files
    - read directory
  - inode_operations
    - create / lookup / unlink / mkdir / rmdir / rename
  - super_operations
    - read/write inodes
  - address_space_operations
    - readpage/writepage for memory-mapped IO
  - file_system_operations
    - read in superblock

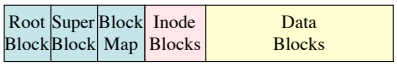
FS storage

- File system is layered on top of a block device
  - Device provides ordered list of blocks
  - Blocks are cached in the *buffer cache*
- File systems access blocks through:
  - getblk() - gets a cached block
  - bread() - reads a block
  - mark_buffer_dirty() / brelse - marks buffer as changed and releases to kernel (which does the writing)

FS overview

- File system layout created in mkfs
  - initializes on-disk data structures
    - superblock
    - datablock map - bitstring of in-use data blocks
    - empty root directory (just links for self & parent)
- Superblock contains file system parameters
  - e.g. block size, # of blocks, # of inodes, inode of root directory
- Changing on-disk layout requires changing mkfs
  - setup_tables()
  - make_root_dir()

cse451_fs layout



- Inodes contain:
  - mode
  - nlinks
  - uid/gid
  - filesize
  - array of block numbers for data
- Directories are:
  - array of:
    - inode # (0 == empty)
    - char name[30]
  - # of entries set by file size

File Layout

- Inode contains array of blocks for the file
- File size is limited by fixed number of fixed size blocks
- # of files is limited by # of inodes

File System Operation

- On load:
  - super.c:cse451_read_super() loads FS structures off disk
- create a file:
  - dir.c:cse451_create() creates directory entry
- lookup file:
  - dir.c:cse451_lookup() scans directory for file name
- read/write
  - uses memory mapped I/O: mmap.c: cse451_readpage(), cse451_writepage()
  - calls super.c:get_block() to read/write a specific block of a file

What you have to do

- Fix at least 2 limitations
  - Increase name length from 30 characters
  - Increase number of files from 8000
  - Increase max file size from 13 kb

File Systems Advanced Topics

- Problems
  - Corruption
    - If you create a file, and you crash before adding it to a directory, what happens?
    - If you add the file to the directory, and crash before creating the file itself, what happens?
    - If you free a block in the bitmap before you delete the file inode, what happens?

Old Fashioned Solution

- fsck (unix) or chkdsk (windows)
  - Walk through the entire file system and check for consistency:
    - All files are in directories
    - All blocks marked busy are in files
    - All directory entries point to files
  - Clean It up!
    - move files without directories to a standard place
    - free up unused blocks
    - etc.
- Problem: slow

Journaling

- Problem stems from interrupting operations
  - .e.g doing 1/2 of a file create or delete
- Solution: do what databases do
  - Write to a log what you are *going* to do
  - Then do it
  - Then write down that you did it
- On a crash
  - Just check what hasn't been finished yet and finish it

Example:

- create file foo/bar
  - allocate blocks for foo
  - allocate inode for foo
  - write to directory entry in bar
  - write data for bar

Drawbacks

- Need to store journal somewhere and write to it before every metadata operation
  - requires additional seeks
  - Can be avoided by delaying metadata writes for a while and only writing to journal

Storage Systems

- Scalability is a big problem today
  - How do you build a file system for 100 TB?
    - Lots of disks
    - Too much I/O for one computer to handle
      - (e.g. PCI bus max bandwidth is 133 megabytes/sec)
- Approaches:
  - Network-attached storage
  - Storage area networks

Network attached storage

- Get rid of the computer
- Put the file system in the disk drives
- Benefits:
  - Removes bottleneck of a server operating system
  - Optimized just for serving up files
  - Used for users / single-machine applications accessing shared data
- Drawbacks
  - Still limited to one machine for a filesystem

Storage area networks

- Put the disk drives on the network directly
- Have computers read & write blocks remotely
- Benefits:
  - Can scale to huge numbers of disks / huge bandwidth
  - Used for clusters accessing shared data
- Drawbacks:
  - Computers must coordinate file system operations to avoid conflicts at the block level
  - Computers must be trusted with access to blocks - file system security is not applied