File System Implementation

Jonathan Geisler

March 22, 2006

Taylor University
CSS

Layers in file system

- Applications
- Logical file system (VFS entry)
- File organization module
- Basic file system (VFS exit)
- Device driver
- Physical device

On-disk data structures

- Boot block
- Partition information
- Directory structure
- FCBs for each file

In-memory data structures

- Partitition information
- Directory structure
- System-wide open files table
- Per-process open files table

In-memory data structures

- Partitition information
- Directory structure
- System-wide open files table
- Per-process open files table

Why do we have these?

File Control Block

- Permissions
  - Owner and group
  - ACLs
- Access/creation/modification dates
- Size
- Physical location

Virtual File System

- Why bother?
- How does it work, logically?
- Notice the book calls this object oriented
  - This existed in the early 80's long before C++
  - This is implemented in Linux, FreeBSD, Solaris, etc. in C!

Directories

Linear List

- Pros
  - Simple
  - Usually already sorted (for listings)
- Cons
  - Inefficient searches, or
  - Insertions & Deletions cumbersome

Hash Table

- Pro: Fast lookup
- Cons
  - Collisions
  - Rehashing

Allocating physical disk space

- Contiguous
  - Great for sequential access
  - Easy for random access
  - Suffers from external fragmentation
  - Hard to determine allocation size *a priori*
- Linked
  - Fine for sequential access
  - Horrible for random access
  - FAT uses this method

Indexed allocation

- The UNIX way
- Fine for sequential access
- Fine for random access
- Needs modification for large files

Free-space management

- Bit vector
  - Pros: Simple & Efficient hardware to find first empty block
  - Con: Too much memory for large (normal?) disk sizes
- Linked list
  - Pro: Minimizes memory usage
  - Con: List traversal expensive
- Grouping
- Counting

Performance tweaks

- Caching
  - Separate page and buffer caches
  - Unified
- Asynchronous writes
- Free-behind/read-ahead

Data recovery

- Microsoft: `diskscan` or `chkdsk`
- Traditional UNIX: `fsck`
- Newer OSes: journaled filesystems

NFS protocols

- Mount
- File operations

NFS file operations

- Normal operations like `read()` and `write()`
- No `open()` or `close()`
- Path translation done on per directory basis

How does this file system work?