Slide 1

Administrivia

- Homework 6 on the Web; due a week from today.
- (Review minute essay from last time.)

Slide 2

Virtual File Systems

- Apparently many possibilities for implementing filesystem abstraction, with the usual tradeoffs. Do we have to choose one, or can different types coexist? The latter . . .
- In Windows, having different filesystems on different logical drives is managed via drive letters.
- In UNIX, current approach is usually a “virtual file system” — basically, an extra layer of abstraction (remember the adage about how that can solve any programming problem).

Slide 3

Filesystem Performance

- Access to disk data is much slower than access to memory — seek time plus rotational delay plus transfer time.
- So, file systems include various optimizations . . .

Slide 4

Improving Filesystem Performance — Caching

- Idea — keep some disk blocks in memory; keep track of which ones are there using hash table (base hash code on device and disk address).
- When cache is full and we must load a new block, which one to replace?

  Could use algorithms based on page replacement algorithms, could even do LRU accurately — though that might be wrong (e.g., want to keep data blocks being filled).
- When should blocks be written out?
  - If block is needed for file system consistency, could write out right away. If block hasn't been written out in a while, also could write out, to avoid data loss in long-running program.
  - Two approaches: "Write-through cache" (Windows) — always write out modified blocks right away. Periodic "sync" to write out (UNIX).

Slide 5

Improving Filesystem Performance — Block Read-Ahead

- Idea — if file is being read sequentially, can read some blocks “ahead”. (Of course, doesn’t help if file is being read non-sequentially. Decide based on recent access patterns.)

Slide 6

Improving Filesystem Performance — Reducing Disk Arm Motion

- Group blocks for each file together — easier if bitmap is used to keep track of free space. If not grouped together — “disk fragmentation” may affect performance.
- If i-nodes are being used, place them so they’re fast to get to (and so maybe we can read an i-node and associated file block together).

Slide 7

Disk Fragmentation

- Idea — if blocks that make up a file are (mostly) contiguous, faster to read them all. If not, “disk fragmentation”.
- How likely is disk fragmentation? Depends on filesystem, strategy for allocating space for files.
- “Defragmenter” utility can be run to correct it. Windows comes with one. Linux doesn’t. The claim is that UNIX and Linux filesystems typically don’t become fragmented unless the disk is close to full.

Slide 8

Example Filesystem — Unix V7

- Filename restriction — each part of path name at most 14 characters.
- So, directory entry is just 14-byte name and i-node number.
- I-nodes are all stored in a contiguous array at the start of the file system (right after boot block and a “superblock” containing additional parameters).
- What’s in each i-node? attributes (permission bits, numeric owner and group ID, timestamps, links count) and list of blocks — last is pointer to more blocks.
- To find a file:
  - Start with root directory — its i-node is in a known place.
  - Scan directory for first part of path, get its i-node, read it, scan for next part of path, etc.
  - Relative path names are handled by including “.” and “..” in each directory, so no special code needed.

Slide 9

UNIX Filesystems — Hard Links versus Symbolic Links

- As mentioned previously, many filesystems provide a mechanism for creating not-strictly-hierarchical relationships among files/folders. UNIX typically has two:
  - "Hard" links allow multiple directory entries to point to the same i-node.
  - "Soft" (symbolic) links are a special type of file containing a pathname (absolute or relative).
- (Why two? Good question. Compare and contrast . . .)

Slide 10

Minute Essay

- What are the advantages/disadvantages of hard links versus symbolic links? are there things one can do that the other can't? are there ways in which one can go wrong and the other can't?
- This wraps up the planned lectures on filesystems. Anything you'd like to hear more about?

Slide 11

Minute Essay Answer

- Hard links can only point to files within the same filesystem. Soft links get around that limitation but can “break” if the file they point to is removed or renamed.