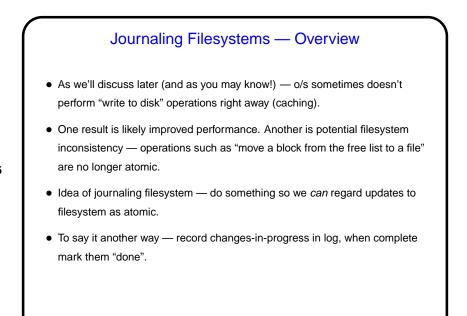


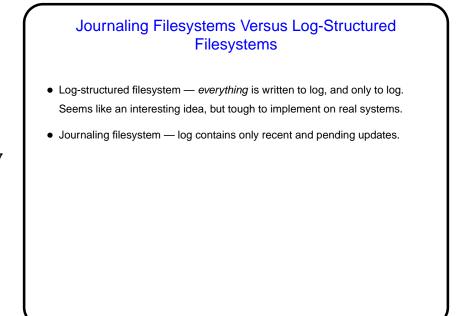
Filesystem Reliability — Consistency Checks Can easily happen that true state of filesystem is represented by a combination of what's on disk and what's in memory — a problem if shutdown is not orderly. Solution is a "fix-up" program (UNIX fsck, Windows scandisk). Kinds of checking we can do: Consistency check: For each block, how many files does it appear in (treating free list as a file)? If other than 1, problem — fix it as best we can. File consistency check: For each file, count number of links to it and compare with number in its i-node. If not equal, change i-node. Etc., etc. — see text.



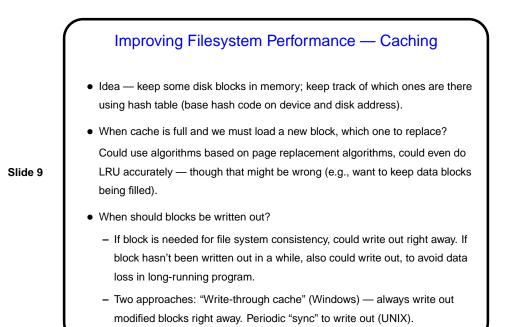
Journaling Filesystems, Continued
Can record "data", "metadata" (directory info, free list, etc.), or both.
"Undo logging" versus "redo logging":

Undo logging: First copy old data to log, then write new data (possibly many blocks) to disk. If something goes wrong during update, "roll back" by copying old data from log.
Redo logging: First write new data to log (i.e., record changes we're going to make), then write new data to disk. If something goes wrong during update, complete the update using data in log.

A key benefit — after a system crash, we should only have to look at the log for incomplete updates, rather than doing a full filesystem consistency check.



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 …

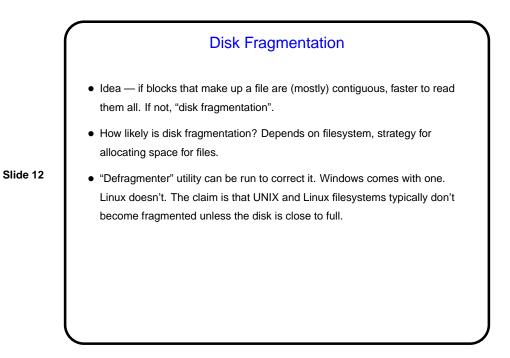


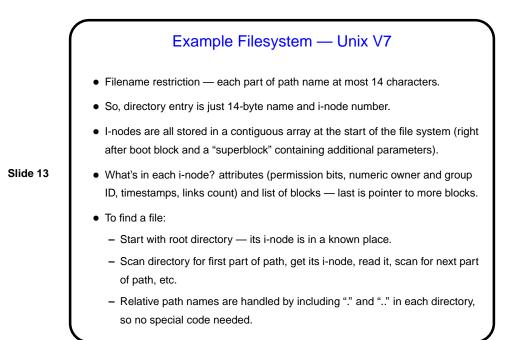
Improving Filesystem Performance — Block **Read-Ahead**

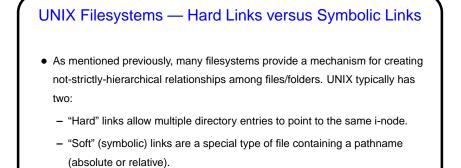
• Idea — if file is being read sequentially, can read some blocks "ahead". (Of course, doesn't help if file is being read non-sequentially. Decide based on recent access patterns.)



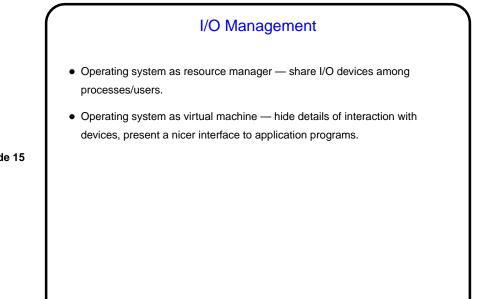
- Group blocks for each file together easier if bitmap is used to keep track of free space. If not grouped together — "disk fragmentation" may affect performance.
- Slide 11
- If i-nodes are being used, place them so they're fast to get to (and so maybe we can read an i-node and associated file block together).







• (Why two? Good question. Compare and contrast ...)



| | I/O Hardware, Revisited |
|----------|---|
| | First, a review of I/O hardware — simplified and somewhat abstract view, mostly focusing on how low-level programs communicate with it. |
| | Many, many kinds of I/O devices — disks, tapes, mice, screens, etc., etc. Can be useful to try to classify as "block devices" versus "character devices". |
| Slide 16 | Many/most devices are connected to CPU via a "device controller" that manages low-level details — so o/s talks to controller, not directly to device. |
| | Interaction between CPU and controllers is via registers in controller (write to tell controller to do something, read to inquire about status), plus (sometimes) data buffer. |
| | Example — parallel port (connected to printers, etc.) has control register (example bit — linefeed), status register (example bit — busy), data register (one byte of data). These map onto the wires connecting the device to the CPU. |

