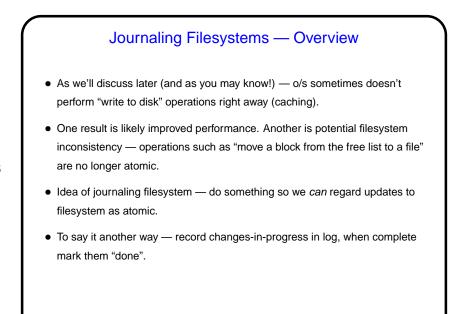


Filesystem Implementation — Directories

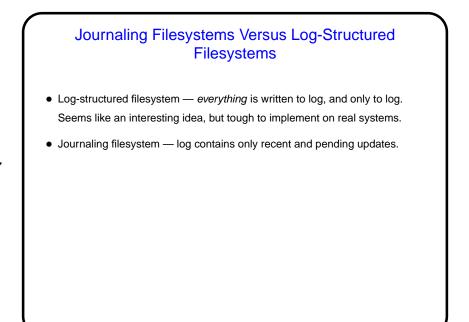
- Many things to consider here whether to keep attribute information in directory, whether to make entries fixed or variable size, etc.
- Also consider whether to allow some sort of sharing (making the hierarchy a directed graph rather than a tree). Different possibilities here; contrast UNIX "hard links" (in which different directory entries point to a common structure describing the file) and "soft (symbolic) links" (in which the link is a special type of file).



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.



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, 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).

