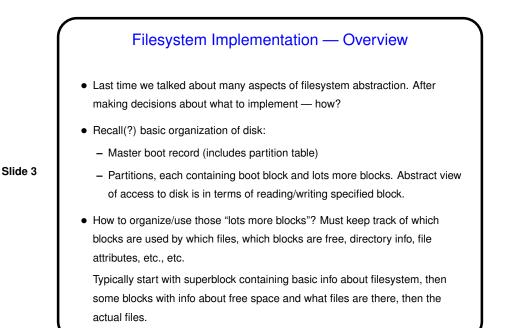
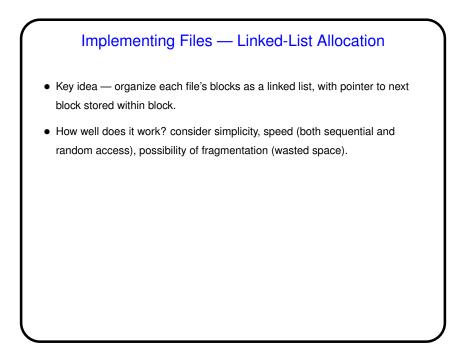


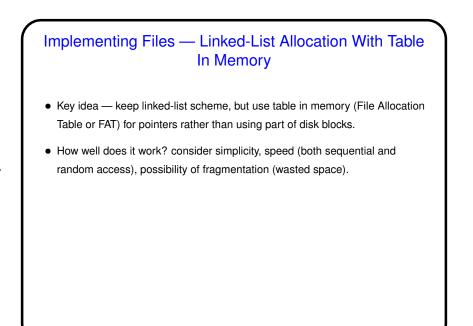
Minute Essay From Last Lecture
Many people mentioned that the homework problems helped them understand material. Good! that's the goal.
One person said "hadn't really thought about this stuff before". Also a good result? and one person said "made me think" — a *very* good result.



## Implementing Files One problem is keeping track of which disk blocks belong to which files. No surprise — there are several approaches. (All assume some outside "directory"-type structure with some information about each file — a starting block, e.g.)



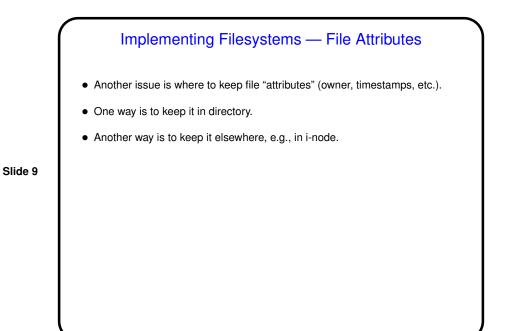




## Implementing Files — I-Nodes

• Key idea — associate with each file a data structure ("index node" or i-node) containing file attributes and disk block numbers, keep in memory for "open" files.

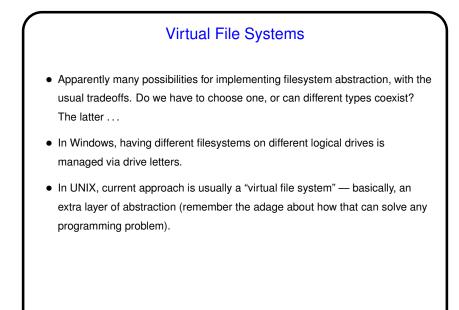
• How well does it work? consider simplicity, speed (both sequential and random access), possibility of fragmentation (wasted space).



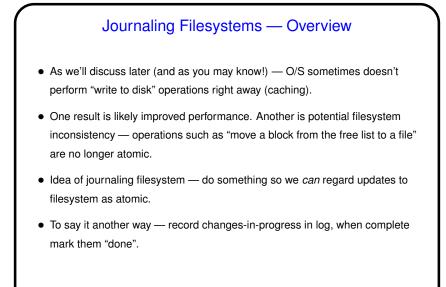
### Filesystem Implementation — Directories

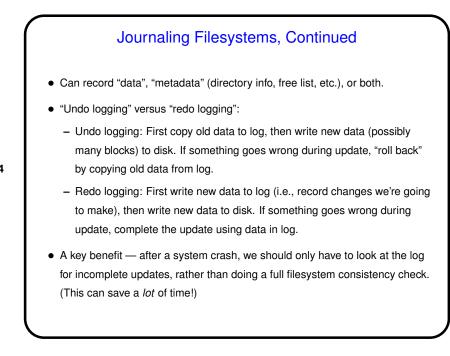
 Many things to consider here — whether to keep attribute information in directory, whether to make entries fixed or variable size, etc.

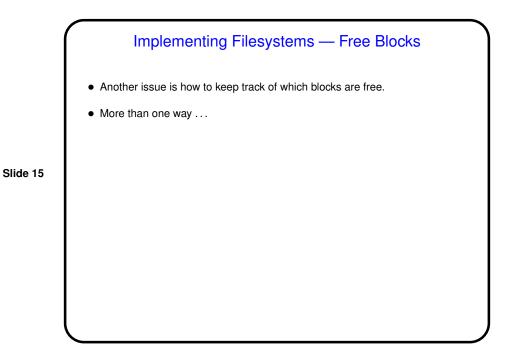
 If directory abstraction is basically hierarchical but allows some way of creating a non-tree directed graph, must figure out how to do that. Windows has "shortcuts"; UNIX has "hard links" (in which different directory entries point to a common structure describing the file) and "soft (symbolic) links" (in which the link is a special type of file).

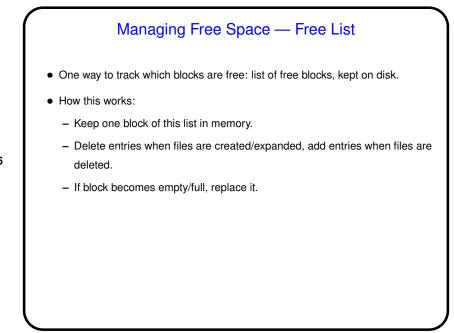


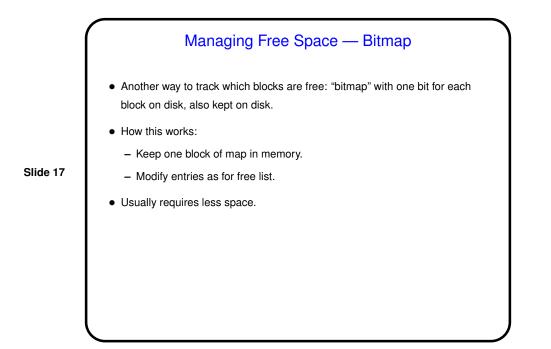
# Log-Structured Filesystems Log-structured filesystem — *everything* is written to log, and only to log. That sounds impractical, but ... Key idea is that these many disk reads are satified from cache anyway, and lots of small writes to disk give poor performance, so it makes more sense to just write (to cache) a log, and periodically save that to disk. Not used much, though, because incompatible with other file systems. Instead ...

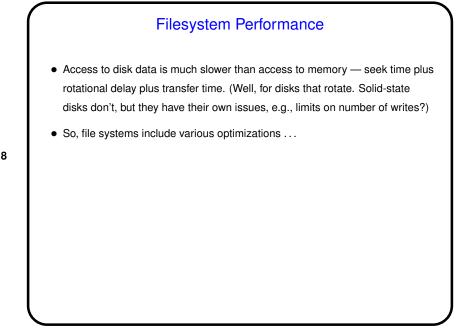


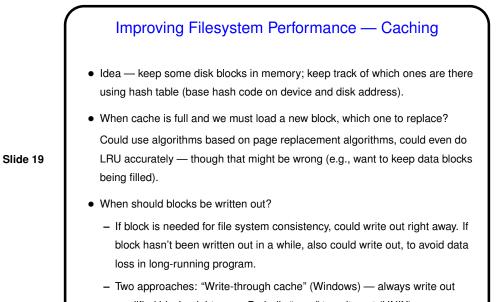










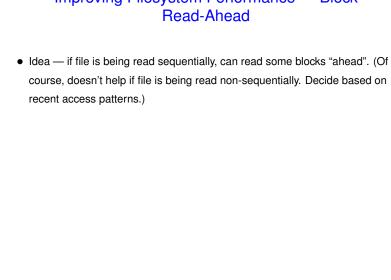


### Improving Filesystem Performance — Block **Read-Ahead**

course, doesn't help if file is being read non-sequentially. Decide based on recent access patterns.)

Slide 20

modified blocks right away. Periodic "sync" to write out (UNIX).





 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 21

• 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).

