Slide 1

Administrivia

- Grading status update:

  Homework 2, Exam 1 almost done. Should finish today. Generally good.

  Homework 3 and 4 soon. Solutions available (Homework 3 today, Homework 4 tomorrow).

  (How many people would like to have grades by e-mail, plus average if possible, before exam?)

- Exam review sheet on Web. Questions about format? This time you're allowed to bring a calculator — though I really don't think you'll need one.

Slide 2

Journaling Filesystems — Overview

- Recall — o/s sometimes doesn't perform "write to disk" operations right away (caching).
- One result is likely improved performance. Another is potential filesystem inconsistency — operations such as "move a block from the free list to a file" are no longer atomic.
- Idea of journaling filesystem — do something so we *can* regard updates to filesystem as atomic.
- To say it another way — record changes-in-progress in log, when complete mark them "done".

Slide 3

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.

Slide 4

Journaling Filesystems Versus Log-Structured Filesystems

- Log-structured filesystem — *everything* is written to log, and only to log. Seems like an interesting idea, but tough to implement with good performance.
- Journaling filesystem — log contains only recent and pending updates.

Slide 5

A Little Review — Common Mistakes from HW 2 / Exam 1

- "Virtual CPU" idea behind process abstraction (implemented by both processes and threads): Each process in effect has *its own* registers (general-purpose, PC, PSW, etc.), but they all access the same physical memory.

  What does this imply for what has to be saved/restored during a context switch?

  What does this imply about what might change if a process is interrupted?
- Deadlock happens when all of a group of processes are waiting for something that only another of the group can do. Is it deadlock if one processes grabs something and just refuses to give it up?

Slide 6

Review — Memory Management

- Interplay between hardware and o/s — how hardware (architecture) affects o/s design decisions.
- Page table entries, including R and M bits (what they're used for).
- Page fault processing.

Slide 7

Review — I/O Management

- Programmed I/O versus interrupt-driven I/O.
- Layered scheme. What needs to be done in kernel (supervisor mode) and what can be done "in user space"? How do we get into supervisor mode anyway? When does "interrupt handler" code run? When does other device-driver code run?

Slide 8

Review — Filesystems

- Different ways of keeping track of which disk blocks belong to a file. Is there a way that's always best? How do you get from a pathname to disk blocks?
- Buffers, caches, etc.

Slide 9

Minute Essay

- None — sign in.
- (Reminder — Homework 4 due by 5pm.)