Slide 1

Administrivia

- Reminder: Homework 1 due Monday. For the programming problem, you're meant to fill in details of some (pseudo?)code shown in the textbook. If you don't have your book yet — borrow one!
- Reminders/requests about homework:

  All homework is considered pledged work. Write "pledged" on hardcopy work, and include it in comments for programming assignments.

  For work submitted by e-mail, please do include the name or number of the course in the subject line of your message, plus something about which assignment it is, to help me get it into the correct folder for grading.

Slide 2

Minute Essay From Last Lecture

- (See notes from last time.)
- Most people more or less got the answers I had in mind. (Apparently no budding lawyers here quibbling about details.) Minimum number running — is there one? Answer might be "it depends".

Slide 3

Processes Versus Threads

- So far I've used "process" in an abstract/general way.
- In typical implementations, though, "process" is more specific — something that has its own address space, list of open files, etc. Often these are called "heavyweight processes".
  - Advantages — such processes don't interfere with each other.
  - Disadvantages — they can't share data, switching between them is expensive ("a lot of state" to save/restore).
- For some applications, might be nice to have something that implements the abstract process idea but allows sharing data and faster context switching — "threads".

Slide 4

Threads

- So, threads are another way to implement the process abstraction.
- Typically, a thread is "owned" by a (heavyweight) process, and all threads owned by a process share some of its state — address space, list of open files.
- However, each thread has a "virtual CPU" (a distinct copy of registers, including program counter).
- Implementation involves data structures similar to process table.
- Advantages / disadvantages (compared to processes)?

Slide 5

Threads, Continued

- Advantages: threads can share data (same address space), switching from thread to thread is fairly fast.
- Disadvantages: sharing data has its hazards (more about this later).

Slide 6

Implementing Threads

- Two basic approaches — “in user space” and “in kernel space” (next two slides).
- Various hybrid schemes also possible.

Slide 7

Implementing Threads “In User Space”

- Basic idea — operating system thinks it’s managing single-threaded processes, all the work of managing multiple threads happens via library calls within each process.
- Advantages / disadvantages?

Slide 8

Implementing Threads “In User Space”, Continued

- Advantages: fewer system calls, hence probably more efficient.
- Disadvantages:
  - If a thread blocks, it may do so in a way that blocks the whole process.
  - Preemptive multitasking is difficult/impossible.
  - Using multiple CPUs is difficult/impossible.

Slide 9

Implementing Threads “In Kernel Space”

- Basic idea — operating system is involved in managing threads, the work of managing multiple threads happens via system calls (rather than user-level library calls).
- Advantages / disadvantages?

Slide 10

Implementing Threads “In Kernel Space”, Continued

- Advantages: avoids the difficulties of implementing in user space.
- Disadvantages: probably less efficient.

Slide 11

Adding Multithreading

- If you've written multithreaded applications — moving from single-threaded to multithreaded not trivial:
  - Figure out how to split up computation among threads.
  - Coordinate threads' actions (including dealing properly with shared variables).
- Similar problems in adding multithreading to systems-level programs:
  - Deal properly with shared variables (including ones that may be hidden).
  - Deal properly with signals/interrupts.

Slide 12

Threads — Example Implementations

- UNIX systems vary as to which they use (see chapter 10 for more info). Early versions of Linux provided no support for kernel-space threading, but there were libraries for the user-space version. Kernel now provides support, but threads apparently basically processes with some different flags allowing them to share memory, etc.
- Windows NT/2000 apparently is such that *all* processes have at least one thread, and the basic scheme is either kernel-space or a hybrid (see chapter 11 for more info).

Slide 13

Minute Essay

- Tell me about your experience with writing programs that involve concurrency — multithreaded, message-passing, communicating over sockets, etc.