Slide 1

Administrivia

- (Via e-mail.)

Slide 2

Threads and Processes

- In the first lecture on threads I said interaction between threads and processes seemed murky to me? I did some Web-searching . . .
- Probably best to think in terms of processes being containers for threads, with every process containing at least one thread.

Slide 3

Implementing Threads Without O/S Support

- Something the textbook doesn't mention: Threads *can* be implemented totally without O/S involvement — "in user space" (as opposed to "in kernel space"):
- O/S only knows about processes; all support for multiple threads happens via libraries.
- Likely more efficient, but has some drawbacks:
- If a thread blocks, it may do so in a way that blocks the whole process.
- Preemptive multitasking difficult/impossible without O/S involvement, as is using multiple CPUs.
- Fairly widely used as first implementation of threads (e.g., in Java).

Slide 4

Multithreaded Programming with "P-Threads"

- POSIX (Portable Operating System Interface) defines interface for multithreaded programming — variously called P-Threads, Pthreads, pthreads.
- Fairly primitive, but likely to be found on wide range of systems.
- A quick tour . . . (Also, I posted on the course Web site some examples you can download and play with.)

Slide 5

P-Threads — Basics

- Declare threads as opaque data type `pthread_t`.
- Create with `pthread_create()` (refer to `man` page).
- Syntax is ugly ugly, but very C-idiomatic:

  Specify function thread should run with function pointer. Pass data to it with single parameter of type `void *`.
- Thread runs until it calls `pthread_exit()` (preferred) or exits.
- Creating thread can wait for thread to finish with `pthread_join`.
- "Hello world" example (under "sample programs" on course Web site).

Slide 6

P-Threads — Locks

- Declare as opaque data type `pthread_mutex_t`.
- Initialize, clean up with `pthread_mutex_init()`; `pthread_mutex_destroy()`.
- Lock, unlock with `pthread_mutex_lock()`, `pthread_mutex_unlock()`.
- Semi-real-world example of use: Multithreaded program to estimate π with numerical integration (under "sample programs" on course Web site).

Slide 7

P-Threads — Semaphores

- Opaque data type `sem_t`.
- Initialize, clean up with `sem_init()`, `sem_destroy()`.
- Post (`up()`) with `sem_post()`.
- Wait (`down()`) with `sem_wait()`.
- Example: Semaphore-based solution to bounded buffer problem (under "sample programs" on course Web site).

Slide 8

P-Threads — Condition Variables

- Opaque data type `pthread_cond_t`.
- Initialize, clean up with `pthread_cond_init()`, `pthread_cond_destroy()`.
- Wait with `pthread_cond_wait()`.
- Signal with `pthread_cond_signal()`.
- Can combine with locks to implement the "monitors" idea.
- Example: Monitor-based solution to bounded buffer problem (under "sample programs" on course Web site).

Slide 9

Minute Essay

- Questions? Do the examples make sense?