Slide 1

Administrivia

- A note for those who weren't here the Wednesday before the holiday: Skim the "Twelve ways to fool the masses" paper linked from the "Useful links" page: tongue-in-cheek, but some useful ideas.

Slide 2

A Little About Multithreaded Programming with POSIX Threads

- POSIX threads ("pthreads"): widely-available set of functions for multithreaded programming, callable from C/C++.
- Same ideas as multithreaded programming with OpenMP and Java, but not as nicely packaged (my opinion). Might be more widely available than OpenMP compilers, though.

Slide 3

POSIX Threads — UE Management

- Create a new thread with `pthread_create()`, specifying function to execute and a single argument. (Yes, this is restrictive — but the single argument could point to a complicated data structure.)
- Thread continues until function terminates. Best to end with call to `pthread_exit()`.

Slide 4

POSIX Threads — Synchronization

- `pthread_join()` waits until another thread finishes — similar to `join` in Java's `Thread` class.
- Various synchronization mechanisms:
  - Mutexes (locks): `pthread_mutex_init()`, `pthread_mutex_destroy()`, `pthread_mutex_lock()`, `pthread_mutex_unlock()`.
  - Condition variables: `pthread_cond_init()`, `pthread_cond_destroy()`, `pthread_cond_wait()`, `pthread_cond_signal()`.
  - Semaphores: `sem_init()`, `sem_destroy()`, `sem_wait()`, `sem_post()`.

Slide 5

POSIX Threads — Communication

- As with other multithreaded programming environments we've looked at, conceptually all threads share access to a single memory space.
- In terms of scoping, though, each thread has access to:
  - Any global variables (shared with other threads).
  - Its single argument (potentially shared with other threads).
  - Any local variables (not shared with other threads — since every call to function creates a new copy).

Slide 6

POSIX Threads — Simple Examples

- "Hello world" example.
- "Hello world" example with delay (to illustrate synchronization).
- Numerical integration example.

Minute Essay

- If you wanted to provide a parallel programming environment on a new architecture or operating system, which do you think would be easier to port, a POSIX threads library or an OpenMP compiler?

Minute Essay Answer

- Probably the POSIX threads library — less code overall, and for both of them you'd have to figure out basic stuff such as thread creation and synchronization.