Slide 1

Administrivia

- Reminder: Homework 4 code due today.

Slide 2

Minute Essay From Last Lecture

- GPGPU (General Purpose Computing on Graphics Processing Units) is the next big thing? Maybe! OpenCL may be emerging as a standard. Somewhat different model of computation.
- X10 and/or other new parallel programming environments? Apparently interest in new environments has picked up again, after falling off for a while. Hm!
- Threads in C++ standard library? Apparently being considered as part of standard in work (C++0x). Support for threading also provided by some third-party libraries (Boost is a free one).

Slide 3

A Little About Multithreaded Programming with POSIX Threads

- POSIX threads ("pthreads"): widely-available set of functions for multithreaded programming, callable from C/C++.

  ("POSIX" is Portable Operating System Interface, a set of IEEE standards defining an API for UNIX-compatible systems. Implemented to varying degrees by most UNIX-like systems; implementations also exist for other systems — e.g., Cygwin for Windows.)

- Same ideas as multithreaded programming with OpenMP and Java, but not as nicely packaged (my opinion). At one time probably more widely available than OpenMP compilers, though that has probably changed with `gcc` OpenMP support.

Slide 4

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 5

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 6

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 7

POSIX Threads — Simple Examples

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

Slide 8

Example Application — Generic Master/Worker Program

- (Look at code.)

Slide 9

Minute Essay

- None — sign in.