Slide 1

Administrivia

- Reminder: Homework 7 due today. If you're not done by end of the day, send me what you have and then send an improved version later — by Monday is okay.
- No more required homeworks, but I might make up an extra-credit assignment?

Slide 2

Minute Essay From Last Lecture

- More votes for multithreaded programming in C, so we'll do that today.

Slide 3

Concurrent Programming

- Several situations in which it's useful to be able to do multiple things "at the same time" (quotation marks is because the idea has been around since the very early days of computers, well before the advent of computers with more than CPU, though that too came surprisingly early, in the mainframe days anyway).
- One use is in situations in which there are several things that could or should be done concurrently — multiple applications "open" at the same time, background activities such as monitoring a network connction, "multitasking" aspects of a GUI program (the user interface, the part that draws what's on the screen, possibly computationally-intensive activities).
- Another use is to speed up computations — "parallel processing" is old term for it.

Slide 4

Concurrent Programming, Continued

- Several ways to implement this idea of processes/concurrency, and all can be tricky in terms of partitioning the work to be done into (semi-)independent "threads of control" and arranging for them to cooperate.
- One way involves "heavy-weight" processes, all on the same computer or on more than one. Requires some form of interprocess communication. Processes normally don't share memory.
- Another way involves "threads", all on the same computer and typically sharing memory.
- (Yet another form of concurrency is making use of a GPU for computing, but that's different enough to not address today.)

Slide 5

Multithreaded Programming — Basic Constructs

- Some way to set up / launch threads.
- Some notion of shared and non-shared variables.
- Some way(s) for threads to cooperate — having one thread wait for another, having one thread interrupt another, mechanisms for "locking" to guarantee one-at-a-time access to shared varaibles.
- How this is implemented — can be included in a programming language (as in Java and Scala, and OpenMP extensions to C etc.), or via libraries (as in POSIX threads library).

Slide 6

Multithreaded Programming in C — 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.)
- Widely supported but fairly primitive. Latest C standard also includes support for threads, I think based on POSIX threads, but as far as I know is not widely implemented. C++11 also includes support for threads, and recent versions of `g++` (the C++ version of `gcc`) implement it.

Slide 7

POSIX Threads

- Functions exist to define threads, start them up, and have one thread wait for another to finish. Creating a thread requires giving it a function to execute, which can accept only a single argument. (Yes, this is restrictive — but the single argument could point to a `struct`). Thread continues until function terminates.
- Various groups of functions exist to synchronize among threads — mutex locks, condition variables, semaphores.
- ("Hello world" program.)

Slide 8

Multithreaded Programming in C — OpenMP

- OpenMP defines standard extensions to C, C++, and Fortran to support multithreaded programming. At one point was considered emerging standard, but possibly with addition of threads to the C++ standard maybe not so much?
- Implemented as a set of "compiler directives" (`#pragma` in C and C++), ignored if compiling with a compiler that doesn't support OpenMP), and a set of functions.
- Possibly originally intended mostly for "parallelizing" loops (distributing iterations among threads) but also included facilities for doing two or more different things concurrently, which has been extended.

Slide 9

OpenMP, Continued

- Basically a fork/join model — “master” thread starts up group of “worker threads”, which run conncurrently until are done, at which point execution continues (sequentially) in master thread.
- Variables are shared unless stated otherwise (with all the risks that implies), but constructs exist to specify thread-private variables and “reductions”, and there are library functions to provide explicit locking.
- (“Hello world” program.)

Slide 10

Multithreaded Programming — Simple Example

- A nice example is using numerical integration to estimate the value of π by approximating the area under the curve

$$\frac{4}{1+x^2}$$

- Sequential version uses a loop, which ”parallelizes” nicely (give each thread some of the iterations).
- (Sequential, pthreads, and OpenMP versions.)

Slide 11

Minute Essay

- Anything noteworthy about Homework 7? difficult, interesting,. . . (I hope if you finished it you now understand pointers and DIY memory management better!)
- Would you be interested in an extra-credit assignment, to be due sometime during finals week?