Slide 1

Administrivia

- *Please do not* reboot the machines in HAS 340! If a previous user has left a machine in the "locked by screensaver" state, you can bail out by pressing control-alt-backspace to restart X (the graphical subsystem) without disturbing background processes.
- Are your prox cards giving you access to the labs? Supposedly all known problems have been resolved.
- Homework 1 on Web; first due date is next Monday.

Slide 2

Multithreaded Programming with OpenMP — Review

- Basic idea — fork/join programming model, all threads share memory.
- Can duplicate code in all threads (`parallel` directive), split a loop among threads (`parallel for`), have different threads do different things (`parallel sections`).

  More details in specification — can combine these in various ways.

  Various ways to assign loop iterations to threads — more about that shortly.

Slide 3

Variables in OpenMP

- Most variables are shared by default.

  Exceptions are variables local to a block within a parallel region, stack (local) variables in subprograms called from parallel region.
- To give each thread a separate copy — `private` clause. `firstprivate` and `lastprivate` can be used to start/end with shared value.
- To create a partial result in each thread and then combine (“reduce”) — `reduction` clause. Operations include sum, product, and/or. No max or min in C/C++.
- (Review numerical integration program.)

Slide 4

Assigning Work to Threads — `schedule` clause

- `static` (with optional chunk size) — divide iterations into fixed-size blocks, distribute evenly among threads.
- `dynamic` (with optional chunk size) — queue of iterations, threads grab blocks of iterations until all done.
- `guided` (with optional chunk size) — like `dynamic`, but with decreasing blocks of iterations.
- `runtime` — get from `OMP_SCHEDULE` environment variable.

Intermezzo — Environment Variables (in `bash`)

- To set environment variable `FOO` for the rest of the session:

  `export FOO=fooval`

  (To set every time you log in, put in `.bash_profile`.)

- To run `bar` with a value for `FOO`:

  `FOO=fooval bar`

Library Functions

- `omp_get_num_threads`, `omp_set_num_threads`, `omp_get_thread_num` — as in examples and appendix.
- `omp_get_wtime` — as in examples and appendix.
- Functions to do locking — more about them shortly.
- Functions to do other things — in specification.

Slide 7

Synchronization Constructs

- `critical` — only one thread at a time executes this block of code. (Example — `synch-2.c` on sample programs page.)
- `barrier` — threads wait here until all have arrived. Implicit barrier at end of parallel region.
- `single` — only one thread executes this block.
- Several others — `atomic`, `flush`, `ordered`, `master`. More about them in the specification.

Slide 8

Locks

- `omp_lock_t` — declares a lock variable.
- `omp_init_lock`, `omp_destroy_lock` — create and destroy.
- `omp_set_lock` — acquire lock (wait if necessary).
- `omp_unset_lock` — release lock.
- Other functions described in specification.
- Example — `synch-3.c` on sample programs page.

Homework 1 Background

- In Homework 1, you will make a first pass at writing a set of programs (one using OpenMP, one using MPI, and one using Java) to solve the following problem. (We'll talk more about it in class after you've tried it.)
- We talked about computing π using numerical integration. Another interesting (surprising?) approach uses a "Monte Carlo" method:

  Consider a square with sides of length 2 (any unit you like), enclosing a circle of radius 1.

  Approximate the area of the circle by "throwing darts" at the square, counting how many fall within the circle, and calculating the ratio of those within the circle to the total number.

  Model "throwing darts" by using pseudorandom number generator to generate coordinates of a point.

Minute Essay

- Running the numerical integration example with different numbers of threads gives different results. Why do you think that happens?

Minute Essay Answer

- The order in which the partial results (produced by the iterations of the loop to compute areas of rectangles) are added together depends on the number of threads and the scheduling — and floating-point arithmetic is not associative (!).