Slide 1

Administrivia

- Homework 1 to be on the Web soon. I will send mail.
- Notes on using OpenMP on the Web, linked from course Web page and my home page. Also notes on GNU compilers.
- (Status of updated appendices.)

Slide 2

OpenMP Programs — Recap/Review

- OpenMP defines some concepts and a set of extensions to three base languages (C, C++, Fortran). These extensions include compiler directives and library functions.
- So, OpenMP programs look like programs in the base language, plus the directives, which are defined in a way that the code still compiles as sequential code even without support for the directives.

Slide 3

Simple Example / Compiling and Executing

- Look at simple programs — `hello.c`, `hello++.cpp` on sample programs page.
- Compile with compiler supporting OpenMP.
- Execute like regular program. Can set environment variable `OMP_NUM_THREADS` to specify number of threads. Default value seems to be one thread per processor.

Slide 4

Sidebar: “Atomic” Operations

- Some discussion last time about different behavior of `printf` and C++ stream output — “are they doing some kind of locking?”
- Interesting question, but possibly a better way to describe it is in terms of atomicity — an “atomic” operation is one that executes as one indivisible operation, without interference from other units of execution. Whether that effect comes from locks or something else maybe we don’t need to know. (Yet?)

Slide 5

Sidebar — GNU Compilers on Classroom/Lab Machines

- Two versions of GNU compiler collection installed this year:
  - Most-recent version available in standard Scientific Linux repositories (4.4.7).
  - Current version available directly from project Web site (4.8.1). Supports many (most?) C++11 additions/changes.
- Default is 4.4.7. To get the newer version, type

  ```
  module load gcc-4.8.1
  ```

  (`module avail` if you don't remember the name)

  and then standard command names (`gcc`, `g++`, etc.) should give you the 4.8.1 version. Also sets up other needed environment.

Slide 6

Sidebar — `make` and makefiles

- Compiling with non-default options (as you must do to compile OpenMP programs with `gcc`) can become tedious.
- `make` can help. Briefly — it's a very old UNIX tool intended to help automate building large programs. Can be used in different ways, but one of them is simply to make it easy to compile with non-default options.
- To use `make`, set up `Makefile` (example linked from "Sample programs" Web page), and then type `make foo` to compile `foo.c` to `foo`.

Slide 7

Sidebar — 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 be sure it's set in terminal-window sessions may need to set it in `.bashrc`.)

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

  ```
  FOO=fooval bar
  ```

Slide 8

How Do Threads Interact?

- With OpenMP, threads share an address space, so they communicate by sharing variables. (Contrast with MPI, to be discussed next, in which processes don't share an address space, so to communicate they must use messages.)
- Sharing variables is more convenient, may seem more natural.
- However, "race conditions" are possible — program's outcome depends on scheduling of threads, often giving wrong results.

  What to do? use synchronization to control access to shared variables. Works, but takes (execution) time, so good performance depends on using it wisely.

Slide 9

Example — Numerical Integration

- Compute π by integrating $\int_0^1 \frac{4}{1+x^2}\, dx$.
- Do this numerically by approximating area under curve by many small rectangles, computing their area, adding results.
- Sequential program fairly straightforward. (`num-int-seq.c` on “sample programs” page).
- “Parallelize” how? (Discuss.)

Slide 10

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 11

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.

Slide 12

Minute Essay

- None — “sign in” (send me e-mail if you have questions or comments).