Live Lecture Chat Window
Wednesday, April 6, 2022

14:51:24 Does OpenMP work with C++ or C#?

It works with C, C++, and Fortran. As far as I know, I does not work with C#.

15:02:08 Will OpenMP actually tell you it will not work? Or just silently fail in mysterious ways?

No, it won't tell you. If you violate, say, the restrictions on inter-loop dependencies, it will likely just quietly mess up.

15:04:15 So you can't do it because it would force the threads to go in order which would eliminate the benefits of the threads

Good way to put it.

15:05:16 Will it just seg fault or would it just get the wrong answer

Typically, it won't seg fault. Seg faults are reserved for catastrophic problems like trying to access a memory location you don't own. It will just get the wrong answer.

One trick I use is to run the code without any threading, then run it with threading and see if I get the same answer.

15:08:07 What's the default if you don't add either at the end?

I believe it's *shared*, but I don't want you trusting defaults.

15:09:39 It will error if you don't include something when you have default(none)?

That's correct. It's a good way to be sure you are not trusting defaults.

15:24:55 So would a use case for *pragma omp single* be a scenario where you're trying to read a file inside of a for loop? Would it just get assigned to the first thread that grabs it?

Yes and yes. That's a good example, and there is no way to predict which thread becomes the Chosen One.

15:31:39 What if one thread gets stuck,. wouldn't that make it potentially fail silently

More like the program hangs silently. Typically "stuck" means that the thread is waiting for something that is not going to happen.

15:34:26 Sections are how you manually allocate tasks to each thread.

Yes. You will get very good at this in Project #3.

15:36:45 Follow up question on that - what would happen if you set a number of threads but your computer actually doesn't have that many?

Remember that “threads” are a software thing, so you can create as many threads as you want (within some large OS-set limit). The question I think you meant is “what would happen if you set a number of threads but your computer actually doesn't have that many *cores*?” The answer is that more than one thread would be assigned to each core. If those cores are using hyperthreading, then it might not be so bad. But, if they are not, then swapping out a thread state incurs more overhead.