Programming Shared-Memory Multiprocessors - Introduction*

Charles Severance
Kevin Dowd

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License †

In here[1], we examined the hardware used to implement shared-memory parallel processors and the software environment for a programmer who is using threads explicitly. In this chapter, we view these processors from a simpler vantage point. When programming these systems in FORTRAN, you have the advantage of the compiler's support of these systems. At the top end of ease of use, we can simply add a flag or two on the compilation of our well-written code, set an environment variable, and voilá, we are executing in parallel. If you want some more control, you can add directives to particular loops where you know better than the compiler how the loop should be executed.[2] First we examine how well-written loops can benefit from automatic parallelism. Then we will look at the types of directives you can add to your program to assist the compiler in generating parallel code. While this chapter refers to running your code in parallel, most of the techniques apply to the vector-processor supercomputers as well.

*Version 1.3: Aug 25, 2010 11:12 am -0500

†http://creativecommons.org/licenses/by/3.0/

[1] "Shared-Memory Multiprocessors - Introduction" <http://cnx.org/content/m32797/latest/>

[2] If you have skipped all the other chapters in the book and jumped to this one, don't be surprised if some of the terminology is unfamiliar. While all those chapters seemed to contain endless boring detail, they did contain some basic terminology. So those of us who read all those chapters have some common terminology needed for this chapter. If you don't go back and read all the chapters, don't complain about the big words we keep using in this chapter!