Chapter 1

What Is C*?

C* (pronounced “sea-star”) is an extension of the C programming language designed to help users program massively parallel distributed-memory computers. In addition, it is a concise and efficient language for programming many other architectures, including those with shared memory, vector processors, pipelining, and superscalar execution units. This chapter and Chapter 2 introduce C*.

1.1 C* and C

C* implements the ANSI standard C language (referred to in this guide as *Standard C*). Programs written in Standard C compile and run correctly under C* (except when they use one of the words that are newly reserved in C*). In addition, C* provides new features to aid in writing programs for massively parallel computers. These features include:

- A method for describing the size and shape of parallel data and for creating parallel variables. Shapes and parallel variables are discussed in Chapters 3, 4, and 9.
- New operators and expressions for parallel data, and new meanings for standard operators that allow them to work with parallel data. Operators are discussed in Chapter 5.
- Methods for choosing the parallel variables, and the specific data points within parallel variables, upon which C* code is to act. These features are discussed in Chapters 4 and 6.

- New kinds of pointers that point to parallel data and to shapes. C* pointers are discussed in Chapter 7.
- Changes to the way functions work so that, for example, a parallel variable can be used as an argument. Chapter 8 describes C* functions.
- Methods for communication among parallel variables. See Chapter 10.
- Library functions that also allow communication among parallel variables. Chapters 11–14 describe these functions.

1.2 C* Implementations

In addition to a general description of how to program in the C* language, this guide provides specifics about two implementations of C*:

- *CM-200 C** — The CM-200 compiler translates a C* program into a serial C program made up of standard serial C code and calls to Paris, the CM-200's parallel instruction set. This code is then passed to the C compiler of the CM-200's front end, which handles it in the normal way to produce an executable load module. The serial C code is executed on the front end; the Paris instructions are executed on the CM. Programs compiled with the CM-200 C* compiler can run on the CM-200, CM-2, and CM-2a Connection Machine systems.
- *CM-5 C** — The CM-5 compiler translates a C* program into assembly code. Serial instructions are executed on the CM-5's partition manager; parallel instructions are executed on its processing nodes or vector units. When compiled with the **`-node`** option, copies of the program run on individual nodes; serial instructions are executed on the node, and parallel instructions are executed on the node's vector units.

There is in addition a Sun-4 implementation of C*. This implementation is based on the CM-5 compiler, but lets you run your programs on a Sun-4 workstation, without CM hardware. Unless otherwise specified, all notes in this manual that apply to the CM-5 implementation also apply to the Sun-4 implementation.

There are some implementation differences between these compilers. The differences are noted in this guide.

1.3 Program Development Facilities

C* uses its own compiler, run-time libraries, and header files.

C* can use standard UNIX programming tools such as **`make`**. In addition, you can execute, debug, and visualize data for a C* program within Prism, the CM's programming environment.

The C* compiler and related program development facilities are described more fully in the *C* User's Guide* for either the CM-200 or the CM-5.