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.

1.1 C* and C

The C* language is based on the standard version of C specified by the American National Standards Institute. (We assume in this guide that you are familiar with C.) C programmers will find most aspects of C* code familiar to them. C language constructs such as data types, operators, structures, pointers, and functions are all maintained in C*; new features of Standard C such as function prototyping are also supported. C* extends C with a small set of new features to aid in writing programs for massively parallel computers. For example:

- C* provides a **`shape`** keyword that lets you describe the size and shape of parallel data. You can then declare *parallel* variables by tagging them with a shape.
- C* lets you operate on parallel variables by providing a few new operators, and new meanings for standard C operators.
- C* adds a **`with`** statement that lets you choose the shape to be used for parallel operations, and a **`where`** statement to restrict operations to certain data points of the shape.
- C* extends Standard C structures, pointers, and functions so that they work with parallel data.

- C* provides ways for parallel variables to interact with other parallel variables, even if the parallel variables are of different shapes.

1.2 C* Implementations

This guide refers to two implementations of the C* language:

- *CM-200 C** — Use this implementation to run your program on a CM-200, CM-200a, CM-2, or CM-2a Connection Machine system.
- *CM-5 C** — Use this implementation to run your program on all models of the CM-5 Connection Machine system.

1.3 Developing a C* Program

C* uses its own compiler, run-time libraries, and header files, as well as some Standard C libraries. Files containing C* source code must have the suffix **`.cs`**.

Within your C* program, you can include calls to library routines such as those in the CMFS library. You can also call subroutines written in CM Fortran.

On a CM-2, CM-2a, CM-200, or CM–200a, you execute the program by attaching to the CM via the **`cmattach`** command, or by submitting the executable load module as a batch request to a queue in the NQS batch system.

On a CM-5, you execute the program just as you would any UNIX executable load model; or you can submit it as a batch request to NQS.

You can use Prism, the CM's graphical programming environment, to help you develop, debug, and analyze the performance of your C* program.