Chapter 6

The Programming Environment: Prism

The Prism programming environment is an integrated Motif-based graphical environment within which users can develop, execute, debug, and analyze the performance of programs written for the Connection Machine system. It provides an easy-to-use, flexible, and comprehensive set of tools for performing all aspects of Connection Machine programming.

Separate versions of Prism are available for working with data parallel and message-passing programs. Most of the functionality is the same; some features are implemented differently, however, taking into account the requirements of the different programming styles.

Users can either load an executable program into Prism, or start from scratch by calling up an editor and a UNIX shell within Prism and using them to write and compile the program.

Once an executable program is loaded into Prism, users can (among other things):

- **Execute the program.** Users can simply start the program running or single-step through it. Execution can be interrupted at any time.
- **Debug the program.** Users can perform standard **`dbx`**-like debugging operations such as setting breakpoints and traces, printing the value of a variable or expression, and displaying and moving through the call stack.
- **Analyze the program's performance**. Data on execution time, broken down by procedures or by lines of source code, may be displayed as histograms. See Section 6.2.
- **Visualize data**. The values of interactively specified variables or expressions may be displayed in a variety of textual and graphical formats. See Section 6.3.

Copyright © 1993 Thinking Machines Corporation

In debugging message-passing programs, users can work with *PN sets*. PN sets are predefined or user-created groups of nodes that can be viewed and operated on as a single entity. For example, the predefined PN set **`error`** contains all nodes in the error state; Prism updates the contents of this set as the program executes. A user could also define a PN set whose nodes fulfill a specified condition — for example, all nodes in which the value of **`x`** is greater than 0. Commands can then be applied to a specific PN set. For example, the user could have the nodes in the set execute the next line of code, or could display the value of a variable in the nodes.

Prism operates on terminals or workstations running the X Window System. A commands-only version is also available for users without access to X. Another option lets X users operate with the familiar commands interface, but send certain output, such as performance data, to X windows.

6.1 Using Prism

Figure 14 shows the main window of Prism, with a data parallel program loaded. It is within this window that users debug and analyze their programs. Users can operate with a mouse, use keyboard equivalents of mouse actions, or issue text commands.

Clicking on items in the *menu bar* along the top of the window displays pulldown menus that provide access to most of Prism's functionality.

Frequently used menu items can be moved to the *tear-off region*, below the menu bar, to make them more accessible.

The *status region* displays messages about the program's status.

The *source window* displays the source code for the executable program. The user can scroll through this source code or display a different source file. When a program stops execution, the source window is automatically updated to show the code currently being executed. The user can click on variables or expressions in the source code to print their values. The source window can also be split, with the assembly code corresponding to the source code appearing in the bottom pane.

The *line-number region* is associated with the source window. Clicking to the right of the line number sets a breakpoint at that line.

November 1993
Copyright © 1993 Thinking Machines Corporation

The *command window* at the bottom of the main window displays messages and output from Prism. The user can also type commands in the command window, rather than use the graphical interface.

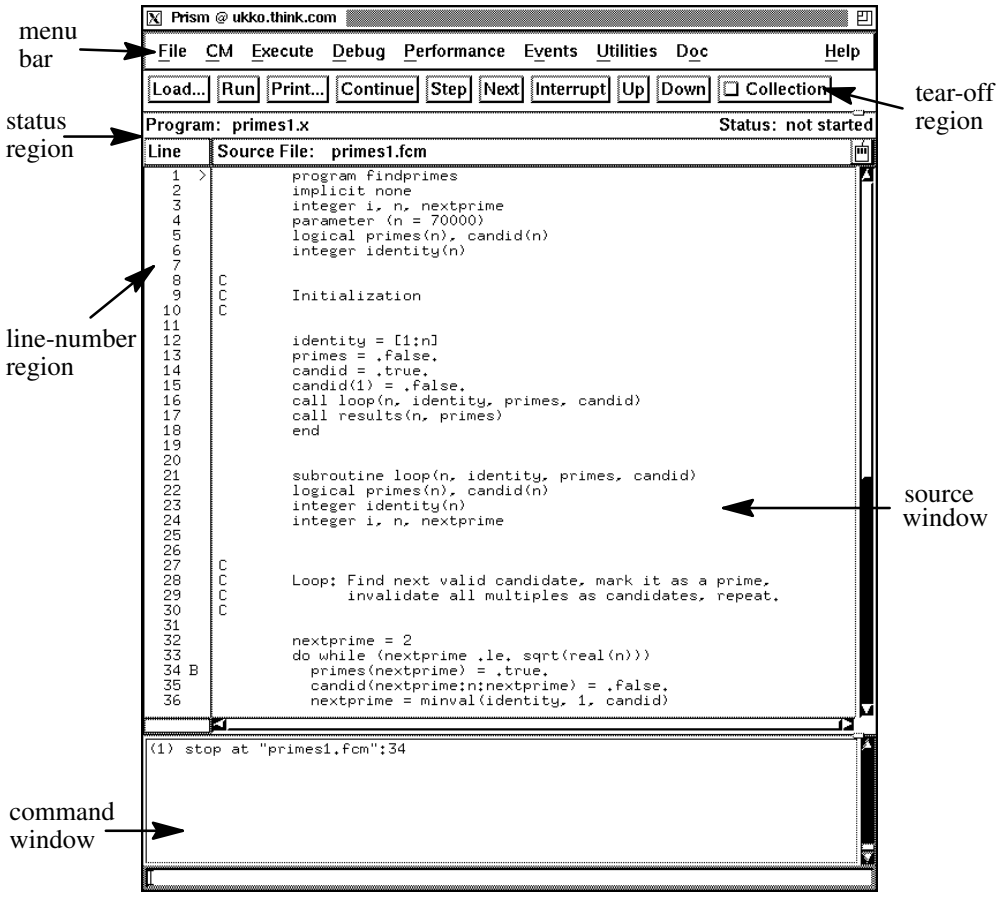

Figure 14. Prism's main window.

6.2 Analyzing Program Performance

In cooperation with the compilers and run-time library routines, Prism provides the performance data essential for effectively analyzing and tuning programs. For data parallel programs, the data includes:

- control processor user and system time
- processing time

Copyright © 1993 Thinking Machines Corporation

- time spent transferring data between control processor and nodes
- time spent in general Data Network communication
- time spent doing specific patterns of Data Network communications, such as nearest-neighbor on a grid
- time spent doing reductions and parallel prefix operations

For message-passing programs, Prism provides performance data separately for each node. The data includes processing time for both the scalar microprocessor and the vector units, as well as time spent performing I/O and various array operations.

The performance data is displayed as histograms and percentages. For each type of time measurement, the user can also see the data broken down for each procedure and each source line in the program. The data on procedures is available in two versions. One gives a flat per-procedure view of the utilization of the resource; the other shows utilization using the dynamic call graph of the program.

6.3 Visualizing Data

When operating on large arrays or parallel variables, it is often important to obtain a visual representation of its data elements. In Prism, the user can create *visualizers* to provide this representation. A wide range of formats is available, including:

- *Text*, where the data is shown as numbers or characters
- *Dither,* where values are displayed as a shading from black to white
- *Colormap*, where each data element is mapped to a single color pixel, based on a range specified by the user
- *Threshold*, where each data element is mapped to a single pixel, either black or white, based on a cutoff value specified by the user
- *Graph,* where values are displayed as a graph, with the index of each data element plotted on the horizontal axis and its value on the vertical axis
- *Surface,* which renders the 3-dimensional contours of a 2-dimensional slice of data
- *Vector,* which displays complex data as vectors

November 1993
Copyright © 1993 Thinking Machines Corporation

A *data navigator* allows manipulation of the display window relative to the data being visualized. If a parallel array is multidimensional, the visualizer displays a slice through the array; the data navigator provides controls for selecting the array axes to be displayed and the position of the slice. The user can update a visualizer or save a snapshot of it.

Figure 15 shows a surface visualizer.

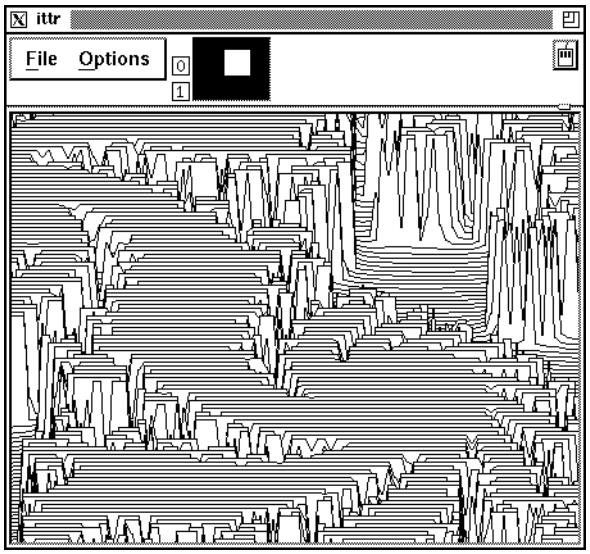

Figure 15. A visualizer.

For message-passing programs, Prism provides an extra dimension for the visualizer; this dimension represents the nodes in a PN set. By moving along this dimension, the user can display in turn the visualizer for each node in the set.

6.4 Using Prism with CMAX

Prism can be used with programs that have been converted from Fortran 77 to CM Fortran via the CMAX Converter. See Chapter 8 for more information on CMAX. Prism provides a *split-screen* option that lets the user view both the CM Fortran source code and the corresponding Fortran 77 source code simultaneously. See Figure 16.

Copyright © 1993 Thinking Machines Corporation

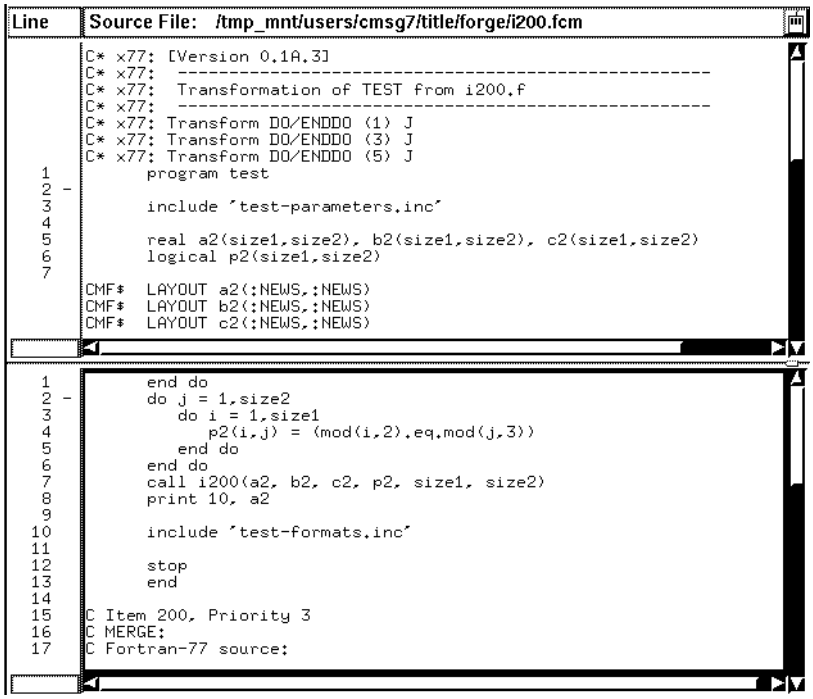

Figure 16. CM Fortran and Fortran 77 source code in a split screen.

Users can debug, visualize data, or obtain performance data in terms of either the CM Fortran or the Fortran 77 source code.

6.5 On-Line Documentation and Help

The CM-5 provides complete on-line documentation for its software. CMview, Thinking Machines Corporation's on-line documentation product, lets users display any CM manual in a format optimized for on-line viewing. Figure 17 shows a sample Table of Contents page from a manual.

Copyright © 1993 Thinking Machines Corporation

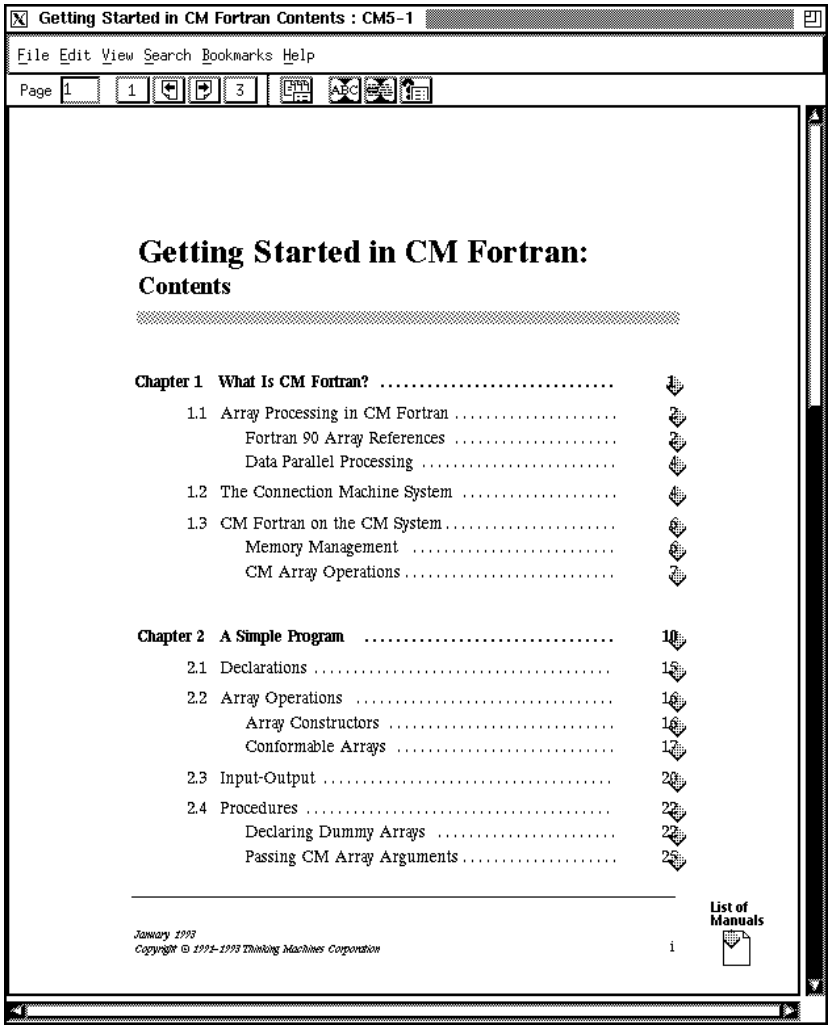

Figure 17. A Table of Contents page displayed on-line via CMview. Users can click on a hypertext marker to display the desired section.

In addition, CMview lets users:

- Follow hypertext links from table of contents and index entries, cross-references, and specially entered hypertext markers.
- Search through the entire collection of manuals for a word or phrase.
- Print all or part of any manual on a laser printer.
- Put their own bookmarks (like dog-earing a page) and notes (like scribbling in the margin) on any page of any manual.

Copyright © 1993 Thinking Machines Corporation

CMview is accessible by a menu selection from within Prism. It is also available by issuing the command **`cmview`** from an xterm within the X Window System.

Prism's own comprehensive help system is based on the same technology. Help is available for each pulldown menu and dialog box. Users in search of more information can follow the hypertext link within the help file to display the section of the on-line Prism manual in which the topic is discussed in detail.

Copyright © 1993 Thinking Machines Corporation