Chapter 2

The Basics

This chapter covers the basic procedures you need to know in order to create, compile, execute, and debug programs on the CM-5. If you have used the UNIX operating system, much of what is covered in this chapter will be familiar to you. If you are not familiar with UNIX, you should begin by reading an introductory book about the operating system.

All the procedures discussed in this chapter can be carried out from within Prism, the CM-5's programming environment; Chapter 3 describes Prism.

Your CM-5 should have sample programs you can use, if you want to try running a program on the computer before writing one yourself. Typically, these programs should be in the directory **`/usr/examples`**; if this directory does not exist, check with your system administrator for the location of sample programs on your system.

In this guide, we refer to a program called **`primes`**, which is a simple data parallel CM Fortran program for calculating and displaying prime numbers. By default, this program, along with its source code, is in the directory **`/usr/examples/prism`** on a partition manager. Check with your system administrator if it isn't there. We provide the source code in Appendix B, in case you can't find it on-line.

We suggest that you copy the source file, **`primes.fcm`**, to your home directory, from which you can compile it and use it within Prism, following the directions we provide in this chapter.

Copyright © 1993–1994 Thinking Machines Corporation

2.1 The User's View of the CM-5

As we mentioned in Chapter 1, a CM-5 has one or more partition managers, each of which controls some number of parallel processing nodes. These partitions operate independently of each other, although data can be passed back and forth between partitions. Your system administrator determines the exact configuration of your site's CM-5.

Each partition manager is a host on a network. You can log in to a partition manager over the network to gain access to the nodes that the partition manager controls, as well as to all the CM-5's I/O devices. Your system administrator can determine which users can log in to a given partition manager.

In Figure 3, the CM-5 has two partition managers, named Mars and Venus; each is currently managing a partition of 256 nodes; this CM-5 has the optional vector-unit hardware, so each partition can also be viewed as containing 1024 vector units. The system also has control processors managing some I/O peripherals, and another control processor that is dedicated as a system console.

You execute your program from the partition manager. The partition manager downloads code to the processors and broadcasts identical memory

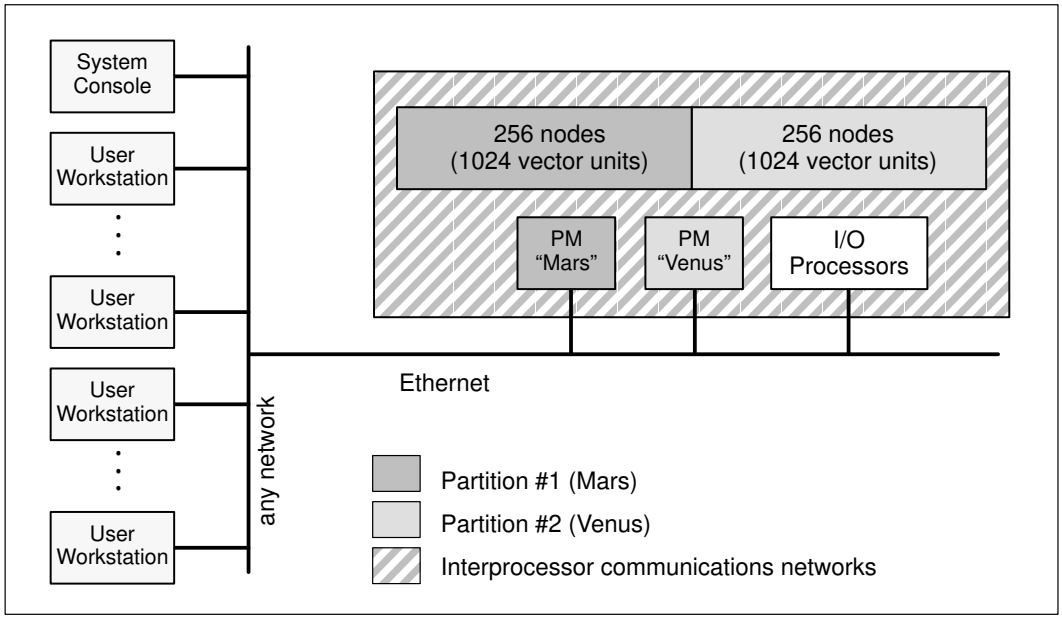

Figure 3. A sample CM-5 system.

Copyright © 1993–1994 Thinking Machines Corporation

maps to each processor (node or vector unit). The processors then execute the code, each acting on its own data and executing computations and branches accordingly. Multiple users can be logged in to a partition manager at the same time, running processes under timesharing. You can also issue commands over the network to submit your program to a batch queue for execution.

2.2 Gaining Access

To gain access to a partition manager, you must know its name. Find out the names of partition managers from your system administrator. The system administrator can also tell you if you have permission to use the partition manager.

From your terminal or workstation on the network, you can gain access to the partition manager via the UNIX command **`rlogin`** or **`rsh`**.

2.2.1 The rlogin Command

Use the **`rlogin`** command to log in to a partition manager remotely across the network. To log in to the partition manager Mars, for example, issue the command

```
% rlogin mars
```

You will then see messages from the partition manager; it will also ask you to confirm your terminal type. For example,

```
term(xterm)
```

asks if you are using an X terminal. If the terminal type is correct, simply press your **Return** key. If it isn't, enter the correct terminal type and press **Return**. (If you don't know your terminal type, ask your system administrator for help.)

You should then see the UNIX/CMOST prompt:

```
%
```


Copyright © 1993–1994 Thinking Machines Corporation

You are now logged in to the partition manager. You can issue any CMOST command and execute programs. For example,

```
% primes
```

executes the program **primes**.

To log out of the partition manager, issue the command

```
% logout
```

2.2.2 The rsh Command

Use the **rsh** command to execute a program or issue a CMOST command without logging in to the partition manager. The **rsh** command creates a "remote shell" on the partition manager. Simply specify the name of the partition manager, followed by the command or program name, on the **rsh** command line. For example,

```
% rsh mars primes
```

executes the program **primes** on Mars.

After the program executes, you are returned to your local UNIX shell.

2.3 Writing and Compiling Your Program

You can create your program as you normally do, on your local computer.

If you are creating a C* or CM Fortran program, you can compile it on the partition manager, or on any computer that has CMOST and the C* or CM Fortran compiler installed. To compile the **primes** program from the source file **primes.fcm**, for example, issue this command:

```
% cmf -cm5 -o primes primes.fcm
```

If you are creating a C or FORTRAN 77 program, you can also compile it on the partition manager, or on any computer that has a **cc** or **f77** compiler.

Generally it's a good idea not to compile on the partition manager (if you have a choice), to avoid overloading it.

Copyright © 1993–1994 Thinking Machines Corporation

2.3.1 Linking

A CM-5 executable program actually consists of two programs, one that executes on the partition manager and the other that executes on the CM processors. CMOST uses a special linker, **cmld**, to create this executable program. If you are developing your program in a high-level language, you typically don't need to deal with **cmld**. You would use it if, for example, you were creating a C* program that calls CM Fortran subroutines.

2.4 Executing Your Program

As shown above, you execute your program as you normally would on a UNIX system: you specify the name of the executable program at the UNIX prompt, or as an argument to the **rsh** command. The program then runs under timesharing on the CM-5.

2.4.1 Providing Accounting Information

On some CM-5 systems, users are required to provide an account ID when running a program, so that time on the system can be charged to the correct project. Obtain the account ID from your system administrator. You then specify this account ID before running a program by setting the UNIX environment variable **CM_ACCOUNT_ID** to your number. For example,

```
% setenv CM_ACCOUNT_ID 1234
```

To avoid having to reissue this command constantly, you can put it in your **.cshrc** file, so that it is executed automatically when you log in or issue an **rsh** command.

Note that this mechanism doesn't work if you are executing in batch mode; see below.

Copyright © 1993–1994 Thinking Machines Corporation

2.4.2 Executing in Batch Mode

You can also run your program in batch mode. You can use the Distributed Job Manager (DJM), the NQS batch system, or standard UNIX batch commands.

UNIX Batch Commands

The CM-5 supports the standard UNIX batch commands **at** and **batch**. For example,

```
% at 0815am Jan 24
at>  primes
at>  Ctrl-d
```

This causes the program **primes** to be executed at 8:15 a.m. on January 24th. *Ctrl-d* means "press the *d* key while holding down the *Control* key." This is the way to signal the end of information you are providing to the standard input.

DJM (Distributed Job Manager)

DJM (Distributed Job Manager) is a batch queuing system and interactive job manager. In it, you submit a request to a queue for execution. A request consists of one or more programs. When your request reaches the head of the queue, it is executed. Queues can be configured in various ways: for example, so that they are active only at certain times, or so that they accept requests only from certain users.

NOTE: If DJM is running at your site, you should typically use it to run your CM-5 programs. Otherwise, you might have difficulty obtaining the resources to run your program.

Use the **jsub** or **jrun** command to submit a job. You can specify the name of the program from the standard input, or in a script file. For example, if the script file's name is **/myname/primes_script**, you would issue the command as follows:

```
% jsub /myname/primes_script
```


CMOST Version 7.3, April 1994
Copyright © 1993–1994 Thinking Machines Corporation

If you don't include the appropriate options (see below), DJM then responds by asking questions about the job:

```
Number of processors (32)?
Estimated CPU time (5min)?
Estimated memory (128M)?
```

(The defaults in parentheses may differ at your site.)

Make your best estimates of the resources your job will need to run. If your estimates are too low, DJM will not effectively schedule your job; if they are too high, the job may be delayed unnecessarily, waiting for the resources to be available. The actual resources consumed by your job are printed at the end of the output file when the job finishes. You can use this information to provide better resource estimates the next time you run your job.

When you have answered DJM's questions, it prints a job ID:

```
Job submitted successfully. Job id is 43.
```

Output from the job is written to the file *myjob*`.o`*nnn*, where *myjob* is the name of the program that ran and *nnn* is its job ID. You can override this by specifying the **`-stdout`** option; this sends output to the standard output.

Use the **`-queue`** option to specify the queue to which the job is to be submitted. If you omit the option, DJM submits the job to the first queue that meets the requirements for your job. You can also use the **`-nproc`**, **`-cputime`**, and **`-mem`** options to specify your job's resources on the command line, rather than having to respond to DJM's questions.

To find out how to obtain status information from DJM, see Section 2.7.3. For complete information on DJM, see the manual *Using DJM on the CM-5*.

The NQS Batch System

NQS is a batch system that your system administrator can set up to control the execution of programs on the CM-5. As in DJM, you submit a request to a queue, which can be configured in various ways. When your request reaches the head of the queue, it is executed.

Use the **`qsub`** command to submit a request to a queue. Specify the name of the queue on the command line, using the **`-q`** option. You can specify the

Copyright © 1993–1994 Thinking Machines Corporation

name of the program from the standard input, as with the **at** command. For example, if the queue's name is **b_queue**, you could type this:

```
% qsub -q b_queue
  primes
  Ctrl-d
```

Alternatively, you can put the program's name in a script file. This is especially useful if you have multiple programs to run, or you have CMOST commands you want to issue along with the program. If the script file's name is **/myname/primes_script**, you would issue the command as follows:

```
% qsub -q b_queue /myname/primes_script
```

Use the **-I** option to specify your account ID, if one is required; obtain the account ID from your system administrator.

To find out how to obtain status information from NQS, see Section 2.7.4. For a complete discussion of NQS, see the manual *NQS for the CM-5*.

2.5 Debugging Your Program

The standard way of debugging both data parallel and message-passing programs on the CM-5 is to use Prism, as described in Chapter 3. Prism supports the CM Fortran, C*, C, and FORTRAN 77 languages.

2.5.1 The Errors File and Core Files

Certain errors can cause cores for some of the nodes to be dumped and an errors file to be generated. For example,

- a segmentation fault on the nodes
- a bus error on the nodes
- division by 0 on the nodes

These in turn can be caused by programming errors. For example, here are some C* errors that can cause core to be dumped:

- dereferencing a bad parallel pointer

CMOST Version 7.3, April 1994
Copyright © 1993–1994 Thinking Machines Corporation

- using an invalid send address
- doing out-of-bounds array referencing of parallel variables
- using an invalid parallel coordinate

Problems of this sort are generally signaled by an obscure CMOST message — for example,

```
CMOST: yellow interrupt
```

The errors file is called **CMTSD_errors**.*pid*, where *pid* is the process ID of the process you were running; it is located in the directory from which you executed your program. The file is generated by the timesharing daemon when a user program crashes; it contains a list of the status of each node (and of the partition manager, if an error was detected there). The file will tell you which nodes crashed, and give you some information about the crash, such as what memory address the node was trying to reference, whether it died because of a segmentation fault, and so on.

If your program was using the CM-5's vector units, additional error files are created in the directory from which you executed the program. Information on vector-unit registers is contained in the files **CMTSD_dp.pn***X*.*pid*, where *X* is the node identifier and *pid* is the process ID. These files contain an ASCII dump of the contents of every register on the vector units, for every node that has a different error. Similarly, vector-unit memory is captured in the files **CMTSD_stack.pn***X*.*pid* and **CMTSD_heap.pn***X*.*pid*.

The core files are named **CMTSD_core_pn***X*.*pid*, where *X* is the node identifier and *pid* is the process ID; they are located in the directory from which you executed your program. You may also see a regular core file for the partition-manager executable program.

To avoid wasting space, only unique cores are dumped. That is, if several nodes have the same error, only the core for the first node with that type of error is dumped. Also, the first node with no error (if there is such a node) will dump core.

You can examine these node core files using Node Prism, as described in Chapter 3.

If you don't want node core files generated, set the environment variable **CM_NO_PN_CORE** (if you are running the C shell) to any value. For example,

```
% setenv CM_NO_PN_CORE 1
```


Copyright © 1993–1994 Thinking Machines Corporation

You can also issue this command:

```
% limit coredumpsize 0
```

Issuing this command prevents the creation of core files from any program, whether or not it runs on the CM-5.

To re-enable the generation of CM-5 core files, unset the environment variable:

```
% unsetenv CM_NO_PN_CORE
```

2.5.2 Fortran Tracebacks

When a Fortran or CM Fortran program dies, it may generate a traceback. The traceback file is called *prog* **.trace**, where *prog* is the name of your program. If there are multiple tracebacks in the file (because the program crashed multiple times), the last traceback is the most recent; it tells you the routine that was executing on the partition manager when the program died.

Note, however, that the information may not be especially valuable, since the partition manager is not necessarily synchronized with the nodes. For example, the partition manager may continue working for a while before an error status from the nodes is propagated to it and the program halts. Therefore, the routine or instruction that is executing on the partition manager when the nodes die may have nothing to do with the error.

2.6 Timing a Program

CMOST provides a timing utility that lets you determine how much time any part of a program takes to execute on the nodes. The timer consists of a set of instructions that you insert at the appropriate places in your program.

Prism also provides performance data about your program. See Chapter 3.

The timing utility has these features:

- It can be used in C* or CM Fortran programs.

Copyright © 1993–1994 Thinking Machines Corporation

- A timer calculates total elapsed time used by the process (or any part of it) and the total amount of time the CM processors are active, with microsecond precision.
- Multiple timers can be active at the same time.
- Timers can be nested. This allows you, for example, to start one timer that will time the entire program, while using other timers to determine how different parts of the program contribute to the overall time.
- You can have up to 64 timers running in a program.

To get timings for the **primes** program, you would include these calls at the appropriate places in the code:

```
CALL CM_TIMER_CLEAR(0)            ! Initialize timer 0
CALL CM_TIMER_START(0)            ! Start timer 0
CALL CM_TIMER_STOP(0)             ! Stop the timer
CALL CM_TIMER_PRINT(0)            ! Print the results
```

For complete information on using the timing utility from CM Fortran or C*, see the user's guides for those languages.

CMMD provides comparable functions for use within message-passing programs. See the *CMMD User's Guide* for more information.

2.7 Getting Information

CMOST provides commands you can use to find out information about the CM-5 and processes running on it. It is especially important to be able to get information about the CM-5's configuration, since the system administrator can easily reconfigure it to add or remove partition managers, change the number of nodes in a partition, and so on.

Copyright © 1993–1994 Thinking Machines Corporation

2.7.1 Finding Out about the CM-5's Configuration

To find out the general configuration of a CM-5, log in to a partition manager and issue the command **cmpartition**:

```
% cmpartition
```

The output might look like this:

```
Name     Partition Manager       Size  State
mars     mars.think.com          128   ACTIVE
venus    venus.think.com         128   ACTIVE
```

This describes a CM-5 system with two partition managers, Mars and Venus, each of which controls 128 nodes.

For more complete information, use the **list -l** option:

```
% cmpartition list -l

CM System "G1"
     256 Processors
     2 Partition Managers
         mars.think.com
         venus.think.com
Available PN Ranges:  All PNs in use
IOP Addresses  480

Name     Partition Manager      Size  State       Nodes     Description
mars     mars.think.com         128   ACTIVE      0-127     Batch only
venus    venus.think.com        128   ACTIVE      128-255
```

This provides the additional useful information that Mars is set up to run batch requests only.

Copyright © 1993–1994 Thinking Machines Corporation

2.7.2 Finding Out about the Status of the System

To find out information about the partition and the programs running on it, use the CMOST command **cmps**, which is modeled after the UNIX command **ps**. The command provides information about the partition on which you execute it. The output looks like this:

```
32 PN System, 29168K mem. free, 6160K VU mem. free, 2 procs, TS-9/13/92-23:00
Daemon up: 1 day,  2:57

USER   PID  CMPID   TIME  TEXT   ILH    ILS   IGS    IGH    VUS    VUH     COMMAND
mob    *900    1  234:13  896K  254K    56K    0K     0K    64K     8K     primes
rhv    7900    0   16:36  200K  230K    56K    0K     0K    64K     0K     wil
```

The first line of the **cmps** output provides general information about the partition, including the number of processing nodes (referred to as *PNs*) it contains and the amount of memory available on the nodes and the vector units.

The columns give information about each process:

- The **PID** and **CMPID** fields give the process's SunOS and CMOST process-ID number.
- The **TIME** field indicates the amount of time that the CMOST time-sharing daemon has made available to the process, regardless of whether the process actually used the nodes.
- The **TEXT** field indicates the total amount of program memory allocated to the process.
- The fields **ILH**, **ILS**, **IGS**, **IGH**, **VUS**, and **VUH** give specifics on stack and heap memory used by the process. Note these points:
  - The numbers reported are the maximum for any single node or (in the case of VUS and VUH), any single vector unit. Thus, in the case of VUS and VUH, the total amount of memory taken up per node is 4 times the total of these two fields.
  - Memory is allocated uniformly across the nodes, even in message-passing processes.

Copyright © 1993–1994 Thinking Machines Corporation

- To calculate the total amount of memory used by a process, you must multiply by the number of nodes or vector units in the partition on which it is running.

For more details on these fields, see the **cmps** man page on-line.

- The **COMMAND** field gives the name of the program being executed.

To find out information about the partition manager and the programs running on it, use the standard UNIX command **ps**.

If you are running under the X Window System, you can use the **xcmps** command to get a graphical display of CM-5 usage on your partition. Sample output is shown in Figure 4. By default, the display is updated every 10 seconds. For more details, see the **xcmps** man page on-line.

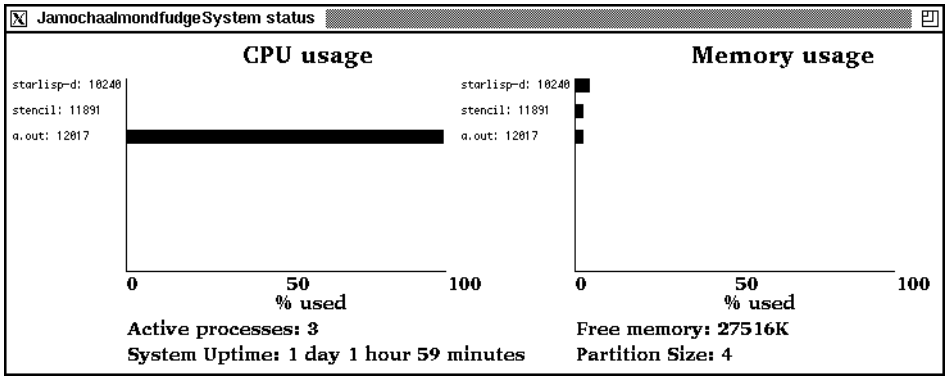

Figure 4. Output of the xcmps command.

CMOST Version 7.3, April 1994
Copyright © 1993–1994 Thinking Machines Corporation

2.7.3 Finding Out about the Status of DJM

Use the **jstat** command to find out information about the Distributed Job Manager (DJM). With no arguments, **jstat** lists the current jobs. For example,

```
% jstat
USER       JID MACH  SERV  PROCS   TIME    MEM  STATUS    COMMAND
sharon      43 cm5   ea     32     0:02   512M  Running   myjob
mark        24 cm5         256   240:00   600M  Que npr   rdt
```

This indicates that user Sharon has a job running on 32 processors of partition manager **ea** of a CM-5; its job ID is 43. The **PROCS**, **TIME**, and **MEM** fields indicate the actual number of processors and amount of CPU time and total amount of memory being used.

User Mark's job is queued; in this case, the **PROCS**, **TIME**, and **MEM** fields indicate the number of processors and amount of CPU time and memory requested when the job was submitted. The status **npr** indicates that the job is not running because the requested number of processors is not available.

Another use of the **jstat** command is to find out the available queues and the resource limits associated with them. Use the **-limits** option to obtain this information. For example,

```
% jstat limits
QUEUE     SRCH  MACH   TIME     MEM    NPROC   DED   FOR   UQUE   URUN   TRUN   TMEM
foreign     50   cm5   20min   128M   unlim   no    yes    999    999    999   unlim
default    100   cm5    1hr      2G     512   no    no       4      4      8   unlim
big        100   cm5   unlim   unlim   unlim   no    no       2      2      3   unlim
ded        100   cm5   unlim   unlim   unlim   yes   no       1      1      2   unlim
```

In the output,

- **SRCH** indicates the order in which DJM searches the queues to determine in which one a job will run. It searches from the lowest-numbered queue to the highest, and submits the job to the first queue that meets the job's requirements.

Copyright © 1993–1994 Thinking Machines Corporation

- **TIME**, **MEM**, and **NPROC** give the maximum usage characteristics of the queue.
- **DED** specifies whether or not the queue is for dedicated jobs (that is, jobs that require exclusive use of the partition).
- **FOR** indicates whether the queue is for "foreign" jobs — that is, jobs that are run directly on the machine without going through DJM.
- **UQUE** and **URUN** indicate the maximum number of jobs an individual user can have queued and running in this queue.
- **TRUN** and **TMEM** indicate the total number of jobs that can be running concurrently in this queue, and the total amount of memory they can consume.

2.7.4 Finding Out about the Status of NQS

To find out information about NQS, issue the command **qstat** on a partition manager. Use the **-u** option to obtain information about requests from a specific user. For example,

```
% qstat -u sharon
```

might produce this output:

```
=========================================================
NQS Version: 2      REQUESTS on diamond
=========================================================
REQUEST     NAME   OWNER   QUEUE   PRI   NICE   CPU     MEM     STATE
0@diamond. primes  sharon  pn_128   32     30   3600   UNLIMITED RUNNING
```

This shows that user Sharon has one request running on the queue **pn_128**. The other fields give information about characteristics of the request.

To find out information about batch queues, use the **-b** option. For example,

```
% qstat -b
```


Copyright © 1993–1994 Thinking Machines Corporation

might produce this response:

```
=========================================================
NQS Version: 2      REQUESTS on diamond.think.com
=========================================================
QUEUE NAME    STATUS   TOTAL   RUNNING  QUEUED   HELD  TRANSITION
---------------------------------------------------------------------
q1            AVAILBL    1       1/1      0        0       0
q2            AVAILBL    2       2/10     0        0       0
```

This indicates that there are two queues on the partition manager **diamond**; they are named **q1** and **q2**. The **TOTAL** field tells you the number of requests in the queue. The other fields give information about the status of the requests in the queue.

2.8 Obtaining On-Line Documentation

2.8.1 CMview

To use the CMview on-line documentation system, you must be running the X Window System. From an Xterm, issue the command **cmview** at the UNIX prompt:

```
% cmview
```

You get a startup message:

```
CMview, Version 1.1
Thinking Machines Corporation On-line Documentation
Copyright (C) 1994 by Thinking Machines Corporation

Starting up WorldView...
```

(*WorldView* is the name of the viewing software.) The CMview welcome screen is then displayed; see Figure 5. You can double-click with your left mouse button on any of the diamond-shaped link markers in this window to start navigating through the CMview system.

Copyright © 1993–1994 Thinking Machines Corporation

If you know which manual you want to view, you can go directly to its hyper-linked table of contents by specifying its keyword on the **cmview** command line. For example, to display the table of contents for this manual, issue this command:

```
% cmview -cm5ug
```

Issue **cmview** with the **-help** argument to display a list of the keywords for all CM-5 user manuals.

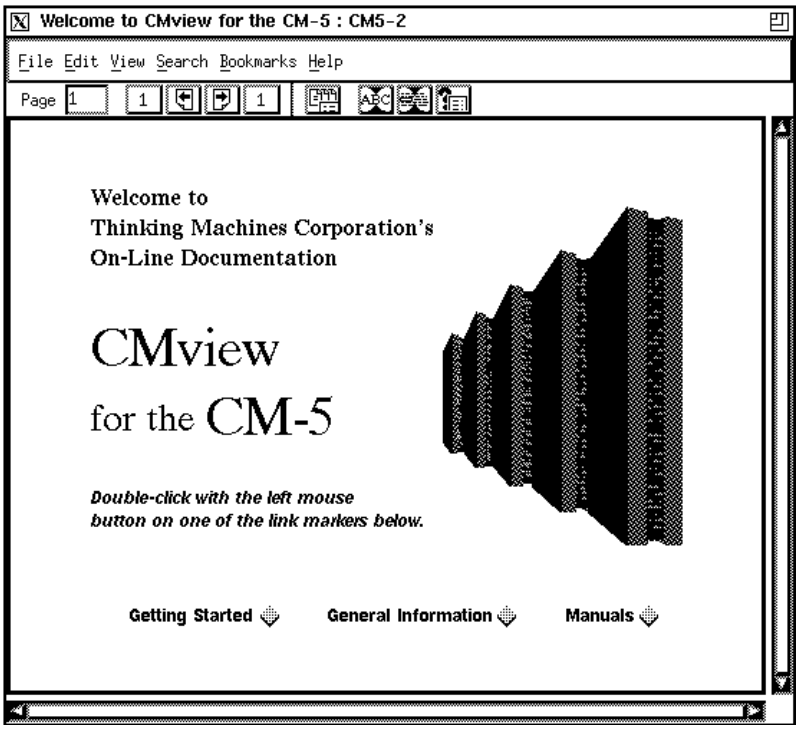

Figure 5. The CMview welcome screen.

To search through the collection of manuals for a word or phrase,

1. Left-click on **Search** in the menu bar of any CMview screen, then click on **Full-Text Query...**.
2. Enter the word or phrase you want to search for in the **Query:** box.

Copyright © 1993–1994 Thinking Machines Corporation

3. Click on **Query**. A list of documents is displayed beneath the **Query:** box. Next to each document is a number that represents the number of "matches" found within the document.

4. Click on the document you want to view, then click on **Open**. The document is displayed, with the first match highlighted in reverse video.

Figure 6 shows a **Full-Text Query** dialog box.

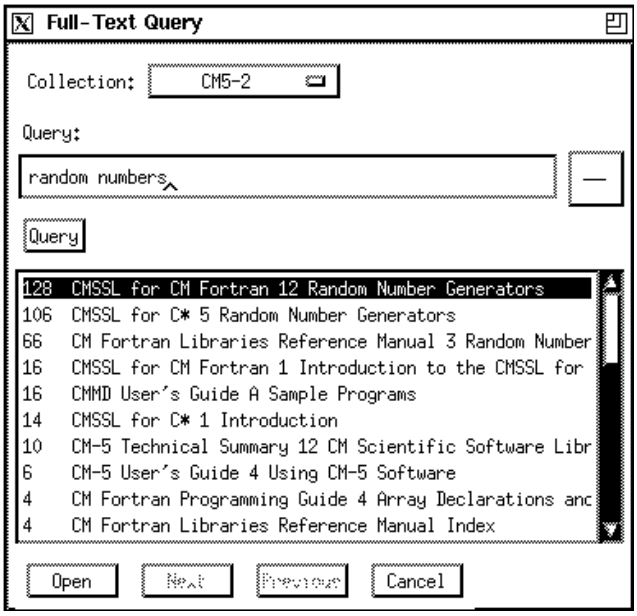

Figure 6. A full-text search in CMview.

2.8.2 Man Pages

Use the standard UNIX **`man`** command to view the man page for a UNIX or CMOST command or routine.

Copyright © 1993–1994 Thinking Machines Corporation

2.8.3 PostScript Files

PostScript versions for most CM-5 user manuals are available, typically in the directory **`/usr/doc`**. If they are not there, ask your system administrator for their location at your site. You can print these PostScript files on an Apple LaserWriter Plus or compatible printer, or view them on-line with PostScript viewing software such as GhostView.

CMOST Version 7.3, April 1994
Copyright © 1993–1994 Thinking Machines Corporation