Appendix A

The CMFS File System

A.1 The CMFS File System

The CMFS file system, developed for the CM-2 and CM-200, is a UNIX-like file system that can reside on the CMIO-bus data-storage devices, including the DataVault, CM-IOPG, and VMEIO host computer. Like the Scalable File System (SFS), this file system exploits the great speed and massive storage capabilities of the CM-5 I/O system.

The CMFS file system is also easy to use, especially if you are familiar with the standard UNIX file system; the two file systems are similar in several ways: they organize files into directories, use pathnames to identify them, and treat I/O devices as files. Unlike a UNIX file system, however, the CMFS file system does not have a single root directory. Rather, each CMIO-bus device can have a CMFS file system directory tree with its own root directory, as Figure 10 illustrates.

As described in Chapter 1, the CM-5 environment variable **`CMFS_PATHTYPE`** determines which file system to attempt to access, given a pathname: to use the CMFS file system, set **`CMFS_PATHTYPE`** to **`cmfs`** or **`mixed`** (see Section 1.4.1 of Chapter 1). The specific CMFS file system host and the directory within it are then defined by either a full pathname or a combination of a relative pathname and environment variables:

Copyright © 1994 Thinking Machines Corporation

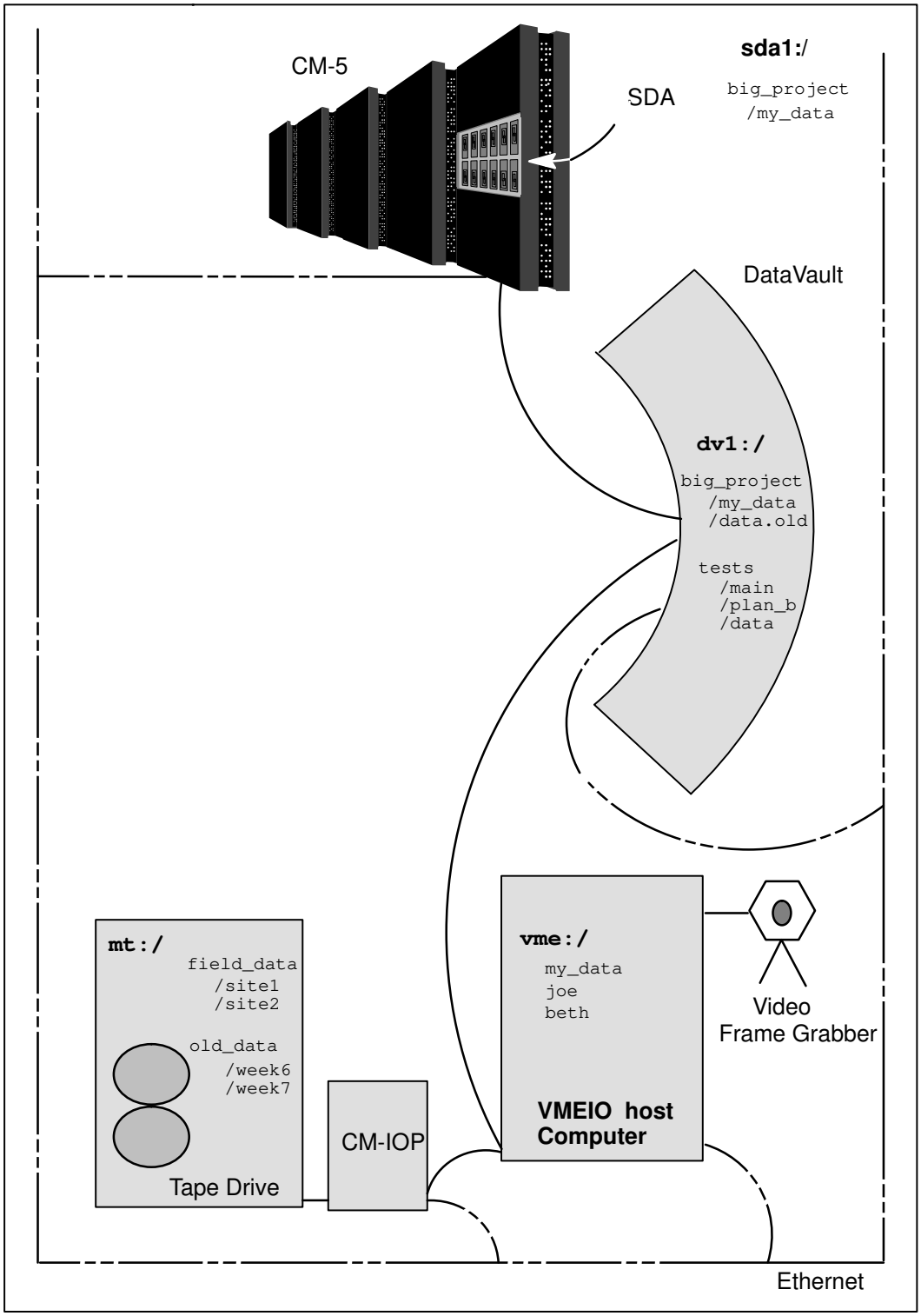

Figure 10. Files within a typical CMFS file system.

Copyright © 1994 Thinking Machines Corporation

- The CMFS file system identifies each data-storage device by its unique hostname — the name of the machine followed by a colon (:). A CMFS file's full pathname, therefore, includes a hostname component — for example, **dv1:/project/data**. (Because the hostname component contains a colon character, the name of the file itself cannot contain a colon.) If you specify a full pathname, therefore, the CM-5 system requires no further information in order to attempt to access the file.

- The CM environment variables **DVHOSTNAME** and **DVWD** define the default CMFS file system host and default working directory, respectively. When you do not give a full pathname, these environment variables are consulted.

  Set **DVHOSTNAME** from a C shell as follows:

  ```
  % setenv DVHOSTNAME default_hostname
  ```

  Set **DVHOSTNAME** from a Bourne shell as follows:

  ```
  $ DVHOSTNAME = default_hostname
  $ export DVHOSTNAME
  ```

  To specify your default current working directory in the CMFS file system, set **DVWD** from a C shell:

  ```
  % setenv DVWD new_working_directory
  ```

  or from a Bourne shell:

  ```
  $ DVWD = new_working_directory
  $ export DVWD
  ```

  new_working_directory may or may not include a hostname component. If it does not, *new_working_directory* is relative to the default CMFS host set by **DVHOSTNAME**. For example, if you set the environment variables as follows:

  ```
  % setenv DVHOSTNAME dv1
  % setenv DVWD /project
  ```

  and later refer to the file **data**, the system will interpret the full name of the file as **dv1:/project/data**. If you want instead to use the file **dv1:/tests/data**, either explicitly specify the file's full pathname or reset **DVWD** to **/tests**. If you want to use the file **dv2:/tests/data**, either explicitly specify the file's full pathname or reset **DVHOSTNAME** to **dv2**.

Copyright © 1994 Thinking Machines Corporation

If you do not set **DVHOSTNAME**, and if **DVWD**'s *new_working_directory* does not include a hostname component, the default CMFS hostname is the one listed with the CM-5 system kernel (see your system administrator). If none is listed with the kernel, the file **/usr/local/etc/dv_hostname** is consulted. If that file is missing, the default CMFS file system is the one on the local host.

For convenience, you can have your **.login** file or your **.cshrc** file set **DVWD** and/or **DVHOSTNAME** (as well as **CMFS_PATHTYPE**) each time you open a shell to work in the CM-5 programming environment.

A.2 CMFS Files

A *CMFS file* is one stored within the CMFS file system, regardless of which kind of computer generated its data and format:

- A *serial CMFS file* is formatted in the conventional manner, as by serial computers, consisting of only one stream of data. A file created on the CM-5 is serial in format to allow it to be easily shared among Connection Machine models and serial computers. A high-performance transposition mechanism automatically "parallelizes" the file's data when it is read into the CM-5, and "serializes" it again when it is written out to the file system.

- *Parallel CMFS files* are indigenous to the Connection Machine model CM-2 and CM-200. Consisting of many streams of data, one stream per CM-2 or CM-200 processor, they are specially formatted to take advantage of the machine's massive parallelism. Although files created on the CM-2 and CM-200 are parallel by default, you can choose to write data to them in serial format, or even to bypass parallel formatting altogether by creating the file in serial format; these options are discussed in the *Connection Machine I/O System Programming Guide, Version 6.1*.

It is easy to share files between a CM-2 or CM-200 and a CM-5 or a serial computer by using a CM-2 or CM-200 transposition routine. See Section A.2.2.

Copyright © 1994 Thinking Machines Corporation

A.2.1 Attribute Files

When a CMFS file is created, the CMFS file server process automatically creates the file's *attribute file*, which is named *.filename* **.attr** (and has permissions that match those of the data file). The first 256 bytes of the attribute file are reserved for use by the CMFS file system. You can use the rest of the attribute file to store information — most often it is specific to the application — that you want to record, back up, or keep in synchronization with the associated file.

The attribute file's original purpose was to store information about how its associated CMFS file's data is layed out in the Connection Machine's processors (collectively, this information is known as the file's *geometry*); this information is crucial to proper data placement on a CM-2 or CM-200, but on a CM-5 a different data layout mechanism is used. Therefore, if you intend to share a CMFS file between a CM-5 and a CM-2 or CM-200, you must be careful to keep track of the file's attribute file (for example, if you copy or move the CMFS file, also copy or move the file's attribute file), but if you do not intend to share the file between Connection Machines, you can choose to ignore CM-5 CMFS attribute files.

A.2.2 Sharing a CMFS File between a CM-5 and a CM-2 or CM-200

Using a CM-5 CMFS file (that is, a CMFS file created on a CM-5 system) on a CM-2 or CM-200 is straightforward: simply copy it to a file system accessible from the CM-2 or CM-200, and open it and read it as described in the *Connection Machine I/O System Programming Guide*, Version 6.1. Then perform any necessary data format conversions, as described below. (The attribute file of a CMFS file created on a CM-5 system contains "generic" geometry information, which enables the file to be compatible with any CM-2 or CM-200 VP set geometry (see Chapter 1 of the *Connection Machine I/O System Programming Guide*, Version 6.1).

Using a CM-2 or CM-200 file on a CM-5 system requires only that the file first be transposed to serial format on the CM-2 or CM-200 — see Chapter 2 of the *Connection Machine I/O System Programming Guide*, Version 6.1 — and its data format converted if necessary, as described below.

Copyright © 1994 Thinking Machines Corporation

Data Format Conversion

- Data used by the CM-2 or CM-200 must be in little-endian (right-to-left) byte order and in IEEE floating-point format. Data used by the CM-5 must be in big-endian (left-to-right) byte order and in IEEE floating-point format. You can perform byte conversion either on the CM-5, using a standard byte-swapping technique, or on the CM-2 or CM-200, using **`CMFS_cm_to_standard_byte_order`**.

- Floating-point data used by the CM-2, CM-200, or CM-5 must be in IEEE format. Therefore, a file shared between the CM-2 or CM-200 and the CM-5 needs conversion only if, for example, it was written directly from a CM-2 or CM-200 VAX front end, or if the CM-2 or CM-200 had previously converted the file's floating-point format via a call to the CM-2/200 CMFS routine **`CMFS_convert_ieee_to_vax`**. If conversion is necessary, read the file into the CM-2 or CM-200 and call the CM-2/200 CMFS routine **`CMFS_convert_vax_to_ieee`** before writing it back out. (See the *Connection Machine I/O System Programming Guide,* Version 6.1, for more information about these conversion routines.)

Copyright © 1994 Thinking Machines Corporation