Chapter 7

Parallel I/O

The utility library procedures in this chapter support CM parallel I/O. Parallel I/O refers to transferring data in multiple streams between the CM processing elements and an external device. The procedures fall into two categories:

- Operations on files of the CM file system (Sections 7.1 through 7.4)
- I/O via sockets and devices, including CM-HIPPI (Section 7.5)

These procedures provide an interface to procedures in the CM file system libraries, allowing you to perform parallel I/O. For more information on the CM file system and libraries, see the CM I/O documentation for CM-5.

Language Comparison

The CM Fortran utility library I/O procedures are, for most purposes, redundant with the CM Fortran language statements **READ** and **WRITE**. Using **READ** and **WRITE** on a file opened with the keyword **PHYSICAL** produces fast I/O, comparable to that produced by the fixed machine size (FMS) procedures, described below. We recommend that you use the **READ** and **WRITE** statements instead of utility library I/O, except in these situations:

- Continue using the utility library procedures to perform parallel I/O via sockets or devices, such as CM-HIPPI. **READ** and **WRITE** support parallel I/O only to the Scalable Disk Array (SDA) or DataVault.
- Use the utility library procedures to read files previously written with utility write procedures.

Copyright © 1994 Thinking Machines Corporation

Restriction

You cannot call the I/O utility library routines from the CM Fortran nodal execution model, the local part of the global/local model, or from the CM Fortran simulator. For more information about CM Fortran execution models, see the *CM Fortran User's Guide*.

7.1 Opening, Closing, and Removing CM Files

The subroutines in this section open, close, or remove (unlink) CM files.

7.1.1 Opening a CM File

CMF_FILE_OPEN opens the CM file specified by **PATH** (a character string) and associates the file with the integer argument **UNIT**. The value returned in **IOSTAT** indicates whether the operation succeeded.

```
CALL CMF_FILE_OPEN( UNIT, PATH, IOSTAT )
```

CMF_FILE_OPEN_READONLY is similar to **CMF_FILE_OPEN**, except that it opens the file in read-only mode. An error is reported if an attempt is made to write to this file.

Unit Numbers

For each CM file to be opened, you choose a unit number in the range 0 through 2999. The number becomes associated with a file when it is used as the **UNIT** argument to **CMF_FILE_OPEN**. You then supply that unit number to other I/O procedures when you wish to operate on this file. The total number of open files is limited by the operating system, not by CM Fortran.

These utility library unit numbers are not related to standard CM Fortran unit numbers, as described in the *CM Fortran Language Reference Manual* for the **READ** and **WRITE** statements.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

Error Status

All the parallel I/O procedures take an **INTEGER*4 IOSTAT** argument, into which the error status of the operation is placed:

- A positive value in **IOSTAT** indicates success.
- A negative value in **IOSTAT** indicates failure.
- For the parallel read utilities only (see below), a zero value in **IOSTAT** indicates an end-of-file condition. For the other utilities, a zero value in **IOSTAT** indicates success.

Other than sign or zero, there is no significance to any of the particular values returned.

7.1.2 Closing a CM File

CMF_FILE_CLOSE closes the file associated with **UNIT**.

```
CALL CMF_FILE_CLOSE( UNIT, IOSTAT )
```

7.1.3 Removing a CM File

CMF_FILE_UNLINK removes the entry for the file specified by **PATH** from the file's directory.

```
CALL CMF_FILE_UNLINK( PATH, IOSTAT )
```

If this entry is the last link to the file and no process has the file open, then the file is deleted and all resources associated with it are reclaimed. If, however, the file is open in any process, the resource reclamation is delayed until the file is closed, even though the directory entry has disappeared.

Copyright © 1994 Thinking Machines Corporation

CMF_FILE_OPEN

Opens a CM file and attaches the file to the **UNIT**.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_FILE_OPEN( UNIT, PATH, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant in the range 0 through 2999.
PATH	A **CHARACTER** string containing the pathname for the file to be opened.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation will be placed. A positive or zero value indicates success; a negative value indicates failure.

Result

None.

Description

This subroutine opens a CM file and attaches the file to the **UNIT**. You must supply this unit number to other CM file system procedures when you wish to operate on this file. Note that the CM file system unit numbers have no relation to standard Fortran unit numbers used with the **OPEN** statement.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

See Also

CMF_FILE_OPEN_READONLY
CMF_FILE_FDOPEN
CMF_FILE_CLOSE
CMF_FILE_UNLINK

Copyright © 1994 Thinking Machines Corporation

CMF_FILE_OPEN_READONLY

Opens a CM file for read-only access and attaches the file to the **UNIT**.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_FILE_OPEN( UNIT, PATH, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant in the range 0 through 2999.
PATH	A **CHARACTER** string containing the pathname for the file to be opened.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation will be placed. A positive or zero value indicates success; a negative value indicates failure.

Result

None.

Description

This subroutine opens a CM file in read-only mode and attaches the file to the **UNIT**. You must supply this unit number to other CM file system procedures when you wish to operate on this file. A run-time error is reported if an attempt is made to write to this file. Note that the CM file system unit numbers have no relation to standard Fortran unit numbers used with the **OPEN** statement.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

See Also

```
CMF_FILE_OPEN
CMF_FILE_FDOPEN
CMF_FILE_CLOSE
CMF_FILE_UNLINK
```

Copyright © 1994 Thinking Machines Corporation

CMF_FILE_CLOSE

Closes a CM file.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_FILE_CLOSE( UNIT, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant containing a valid unit number. The unit number is established in the call to **CMF_FILE_OPEN** that creates the file in the CM file system. This **UNIT** number has no relation to the standard Fortran unit numbers used in the **OPEN** statement.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation will be placed. A positive or zero value indicates success, and a negative value indicates failure.

Result

None.

Description

Closes a file in the CM file system.

See Also

CMF_FILE_FDOPEN
CMF_FILE_OPEN
CMF_FILE_UNLINK

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

CMF_FILE_UNLINK

Removes a file from a directory.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_FILE_UNLINK( PATH, IOSTAT )
```

Arguments

PATH	A **CHARACTER** string containing the pathname of the file to be removed.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation will be placed. A positive or zero value indicates success; a negative value indicates failure.

Result

None.

Description

CMF_FILE_UNLINK removes the entry for the file **PATH** from the file's directory. If this entry was the last link to the file and no process has the file open, then the file is deleted and all resources associated with the file are reclaimed. If, however, the file was open in any process, the actual resource reclamation is delayed until the file is closed, even though the directory entry has disappeared.

See Also

CMF_FILE_CLOSE
CMF_FILE_OPEN

Copyright © 1994 Thinking Machines Corporation

7.2 Reading and Writing CM Files

The CM Fortran utility library procedures read or write CM arrays in parallel, that is, in multiple streams directly between the memory of CM processors and a CM file on a storage device.

These procedures are available in three variants, reflecting different tradeoffs between speed and flexibility. The variants are distinguished by suffix (or lack of): no-suffix or generic, FMS, or SO. They take the same arguments.

```
CALL CMF_CM_ARRAY_TO_FILE    ( UNIT, ARRAY, IOSTAT )
CALL CMF_CM_ARRAY_FROM_FILE  ( UNIT, ARRAY, IOSTAT )

CALL CMF_CM_ARRAY_TO_FILE_FMS     ...
CALL CMF_CM_ARRAY_FROM_FILE_FMS   ...

CALL CMF_CM_ARRAY_TO_FILE_SO      ...
CALL CMF_CM_ARRAY_FROM_FILE_SO    ...
```

Always read a file with the same variant that was used to write it.

Arguments

UNIT — Integer variable, parameter, or literal constant that is a valid unit number in the range 0:2999. This is the unit number that became associated with a file by the initial call to **CMF_FILE_OPEN** (see Section 7.1).

ARRAY — CM array of any type and layout. This array is the source or destination of the I/O operation.

IOSTAT — **INTEGER*4** variable. The value returned in this argument indicates the error status of the operation:

- A positive value in **IOSTAT** indicates success.
- A negative value in **IOSTAT** indicates failure.
- For the parallel read utilities only, a zero value in **IOSTAT** indicates an end-of-file condition. For the other utilities, a zero value in **IOSTAT** indicates success.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

NOTE

Like all procedures in the utility library, these I/O procedures cannot be used with any array that is aligned with another array of higher rank or aligned with nonzero axis offset(s) with any other array.

7.2.1 Portability and Speed

The three sets of read/write procedures give different combinations of speed and portability. The FMS (fixed machine size) routines are the fastest but the least flexible. The SO (serial order) routines are slower but the most portable across CM configurations and execution models. The generic (no-suffix) routines are a compromise between the two for general-purpose use.

The generic and FMS procedures treat file data in a *parallel* order, which we call the *geometry* of the file. File geometry reflects the shape and layout of the first array written to that file. In consequence:

- All subsequent writes to a parallel-ordered file must be of arrays of the same shape and layout as the first, and any read operation from the file must be to an array of the same shape and layout.
- Parallel-ordered files may contain extraneous data (padding) in scattered locations. As long as you observe the restrictions on using the FMS and generic routines (summarized in the table below), the padding is handled transparently when the file is read.

In the interest of speed, the FMS procedures impose the further restriction that write and read operations of a CM file must be from the same execution model, the same machine size (number of processing elements), and, if using SDA, the same SDA size. Hence the term *fixed machine size*.

Copyright © 1994 Thinking Machines Corporation

The generic procedures, in contrast, are limited only by array shape and layout. They can write and then read from different execution models and partition sizes.

The SO procedures treat file data in *serial* order. Serial ordering is the same as array-element ordering and the same as the output of the Fortran **WRITE** statement. For example, the SO utility stores the elements of array **A(2,3)** in the following order:

```
A(1,1)
A(2,1)
A(1,2)
A(2,2)
A(1,3)
A(2,3)
```

Unlike the generic and FMS variants, the SO procedures do not pad files. Because they read and write only the array elements, not any extraneous data, these utilities operate independently of array shape and layout, and are completely portable across CM execution models and partition sizes. The SO utilities are also compatible with parallel I/O via sockets and devices (Section 7.5).

NOTE: Always read a CM file with the same mechanism that was used to write it. That is, read with **READ** if the file was written with **WRITE**; read with **CMF_CM_ARRAY_FROM_FILE_SO** if the file was written with **CMF_CM_ARRAY_TO_FILE_SO**, and so on. Mixing **READ** and **WRITE** with the SO utilities may cause an end-of-file error and is not supported.

The following table summarizes the behavior of the three variants of the parallel read/write utility procedures. The "portability" entry refers to restrictions on subsequent reads and writes of a CM file after the first array has been written to it.

Copyright © 1994 Thinking Machines Corporation

Table 2. Variants of `CMF_CM_ARRAY_TO/FROM_FILE` on CM-5.

	FMS	Generic	SO
File order	parallel	parallel	serial
Padding	yes	yes	no
Portability	same partition size same exec. model same array shape same array layout	any partition size any exec. model same array shape same array layout	any partition size any exec. model any array shape any array layout

Copyright © 1994 Thinking Machines Corporation

CMF_CM_ARRAY_TO_FILE

Writes the contents of an array to a CM file.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_CM_ARRAY_TO_FILE( UNIT, SOURCE, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant containing a valid unit number. The unit number is specified in the call to **CMF_FILE_OPEN** that creates the file in the CM file system. This **UNIT** number has no relation to the standard Fortran unit numbers used in the **OPEN** statement.
SOURCE	A CM array of any type.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation will be placed. A positive or zero value indicates success, and a negative value indicates failure.

Result

None.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

Description

This subroutine writes the contents of the CM array **SOURCE** to the CM file specified by **UNIT**. The array is written to the file in a parallel order that reflects the geometry of the array. This allows **CMF_CM_ARRAY_TO_FILE** to transfer the array substantially faster than **CMF_CM_ARRAY_TO_FILE_SO**. However, files written with **CMF_CM_ARRAY_TO_FILE** cannot be transferred outside the CM file system and are subject to the restriction that all arrays written to the file must have the same shape as the first array written to the file.

More specialized parallel order files can be written and read with **CMF_CM_ARRAY_TO/FROM_FILE_FMS**. These subroutines write and read arrays to CM files more quickly than **CMF_CM_ARRAY_TO/FROM_FILE**, but they are more severely restricted in their use. In particular, a file must be read on the same size partition size as it was written from.

Serial order files are written and read with **CMF_CM_ARRAY_TO/FROM_FILE_SO**. Such files can be transferred outside the CM file system (for instance, to a serial UNIX file server) or directly to an I/O device such as a HIPPI interface or a CM socket.

Restriction

- The CM Fortran utility library procedures do not operate correctly on arrays that have been aligned with other arrays of greater rank or with other arrays of the same rank but with offsets for the individual axes.

See Also

CMF_CM_ARRAY_FROM_FILE
CMF_CM_ARRAY_FROM_FILE_FMS
CMF_CM_ARRAY_FROM_FILE_SO
CMF_CM_ARRAY_TO_FILE_FMS
CMF_CM_ARRAY_TO_FILE_SO

Copyright © 1994 Thinking Machines Corporation

CMF_CM_ARRAY_FROM_FILE

Reads an array from a CM file.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_CM_ARRAY_FROM_FILE( UNIT, DEST, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant containing a valid unit number. The unit number is specified in the call to **CMF_FILE_OPEN** that creates the file in the CM file system. This **UNIT** number has no relation to standard Fortran unit numbers used in the **OPEN** statement.
DEST	A CM array of any type. The **DEST** array must be identical in shape and type to the array that is to be transferred from the file.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation will be placed. A positive value indicates success, a value of zero indicates an end-of-file condition, and a negative value indicates failure.

Result

None.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

Description

This subroutine reads an array from the CM file specified by **UNIT** into the CM array specified by **DEST**. The file must have been written by **CMF_CM_ARRAY_TO_FILE**. **CMF_CM_ARRAY_TO_FILE** writes the array to the file in a parallel order that reflects the geometry of the array. The files written by **CMF_CM_ARRAY_TO_FILE** cannot be transferred outside the CM system and are subject to the following restrictions:

- The **DEST** array must be identical in shape and type to the array in the file.
- Files read by **CMF_CM_ARRAY_FROM_FILE** must have been written by **CMF_CM_ARRAY_TO_FILE**. The partition size, array layout, and execution model can be different between the write and read operations.

More specialized parallel order files can be written and read with **CMF_CM_ARRAY_TO/FROM_FILE_FMS**. These two subroutines write and read arrays to CM files more quickly than **CMF_CM_ARRAY_TO/FROM_FILE**, but they are more restricted in their use. In particular, a file must be read on the same size partition as it was written from.

Serial order files are written with **CMF_CM_ARRAY_TO/FROM_FILE_SO**. Such files can be transferred outside the CM file system (for instance, to a serial UNIX file server) or directly to an I/O device such as a HIPPI interface or a CM socket.

Restriction

- The CM Fortran utility library procedures do not operate correctly on arrays that have been aligned with other arrays of greater rank or with other arrays of the same rank but with offsets for the individual axes.

See Also

CMF_CM_ARRAY_FROM_FILE_FMS
CMF_CM_ARRAY_FROM_FILE_SO
CMF_CM_ARRAY_TO_FILE
CMF_CM_ARRAY_TO_FILE_FMS
CMF_CM_ARRAY_TO_FILE_SO

Copyright © 1994 Thinking Machines Corporation

CMF_CM_ARRAY_TO_FILE_FMS

Writes the contents of an array to a CM file for a fixed machine size.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_CM_ARRAY_TO_FILE_FMS( UNIT, SOURCE, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant containing a valid unit number. The unit number is specified in the call to **CMF_FILE_OPEN** that creates the file in the CM file system. This **UNIT** number has no relation to standard Fortran unit numbers used in the **OPEN** statement.
SOURCE	A CM array of any type.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation will be placed. A positive or zero value indicates success, and a negative value indicates failure.

Result

None.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

Description

This subroutine writes the contents of the CM array **SOURCE** to the CM file specified by **UNIT**. The array is written to the file in a parallel order that reflects the geometry of the array, the array layout, and the size of the machine executing the program. This allows **CMF_CM_ARRAY_TO_FILE_FMS** to transfer the array substantially faster than **CMF_CM_ARRAY_TO_FILE** or **CMF_CM_ARRAY_TO_FILE_SO**. However, parallel order files cannot be transferred outside the CM file system, and FMS files are subject to the following restrictions when being used on the CM:

- All arrays written to the file must have the same shape and layout as the first array written.
- The array must be read by **CMF_CM_ARRAY_FROM_FILE_FMS** under the same execution model (SPARC nodes or vector units) that was used in writing the file.
- The array must be read into a partition with the same physical size as the one that was used to write the file.
- The array from the file must be read into an array on the CM with the same shape, type, and layout.

More general parallel order files that have some of the performance advantages of FMS files but less severe restrictions can be written and read with **CMF_CM_ARRAY_TO/FROM_FILE**.

Serial order files that can be transferred outside the CM file system or directly to an I/O device such as a HIPPI interface or a CM socket, can be written and read with **CMF_CM_ARRAY_TO/FROM_FILE_SO**.

Restriction

- The CM Fortran utility library procedures do not operate correctly on arrays that have been aligned with other arrays of greater rank or with other arrays of the same rank but with offsets for the individual axes.

Copyright © 1994 Thinking Machines Corporation

See Also

CMF_CM_ARRAY_FROM_FILE
CMF_CM_ARRAY_FROM_FILE_FMS
CMF_CM_ARRAY_FROM_FILE_SO
CMF_CM_ARRAY_TO_FILE
CMF_CM_ARRAY_TO_FILE_SO

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

CMF_CM_ARRAY_FROM_FILE_FMS

Reads an array from a CM file to a CM array for a fixed machine size.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_CM_ARRAY_FROM_FILE_FMS( UNIT, DEST, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant containing a valid unit number. The unit number is specified in the call to **CMF_FILE_OPEN** that creates the file in the CM file system. This **UNIT** number has no relation to standard Fortran unit numbers used in the **OPEN** statement.
DEST	A CM array of any type. The **DEST** array must be identical in shape, type, and layout to the array that is to be transferred from the file.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation will be placed. A positive value indicates success, a value of zero indicates an end-of-file condition, and a negative value indicates failure.

Result

None.

Copyright © 1994 Thinking Machines Corporation

Description

This subroutine reads the contents of an array from the file specified by **UNIT** and stores it in the **DEST** CM array. The file must have been written by **CMF_CM_ARRAY_TO_FILE_FMS**. **CMF_CM_ARRAY_TO_FILE_FMS** writes the file in a parallel order that reflects the geometry of the array, the array layout, and the size of the partition executing the program. This allows **CMF_CM_ARRAY_FROM_FILE_FMS** to transfer the array substantially faster than **CMF_CM_ARRAY_FROM_FILE** or **CMF_CM_ARRAY_FROM_FILE_SO**. However, parallel order files cannot be transferred outside the CM file system, and FMS files are subject to the following restrictions when being read back into the CM:

- The partition used to read the array must be the same physical size as the one that was used to write the file. In addition, the same execution model (SPARC nodes or vector units) must be used when writing and reading.
- As mentioned in the description of the **DEST** argument above, the destination array on the CM and the array that is to be transferred from the file must be identical in shape, type, and layout.

More general parallel order files that have some of the performance advantages of FMS files but less severe restrictions can be written with **CMF_CM_ARRAY_TO_FILE** and read with **CMF_CM_ARRAY_FROM_FILE**.

Serial order files that can be transferred outside the CM file system or directly to an I/O device such as a HIPPI interface or a CM socket, can be written with **CMF_CM_ARRAY_TO_FILE_SO** and read with **CMF_CM_ARRAY_FROM_FILE_SO**.

Restriction

- The CM Fortran utility library procedures do not operate correctly on arrays that have been aligned with other arrays of greater rank or with other arrays of the same rank but with offsets for the individual axes.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

See Also

```
CMF_CM_ARRAY_FROM_FILE
CMF_CM_ARRAY_FROM_FILE_SO
CMF_CM_ARRAY_TO_FILE
CMF_CM_ARRAY_TO_FILE_FMS
CMF_CM_ARRAY_TO_FILE_SO
```


Copyright © 1994 Thinking Machines Corporation

CMF_CM_ARRAY_TO_FILE_SO

Writes the contents of an array to a CM file in serial order.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_CM_ARRAY_TO_FILE_SO( UNIT, SOURCE, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant containing a valid unit number. The unit number is specified in the call to **CMF_FILE_OPEN** that creates the file in the the CM file system. This **UNIT** number has no relation to standard Fortran unit numbers used in the **OPEN** statement.
SOURCE	A CM array of any type, shape, or layout.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation will be placed. A positive or zero value indicates success, and a negative value indicates failure.

Result

None.

Description

CMF_CM_ARRAY_TO_FILE_SO writes the contents of the **SOURCE** array to a CM file specified by **UNIT** in normal Fortran (or "serial") order. These arrays must be read back into CM memory with **CMF_CM_ARRAY_FROM_FILE_SO**.

NOTE: Use **CMF_CM_ARRAY_TO_FILE_SO** only to write arrays that are to be read with **CMF_CM_ARRAY_FROM_FILE_SO**. Do not use it to write arrays to be read by the CM Fortran **READ** statement.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

The array elements are stored in a serial order file in the same order as those written with the Fortran **WRITE** statement. For example, the array **A(2,3)** is stored in the following order:

A(1,1)
A(2,1)
A(1,2)
A(2,2)
A(1,3)
A(2,3)

Files containing arrays in serial order are portable to any CM configuration and may also be transferred outside the CM system to other file systems. However, arrays that are written by **CMF_CM_ARRAY_TO_FILE_SO** directly to a device (a HIPPI interface or a CM socket) may contain some "padding." The padding consists of extra elements added to the array when it is allocated in CM memory or when the I/O system writes it out. The padding is handled transparently by the CM Fortran I/O utilities that read and write in parallel order (**CMF_CM_ARRAY_TO/FROM_FILE** and **CMF_CM_ARRAY_TO/FROM_FILE_FMS**), and the padding is stripped from the arrays when they are written to a file in serial order by **CMF_CM_ARRAY_TO_FILE_SO**. However, when **CMF_CM_ARRAY_TO_FILE_SO** is used to write an array to a device, extraneous data may be added to the end of the array. Padding is not present if a device is written from an array whose size (number of elements) is a power of 2 and an integer multiple of the size of the partition (number of nodes) executing the program.

Restriction

- The CM Fortran utility library procedures do not operate correctly on arrays that have been aligned with other arrays of greater rank or with other arrays of the same rank but with offsets for the individual axes.

Copyright © 1994 Thinking Machines Corporation

See Also

CMF_CM_ARRAY_FROM_FILE
CMF_CM_ARRAY_FROM_FILE_FMS
CMF_CM_ARRAY_FROM_FILE_SO
CMF_CM_ARRAY_TO_FILE
CMF_CM_ARRAY_TO_FILE_FMS

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

CMF_CM_ARRAY_FROM_FILE_SO

Reads an array from a CM file to a CM array in serial order.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_CM_ARRAY_FROM_FILE_SO( UNIT, DEST, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant containing a valid unit number. The unit number is specified in the call to **CMF_FILE_OPEN** that creates the file in the CM file system. This **UNIT** number has no relation to standard Fortran unit numbers used in the **OPEN** statement.
DEST	A CM array of any type, shape, or layout.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation will be placed. A positive value indicates success and specifies the number of bytes read from the file, a value of zero indicates an end-of-file condition, and a negative value indicates failure.

Result

None.

Description

CMF_CM_ARRAY_FROM_FILE_SO reads an array from the CM file specified by **UNIT** into the CM array **DEST**. This procedure expects the array to be in normal Fortran array-element (or "serial") order, that is, an array written with **CMF_CM_ARRAY_TO_FILE_SO**.

Copyright © 1994 Thinking Machines Corporation

NOTE: Use **CMF_CM_ARRAY_FROM_FILE_SO** only to read arrays written by **CMF_CM_ARRAY_TO_FILE_SO**. Do not use it to read arrays written by the CM Fortran **WRITE** statement.

Restriction

- The CM Fortran utility library procedures do not operate correctly on arrays that have been aligned with other arrays of greater rank or with other arrays of the same rank but with offsets for the individual axes.

See Also

CMF_CM_ARRAY_FROM_FILE
CMF_CM_ARRAY_FROM_FILE_FMS
CMF_CM_ARRAY_TO_FILE
CMF_CM_ARRAY_TO_FILE_FMS
CMF_CM_ARRAY_TO_FILE_SO

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

7.3 Manipulating CM Files

The procedures in this section rewind, seek within, or truncate a CM file.

7.3.1 Determining File Geometry

All seek, rewind, and truncate operations on CM files must be preceded by a read or write operation. It is necessary first to determine the geometry of a newly opened file, even a serial-order file, by performing a read or write of the file.

A further restriction is that the element size in any file manipulation (rewind, seek, or truncate) must be the same as the element size in the read or write operation that initially determined the file's geometry in that session.

7.3.2 Rewinding a File

CMF_FILE_REWIND moves the file pointer to the beginning of a CM file.

```
CALL CMF_FILE_REWIND( UNIT, IOSTAT )
```

7.3.3 Seeking within a File

Four procedures serve to reposition the file pointer in a CM file:

```
CALL CMF_FILE_LSEEK( UNIT, OFFSET, IOSTAT )

CALL CMF_FILE_LSEEK_FMS( UNIT, OFFSET, IOSTAT )

CALL CMF_FILE_LLSEEK( UNIT, OFFSET, IOSTAT )

OFFSET = CMF_SIZEOF_ARRAY_ELEMENT( ARRAY )
```

CMF_FILE_LSEEK and **CMF_FILE_LLSEEK** operate on files written with the generic and SO write procedures; use **CMF_FILE_LSEEK_FMS** on files written with the FMS write procedure.

Copyright © 1994 Thinking Machines Corporation

The seek utilities operate slightly differently on serial-ordered files (those written with the SO procedure), compared with parallel-ordered files (those written with the generic or FMS procedure).

- In serial-ordered files, **CMF_FILE_LSEEK** and **CMF_FILE_LLSEEK** can move the file pointer either to an array boundary or to an arbitrary element. For the **OFFSET** argument, use the number of bytes in the array's element type times the number of elements to traverse. (The **LLSEEK** variant takes a 64-bit offset argument, supported on the CM-5 with vector units only.)

- In parallel-ordered files, **CMF_FILE_LSEEK**, **CMF_FILE_LLSEEK**, and **CMF_FILE_LSEEK_FMS** move the file pointer *only* from one array to another within a file. You cannot move it to an arbitrary element. To compute the offset, you need not specify the size of the array(s), since this information is contained in the file geometry. You need specify only the size of an array's elements, using **CMF_SIZEOF_ARRAY_ELEMENT**.

For example, suppose that a parallel-ordered file associated with unit 29 was created with three successive writes, of array **A**, then array **B**, then array **C**. To position the file pointer to the beginning of array **B**, use:

```
CALL CMF_FILE_REWIND( 29, IOSTAT )
SIZEOF_A = CMF_SIZEOF_ARRAY_ELEMENT( A )
CALL CMF_FILE_LSEEK( 29, SIZEOF_A, IOSTAT )
```

To move the file pointer to the beginning of array **C**, add the return values of **CMF_SIZEOF_ARRAY_ELEMENT** for the two arrays to be traversed:

```
CMF_FILE_REWIND( 29, IOSTAT )
SIZEOF_A = CMF_SIZEOF_ARRAY_ELEMENT( A )
SIZEOF_B = CMF_SIZEOF_ARRAY_ELEMENT( B )
CMF_FILE_LSEEK( 29, SIZEOF_A + SIZEOF_B, IOSTAT )
```

To read arrays **A** and **C**:

```
CALL CMF_FILE_REWIND( 29, IOSTAT )
CALL CMF_CM_ARRAY_FROM_FILE( 29, DEST_A, IOSTAT )
CALL CMF_FILE_LSEEK( 29, SIZEOF_B, IOSTAT )
CALL CMF_CM_ARRAY_FROM_FILE( 29, DEST_C, IOSTAT )
```

If these arrays had been written with **CMF_CM_ARRAY_TO_FILE_FMS**, you would have used the FMS variant of the seek and read procedures.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

The type of the **OFFSET** argument to **CMF_FILE_LSEEK** is **INTEGER*4**. This permits an offset in the range of 2 gigabytes. On CM systems that support the **INTEGER*8** type, you can move the file pointer over a larger range by using the comparable procedure **CMF_FILE_LLSEEK**, which takes an **INTEGER*8** variable as the **OFFSET** argument.

Since **CMF_SIZEOF_ARRAY_ELEMENT** returns an **INTEGER*4** result, the result must be coerced to **INTEGER*8** before being passed as an argument to **CMF_FILE_LLSEEK**. This can be done with the intrinsic function **INT**, supplying the bit-length **64** as the **KIND** argument:

```
... INT(CMF_SIZEOF_ARRAY_ELEMENT(ARRAY),64)
```

or by assignment of the function result to an **INTEGER*8** temp.

```
INTEGER*8 TEMP
...
TEMP = CMF_SIZEOF_ARRAY_ELEMENT( ARRAY )
CALL CMF_FILE_LLSEEK( UNIT, TEMP, IOSTAT )
```

7.3.4 Changing the Size of a File

CMF_FILE_TRUNCATE and **CMF_FILE_LTRUNCATE** increase or decrease the size of a CM file:

```
CALL CMF_FILE_TRUNCATE( UNIT, LENGTH, IOSTAT )

CALL CMF_FILE_LTRUNCATE( UNIT, LENGTH, IOSTAT )

LENGTH = CMF_SIZEOF_ARRAY_ELEMENT( ARRAY )
```

These subroutines change the size of the file specified by **UNIT** to **LENGTH**. If the file is smaller than **LENGTH**, it is extended to **LENGTH**. If the file is larger than **LENGTH**, it is truncated and the extra data is lost. The file must be open when **CMF_FILE_TRUNCATE** or **CMF_FILE_LTRUNCATE** is called.

CMF_FILE_TRUNCATE accepts an **INTEGER*4** value for **LENGTH**. This allows a length in the range of 2 gigabytes. On CM systems that support the **INTEGER*8** type, you can specify a longer length by using **CMF_FILE_LTRUNCATE**, which accepts an **INTEGER*8** value for **LENGTH**.

Like the seek procedures described above, **CMF_FILE_TRUNCATE** and **CMF_FILE_LTRUNCATE** behave slightly differently with serial-ordered and

Copyright © 1994 Thinking Machines Corporation

parallel-ordered files, and you compute the **LENGTH** argument differently for the two kinds of files.

- **CMF_FILE_TRUNCATE** and **CMF_FILE_LTRUNCATE** can extend or truncate a serial-ordered file either by whole arrays or by an arbitrary number of array elements. For the **LENGTH** argument, supply the number of bytes in the array's element type times the number of elements desired.

- **CMF_FILE_TRUNCATE** and **CMF_FILE_LTRUNCATE** can extend or truncate a parallel-ordered file only by whole arrays, not by an arbitrary number of array elements. To compute the **LENGTH** argument, you need not specify the size of the array(s), since this information is contained in the file geometry. Specify only the size of array elements, using **CMF_SIZEOF_ARRAY_ELEMENT**.

You compute the **LENGTH** argument for parallel-ordered files in the same way as the **OFFSET** argument for the seek procedures, shown above. And, as with the seek procedures, you can extend or truncate a file by more than one array by invoking **CMF_SIZEOF_ARRAY_ELEMENT** on several arrays in succession, adding the returned values, and supplying the result as the **LENGTH** argument.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

CMF_FILE_REWIND

Moves a file pointer to the beginning of a CM file.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_FILE_REWIND( UNIT, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant containing a valid unit number. The unit number is established in the call to **CMF_FILE_OPEN** that creates the file in the CM file system. This **UNIT** number has no relation to the standard Fortran unit numbers used in the **OPEN** statement.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation will be placed. A positive or zero value indicates success; a negative value indicates failure.

Result

None.

Description

This subroutine moves the file pointer for the CM file associated with the **UNIT** number to the beginning of that file. You can use it to reset the file pointer before setting it to a specific location in the file with **CMF_FILE_LSEEK**, **CMF_FILE_LSEEK_FMS**, or **CMF_FILE_LLSEEK**. See the description of **CMF_FILE_LSEEK** for more information.

Restriction

- Before calling **CMF_FILE_REWIND** on a newly opened file, you must first perform a read or write on the file.

Copyright © 1994 Thinking Machines Corporation

See Also

CMF_FILE_LSEEK
CMF_FILE_LSEEK_FMS
CMF_FILE_OPEN

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

CMF_FILE_LSEEK

Offsets the file pointer a specified distance within a CM file.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_FILE_LSEEK( UNIT, OFFSET, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant containing a valid unit number. The unit number is established in the call to **CMF_FILE_OPEN** that creates the file in the CM file system. This **UNIT** number has no relation to the standard Fortran unit numbers used in the **OPEN** statement.
OFFSET	**INTEGER*4**. An offset from the current position in the specified file. This argument is specified differently for serial order and parallel order files. See the **Description** below for more information.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation is placed. A positive or zero value indicates success; a negative value indicates failure.

Result

None.

Description

This subroutine moves the file pointer within a CM file. The new position is computed by adding the specified offset to the current position of the file pointer.

Copyright © 1994 Thinking Machines Corporation

Seeking in Serial Order Files

For serial order files (those written with **CMF_CM_ARRAY_TO_FILE_SO**), the offset is given in bytes. To calculate the offset, multiply the number of bytes in the array's element type by the number of elements to traverse. This allows you to seek to an arbitrary element in the file.

The **INTEGER*4** type of the **OFFSET** argument permits an offset in the range of 2 gigabytes. On CM systems that support the **INTEGER*8** type, you can move the file pointer over a larger range by using the comparable procedure **CMF_FILE_LLSEEK**, which takes an **INTEGER*8** variable as the **OFFSET** argument.

Seeking in Parallel Order Files

Before performing a seek operation on a parallel order file (one written with **CMF_CM_ARRAY_TO_FILE**), you must perform a read or write operation on the file to establish its geometry.

For parallel order files, you can only seek to the beginning of whole arrays. To compute the offset, you need not specify the size of the array, since this information is contained in the file geometry. You need specify only the size of an array's elements using the utility function **CMF_SIZEOF_ARRAY_ELEMENT**. To seek over multiple arrays, call the function **CMF_SIZEOF_ARRAY_ELEMENT** on each array and add the results (see the example below).

Example

These examples illustrate the use of **CMF_SIZEOF_ARRAY_ELEMENT** to seek over parallel order files. For these examples, assume that a file associated with unit 29 has had three arrays written to it: **A**, **B**, and then **C**. Assume also that we have determined **SIZEOF_A** and **SIZEOF_B** by calling **CMF_SIZEOF_ARRAY_ELEMENT** on each array. Then, to position the file pointer to the beginning of array **A**, call

```
CALL CMF_FILE_REWIND( 29, IOSTAT )
```

To position the file pointer to the beginning of array **B**, use:

```
CALL CMF_FILE_REWIND( 29, IOSTAT )
CALL CMF_FILE_LSEEK( 29, SIZEOF_A, IOSTAT )
```

To position the file pointer to the beginning of array **C**, use:

```
CALL CMF_FILE_REWIND( 29, IOSTAT )
CALL CMF_FILE_LSEEK( 29, SIZEOF_A + SIZEOF_B, IOSTAT )
```


Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

To read arrays **A** and **C**:

```
CALL CMF_FILE_REWIND( 29, IOSTAT )
CALL CMF_CM_ARRAY_FROM_FILE ( 29, DEST_ARRAY, IOSTAT )
CALL CMF_FILE_LSEEK( 29, SIZEOF_B, IOSTAT )
CALL CMF_CM_ARRAY_FROM_FILE( 29, DEST_ARRAY, IOSTAT )
```

Restrictions

- For parallel order files, the element size of a seek operation must be the same as the element size of the read or write operation that established the geometry of the file when it was first opened.
- If the file was written with **CMF_CM_ARRAY_TO_FILE_FMS**, you must use **CMF_FILE_LSEEK_FMS** to perform a seek operation on it.

See Also

CMF_FILE_LLSEEK
CMF_FILE_LSEEK_FMS
CMF_FILE_OPEN
CMF_SIZEOF_ARRAY_ELEMENT

Copyright © 1994 Thinking Machines Corporation

CMF_FILE_LSEEK_FMS

Moves the file pointer a specified distance in a file written by **CMF_CM_ARRAY_TO_FILE_FMS**.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_FILE_LSEEK_FMS( UNIT, OFFSET, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant containing a valid unit number. The unit number is established in the call to **CMF_FILE_OPEN** that creates the file in the CM file system. This **UNIT** number has no relation to the standard Fortran unit numbers used in the **OPEN** statement.
OFFSET	**INTEGER*4**. An offset from the current position in the specified file. See the **Description** and **Example** sections below for information on how to specify this argument.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation is placed. A positive or zero value indicates success; a negative value indicates failure.

Result

None.

Description

This subroutine moves the file pointer to array boundaries within a CM file that has been written by **CMF_CM_ARRAY_TO_FILE_FMS**. The new position is computed by adding the specified offset to the current position of the file pointer. See the description of **CMF_CM_ARRAY_TO_FILE_FMS** for details on FMS procedures.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

Before performing a seek operation on any parallel order CM file, you must perform a read or write operation on the file to establish its geometry. Note that the element size of any later file operation, such as seeking, must be the same as the element size of the read or write operation that established the geometry of the file when it was first opened.

To compute the offset, you need not specify the size of the array, since this information is contained in the file geometry. You need specify only the size of an array's elements using the function **CMF_SIZEOF_ARRAY_ELEMENT**. To seek over multiple arrays, call the utility function **CMF_SIZEOF_ARRAY_ELEMENT** on each array and add the results. (See the example given below.)

Example

These examples illustrate the use of **CMF_SIZEOF_ARRAY_ELEMENT** to seek over FMS parallel order files. For these examples, assume that **CMF_CM_ARRAY_TO_FILE_FMS** has been used to write three arrays (**A**, **B**, and then **C**) to the file associated with unit 29. Assume also that we have determined **SIZEOF_A** and **SIZEOF_B** by calling **CMF_SIZEOF_ARRAY_ELEMENT** on each array. Then, to position the file pointer to the beginning of array **A**, call

```
CALL CMF_FILE_REWIND( 29, IOSTAT )
```

To position the file pointer to the beginning of array **B**, use:

```
CALL CMF_FILE_REWIND( 29, IOSTAT )
CALL CMF_FILE_LSEEK_FMS( 29, SIZEOF_A, IOSTAT )
```

To position the file pointer to the beginning of array **C**, use:

```
CALL CMF_FILE_REWIND( 29, IOSTAT )
CALL CMF_FILE_LSEEK_FMS( 29, SIZEOF_A + SIZEOF_B, IOSTAT )
```

To read arrays **A** and **C**:

```
CALL CMF_FILE_REWIND( 29, IOSTAT )
CALL CMF_CM_ARRAY_FROM_FILE_FMS ( 29, DEST_ARRAY, IOSTAT )
CALL CMF_FILE_LSEEK_FMS( 29, SIZEOF_B, IOSTAT )
CALL CMF_CM_ARRAY_FROM_FILE_FMS ( 29, DEST_ARRAY, IOSTAT )
```


Copyright © 1994 Thinking Machines Corporation

Restriction

- If the file was written with **CMF_CM_ARRAY_TO_FILE_SO** or **CMF_CM_ARRAY_TO_FILE**, use **CMF_FILE_LSEEK** or **CMF_FILE_LLSEEK** to perform a seek operation on it.

See Also

CMF_CM_ARRAY_TO_FILE_FMS
CMF_FILE_LSEEK
CMF_FILE_LLSEEK
CMF_FILE_OPEN
CMF_SIZEOF_ARRAY_ELEMENT

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

CMF_FILE_LLSEEK

Offsets the file pointer a specified distance within a CM file.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_FILE_LLSEEK( UNIT, OFFSET, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant containing a valid unit number. The unit number is established in the call to **CMF_FILE_OPEN** that creates the file in the CM file system. This **UNIT** number has no relation to the standard Fortran unit numbers used in the **OPEN** statement.
OFFSET	**INTEGER*8**. An offset from the current position in the specified file. This argument is specified differently for serial order and parallel order files. See the **Description** below for more information.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation is placed. A positive or zero value indicates success; a negative value indicates failure.

Result

None.

Description

This subroutine moves the file pointer within a CM file. The new position is computed by adding the specified offset to the current position of the file pointer.

Copyright © 1994 Thinking Machines Corporation

Seeking in Serial Order Files

For serial order files (those written with **CMF_CM_ARRAY_TO_FILE_SO**), the offset is given in bytes. To calculate the offset, multiply the number of bytes in the array's element type by the number of elements to traverse. This allows you to seek to an arbitrary element in the file.

The **INTEGER*8** type of the **OFFSET** argument permits an offset larger than that accepted by the comparable procedure **CMF_FILE_LSEEK**. The latter procedure is supported on all CM systems, but the **OFFSET** argument is limited by its 32-bit type to 2 gigabytes. **CMF_FILE_LLSEEK** permits moving the file pointer over a larger range, but note the restriction below on CM system support for the **INTEGER*8** type.

Seeking in Parallel Order Files

Before performing a seek operation on a parallel order file (one written with **CMF_CM_ARRAY_TO_FILE**), you must perform a read or write operation on the file to establish its geometry.

For parallel order files, you can only seek to the beginning of whole arrays. To compute the offset, you need not specify the size of the array, since this information is contained in the file geometry. You need specify only the size of an array's elements using the utility function **CMF_SIZEOF_ARRAY_ELEMENT**. To seek over multiple arrays, call the function **CMF_SIZEOF_ARRAY_ELEMENT** on each array and add the results. (See the example given under **CMF_FILE_LSEEK**.)

Since **CMF_SIZEOF_ARRAY_ELEMENT** returns an **INTEGER*4** result, you must coerce the result to **INTEGER*8** before passing it as an argument to **CMF_FILE_LLSEEK**.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

Example

The use of **CMF_SIZEOF_ARRAY_ELEMENT** to seek over parallel order files is shown under the description of **CMF_FILE_LSEEK**. To use **CMF_FILE_LLSEEK** for this purpose, you must coerce the result of **CMF_SIZEOF_ARRAY_ELEMENT** to **INTEGER*8**. This can be done with the intrinsic function **INT**, supplying the bit-length **64** as the **KIND** argument:

```
... INT(CMF_SIZEOF_ARRAY_ELEMENT(ARRAY),64)
```

or by assignment of the function result to an **INTEGER*8** temp.

```
INTEGER*8 TEMP
...
TEMP = CMF_SIZEOF_ARRAY_ELEMENT( ARRAY )
CALL CMF_FILE_LLSEEK( UNIT, TEMP, IOSTAT )
```

Restrictions

- The **INTEGER*8** type is supported only on CM-5 systems with vector units.
- For parallel order files, the element size of a seek operation must be the same as the element size of the read or write operation that established the geometry of the file when it was first opened.
- If the file was written with **CMF_CM_ARRAY_TO_FILE_FMS**, you must use **CMF_FILE_LSEEK_FMS** to perform a seek operation on it.

See Also

CMF_FILE_LSEEK
CMF_FILE_LSEEK_FMS
CMF_FILE_OPEN
CMF_SIZEOF_ARRAY_ELEMENT

Copyright © 1994 Thinking Machines Corporation

CMF_SIZEOF_ARRAY_ELEMENT

Returns the size of an array for use with the parallel I/O procedures **CMF_FILE_LSEEK**, **CMF_FILE_LLSEEK**, **CMF_FILE_LSEEK_FMS**, **CMF_FILE_TRUNCATE**, and **CMF_FILE_LTRUNCATE**.

Syntax

```
INCLUDE 'CMF_defs.h'

LENGTH = CMF_SIZEOF_ARRAY_ELEMENT( ARRAY )
```

Arguments

ARRAY	A CM array of any type.

Result

Integer scalar of type **INTEGER*4**.

Description

This function returns the size of a single array element of **ARRAY**.

This return value can be passed to **CMF_FILE_LSEEK**, **CMF_FILE_LSEEK_FMS**, **CMF_FILE_TRUNCATE**, or **CMF_FILE_LTRUNCATE** to specify the length of an array. To pass the return value to **CMF_FILE_LLSEEK** or **CMF_FILE_LTRUNCATE**, first coerce it to type **INTEGER*8**; see the man page for **CMF_FILE_LLSEEK** or **CMF_FILE_LTRUNCATE** for more information. **CMF_SIZEOF_ARRAY_ELEMENT** can be called for multiple arrays stored in a file and the return values added together to compute file positions for these procedures.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

See Also

```
CMF_FILE_LSEEK
CMF_FILE_LSEEK_FMS
CMF_FILE_TRUNCATE
CMF_FILE_LTRUNCATE
```

Copyright © 1994 Thinking Machines Corporation

CMF_FILE_TRUNCATE

Change the size of a CM file.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_FILE_TRUNCATE( UNIT, LENGTH, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant containing a valid unit number. The unit number is established in the call to **CMF_FILE_OPEN** that creates the file in the CM file system. This **UNIT** number has no relation to the standard Fortran unit numbers used in the **OPEN** statement.
LENGTH	An **INTEGER*4** specifying the new length of the file. This argument is specified differently for serial order and parallel order files. See the **Description** section below for more information.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation is placed. A positive or zero value indicates success; a negative value indicates failure.

Result

None.

Description

This subroutine changes the size of the specified CM file to **LENGTH**. The file is extended or shortened by the difference between the **LENGTH** argument and the file's current length. If the file was previously larger than **LENGTH**, the extra data is lost. If the file was previously smaller than **LENGTH**, the file is extended to **LENGTH**.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

The **INTEGER*4** type of the **LENGTH** argument permits a length in the range of 2 gigabytes. On CM systems that support the **INTEGER*8** type, you can specify a longer length by using the comparable procedure **CMF_FILE_LTRUNCATE**, which takes an **INTEGER*8** variable as the **LENGTH** argument.

Deriving LENGTH for Serial Order Files

For serial order files (those written with **CMF_CM_ARRAY_TO_FILE_SO**), the length is given in bytes. To calculate the length for the file, multiply the number of bytes in the array's element type by the number of array elements to be contained in the file.

Deriving LENGTH for Parallel Order Files

Before performing a truncate operation on a parallel order file (one written with **CMF_CM_ARRAY_TO_FILE** or **CMF_CM_ARRAY_TO_FILE_FMS**), you must perform a read or write operation on the file to establish its geometry.

Parallel order files can only be reduced or enlarged by whole arrays. To compute the length of the array, you need not specify the size of the array, since this information is contained in the file geometry. You need specify only the size of an array's elements using the function **CMF_SIZEOF_ARRAY_ELEMENT**. You can increase or decrease the size of the file by more than one array by calling **CMF_SIZEOF_ARRAY_ELEMENT** on several arrays in succession, adding the returned values together, and supplying the cumulative result as the **LENGTH** argument. (See the example given under the description of **CMF_FILE_LSEEK**.)

Restrictions

- **CMF_FILE_TRUNCATE** requires the file to be open for writing.
- For parallel order files, the element size of a truncate operation must be the same as the element size of the read or write operation that established the geometry of the file when it was first opened.

Copyright © 1994 Thinking Machines Corporation

See Also

CMF_FILE_LTRUNCATE
CMF_FILE_LSEEK
CMF_FILE_LSEEK_FMS
CMF_FILE_OPEN
CMF_FILE_REWIND
CMF_SIZEOF_ARRAY_ELEMENT

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

CMF_FILE_LTRUNCATE

Change the size of a CM file.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_FILE_LTRUNCATE( UNIT, LENGTH, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant containing a valid unit number. The unit number is established in the call to **CMF_FILE_OPEN** that creates the file in the CM file system. This **UNIT** number has no relation to the standard Fortran unit numbers used in the **OPEN** statement.
LENGTH	An **INTEGER*8** specifying the new length of the file. This argument is specified differently for serial order and parallel order files. See the **Description** section below for more information.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation is placed. A positive or zero value indicates success; a negative value indicates failure.

Result

None.

Description

This subroutine changes the size of the specified CM file to **LENGTH**. The file is extended or shortened by the difference between the **LENGTH** argument and the file's current length. If the file was previously larger than **LENGTH**, the extra data is lost. If the file was previously smaller than **LENGTH**, the file is extended to **LENGTH**.

Copyright © 1994 Thinking Machines Corporation

The **INTEGER*8** type of the **LENGTH** argument permits a length larger than that accepted by the comparable procedure **CMF_FILE_TRUNCATE**. The latter procedure is supported on all CM systems, but the **LENGTH** argument is limited by its 32-bit type to 2 gigabytes. **CMF_FILE_LTRUNCATE** permits a longer file length, but note the restriction below on CM system support for the **INTEGER*8** type.

Deriving LENGTH for Serial Order Files

For serial order files (those written with **CMF_CM_ARRAY_TO_FILE_SO**), the length is given in bytes. To calculate the length for the file, multiply the number of bytes in the array's element type by the number of array elements to be contained in the file.

Deriving LENGTH for Parallel Order Files

Before performing a truncate operation on a parallel order file (one written with **CMF_CM_ARRAY_TO_FILE** or **CMF_CM_ARRAY_TO_FILE_FMS**), you must perform a read or write operation on the file to establish its geometry.

Parallel order files can only be reduced or enlarged by whole arrays. To compute the length of the array, you need not specify the size of the array, since this information is contained in the file geometry. You need specify only the size of an array's elements using the function **CMF_SIZEOF_ARRAY_ELEMENT**. You can increase or decrease the size of the file by more than one array by calling **CMF_SIZEOF_ARRAY_ELEMENT** on several arrays in succession, adding the returned values together, and supplying the cumulative result as the **LENGTH** argument. (See the example given under the description of **CMF_FILE_LSEEK**.)

Restrictions

- The **INTEGER*8** type is supported only on CM-5 systems with vector units.
- **CMF_FILE_LTRUNCATE** requires the file to be open for writing.
- For parallel order files, the element size of a truncate operation must be the same as the element size of the read or write operation that established the geometry of the file when it was first opened.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

See Also

CMF_FILE_TRUNCATE
CMF_FILE_LSEEK
CMF_FILE_LSEEK_FMS
CMF_FILE_OPEN
CMF_FILE_REWIND
CMF_SIZEOF_ARRAY_ELEMENT

Copyright © 1994 Thinking Machines Corporation

7.4 Example of CM File Operations

The following program writes five arrays into a file and then reads the third one:

```
      PROGRAM READ_RECORD
      INTEGER FILE_UNIT, IOSTAT, RECORD
      REAL A(8192)
      DOUBLE PRECISION B(8192)
      COMPLEX C(8192), DEST(8192)
      DOUBLE COMPLEX D(8192)
      LOGICAL E(8192)
      INCLUDE 'CMF_defs.h'

C Initialize variables

      FILE_UNIT = 13
      IOSTAT    = 0
      A = [1:8192]
      B = [1:8192]
      C = CMPLX([1:8192],[1:8192])
      D = DCMPLX([1:8192],[1:8192])
      E = MOD([1:8192],2).EQ.0
      DEST = 0.0

C Open a file and write to it; add failure tests to each
C operation if desired.

      CALL CMF_FILE_OPEN(FILE_UNIT,'my-file',IOSTAT)
      IF (IOSTAT<0) PRINT *,"File open failed",IOSTAT

      CALL CMF_CM_ARRAY_TO_FILE(FILE_UNIT,A,IOSTAT)
      CALL CMF_CM_ARRAY_TO_FILE(FILE_UNIT,B,IOSTAT)
      CALL CMF_CM_ARRAY_TO_FILE(FILE_UNIT,C,IOSTAT)
      CALL CMF_CM_ARRAY_TO_FILE(FILE_UNIT,D,IOSTAT)
      CALL CMF_CM_ARRAY_TO_FILE(FILE_UNIT,E,IOSTAT)

C Rewind the file

      CALL CMF_FILE_REWIND(FILE_UNIT,IOSTAT)
```


Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

```
C Compute the offset to the third record

      RECORD = CMF_SIZEOF_ARRAY_ELEMENT(A) +
     $     CMF_SIZEOF_ARRAY_ELEMENT(B)

C Seek to the third record

      CALL CMF_FILE_LSEEK(FILE_UNIT,RECORD,IOSTAT)

C Read the third record into array DEST

      CALL CMF_CM_ARRAY_FROM_FILE(FILE_UNIT,DEST,IOSTAT)

      STOP
      END
```


Copyright © 1994 Thinking Machines Corporation

7.5 Parallel I/O via Devices and Sockets

The serial-order read and write utilities described above for the CM file system can also be used to transfer data via the CM-HIPPI or VME interfaces. In these cases the "file" is either a CM-HIPPI device or a CM socket, respectively. Operations on these devices require you to access the lower-level CM I/O library CMFS, as described in the CM I/O and CM-HIPPI documentation.

7.5.1 Translating between File Descriptors and Unit Numbers

The CMFS procedures use file or socket descriptors to specify the desired "file." Two CM Fortran utility procedures translate between these descriptors and the unit numbers required by the CM Fortran utility I/O procedures.

One subroutine associates a CMFS file or socket descriptor of a previously opened "file" (or device) with a CM Fortran unit number.

```
CALL CMF_FILE_FDOPEN( UNIT, CMFS_FD, IOSTAT )
```

Both **UNIT** and **CMFS_FD** are input values; the procedure simply establishes an association between them. You can then call the CM Fortran utility read/write procedures, **CMF_CM_ARRAY_TO/FROM_FILE_SO**.

The other subroutine determines the CMFS file or socket descriptor that is already associated with a CM Fortran unit number:

```
CALL CMF_FILE_GET_FD( UNIT, CMFS_FD, IOSTAT )
```

UNIT is an input value; the value returned in **CMFS_FD** is the CMFS descriptor associated with it. This procedure is useful if you wish to use the descriptor in calls to the CMFS routines.

7.5.2 I/O via Devices

To write or read data via devices, use the serial-order ("SO") I/O procedures.

Although the serial-order I/O procedures do not pad CM files, they *do* sometimes add extraneous data at the end of an array being written to a device. If you do not wish to deal with padding explicitly in the program, you can avoid it by observing certain restrictions on array size. These restrictions are reported in the documentation for CM-HIPPI.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

CMF_FILE_FDOPEN

Associates CM file or socket descriptor with a CM Fortran unit number. Both the descriptor and the unit number are input values.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_FILE_FDOPEN( UNIT, CMFS_FD, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant containing a valid unit number in the range 0 through 2999. This is the CM Fortran unit number to be associated with **CMFS_FD**. This unit number has no relation to standard Fortran unit numbers used in the **OPEN** statement.
CMFS_FD	**INTEGER*4**. A CMFS (CM file system) file or socket descriptor.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation is placed. A positive or zero value indicates success; a negative value indicates failure.

Result

None.

Description

This subroutine associates the descriptor, **CMFS_FD**, of an open CMFS file or a CM socket with the CM Fortran unit number **UNIT**. You can then use **UNIT** as an argument to **CMF_CM_ARRAY_TO/FROM_FILE_SO** to perform I/O to CM-HIPPI, VME, or CM sockets from within a CM Fortran program.

For more information on using the Connection Machine file systems, see your CM I/O system documentation.

Copyright © 1994 Thinking Machines Corporation

For more information on using the CM-HIPPI interface, see your CM-HIPPI documentation.

See Also

CMF_FILE_GET_FD

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

CMF_FILE_GET_FD

Determines the CMFS file or socket descriptor previously associated with a specified CM Fortran unit.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_FILE_GET_FD( UNIT, CMFS_FD, IOSTAT )
```

Arguments

UNIT	An **INTEGER*4** variable, parameter, or literal constant containing a valid unit number. The unit number is assigned by the user to a file when it is created with **CMF_FILE_OPEN** or **CMF_FILE_FDOPEN**; it has no relation to standard Fortran unit numbers used with the **OPEN** statement.
CMFS_FD	An **INTEGER*4** output argument. The CMFS file or socket descriptor is returned in this variable.
IOSTAT	An **INTEGER*4** variable into which the status of the I/O operation is placed. A positive or zero value indicates success; a negative value indicates failure.

Result

None.

Description

This subroutine returns, in the argument **CMFS_FD**, the CMFS (CM file system) file or socket descriptor associated with the CM Fortran unit number **UNIT**. This allows you to determine the file descriptor previously associated with **UNIT**, for example with **CMF_FILE_FDOPEN**, so that the file descriptor can be used in calls to the low-level routines of the CMFS (CM File System) library.

Copyright © 1994 Thinking Machines Corporation

For more information on using the Connection Machine file systems, see your CM I/O system documentation.

See Also

CMF_FILE_FDOPEN

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation