Chapter 9

cmf77 Library

The library **libcmf77.a** provides an interface to OS system calls, comparable to Sun Microsystems' library **libF77.a**. The library requires no special header (include) file or explicit linking.

The CM Fortran library contains:

ACCESS	GETENV	LOC
ALARM	GETFD	LSTAT
CHDIR	GETGID	LTIME
CHMOD	GETLOG	PERROR
CTIME	GETPID	QSORT
DRAND	GETUID	RAND
DTIME	GMTIME	RENAME
ETIME	HOSTNM	RINDEX
EXIT	IARGC	SIGNAL
FDATE	IDATE	SLEEP
FFREE	IERRNO	STAT
FLUSH	IRAND	SYMLNK
FMALLOC	ITIME	SYSTEM
FORK	KILL	TIME
GERROR	LINK	UNLINK
GETARG	LNBLNK	WAIT
GETCWD		

All these procedures have on-line man pages. To avoid name conflicts with the Sun man pages, specify the name in uppercase to the **man** or **cmman** command.

Copyright © 1994 Thinking Machines Corporation

9.1 Comparison with Sun's F77 Library

There are a few differences between the contents of **libcmf77.a** and Sun's **libF77.a**.

- Sun's library includes the functions **INDEX** and **LEN**. These are intrinsic functions in CM Fortran (not library functions in **libcmf77.a**). They are described in the *CM Fortran Language Reference Manual*, and their man pages are available on-line.
- The CM Fortran library **libcmf77.a** does include **RINDEX** and **LNBLNK**, and CM Fortran provides separate man pages for them.

9.2 Managing Front-End Storage

The subroutines **FMALLOC** and **FFREE** in **libcmf77.a** provide an interface from CM Fortran to the standard **malloc** and **free** functionality for managing front-end storage.

```
SUBROUTINE FMALLOC( NB, POINTER )
INTEGER NB, POINTER

SUBROUTINE FFREE( POINTER )
```

FMALLOC allocates a block of front-end memory of at least **NB** bytes, suitably aligned for the storage of any data type. It returns the address of the start of that area in **POINTER**. If the allocation is not successful, it returns zero in **POINTER**. The subroutine **FFREE** releases a block of memory previously allocated by **FMALLOC**.

The value returned in **POINTER** by **FMALLOC** is used with the **%VAL** operator to pass the argument by value to a subprogram. The **%VAL** operator is a predefined argument-list function: it indicates that the argument following it is a pointer that is to be passed by value. The pointer must be a scalar value of type **INTEGER**, **LOGICAL**, or **REAL**.

Version 2.1, January 1994
Copyright © 1994 Thinking Machines Corporation

The following example shows the use of **FMALLOC** with **%VAL** to allocate front-end storage and treat it as an array:

```
PROGRAM MALLOC_ME
PARAMETER (M=3,NB=M*4)
INTEGER POINTER

CALL FMALLOC (NB, POINTER)
PRINT *,POINTER

CALL DO_IT(%VAL(POINTER), M)

CALL FFREE (POINTER)
END

SUBROUTINE DO_IT(ARRAY,M)
REAL ARRAY(M)

ARRAY(2) = 14.333
PRINT *,ARRAY(2)
END
```

9.3 Use of FLUSH

As of Version 2.1, calls to **FLUSH** are unnecessary. Support for **FLUSH** will be removed in a future release. We recommend that users remove calls to **FLUSH** from their CM Fortran codes.

Copyright © 1994 Thinking Machines Corporation