**The LFI file access library**
**F. Bouttier and J. Clochard**
**revised in June 2002 on cycle 24t1**

# Contents



# 1 Introduction

The LFI library is used to manipulate direct-access model files at Météo-France. It is currently used for the native files of Arpège, Aladin and Méso-NH models, and for Arpège/Aladin ancillary files (climatologies, postprocessing, Aladin coupling). Arpège/Aladin use an overlayer of LFI called the FA (Fichier Arpège) interface. Méso-NH packages LFI files with ascii files using the `cpio` Unix command to create FM (Fichier Méso-NH) files.

In the Arpège/IFS environment, the LFI library source code is held in directory `lfi/` of the auxiliary library (ClearCase project `xrd`).

This documentation only describes LFI subroutines for common use; internal routines are not listed. The Doctor variable naming convention is used throughout (argument types: K=integer, LD=logical, CD=string)

### 1.1 principle:

The principle of LFI is to manage synchronous, direct-access fortran file I/O using subroutine calls and indexation by name (16-character (max) strings; the blank string is not allowed). The index table (a list of record name/offset pairs) can be found at the beginning of the file (tip: use the `od -c` Unix command to quickly examine the index of an LFI file). An LFI file contains certain metadata, such as the creation date, which means that binary comparison (`cmp` Unix command) cannot be used to compare LFI files. One cannot extract or concatenate the fields from two LFI files without writing a dedicated program. An LFI file can hold any kind of data, not just fields. An LFI file can grow as required. Records can be edited or deleted. The LFI library supports several levels of verbosity and tolerance to errors.

### 1.2 portability:

The LFI library itself is written in ANSI Fortran 77, so it compiles on most platforms. The machine characteristics are handled as PARAMETERs, either in include file `lficom0.h` for Fortran 77 compilation, or in include file `precision.h` for Fortran 90. A few Cray-specific externals shall be replaced by dummies (they are only used for Cray multitasking)

Unlike e.g. GRIB files, the LFI files themselves are not, in general, portable, because the LFI interface does not handle the encoding of data itself. Data is written as it is passed through LFI Fortran subroutine arguments. Most portability problems can be solved using an appropriate machine-dependent Fortran-90 KIND specification.

## 2 Basic routines

### 2.1 LFIOUV: open an LFI file

```
SUBROUTINE LFIOUV ( KREP, KNUMER, LDNOMM, CDNOMF, CDSTTO, LDERFA,
                    LDIMST, KNIMES, KNBARP, KNBARI )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

**LDNOMM (in)** true if filename is specified

**CDNOMF (in)** filename (only used if LDNOMM)

**CDSTTO (in)** file open status ('OLD','NEW','UNKNOWN','SCRATCH')

**LDERFA (in)** makes errors fatal

**LDIMST (in)** activates printing statistics on close

**KNIMES (in)** verbosity level (0 lowest, 2 highest)

**KNBARP (in)** planned number of records. Only used on file creation. This is not a hard limit.

**KNBARI (out)** number of records in file (zero if file is being created)

## 2.2 LFIFER: close an LFI file

```
SUBROUTINE LFIFER ( KREP, KNUMER, CDSTTC )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

**CDSTTC (in)** file close status ('KEEP', usually)

## 2.3 LFIECR: write data to file

Writes a block of data from memory to an (already opened) LFI file record.

```
SUBROUTINE LFIECR ( KREP, KNUMER, CDNOMA, KTAB, KLONG )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

**CDNOMA (in)** record name to write (16 chars max)

**KTAB (in)** address of first word to write (typically, name of the 1-D array holding data)

**KLONG (in)** number of words to write

## 2.4 LFILEC: read data from file

Reads an (already opened) LFI file record as a block of data into memory.

```
SUBROUTINE LFILEC ( KREP, KNUMER, CDNOMA, KTAB, KLONG )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

**CDNOMA (in)** record name to read

**KTAB (in)** address of first word to read (typically, name of a big enough 1-D array)

**KLONG (in)** number of words to read

# 3 Informative subroutines

## 3.1 LFILAF: list file contents

Lists record names and lengths in physical order, plus optional file info.

```
SUBROUTINE LFILAF ( KREP, KNUMER, LDTOUT )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

**LDTOUT (in)** activates printing additional information

### 3.2 LFINFO: return index and existence of a record

```
SUBROUTINE LFINFO ( KREP, KNUMER, CDNOMA, KLONG, KPOSEX )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

**CDNOMA (in)** record name

**KLONG (out)** record length (or zero if nonexistent)

**KPOSEX (out)** position of next record (or zero if nonexistent)

### 3.3 LFINAF: print detailed information on file structure

```
SUBROUTINE LFINAF ( KREP, KNUMER, KNALDO, KNTROU, KNARES, KNAMAX )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

**KNALDO (out)** number of actual records

**KNTROU (out)** number of dataless records

**KNARES (out)** number of available positions in record index table

**KNAMAX (out)** max number of all records that can be written to the file

### 3.4 LFISTA: print file statistics

Prints stats about the file handling since it was opened.

```
SUBROUTINE LFISTA ( KREP, KNUMER )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

## 4 Sequential file access

### 4.1 LFICAS: return information about next record

```
SUBROUTINE LFICAS ( KREP, KNUMER, CDNOMA, KLONG, KPOSEX, LDAVAN )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

**CDNOMA (out)** name of next record (`' '` on EOF)

**KLONG (out)** length of next record (zero on EOF)

**KPOSEX (out)** index of the first word in next record

**LDAVAN (in)** go to next record after processing; put .false. if the record is going to be read by LFICAS.

## 4.2 LFILAS: read next record

An error

```
SUBROUTINE LFILAS ( KREP, KNUMER, CDNOMA, KTAB, KLONG )
```

**KREP (out)** return code (zero means ok). An error will be returned the requested record does not exist.

**KNUMER (in)** Fortran logical unit of LFI file

**CDNOMA (out)** name of record to read

**KTAB (in)** address of first word to read (typically, name of a big enough 1-D array)

**KLONG (in)** length of record to read

## 4.3 LFICAP: return information about previous record

Like LFICAS, but backwards.

```
SUBROUTINE LFICAP ( KREP, KNUMER, CDNOMA, KLONG, KPOSEX, LDRECU )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

**CDNOMA (out)** name of previous record (`' '` on BOF)

**KLONG (out)** length of previous record (zero on BOF)

**KPOSEX (out)** index of the first word in previous record

**LDRECU (in)** go to previous record after processing; put .false. if the record is going to be read by LFILAP.

## 4.4 LFILAP: read previous record

Like LFILAS, but backwards (duh).

```
SUBROUTINE LFILAP ( KREP, KNUMER, CDNOMA, KTAB, KLONG )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

**CDNOMA (out)** name of record to read

**KTAB (in)** address of first word to read (typically, name of a big enough 1-D array)

**KLONG (in)** length of record to read

## 4.5 LFIPOS: rewind file

The next call to LFICAS/LFILAS (resp LFICAP/LFILAP) will apply to first (resp last) record.

```
SUBROUTINE LFIPOS ( KREP, KNUMER )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

# 5 Miscellaneous

## 5.1 LFIREN: rename a record

```
SUBROUTINE LFIREN ( KREP, KNUMER, CDNOM1, CDNOM2 )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

**CDNOM1 (in)** old record name

**CDNOM2 (in)** new record name

## 5.2 LFISUP: delete a record

```
SUBROUTINE LFISUP ( KREP, KNUMER, CDNOMA, KLONUT )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

**CDNOMA (in)** record name

**KLONUT (out)** released record length

# 6 Tuning verbosity and error handling

## 6.1 LFIERF: make errors fatal or no

```
SUBROUTINE LFIERF ( KREP, KNUMER, LDERFA )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

**LDERFA (in)** if true, errors cause the LFI library to stop internally. overridden if LFINEG level is 0 or 2.

## 6.2 LFINIM: set verbosity level

```
SUBROUTINE LFINIM ( KREP, KNUMER, KNIMES )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

**KNIMES (in)** verbosity level (0=min, 2=max). Overridden if LFINMG level is 0 or 2.

## 6.3 LFIMST: print statistics on close

```
SUBROUTINE LFIMST ( KREP, KNUMER, LDIMST )
```

**KREP (out)** return code (zero means ok)

**KNUMER (in)** Fortran logical unit of LFI file

**LDIMST (in)** activates the printing of file statistics when it is closed. Overidden by NISTAG (see routine LFINSG) if the latter is 0 or 2.

### 6.4 LFINEG: adjust global error handling

`SUBROUTINE LFINEG( KNIVAU )`

**KNIVAU (in)** 0=any error is fatal, 1= (default) as specified for each file, 2=no error is fatal (but errors are still reflected in return codes)

### 6.5 LFINMG: adjust global verbosity

`SUBROUTINE LFINMG ( KNIVAU )`

**KNIVAU (in)** 0=print few messages, 1= (default) as specified for each file, 2=print any message.

### 6.6 LFINSG: print all or no statistics

`SUBROUTINE LFINSG ( KNIVAU )`

**KNIVAU (in)** 0=do not print any statistics, 1= (default) as specified for each file, 2= print all statistics.

### 6.7 LDEBUG: print debugging info

`SUBROUTINE LFIDEB ( LDEBUG )`

**LDEBUG (in)** activates printing debugging information (default=false)

## 7 Error codes

**>0** system error code returned from OPEN, CLOSE, READ, WRITE or INQUIRE call

**0** no error

**-1** logical unit not opened

**-2** argument out of (0-2) range

**-3** bad locking option

**-4** bad multitasking setup

**-5** already opened logical unit

**-6** too many logical units are in use

**-7** wrong STATUS option for open

**-8** explicit filename is required for OLD or NEW file

**-9** nonexistent OLD file, or preexistent NEW file

**-10** file/software version incompatibility

**-11** file was not properly closed

**-12** cannot read first record

**-13** already opened file with another logical unit

**-14** wrong integer argument (negative or null)

**-15** wrong character argument (too long)

**-16** internal table/record file inconsistency

**-17** too many records in file

**-18** blank name is not allowed

**-19** cannot KEEP a file with status SCRATCH

**-20** record name not found in file

**-21** actual record is longer in file than specified

**-22** actual record is shorter in file than specified

**-23** end of file reached

**-24** record name too long

**-25** already existing record name

**-26** beginning of file reached