Author(s) : J. Zoll **Library:** KERNLIB, VAX and UNIX systems only
Submitter : **Submitted:** 31.10.1991
Language : Fortran + C **Revised:** 01.04.1994

Handle Unix Disk Files

The routines of this package are an interface to the C library functions open, read, write, lseek, close, to permit a Fortran program to handle an unstructured Unix file as a string of bytes. Both sequential and direct-access READ / WRITE can be done.

New files are opened with the default permissions 644; one may set different permissions by calling `CIPERM` just before calling `CIOPEN`, which resets to the default after every call.

One parameter is common to almost all routines : `LUNDES` is the file-descriptor of C to identify the file; with CIOPEN this is an output parameter, for all other routines it is an input parameter.

Structure:

`SUBROUTINE` subprograms
User Entry Names: `CIOPEN`, `CIGET`, `CIGETW`, `CIPUT`, `CIPUTW`, `CISIZE`, `CITELL`, `CISEEK`, `CIREW`, `CICLOS`, `CIPERM`
Files Referenced: Parameter

Usage:

Note: the symbol * designates output parameters.

Open a file

```
    CALL CIOPEN(LUNDES, CHMODE, CHNAME, ISTAT)

        LUNDES* file-descriptor returned

        CHMODE  CHARACTER string selecting the IO mode :

                = 'r'   open for reading
                  'r+'  open for read/write
                  'w'   create or truncate for writing
                  'w+'  open for write/read, create or truncate
                  'a'   append
                  'a+'  open for append/read

        CHNAME  name of the file, of type CHARACTER
        ISTAT*  status, =zero if success
```

For example, create a new file in the current directory :

```
        CALL CIOPEN(LUNDES, 'w', 'concert.car', ISTAT)
        IF(ISTAT .NE. 0)          GO TO trouble
```

Read next string of bytes

```
CALL CIGET (LUNDES, CHBUF, NBDO, NBDONE, ISTAT)

    CHBUF*  text vector to be read into
     NBDO   maximum number of bytes to be read
   NBDONE*  number of bytes actually read
    ISTAT*  status, = zero if success,
            = -1   if end-of-file
```

Read next string of full words

```
CALL CIGETW(LUNDES, MBUF, NWDO, NWDONE, ISTAT)

     MBUF*  vector to be read into
     NWDO   maximum number of words to be read
   NWDONE*  number of words actually read
    ISTAT*  status, = zero if success,
            = -1   if end-of-file
```

A full word is normally 4 bytes; on the CRAY it is 8 bytes.
To simulate direct-access reading one has to call `CISEEK` first.
For example:

```
To read the next 2048 bytes:

 << starting at byte 8193 :
    CALL CISEEK(LUNDES, 8192, ISTAT)
    IF(ISTAT .NE. 0)              GO TO trouble   >>

    CALL CIGET(LUNDES, CHBUF, 2048, NBDONE, ISTAT)
    IF(ISTAT .EQ. -1)         GO TO eof
    IF(ISTAT .NE. 0)          GO TO trouble
```

Write next string of bytes

```
CALL CIPUT(LUNDES, CHBUF, NBDO, ISTAT)

    CHBUF   text vector to be written, NBDO bytes
    ISTAT*  status, =zero if success
```

Write next string of full words

```
CALL CIPUTW(LUNDES, MBUF, NWDO, ISTAT)

     MBUF   vector to be written, NWDO words
    ISTAT*  status, =zero if success
```

Get the size of the file

```
CALL CISIZE(LUNDES, NBYTT, ISTAT)

    NBYTT*  number of bytes on the file
    ISTAT*  status, =zero if success

Careful : this will position the file to the end.
```

Get the current file position

```
CALL CITELL(LUNDES, NBYTC, ISTAT)

    NBYTC*  number of bytes before current
    ISTAT*  status, =zero if success
```

Set the current file position

```
CALL CISEEK(LUNDES, NBYTC, ISTAT)

    NBYTC   number of bytes before current
    ISTAT*  status, =zero if success
```

For example :

```
CALL CISEEK(LUNDES, 0, ISTAT)    position to start-of-file
CALL CISEEK(LUNDES, 8, ISTAT)    position to 9th byte
use CISIZE to position to end-of-file
```

Rewind the file

```
CALL CIREW(LUNDES)
```

Close the file

```
CALL CICLOS(LUNDES)
```

Set the permissions for the next open

```
CALL CIPERM(IPERM)

    IPERM   the permissions as a decimal integer,
            as returned by STATF (Z265) for example
```

For example (using `NCOCTI` of `M432`) :

```
CALL CIPERM(NCOCTI('664'))

      set read for everybody, and write for owner and group.
```

Note: formally the buffer for reading and writing should be of type `CHARACTER` for `CIGET` and `CIPUT`, and of type `INTEGER` for `CIGETW` and `CIPUTW`. On most machines there is no difference, but on the VAX this must be observed, because the parameter passing mechanisme differs crucially for the two cases. Also, on the CRAY there would be problems if one were using `CIGETW` to read into a Character address other than a word boundary.

•