Author(s) : see below
Library: KERNLIB
Submitter :
Submitted: 19.09.1991
Language : Fortran + C
Revised: 01.04.1994

Immediate Interface Routines to the C Library

Authors: F. Carminati, M. Marquina, A. Rademakers, J. Shiers, J. Zoll.

The routines of this package are Fortran callable routines which in turn call their corresponding C Library routines, after having taken care of the Fortran way of passing parameters.

The names of the interface routines are exactly the names of the C functions with the letter F added; the parameters are in one-to-one correspondence with the C functions; thus "`man <name>`" gives the exact details also for the interface routine.

Most Fortran systems on Unix machines are clever, they protect the Fortran user against name-clashes with the C library, for example a "`CALL RENAME (...)`" compiles as a reference to "`rename_`" (or to "`RENAME`" on the Cray).

If this is not strictly true, and/or if moreover the Fortran Run-time library does itself contain an interface routine "`rename`" then there might be trouble because it is not obvious which "`rename`" will be linked to the interface routine `RENAMEF`. The IBM 6000 machine has succeeded in creating this problem, it has both "`rename`" and "`rename_`" on the Fortran Run-time library. In this case one has to give an explicite `-lc` on the link statement to ensure that the C library is searched before the Fortran library (but after the Kernlib library).

Structure:

`SUBROUTINE` and `FUNCTION` subprograms
User Entry Names: `ACCESSF`, `CHDIRF`, `CTIMEF`, `EXITF`, `GETENVF`, `GETGIDF`, `GETPIDF`, `GETUIDF`, `GETWDF`, `GMTIMEF`, `KILLF`, `LSTATF`, `PERRORF`, `READLNF`, `RENAMEF`, `SETENVF`, `SLEEPF`, `STATF`, `SYSTEMF`, `UNLINKF`
`COMMON` Block Names and Lengths: `/SLATE/ ISLATE(40)`

Usage:

The types of all variables and functions follow from the Fortran default typing convention (unless typed explicitly), except that variables starting with the letters `CH` are of type `CHARACTER`.

The symbol `*` designates an output parameter.

For convenience, routines which return a `CHARACTER` string also return the occupied useful length of this string in `ISLATE(1)` of `/SLATE/`.

'access' — determine accessibility of file

```
LOGICAL ACCESSF
truth = ACCESSF(CHNAME,MODE)

  CHNAME  the path-name of the file
    MODE  a bit pattern specifying the type of access:

          bit 1 (1): execution permission
              2 (2): write permission
              3 (4): read permission
           all zero: existence
```

'chdir' — set current working directory

```
INTEGER CHDIRF
ISTAT = CHDIRF(CHNAME)

  CHNAME  the path-name of the new working directory
   ISTAT  function value returns zero if successful.
```

'ctime' — convert encoded time to ASCII

```
CHARACTER CHTIME*24
CALL CTIMEF(ITIME, CHTIME)

   ITIME  encoded time (as returned by STATF)
 CHTIME*  decoded time string of length 24
```

'exit' — terminate the process with a status code

```
CALL EXITF(IRC)
```

stops setting return status `IRC`. This should not be used for normal run termination. On the IBM VM this had to be implemented with a computed `GOTO`, hence if `IRC` > `20` a `STOP 255` is executed.

On the Unix machines `IRC` will appear in the shell variable "status" which is reset after execution of each command, thus for more complicated logic the value of status has to be saved like (in the C shell):

```
set rc = $status
if   (rc != 0)  then
  if (rc == 1)  then
    echo '  not quite happy, but continue'
   else
    echo '  stop for trouble'
    exit
   endif
 endif
```

’getenv’ — get the text of an environment variable

```
CHARACTER CHTEXT*(big enough)
CALL GETENVF(CHNAME, CHTEXT)

  CHNAME  the name of the environment variable,
 CHTEXT*  returns its value, with blank-fill
          ISLATE(1) occupied length, =0 if not found
```

’getgid’ — get group identification

```
CALL GETGIDF(IDG)

  IDG  returns the real group ID of the current process.
```

’getpid’ — get process identification

```
CALL GETPIDF(IDP)

  IDP  returns the process ID of the current process.
```

’getuid’ - get user identification

```
CALL GETUIDF(IDU)

  IDU  returns the real user ID of the current process.
```

’getwd’ — get the path-name of the working directory

```
CHARACTER CHTEXT*(big enough)
CALL GETWDF(CHTEXT)

 CHTEXT*  returns the path-name, with blank-fill
          ISLATE(1) occupied length, =0 if not found
```

’gmtime’ — blow encoded time to time elements for Greenwich Mean Time

```
INTEGER ITMELS(9)
CALL GMTIMEF(ITIME, ITMELS)

   ITIME  encoded time (as returned by STATF)
 ITMELS*  decoded time elements:
          (1) sec, (2) min, (3) hour, (4) day, (5) month, (6) year,
          (7) weekday, (8) yearday, (9) isdst
```

’kill’ — send a signal to a process

```
ISTAT = KILLF(IPID,ISIG)

    IPID  process ID
    ISIG  signal number
   ISTAT  function value returns zero if successful.
```

'perror' — print message for the most recent C Library error

```
CALL PERRORF(CHTEXT)

  CHTEXT  the text to be printed before the error message
```

'readlink' — read value of a symbolic link

```
INTEGER   READLNF
CHARACTER VAL*(big enough)

NCH = READLNF(CHNAME,VAL)

      CHNAME  path-name of the link
  VAL(1:NCH)  returns the value of the link
         NCH  useful length returned,
              = -1 if trouble, PERRORF may be called.
```

'rename' — rename a file

```
INTEGER RENAMEF
ISTAT = RENAMEF(CHFROM,CHTO)

  CHFROM  old file name
    CHTO  new file name
   ISTAT  function value returns zero if successful.
```

'setenv' - set environment variable

```
INTEGER SETENVF

ISTAT = SETENVF(CHNAME,CHVAL)

   CHNAME  name of the environment variable
    CHVAL  its value to be set
    ISTAT  function value returns zero if succesful.
```

On machines where the `setenv` function of system BSD is not available, `putenv` is used instead on a string constructed from `CHNAME` and `CHVAL` in allocated memory, hence one should avoid re-defining the same variable very many times.

'sleep' — suspend execution

```
CALL SLEEPF(NSECS)

   NSECS  number of seconds to wait
```

'stat' — get file status

```
INTEGER INFO(12)
INTEGER STATF

ISTAT = STATF(CHNAME, INFO)

  CHNAME  path-name of the file
   INFO*  information returned
   ISTAT  function value returns zero if successful.
```

This routine returns the properties of a given file in a 12-word integer vector:

```
INFO(1) =  dev       device inode resides on
INFO(2) =  ino       this inode's number
INFO(3) =  mode      protection
INFO(4) =  nlink     number or hard links to the file
INFO(5) =  uid       user-id of owner
INFO(6) =  gid       group-id of owner
INFO(7) =  size      total size of file
INFO(8) =  atime     file last access time
INFO(9) =  mtime     file last modify time
INFO(10) = ctime     file last status change time
INFO(11) = blksize   optimal blocksize for file system i/o ops
INFO(12) = blocks    actual number of blocks allocated
```

On machines where `'blksize'` and `'blocks'` are not available (like Silicon Graphics) the words `INFO(11/12)` will always be zero.

'lstat' — get file status

`LSTATF` is like `STATF` except in the case where the named file is a symbolic link, in which case `LSTATF` returns information about the link, while `STATF` returns information about the file the link references.

For convenience `LSTATF` stores into `/SLATE/` some information about the nature of `CHNAME`:

```
ISLATE(1) = 0  if CHNAME is a regular file
ISLATE(2) = 0  if CHNAME is a symbolic link
ISLATE(3) = 0  if CHNAME is a directory
```

'system' — issue a shell command

```
INTEGER SYSTEMF
ISTAT = SYSTEMF(CHTEXT)

  CHTEXT  the command to be executed
   ISTAT  returns the exit status of the shell
```

'unlink' — remove directory entry

```
INTEGER UNLINKF
ISTAT = UNLINKF(CHNAME)

  CHNAME  the path-name of the file to be unlinked
   ISTAT  function value returns zero if successful.
```

Normally this deletes file `CHNAME`. If `CHNAME` is a soft link, the link is deleted, but not the file pointed to.

Notes:

The routine `SIGNALF`, which belongs to this family, will be described separately in the next paper

These routines have also been implemented on some machines which are not running Unix. The present state is as follows:
VAX system `VMS` has :

```
CHDIRF, EXITF, GETENVF, GETWDF, RENAMEF, SLEEPF, SYSTEMF
```

Presently `GETENVF` looks in the symbol table, except if the name of the environment variable is `"HOME"` for which it will return the value of the logical name `SYS$LOGIN`.
Some other routines are available through the C run-time library.
IBM 3090 system `VM/CMS` has :

```
CHDIRF, CTIMEF, EXITF, GETENVF, GETPIDF, GETWDF, GMTIMEF,
KILLF, PERRORF, RENAMEF, SLEEPF, STATF, SYSTEMF
```

•