f90_unix_file: Unix File Operations Module

March 11, 2024

1 Name

`f90_unix_file` — Module of Unix file operations

2 Usage

USE F90_UNIX_FILE

This module contains part of a Fortran API to functions detailed in ISO/IEC 9945-1:1990 Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API) [C Language].

The functions in this module are from Section 5.6 File Characteristics.

Error handling is described in `F90_UNIX_ERRNO`. Note that for procedures with an optional `ERRNO` argument, if an error occurs and `ERRNO` is not present, the program will be terminated.

All the procedures in this module are generic; some may be specific but this is subject to change in a future release.

3 Synopsis

Parameters

`F_OK`, `ID_KIND` (from `F90_UNIX_ENV`), `MODE_KIND` (from `F90_UNIX_DIR`), `R_OK`, `S_IRGRP`, `S_IROTH`, `S_IRUSR`, `S_IRWXG`, `S_IRWXO`, `S_IRWXU`, `S_ISGID`, `S_ISUID`, `S_IWGRP`, `S_IWOTH`, `S_IWUSR`, `S_IXGRP`, `S_IXOTH`, `S_IXUSR`, `TIME_KIND` (from `F90_UNIX_ENV`), `W_OK`, `X_OK`.

Derived Types

`STAT_T`, `UTIMBUF`.

Generic Procedures

`ACCESS`, `CHMOD`, `CHOWN`, `FSTAT`, `STAT`, `ISBLK`, `ISCHR`, `ISDIR`, `ISFIFO`, `ISREG`, `UTIME`.

Note that some of these generic procedures might also be specific; these characteristics could change in a future release.

4 Parameter Description

```
INTEGER(int32),PARAMETER :: F_OK
```

Flag for requesting file existence check (see `ACCESS`).

```
USE F90_UNIX_ENV,ONLY:ID_KIND
```

See `F90_UNIX_ENV` for a description of this parameter.

```
USE F90_UNIX_DIR,ONLY:MODE_KIND
```

See F90_UNIX_DIR for a description of this parameter.

```
INTEGER(int32),PARAMETER :: R_OK
```

Flag for requesting file readability check (see ACCESS).

```
INTEGER(MODE_KIND),PARAMETER :: S_IRGRP
```

File mode bit indicating group read permission (see STAT_T).

```
INTEGER(MODE_KIND),PARAMETER :: S_IROTH
```

File mode bit indicating other read permission (see STAT_T).

```
INTEGER(MODE_KIND),PARAMETER :: S_IRUSR
```

File mode bit indicating user read permission (see STAT_T).

```
INTEGER(MODE_KIND),PARAMETER :: S_IRWXG
```

Mask to select the group accessibility bits from a file mode (see STAT_T).

```
INTEGER(MODE_KIND),PARAMETER :: S_IRWXO
```

Mask to select the other accessibility bits from a file mode (see STAT_T).

```
INTEGER(MODE_KIND),PARAMETER :: S_IRWXU
```

Mask to select the user accessibility bits from a file mode (see STAT_T).

```
INTEGER(MODE_KIND),PARAMETER :: S_ISGID
```

File mode bit indicating that the file is set-group-ID (see STAT_T).

```
INTEGER(MODE_KIND),PARAMETER :: S_ISUID
```

File mode bit indicating that the file is set-user-ID (see STAT_T).

```
INTEGER(MODE_KIND),PARAMETER :: S_IWGRP
```

File mode bit indicating group write permission (see STAT_T).

```
INTEGER(MODE_KIND),PARAMETER :: S_IWOTH
```

File mode bit indicating other write permission (see STAT_T).

```
INTEGER(MODE_KIND),PARAMETER :: S_IWUSR
```

File mode bit indicating user write permission (see STAT_T).

```
INTEGER(MODE_KIND),PARAMETER :: S_IXGRP
```

File mode bit indicating group execute permission (see STAT_T).

```
INTEGER(MODE_KIND),PARAMETER :: S_IXOTH
```

File mode bit indicating other execute permission (see STAT_T).

```
INTEGER(MODE_KIND),PARAMETER :: S_IXUSR
```

File mode bit indicating user execute permission (see STAT_T).

```
USE F90_UNIX_ENV,ONLY :: TIME_KIND
```

See F90_UNIX_ENV for a description of this parameter.

```
INTEGER(int32),PARAMETER :: W_OK
```

Flag for requesting file writability check (see ACCESS).

```
INTEGER(int32),PARAMETER :: X_OK
```

Flag for requesting file executability check (see ACCESS).

5 Derived-Type Description

```
TYPE stat_t
    INTEGER(MODE_KIND) st_mode
    INTEGER(...) st_ino
    INTEGER(...) st_dev
    INTEGER(...) st_nlink
    INTEGER(id_kind) st_uid
    INTEGER(id_kind) st_gid
    INTEGER(...) st_size
    INTEGER(TIME_KIND) st_atime, st_mtime, st_ctime
END TYPE
```

Derived type holding file characteristics.

ST_MODE File mode (read/write/execute permission for user/group/other, plus set-group-ID and set-user-ID bits).

ST_INO File serial number.

ST_DEV ID for the device on which the file resides.

ST_NLINK
The number of links (see F90_UNIX_DIR, LINK operation) to the file.

ST_UID User number of the file's owner.

ST_GID Group number of the file.

ST_SIZE File size in bytes (regular files only).

ST_ATIME
Time of last access.

ST_MTIME
Time of last modification.

ST_CTIME
Time of last file status change.

```
TYPE UTIMBUF
    INTEGER(time_kind) actime, modtime
END TYPE
```

Data type holding time values for communication to UTIME. ACTIME is the new value for ST_ATIME, MODTIME is the new value for ST_MTIME.

6 Procedure Description

```
PURE SUBROUTINE access(path,amode,errno)
    CHARACTER(*),INTENT(IN) :: path
    INTEGER(*),INTENT(IN) :: amode
    INTEGER(error_kind),INTENT(OUT) :: errno
```

Checks file accessibility according to the value of AMODE; this should be F_OK or a combination of R_OK, W_OK and X_OK. In the latter case the values may be combined by addition or the intrinsic function IOR.

The result of the accessibility check is returned in ERRNO, which receives zero for success (i.e. the file exists for F_OK, or all the accesses requested by the R_OK et al combination are allowed) or an error code indicating the reason for access rejection. Possible rejection codes include EACCES, ENAMETOOLONG, ENOENT, ENOTDIR and EROFS (see F90_UNIX_ERRNO).

If the value of AMODE is invalid, error EINVAL is returned.

Note that most ACCESS enquiries are equivalent to an INQUIRE statement, in particular:

CALL ACCESS(PATH,F_OK,ERRNO)
returns success (ERRNO==0) if and only if
INQUIRE(FILE=PATH,EXIST=LVAR) would set LVAR to .TRUE.;

CALL ACCESS(PATH,R_OK,ERRNO)
returns success (ERRNO==0) if and only if
INQUIRE(FILE=PATH,READ=CHVAR) would set CHVAR to 'YES';

`CALL ACCESS(PATH,W_OK,ERRNO)`
: returns success (`ERRNO==0`) if and only if `INQUIRE(FILE=PATH,WRITE=CHVAR)` would set `CHVAR` to `'YES'`;

`CALL ACCESS(PATH,IOR(W_OK,R_OK),ERRNO)`
: returns success (`ERRNO==0`) if and only if `INQUIRE(FILE=PATH,READWRITE=CHVAR)` would set `CHVAR` to `'YES'`.

The only differences being that `ACCESS` returns a reason for rejection, and can test file executability.

```
SUBROUTINE CHMOD(PATH,MODE,ERRNO)
    CHARACTER(*),INTENT(IN) :: PATH
    INTEGER(*),INTENT(IN) :: MODE
    INTEGER(error_kind),OPTIONAL,INTENT(OUT) :: ERRNO
```

Sets the file mode (`ST_MODE`) to `MODE`.

Possible errors include `EACCES`, `ENAMETOOLONG`, `ENOTDIR`, `EPERM` and `EROFS` (see `F90_UNIX_ERRNO`).

```
SUBROUTINE CHOWN(PATH,OWNER,GROUP,ERRNO)
    CHARACTER(*),INTENT(IN) :: PATH
    INTEGER(id_kind),INTENT(IN) :: OWNER, GROUP
    INTEGER(error_kind),OPTIONAL,INTENT(OUT) :: ERRNO
```

Changes the owner (`ST_UID`) of file `PATH` to `OWNER`, and the group number (`ST_GID`) of the file to `GROUP`.

Possible errors include `EACCES`, `EINVAL`, `ENAMETOOLONG`, `ENOTDIR`, `ENOENT`, `EPERM` and `EROFS` (see `F90_UNIX_ERRNO`).

```
SUBROUTINE FSTAT(LUNIT,BUF,ERRNO)
    INTEGER(*),INTENT(IN) :: LUNIT
    TYPE(stat_t),INTENT(OUT) :: BUF
    INTEGER(error_kind),OPTIONAL,INTENT(OUT) :: ERRNO
```

`BUF` receives the characteristics of the file connected to logical unit `LUNIT`.

If `LUNIT` is not a valid logical unit number or is not connected to a file, error `EBADF` is raised (see `F90_UNIX_ERRNO`).

```
PURE LOGICAL(word) FUNCTION isblk(mode)
    INTEGER(mode_kind),INTENT(IN) :: mode
```

Returns `.TRUE.` if and only if the `MODE` value indicates that the file is a "block device".

```
PURE LOGICAL(word) FUNCTION ischr(mode)
    INTEGER(mode_kind),INTENT(IN) :: mode
```

Returns `.TRUE.` if and only if the `MODE` value indicates that the file is a "character device".

```
PURE LOGICAL(word) FUNCTION isdir(mode)
    INTEGER(mode_kind),INTENT(IN) :: mode
```

Returns .TRUE. if and only if the MODE value indicates that the file is a directory (or folder).

```
PURE LOGICAL(word) FUNCTION isfifo(mode)
    INTEGER(mode_kind),INTENT(IN) :: mode
```

Returns .TRUE. if and only if the MODE value indicates that the file is a "FIFO" (named or unnamed pipe).

```
PURE LOGICAL(word) FUNCTION isreg(mode)
    INTEGER(mode_kind),INTENT(IN) :: mode
```

Returns .TRUE. if and only if the MODE value indicates that the file is a "regular" (i.e. normal) file.

```
SUBROUTINE STAT(PATH,BUF,ERRNO)
    CHARACTER(*),INTENT(IN) :: PATH
    TYPE(stat_t),INTENT(OUT) :: BUF
    INTEGER(error_kind),OPTIONAL,INTENT(OUT) :: ERRNO
```

BUF receives the characteristics of the file PATH.

Possible errors include EACCES, ENAMETOOLONG, ENOENT and ENOTDIR (see F90_UNIX_ERRNO).

```
SUBROUTINE utime(path,times,errno)
    CHARACTER(*),INTENT(IN) :: path
    TYPE(utimbuf),OPTIONAL,INTENT(IN) :: times
    INTEGER(error_kind),OPTIONAL,INTENT(OUT) :: errno
```

Set the access and modification times of the file named by PATH to those specified by the ACTIME and MODTIME components of TIMES respectively.

Possible errors include EACCES, ENAMETOOLONG, ENOENT, ENOTDIR, EPERM and EROFS (see F90_UNIX_ERRNO).

7 See Also

f90_unix_dir(3), **f90_unix_env**(3), **f90_unix_errno**(3), **intro**(3), **nag_modules**(3), **nagfor**(1).

8 Bugs

Please report any bugs found to 'support@nag.co.uk' or 'support@nag.com', along with any suggestions for improvements.