f90_unix_dir: Unix Directory Functions Module

March 8, 2024

1 Name

`f90_unix_dir` — Module of Unix directory functions

2 Usage

USE F90_UNIX_DIR

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 procedures in this module are from sections 5.2: Working Directory, 5.3.3 Set File Creation Mask, 5.3.4 Link to a File, 5.4 Special File Creation and 5.5 File Removal.

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 both generic and specific.

3 Synopsis

Parameters

`MODE_KIND`.

Procedures

`CHDIR`, `GETCWD`, `LINK`, `MKDIR`, `MKFIFO`, `RENAME`, `RMDIR`, `UMASK`, `UNLINK`.

4 Parameter Description

```
INTEGER,PARAMETER :: MODE_KIND
```

The integer kind used to represent file permissions (see ISO/IEC 9945-1). Parameters for specific permissions are contained in `F90_UNIX_FILE`.

5 Procedure Description

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

Sets the current working directory to `PATH`. Note that any trailing blanks in `PATH` may be significant. If `ERRNO` is present it receives the error status.

Possible error conditions include `EACCES`, `ENAMETOOLONG`, `ENOTDIR` and `ENOENT` (see `F90_UNIX_ERRNO`).

```
SUBROUTINE GETCWD(PATH,LENPATH,ERRNO)
CHARACTER(*),OPTIONAL,INTENT(OUT) :: PATH
INTEGER(int32),OPTIONAL,INTENT(OUT) :: LENPATH
INTEGER(error_kind),OPTIONAL,INTENT(OUT) :: ERRNO
```

Accesses the current working directory information. If `PATH` is present, it receives the name of the current working directory, blank-padded or truncated as appropriate if the length of the current working directory name differs from that of `PATH`. If `LENPATH` is present, it receives the length of the current working directory name. If `ERRNO` is present it receives the error status.

If neither `PATH` nor `LENPATH` is present, error `EINVAL` is raised. If the path to current working directory cannot be searched, error `EACCES` is raised. If `PATH` is present and `LENPATH` is not present, and `PATH` is shorter than the current working directory name, error `ERANGE` is raised. (See `F90_UNIX_ERRNO`).

```
SUBROUTINE LINK(EXISTING,NEW,ERRNO)
CHARACTER(*),INTENT(IN) :: EXISTING,NEW
INTEGER(error_kind),OPTIONAL,INTENT(OUT) :: ERRNO
```

Creates a new link (with name given by `NEW`) for an existing file (named by `EXISTING`).

Possible errors include `EACCES`, `EEXIST`, `EMLINK`, `ENAMETOOLONG`, `ENOENT`, `ENOSPC`, `ENOTDIR`, `EPERM`, `EROFS`, `EXDEV` (see `F90_UNIX_ERRNO`).

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

Creates a new directory with name given by `PATH` and mode `MODE` (see `F90_UNIX_FILE` for mode values). Note that any trailing blanks in `PATH` may be significant.

Possible errors include `EACCES`, `EEXIST`, `EMLINK`, `ENAMETOOLONG`, `ENOENT`, `ENOSPC`, `ENOTDIR` and `EROFS` (see `F90_UNIX_ERRNO`).

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

Creates a new FIFO special file with name given by `PATH` and mode `MODE`. Note that any trailing blanks in `PATH` may be significant.

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

```
SUBROUTINE RENAME(OLD,NEW,ERRNO)
CHARACTER(*),INTENT(IN) :: OLD
CHARACTER(*),INTENT(IN) :: NEW
INTEGER(error_kind),OPTIONAL,INTENT(OUT) :: ERRNO
```

Changes the name of the file `OLD` to `NEW`. Any existing file `NEW` is first removed. Note that any trailing blanks in `OLD` or `NEW` may be significant.

Possible errors include `EACCES`, `EBUSY`, `EEXIST`, `ENOTEMPTY`, `EINVAL`, `EISDIR`, `ENAMETOOLONG`, `EMLINK`, `ENOENT`, `ENOSPC`, `ENOTDIR`, `EROFS` and `EXDEV` (see `F90_UNIX_ERRNO`).

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

Removes the directory `PATH`. Note that any trailing blanks in `PATH` may be significant.

Possible errors include `EACCES`, `EBUSY`, `EEXIST`, `ENOTEMPTY`, `ENAMETOOLONG`, `ENOENT`, `ENOTDIR` and `EROFS` (see `F90_UNIX_ERRNO`).

```
SUBROUTINE UMASK(CMASK,PMASK)
INTEGER(mode_kind),INTENT(IN) :: CMASK
INTEGER(mode_kind),OPTIONAL,INTENT(OUT) :: PMASK
```

Sets the file mode creation mask of the calling process to `CMASK`. If `PMASK` is present it receives the previous value of the mask.

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

Deletes the file `PATH`. Note that any trailing blanks in `PATH` may be significant.

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

6 See Also

f90_kind(3), **f90_unix_errno**(3), **f90_unix_file**(3), **intro**(3), **nag_modules**(3), **nagfor**(1).

7 Bugs

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