NAME

opendir - open a directory

SYNOPSIS

```
#include <dirent.h>

DIR *opendir(const char *dirname);
```

DESCRIPTION

The *opendir*() function shall open a directory stream corresponding to the directory named by the *dirname* argument. The directory stream is positioned at the first entry. If the type **DIR** is implemented using a file descriptor, applications shall only be able to open up to a total of {OPEN_MAX} files and directories.

RETURN VALUE

Upon successful completion, *opendir*() shall return a pointer to an object of type **DIR**. Otherwise, a null pointer shall be returned and *errno* set to indicate the error.

ERRORS

The *opendir*() function shall fail if:

[EACCES]
: Search permission is denied for the component of the path prefix of *dirname* or read permission is denied for *dirname*.

[ELOOP]
: A loop exists in symbolic links encountered during resolution of the *dirname* argument.

[ENAMETOOLONG]
: The length of the *dirname* argument exceeds {PATH_MAX} or a pathname component is longer than {NAME_MAX}.

[ENOENT]
: A component of *dirname* does not name an existing directory or *dirname* is an empty string.

[ENOTDIR]
: A component of *dirname* is not a directory.

The *opendir*() function may fail if:

[ELOOP]
: More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the

dirname argument.

[EMFILE]
: {OPEN_MAX} file descriptors are currently open in the calling process.

[ENAMETOOLONG]
: As a result of encountering a symbolic link in resolution of the *dirname* argument, the length of the substituted pathname string exceeded {PATH_MAX}.

[ENFILE]
: Too many files are currently open in the system.

The following sections are informative.

EXAMPLES

Open a Directory Stream

The following program fragment demonstrates how the *opendir*() function is used.

```
#include <sys/types.h>
#include <dirent.h>
#include <libgen.h>
...
    DIR *dir;
    struct dirent *dp;
...
    if ((dir = opendir (".")) == NULL) {
        perror ("Cannot open .");
        exit (1);
    }


    while ((dp = readdir (dir)) != NULL) {
...
```

APPLICATION USAGE

The *opendir*() function should be used in conjunction with *readdir()*, *closedir()*, and *rewinddir()* to examine the contents of the directory (see the EXAMPLES section in *readdir()*). This method is recommended for portability.

RATIONALE

Based on historical implementations, the rules about file descriptors apply to directory streams as well. However, this volume of IEEE Std 1003.1-2001 does not mandate that the directory stream be implemented using file descriptors. The description of *closedir()* clarifies that if a file descriptor is used for the directory stream, it is mandatory that *closedir()* deallocate the file descriptor. When a file descriptor is used to implement the directory stream, it behaves as if the FD_CLOEXEC had been set for the file descriptor.

The directory entries for dot and dot-dot are optional. This volume of IEEE Std 1003.1-2001 does not provide a way to test *a priori* for their existence because an application that is portable must be written to look for (and usually ignore) those entries. Writing code that presumes that they are the first two entries does not always work, as many implementations permit them to be other than the first two entries, with a "normal" entry preceding them. There is negligible value in providing a way to determine what the implementation does because the code to deal with dot and dot-dot must be written in any case and because such a flag would add to the list of those flags (which has proven in itself to be objectionable) and might be abused.

Since the structure and buffer allocation, if any, for directory operations are defined by the implementation, this volume of IEEE Std 1003.1-2001 imposes no portability requirements for erroneous program constructs, erroneous data, or the use of unspecified values such as the use or referencing of a *dirp* value or a **dirent** structure value after a directory stream has been closed or after a *fork()* or one of the *exec* function calls.

FUTURE DIRECTIONS

None.

SEE ALSO

closedir(), *lstat()*, *readdir()*, *rewinddir()*, *symlink()* , the Base Definitions volume of IEEE Std 1003.1-2001, *<dirent.h>*, *<limits.h>*, *<sys/types.h>*

CHANGE HISTORY

First released in Issue 2.

Issue 6

In the SYNOPSIS, the optional include of the *<sys/types.h>* header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include *<sys/types.h>* has been removed. Although *<sys/types.h>* was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

End of informative text.

[Main Index | XBD | XCU | XSH | XRAT]