NAME

opendir, **fdopendir**, **readdir**, **readdir_r**, **telldir**, **seekdir**, **rewinddir**, **closedir**, **fdclosedir**, **dirfd** - directory operations

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <dirent.h>

*DIR **
opendir(*const char *filename*);

*DIR **
fdopendir(*int fd*);

*struct dirent **
readdir(*DIR *dirp*);

int
readdir_r(*DIR *dirp, struct dirent *entry, struct dirent **result*);

long
telldir(*DIR *dirp*);

void
seekdir(*DIR *dirp, long loc*);

void
rewinddir(*DIR *dirp*);

int
closedir(*DIR *dirp*);

int
fdclosedir(*DIR *dirp*);

int
dirfd(*DIR *dirp*);

DESCRIPTION

The readdir_r() interface is deprecated because it cannot be used correctly unless {*NAME_MAX*} is a fixed value.

The **opendir**() function opens the directory named by *filename*, associates a *directory stream* with it and returns a pointer to be used to identify the *directory stream* in subsequent operations. The pointer NULL is returned if *filename* cannot be accessed, or if it cannot malloc(3) enough memory to hold the whole thing.

The **fdopendir**() function is equivalent to the **opendir**() function except that the directory is specified by a file descriptor *fd* rather than by a name. The file offset associated with the file descriptor at the time of the call determines which entries are returned.

Upon successful return from **fdopendir**(), the file descriptor is under the control of the system, and if any attempt is made to close the file descriptor, or to modify the state of the associated description other than by means of **closedir**(), **readdir**(), **readdir_r**(), or **rewinddir**(), the behavior is undefined. Upon calling **closedir**() the file descriptor is closed. The FD_CLOEXEC flag is set on the file descriptor by a successful call to **fdopendir**().

The **readdir**() function returns a pointer to the next directory entry. The directory entry remains valid until the next call to **readdir**() or **closedir**() on the same *directory stream*. The function returns NULL upon reaching the end of the directory or on error. In the event of an error, *errno* may be set to any of the values documented for the getdirentries(2) system call.

The **readdir_r**() function provides the same functionality as **readdir**(), but the caller must provide a directory *entry* buffer to store the results in. The buffer must be large enough for a *struct dirent* with a *d_name* array with {*NAME_MAX*} + 1 elements. If the read succeeds, *result* is pointed at the *entry*; upon reaching the end of the directory *result* is set to NULL. The **readdir_r**() function returns 0 on success or an error number to indicate failure.

The **telldir**() function returns a token representing the current location associated with the named *directory stream*. Values returned by **telldir**() are good only for the lifetime of the DIR pointer, *dirp*, from which they are derived. If the directory is closed and then reopened, prior values returned by **telldir**() will no longer be valid. Values returned by **telldir**() are also invalidated by a call to **rewinddir**().

The **seekdir**() function sets the position of the next **readdir**() operation on the *directory stream*. The new position reverts to the one associated with the *directory stream* when the **telldir**() operation was performed.

The **rewinddir**() function resets the position of the named *directory stream* to the beginning of the

directory.

The **closedir**() function closes the named *directory stream* and frees the structure associated with the *dirp* pointer, returning 0 on success. On failure, -1 is returned and the global variable *errno* is set to indicate the error.

The **fdclosedir**() function is equivalent to the **closedir**() function except that this function returns directory file descriptor instead of closing it.

The **dirfd**() function returns the integer file descriptor associated with the named *directory stream*, see open(2).

EXAMPLES

Sample code which searches a directory for entry ‘‘name’’ is:

```
dirp = opendir(".");
if (dirp == NULL)
        return (ERROR);
len = strlen(name);
while ((dp = readdir(dirp)) != NULL) {
        if (dp->d_namlen == len && strcmp(dp->d_name, name) == 0) {
                (void)closedir(dirp);
                return (FOUND);
        }
}
(void)closedir(dirp);
return (NOT_FOUND);
```

ERRORS

The **opendir**() function will fail if:

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

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

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

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

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

The **fdopendir**() function will fail if:

[EBADF] The *fd* argument is not a valid file descriptor open for reading.

[ENOTDIR] The descriptor *fd* is not associated with a directory.

The **readdir**() and **readdir_r**() functions may also fail and set *errno* for any of the errors specified for the routine getdents(2).

The **telldir**() function may also fail and set *errno* for any of the errors specified for the routine realloc(3).

The **closedir**() function may also fail and set *errno* for any of the errors specified for the routine close(2).

SEE ALSO

close(2), lseek(2), open(2), read(2), dir(5)

STANDARDS

The **closedir**(), **dirfd**(), **fdopendir**(), **opendir**(), **readdir**(), **readdir_r**(), **rewinddir**(), **seekdir**() and **telldir**() functions are expected to conform to IEEE Std 1003.1-2008 ("POSIX.1"). The **fdclosedir**() function and the *d_off*, *d_reclen* and *d_type* fields of *struct dirent* are non-standard, and should not be used in portable programs.

HISTORY

The **opendir**(), **readdir**(), **telldir**(), **seekdir**(), **rewinddir**(), **closedir**(), and **dirfd**() functions appeared in 4.2BSD. The **fdopendir**() function appeared in FreeBSD 8.0. **fdclosedir**() function appeared in FreeBSD 10.0.

BUGS

The behaviour of **telldir**() and **seekdir**() is likely to be wrong if there are parallel unlinks happening and the directory is larger than one page. There is code to ensure that a **seekdir**() to the location given by a **telldir**() immediately before the last **readdir**() will always set the correct location to return the same value as that last **readdir**() performed. This is enough for some applications which want to "push back the last entry read", e.g., Samba. Seeks back to any other location, other than the beginning of the directory, may result in unexpected behaviour if deletes are present. It is hoped that this situation will be resolved with changes to **getdirentries**() and the VFS.