



Linux Ubuntu 26.04 Manual Pages on command 'readdir.3'

\$ man readdir.3

readdir(3) Library Functions Manual readdir(3)

NAME

readdir - read a directory

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <dirent.h>

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

DESCRIPTION

The `readdir()` function returns a pointer to a `dirent` structure representing the next directory entry in the directory stream pointed to by `dirp`. It returns `NULL` on reaching the end of the directory stream or if an error occurred.

In the glibc implementation, the `dirent` structure is defined as follows:

```
struct dirent {
    ino_t      d_ino;    /* Inode number */
    off_t      d_off;    /* Not an offset; see below */
    unsigned short d_reclen; /* Length of this record */
    unsigned char d_type;  /* Type of file; not supported
                           by all filesystem types */
    char        d_name[256]; /* Null-terminated filename */
};
```

The only fields in the `dirent` structure that are mandated by POSIX.1 are `.d_name` and `.d_ino`.

The other fields are unstandardized, and not present on all systems; see VERSIONS.

The fields of the `dirent` structure are as follows:

.d_ino This is the inode number of the file in the filesystem containing the directory on which `readdir()` was called. If the directory entry is a mount point, then `.d_ino` differs from `.st_ino` returned by `stat(2)` on this file: `.d_ino` is the inode number of the mount point, while `.st_ino` is the inode number of the root directory of the mounted filesystem.

.d_off The value returned in `.d_off` is the same as would be returned by calling `telldir(3)` at the current position in the directory stream. Be aware that despite its type and name, the `.d_off` field is seldom any kind of directory offset on modern filesystems. Applications should treat this field as an opaque value, making no assumptions about its contents; see also `telldir(3)`.

.d_reclen

This is the size (in bytes) of the returned record. This may not match the size of the structure definition shown above; see `VERSIONS`.

.d_type

This field contains a value indicating the file type, making it possible to avoid the expense of calling `lstat(2)` if further actions depend on the type of the file.

When a suitable feature test macro is defined (`_DEFAULT_SOURCE` since glibc 2.19, or `_BSD_SOURCE` on glibc 2.19 and earlier), glibc defines the following macro constants for the value returned in `.d_type`:

`DT_BLK` This is a block device.

`DT_CHR` This is a character device.

`DT_DIR` This is a directory.

`DT_FIFO` This is a named pipe (FIFO).

`DT_LNK` This is a symbolic link.

`DT_REG` This is a regular file.

`DT SOCK` This is a UNIX domain socket.

`DT_UNKNOWN` The file type could not be determined.

Currently, only some filesystems (among them: Btrfs, ext2, ext3, and ext4) have full support for returning the file type in `.d_type`. All applications must properly handle a return of `DT_UNKNOWN`.

.d_name

This field contains the null terminated filename; see `VERSIONS`.

The `.d_name` field

The `dirent` structure definition shown above is taken from the `glibc` headers, and shows the `.d_name` field with a fixed size.

Warning: applications should avoid any dependence on the size of the `.d_name` field. POSIX defines it as `char d_name[]`, a character array of unspecified size, with at most `NAME_MAX` characters preceding the terminating null byte (`'\0'`).

POSIX.1 explicitly notes that this field should not be used as an lvalue. The standard also notes that the use of `sizeof(.d_name)` is incorrect; use `strlen(.d_name)` instead. (On some systems, this field is defined as `char d_name[1]!`) By implication, the use `sizeof(struct dirent)` to capture the size of the record including the size of `.d_name` is also incorrect.

Note that while the call

```
fpathconf(fd, _PC_NAME_MAX)
```

returns the value 255 for most filesystems, on some filesystems (e.g., CIFS, Windows SMB servers), the null-terminated filename that is (correctly) returned in `.d_name` can actually exceed this size. In such cases, the `.d_reclen` field will contain a value that exceeds the size of the `glibc` `dirent` structure shown above.

STANDARDS

POSIX.1-2008.

HISTORY

POSIX.1-2001, SVr4, 4.3BSD.

NOTES

A directory stream is opened using `opendir(3)`.

The order in which filenames are read by successive calls to `readdir()` depends on the filesystem implementation; it is unlikely that the names will be sorted in any fashion.

SEE ALSO

`getdents(2)`, `read(2)`, `closedir(3)`, `dirfd(3)`, `ftw(3)`, `offsetof(3)`, `opendir(3)`, `readdir_r(3)`, `rewinddir(3)`, `scandir(3)`, `seekdir(3)`, `telldir(3)`