NAME

readdir, readdir_r − read a directory

SYNOPSIS

#include <dirent.h>

struct dirent *readdir(DIR *_dirp_);

int readdir_r(DIR *_dirp_, struct dirent *_entry_, struct dirent **_result_);

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

readdir_r():
_POSIX_C_SOURCE >= 1 || _XOPEN_SOURCE || _BSD_SOURCE || _SVID_SOURCE || _POSIX_SOURCE

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.

On Linux, the *dirent* structure is defined as follows:

```
struct dirent {
   ino_t          d_ino;       /* inode number */
   off_t          d_off;       /* not an offset; see NOTES */
   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]; /* filename */
};
```

The only fields in the *dirent* structure that are mandated by POSIX.1 are: *d_name*[], of unspecified size, with at most **NAME_MAX** characters preceding the terminating null byte ('\0'); and (as an XSI extension) *d_ino*. The other fields are unstandardized, and not present on all systems; see NOTES below for some further details.

The data returned by **readdir**() may be overwritten by subsequent calls to **readdir**() for the same directory stream.

The **readdir_r**() function is a reentrant version of **readdir**(). It reads the next directory entry from the directory stream *dirp*, and returns it in the caller-allocated buffer pointed to by *entry*. (See NOTES for information on allocating this buffer.) A pointer to the returned item is placed in **result*; if the end of the directory stream was encountered, then NULL is instead returned in **result*.

RETURN VALUE

On success, **readdir**() returns a pointer to a *dirent* structure. (This structure may be statically allocated; do not attempt to **free**(3) it.) If the end of the directory stream is reached, NULL is returned and *errno* is not changed. If an error occurs, NULL is returned and *errno* is set appropriately.

The **readdir_r**() function returns 0 on success. On error, it returns a positive error number (listed under ERRORS). If the end of the directory stream is reached, **readdir_r**() returns 0, and returns NULL in **result*.

ERRORS

EBADF
Invalid directory stream descriptor *dirp*.

ATTRIBUTES

Multithreading (see pthreads(7))

The **readdir**() function is not thread-safe.

The **readdir_r**() function is thread-safe.

CONFORMING TO

SVr4, 4.3BSD, POSIX.1-2001.

NOTES

Only the fields *d_name* and *d_ino* are specified in POSIX.1-2001. The remaining fields are available on many, but not all systems. Under glibc, programs can check for the availability of the fields not defined in POSIX.1 by testing whether the macros **_DIRENT_HAVE_D_NAMLEN**, **_DIRENT_HAVE_D_RECLEN**, **_DIRENT_HAVE_D_OFF**, or **_DIRENT_HAVE_D_TYPE** are defined.

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).

Other than Linux, the *d_type* field is available mainly only on BSD systems. This field makes it possible to avoid the expense of calling **lstat**(2) if further actions depend on the type of the file. If the **_BSD_SOURCE** feature test macro is defined, then 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 is unknown.

If the file type could not be determined, the value **DT_UNKNOWN** is returned in *d_type*.

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**.

Since POSIX.1 does not specify the size of the *d_name* field, and other nonstandard fields may precede that field within the *dirent* structure, portable applications that use **readdir_r**() should allocate the buffer whose address is passed in *entry* as follows:

```
name_max = pathconf(dirpath, _PC_NAME_MAX);
if (name_max == −1)         /* Limit not defined, or error */
   name_max = 255;          /* Take a guess */
len = offsetof(struct dirent, d_name) + name_max + 1;
entryp = malloc(len);
```

(POSIX.1 requires that *d_name* is the last field in a *struct dirent*.)

SEE ALSO

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

COLOPHON

This page is part of release 3.74 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at http://www.kernel.org/doc/man−pages/.