NAME

readdir_r − read a directory

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

```
#include <dirent.h>

[[deprecated]] int readdir_r(DIR *restrict dirp,
                             struct dirent *restrict entry,
                             struct dirent **restrict result);
```

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

readdir_r():
_POSIX_C_SOURCE
|| /* glibc <= 2.19: */ _BSD_SOURCE || _SVID_SOURCE

DESCRIPTION

This function is deprecated; use **readdir**(3) instead.

The **readdir_r**() function was invented as a reentrant version of **readdir**(3). It reads the next directory entry from the directory stream *dirp*, and returns it in the caller-allocated buffer pointed to by *entry*. For details of the *dirent* structure, see **readdir**(3).

A pointer to the returned buffer is placed in**result*; if the end of the directory stream was encountered, then NULL is instead returned in**result*.

It is recommended that applications use **readdir**(3) instead of **readdir_r**(). Furthermore, since glibc 2.24, glibc deprecates **readdir_r**(). The reasons are as follows:

- On systems where **NAME_MAX** is undefined, calling **readdir_r**() may be unsafe because the interface does not allow the caller to specify the length of the buffer used for the returned directory entry.
- On some systems, **readdir_r**() can't read directory entries with very long names. When the glibc implementation encounters such a name, **readdir_r**() fails with the error **ENAMETOOLONG** *after the final directory entry has been read*. On some other systems,**r eaddir_r**() may return a success status, but the returned *d_name* field may not be null terminated or may be truncated.
- In the current POSIX.1 specification (POSIX.1-2008), **readdir**(3) is not required to be thread-safe. However, in modern implementations (including the glibc implementation), concurrent calls to **readdir**(3) that specify different directory streams are thread-safe. Therefore, the use of **readdir_r**() is generally unnecessary in multithreaded programs. In cases where multiple threads must read from the same directory stream, using **readdir**(3) with external synchronization is still preferable to the use of **readdir_r**(), for the reasons given in the points above.
- It is expected that a future version of POSIX.1 will make **readdir_r**() obsolete, and require that **readdir**(3) be thread-safe when concurrently employed on different directory streams.

RETURN VALUE

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,**r eaddir_r**() returns 0, and returns NULL in**result*.

ERRORS

EBADF
Invalid directory stream descriptor *dirp*.

ENAMETOOLONG
A directory entry whose name was too long to be read was encountered.

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes**(7).

Interface	Attribute	Value
readdir_r()	Thread safety	MT-Safe

STANDARDS

POSIX.1-2001, POSIX.1-2008.

SEE ALSO

readdir(3)