

**NAME**

readdir\_r - read a directory

**SYNOPSIS**

```
#include <dirent.h>
```

```
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 || /* Glibc versions <= 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 version 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, **readdir\_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, **readdir\_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
<b>readdir_r()</b>	Thread safety	MT-Safe

**CONFORMING TO**

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

**SEE ALSO**

[readdir\(3\)](#)

**COLOPHON**

This page is part of release 4.10 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 <https://www.kernel.org/doc/man-pages/>.