



[IBM i](#) / [Change version](#)  [Feedback](#)  [Product list](#)

readdir_r()--Read Directory Entry

Last Updated: 2021-04-13

Syntax

```
#include <sys/types.h>
```

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

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

Service Program Name: QP0LLIB1

Default Public Authority: *USE

Threadsafe: Conditional; see [Usage Notes](#).

The **readdir_r()** function initializes the dirent structure that is referenced by *entry* to represent the next directory entry in the directory stream that is associated with *dirp*. The **readdir_r()** function then stores a pointer to the *entry* structure at the location referenced by *result*.

The storage pointed to by *entry* must be large enough for a dirent structure.

If the call to **readdir_r()** actually reads the directory, the access time of the directory is updated.

The **readdir_r()** function converts the directory entry name into the CCSID (coded character set identifier) of the job at the time of the call to **opendir()**, or to the CCSID specified on the call to **Qlg0pendir()**. If the directory entry name cannot be represented in that CCSID, then that directory entry will not be returned by **readdir_r()** and no error indication will occur.

Qlg0pendir() allows the CCSID to be specified in the `Qlg_Path_Name_T` structure. See [Qlg0pendir\(\)](#)--Open Directory (using NLS-enabled path name) for more information.

Parameters

dirp

(Input) A pointer to a DIR that refers to the open directory stream to be read. This pointer is returned by **opendir()** or **Qlg0pendir()** (see [opendir\(\)](#)--Open Directory or [Qlg0pendir\(\)](#)--Open Directory (using NLS-enabled path name)).

entry

> (Output) A pointer to a dirent structure in which the directory entry is to be placed.

result

(Output) A pointer to a pointer to a dirent structure. Upon successfully reading a directory entry, this dirent pointer is set to the same value as *entry*. Upon reaching the end of the directory stream, this pointer will be set to NULL.

A dirent structure has the following contents:

char	d_reserved1[16]	Reserved.
unsigned int	d_fileno_gen_id	The generation ID associated with the file ID.

ino_t	d_fileno	The file ID of the file. This number uniquely identifies the object within a file system.
unsigned int	d_reclen	The length of the directory entry in bytes.
int	d_reserved3	Reserved.
char	d_reserved4[6]	Reserved.
char	d_reserved5[2]	Reserved.
qlg_nls_t	d_nlsinfo	National language information about d_name. The following fields are defined:
		<i>int ccsid</i> CCSID of the characters in the d_name field.
		<i>char country_id[2]</i> Country or region identifier that is associated with the d_name field.
		<i>char language_id[3]</i> Language identifier that is associated with the d_name field.
		<i>char nls_reserved[3]</i> Reserved.
unsigned int	d_namelen	The length of the name in bytes, excluding the null terminator.
char	d_name[640]	A string that gives the name of a file in the directory. This string ends in a terminating null, and has a maximum length of {NAME_MAX} bytes, not including the terminating NULL (see pathconf()--Get Configurable Path Name Variables).

Authorities

No authorization is required. Authorization is verified during **opendir()**.

Return Value

0

readdir_r() was successful. The *result* parameter points to one of the following:

- A pointer to a *dirent* structure that describes the next directory entry in the directory stream. This will be the same value as the *entry* parameter.
- A NULL pointer. **readdir_r()** reached the end of the directory stream.

error code

readdir_r() was not successful. This value is set to the same value as the *errno* global variable.

Error Conditions

If **readdir_r()** is not successful, *errno* usually indicates one of the following errors. Under some conditions, *errno* could indicate an error other than those listed here.

Error condition	Additional information
<i>[EACCES]</i>	If you are accessing a remote file through the Network File System, update operations to file permissions at the server are not reflected at the client until updates to data that is stored locally by the Network File System take place. (Several options on the Add Mounted File System (ADDMFS) command determine the time between refresh operations of local data.) Access to a remote file may also fail due to different mappings of user IDs (UID) or group IDs (GID) on the local and remote systems.
<i>[EAGAIN]</i>	
<i>[EBADFID]</i>	
<i>[EBADF]</i>	
<i>[EBUSY]</i>	
<i>[EDAMAGE]</i>	
<i>[EFAULT]</i>	
<i>[EINVAL]</i>	
<i>[EIO]</i>	
<i>[ENOSPC]</i>	
<i>[ENOTAVAIL]</i>	

[[ENOTSAFE](#)]

[[ESTALE](#)]

If you are accessing a remote file through the Network File System, the file may have been deleted at the server.

[[EUNKNOWN](#)]

If interaction with a file server is required to access the object, *errno* could indicate one of the following errors:

Error condition	Additional information
-----------------	------------------------

[[EADDRNOTAVAIL](#)]

[[ECONNABORTED](#)]

[[ECONNREFUSED](#)]

[[ECONNRESET](#)]

[[EHOSTDOWN](#)]

[[EHOSTUNREACH](#)]

[[ENETDOWN](#)]

[[ENETRESET](#)]

[[ENETUNREACH](#)]

[ESTALE]

If you are accessing a remote file through the Network File System, the file may have been deleted at the server.

*[ETIMEDOUT]**[EUNATCH]*

Error Messages

The following messages may be sent from this function:

Message ID	Error Message Text
CPE3418 E	Possible APAR condition or hardware failure.
CPFA0D4 E	File system error occurred. Error number &1.
CPF3CF2 E	Error(s) occurred during running of &1 API.
CPF9872 E	Program or service program &1 in library &2 ended. Reason code &3.

Usage Notes

1. This function will fail with error code [ENOTSAFE] when all the following conditions are true:
 - Where multiple threads exist in the job.
 - The object on which this function is operating resides in a file system that is not threadsafe. Only the following file systems are threadsafe for this function:
 - "Root" (/)
 - QOpenSys
 - User-defined
 - QNTC
 - QSYS.LIB
 - Independent ASP QSYS.LIB
 - QOPT
 - Network File System
 - QFileSvr.400
2. **readdir_r()** is threadsafe only when directed to a directory in a threadsafe file system.
3. If the *dirp* argument that is passed to **readdir_r()** does not refer to an open directory stream, **readdir_r()** returns the [EBADF] error.
4. **readdir_r()** caches multiple directory entries to improve performance. This means the directory is not actually read on each call to **readdir_r()**. As a result, files that are added to the directory after **opendir()** or

rewinddir() may not be returned on calls to **readdir_r()**, and files that are removed may still be returned on calls to **readdir_r()**.

5. **readdir_r()** also returns directory entries for dot (.) and dot-dot (..) subdirectories.

6. QSYS.LIB and Independent ASP QSYS.LIB File System Differences

Calls to **readdir_r()** that update the access time of the directory use the normal rules that apply to libraries and database files. At most, the access time is updated once per day.

7. QDLS File System Differences

The access time of the directory is updated on **opendir()**. The access time is not affected by **readdir_r()**.

When objects in QDLS are accessed, the country or region ID and language ID of the directory entry name are always set to the country or region ID and language ID of the system.

When a **readdir_r()** operation specifies the /QDLS directory, the user must have *USE authority to each object in the /QDLS directory (that is, *USE authority to each object immediately below QDLS in the directory hierarchy). A directory entry is returned only for those objects for which the user has *USE authority. If the **readdir_r()** operation specifies a directory below QDLS, a directory entry is returned for all objects, even if the user does not have *USE authority for some of the objects.

8. QOPT File System Differences

The access time of the directory is not updated on a **readdir_r()** operation.

Related Information

- The <sys/types.h> file (see [Header Files for UNIX®-Type Functions](#))
- The <dirent.h> file (see [Header Files for UNIX-Type Functions](#))
- [opendir\(\)](#)--Open Directory
- [QlgOpendir\(\)](#)--Open Directory (using NLS-enabled path name)
- [readdir\(\)](#)--Read Directory Entry
- [QlgReaddir\(\)](#)--Read Directory Entry (using NLS-enabled path name)
- [QlgReaddir_r\(\)](#)--Read Directory Entry (using NLS-enabled path name)
- [readdir_r_ts64\(\)](#)--Read Directory Entry
- [rewinddir\(\)](#)--Reset Directory Stream to Beginning
- [closedir\(\)](#)--Close Directory
- [pathconf\(\)](#)--Get Configurable Path Name Variables

Example

The following example reads the contents of the "root" (/) directory.

Note: By using the code examples, you agree to the terms of the [Code license and disclaimer information](#).

```
#include <sys/types.h>
#include <dirent.h>
#include <errno.h>
#include <stdio.h>

main() {
    int return_code;
```

```
DIR *dir;
struct dirent entry;
struct dirent *result;

if ((dir = opendir("/")) == NULL)
    perror("opendir() error");
else {
    puts("contents of root:");
    for (return_code = readdir_r(dir, &entry, &result);
        result != NULL && return_code == 0;
        return_code = readdir_r(dir, &entry, &result))
        printf("  %s\n", entry.d_name);
    if (return_code != 0)
        perror("readdir_r() error");
    closedir(dir);
}
}
```

Output:

contents of root:

```
.
..
QSYS.LIB
QDLS
QOpenSys
QOPT
home
```

API introduced: V3R1

[[Back to top](#) | [UNIX-Type APIs](#) | [APIs by category](#)]