



*Full credit is given to the above companies including the OS that this PDF file was generated!*

### ***Rocky Enterprise Linux 9.2 Manual Pages on command 'errno.3'***

#### ***\$ man errno.3***

ERRNO(3)                   Linux Programmer's Manual                   ERRNO(3)

#### NAME

errno - number of last error

#### SYNOPSIS

```
#include <errno.h>
```

#### DESCRIPTION

The `<errno.h>` header file defines the integer variable `errno`, which is set by system calls and some library functions in the event of an error to indicate what went wrong.

#### errno

The value in `errno` is significant only when the return value of the call indicated an error (i.e., -1 from most system calls; -1 or NULL from most library functions); a function that succeeds is allowed to change `errno`. The value of `errno` is never set to zero by any system call or library function.

For some system calls and library functions (e.g., `getpriority(2)`), -1 is a valid return on success. In such cases, a successful return can be distinguished from an error return by setting `errno` to zero before

the call, and then, if the call returns a status that indicates that an error may have occurred, checking to see if `errno` has a nonzero value. `errno` is defined by the ISO C standard to be a modifiable lvalue of type `int`, and must not be explicitly declared; `errno` may be a macro. `errno` is thread-local; setting it in one thread does not affect its value in any other thread.

## Error numbers and names

Valid error numbers are all positive numbers. The `<errno.h>` header file defines symbolic names for each of the possible error numbers that may appear in `errno`.

All the error names specified by POSIX.1 must have distinct values, with the exception of `EAGAIN` and `EWOULDBLOCK`, which may be the same.

On Linux, these two have the same value on all architectures.

The error numbers that correspond to each symbolic name vary across UNIX systems, and even across different architectures on Linux. Therefore, numeric values are not included as part of the list of error names below. The `perror(3)` and `strerror(3)` functions can be used to convert these names to corresponding textual error messages.

On any particular Linux system, one can obtain a list of all symbolic error names and the corresponding error numbers using the `errno(1)` command (part of the `moreutils` package):

```
$ errno -l
EPERM 1 Operation not permitted
ENOENT 2 No such file or directory
ESRCH 3 No such process
EINTR 4 Interrupted system call
EIO 5 Input/output error
...
```

The `errno(1)` command can also be used to look up individual error numbers and names, and to search for errors using strings from the error description, as in the following examples:

```
$ errno 2
ENOENT 2 No such file or directory
```

\$ errno ESRCH

ESRCH 3 No such process

\$ errno -s permission

EACCES 13 Permission denied

#### List of error names

In the list of the symbolic error names below, various names are marked as follows:

- \* POSIX.1-2001: The name is defined by POSIX.1-2001, and is defined in later POSIX.1 versions, unless otherwise indicated.
- \* POSIX.1-2008: The name is defined in POSIX.1-2008, but was not present in earlier POSIX.1 standards.
- \* C99: The name is defined by C99.

Below is a list of the symbolic error names that are defined on Linux:

E2BIG       Argument list too long (POSIX.1-2001).  
EACCES       Permission denied (POSIX.1-2001).  
EADDRINUSE   Address already in use (POSIX.1-2001).  
EADDRNOTAVAIL Address not available (POSIX.1-2001).  
EAFNOSUPPORT Address family not supported (POSIX.1-2001).  
EAGAIN       Resource temporarily unavailable (may be the same value  
              as EWOULDBLOCK) (POSIX.1-2001).  
EALREADY     Connection already in progress (POSIX.1-2001).  
EBADE        Invalid exchange.  
EBADF        Bad file descriptor (POSIX.1-2001).  
EBADFD       File descriptor in bad state.  
EBADMSG      Bad message (POSIX.1-2001).  
EBADR        Invalid request descriptor.  
EBADRQC      Invalid request code.  
EBADSLT      Invalid slot.  
EBUSY        Device or resource busy (POSIX.1-2001).  
ECANCELED    Operation canceled (POSIX.1-2001).  
ECHILD       No child processes (POSIX.1-2001).  
ECHRNG       Channel number out of range.  
ECOMM        Communication error on send.

ECONNABORTED Connection aborted (POSIX.1-2001).

ECONNREFUSED Connection refused (POSIX.1-2001).

ECONNRESET Connection reset (POSIX.1-2001).

EDEADLK Resource deadlock avoided (POSIX.1-2001).

EDEADLOCK On most architectures, a synonym for EDEADLK. On some architectures (e.g., Linux MIPS, PowerPC, SPARC), it is a separate error code "File locking deadlock error".

EDESTADDRREQ Destination address required (POSIX.1-2001).

EDOM Mathematics argument out of domain of function (POSIX.1, C99).

EDQUOT Disk quota exceeded (POSIX.1-2001).

EEXIST File exists (POSIX.1-2001).

EFAULT Bad address (POSIX.1-2001).

EFBIG File too large (POSIX.1-2001).

EHOSTDOWN Host is down.

EHOSTUNREACH Host is unreachable (POSIX.1-2001).

EHWPOISON Memory page has hardware error.

EIDRM Identifier removed (POSIX.1-2001).

EILSEQ Invalid or incomplete multibyte or wide character (POSIX.1, C99).

The text shown here is the glibc error description; in POSIX.1, this error is described as "Illegal byte sequence".

EINPROGRESS Operation in progress (POSIX.1-2001).

EINTR Interrupted function call (POSIX.1-2001); see signal(7).

EINVAL Invalid argument (POSIX.1-2001).

EIO Input/output error (POSIX.1-2001).

EISCONN Socket is connected (POSIX.1-2001).

EISDIR Is a directory (POSIX.1-2001).

EISNAM Is a named type file.

EKEYEXPIRED Key has expired.

EKEYREJECTED Key was rejected by service.

EKEYREVOKED Key has been revoked.

EL2HLT Level 2 halted.

EL2NSYNC Level 2 not synchronized.

EL3HLT Level 3 halted.

EL3RST Level 3 reset.

ELIBACC Cannot access a needed shared library.

ELIBBAD Accessing a corrupted shared library.

ELIBMAX Attempting to link in too many shared libraries.

ELIBSCN .lib section in a.out corrupted

ELIBEXEC Cannot exec a shared library directly.

ELNRANGE Link number out of range.

ELOOP Too many levels of symbolic links (POSIX.1-2001).

EMEDIUMTYPE Wrong medium type.

EMFILE Too many open files (POSIX.1-2001). Commonly caused by exceeding the RLIMIT\_NOFILE resource limit described in `getrlimit(2)`. Can also be caused by exceeding the limit specified in `/proc/sys/fs/nr_open`.

EMLINK Too many links (POSIX.1-2001).

EMSGSIZE Message too long (POSIX.1-2001).

EMULTIHOP Multihop attempted (POSIX.1-2001).

ENAMETOOLONG Filename too long (POSIX.1-2001).

ENETDOWN Network is down (POSIX.1-2001).

ENETRESET Connection aborted by network (POSIX.1-2001).

ENETUNREACH Network unreachable (POSIX.1-2001).

ENFILE Too many open files in system (POSIX.1-2001). On Linux, this is probably a result of encountering the `/proc/sys/fs/file-max` limit (see `proc(5)`).

ENOANO No anode.

ENOBUFS No buffer space available (POSIX.1 (XSI STREAMS operation)).

ENODATA No message is available on the STREAM head read queue (POSIX.1-2001).

ENODEV No such device (POSIX.1-2001).

ENOENT No such file or directory (POSIX.1-2001).  
Typically, this error results when a specified pathname does not exist, or one of the components in the directory prefix of a pathname does not exist, or the specified pathname is a dangling symbolic link.

ENOEXEC Exec format error (POSIX.1-2001).

ENOKEY Required key not available.

ENOLCK No locks available (POSIX.1-2001).

ENOLINK Link has been severed (POSIX.1-2001).

ENOMEDIUM No medium found.

ENOMEM Not enough space/cannot allocate memory (POSIX.1-2001).

ENOMSG No message of the desired type (POSIX.1-2001).

ENONET Machine is not on the network.

ENOPKG Package not installed.

ENOPROTOPT Protocol not available (POSIX.1-2001).

ENOSPC No space left on device (POSIX.1-2001).

ENOSR No STREAM resources (POSIX.1 (XSI STREAMS option)).

ENOSTR Not a STREAM (POSIX.1 (XSI STREAMS option)).

ENOSYS Function not implemented (POSIX.1-2001).

ENOTBLK Block device required.

ENOTCONN The socket is not connected (POSIX.1-2001).

ENOTDIR Not a directory (POSIX.1-2001).

ENOTEMPTY Directory not empty (POSIX.1-2001).

ENOTRECOVERABLE State not recoverable (POSIX.1-2008).

ENOTSOCK Not a socket (POSIX.1-2001).

ENOTSUP Operation not supported (POSIX.1-2001).

ENOTTY Inappropriate I/O control operation (POSIX.1-2001).

ENOTUNIQ Name not unique on network.

ENXIO No such device or address (POSIX.1-2001).

EOPNOTSUPP Operation not supported on socket (POSIX.1-2001).

(ENOTSUP and EOPNOTSUPP have the same value on Linux, but according to POSIX.1 these error values should be distinct.)

E\_OVERFLOW Value too large to be stored in data type  
(POSIX.1-2001).

E\_OWNERDEAD Owner died (POSIX.1-2008).

E\_PERM Operation not permitted (POSIX.1-2001).

E\_PFNOSUPPORT Protocol family not supported.

E\_PIPE Broken pipe (POSIX.1-2001).

E\_PROTO Protocol error (POSIX.1-2001).

E\_PROTONOSUPPORT Protocol not supported (POSIX.1-2001).

E\_PROTOTYPE Protocol wrong type for socket (POSIX.1-2001).

E\_RANGE Result too large (POSIX.1, C99).

E\_REMCHG Remote address changed.

E\_REMOTE Object is remote.

E\_REMOTEIO Remote I/O error.

E\_RESTART Interrupted system call should be restarted.

E\_RFKILL Operation not possible due to RF-kill.

E\_ROFS Read-only filesystem (POSIX.1-2001).

E\_SHUTDOWN Cannot send after transport endpoint shutdown.

E\_SPIPE Invalid seek (POSIX.1-2001).

E\_SOCKETNOSUPPORT Socket type not supported.

E\_SRCH No such process (POSIX.1-2001).

E\_STALE Stale file handle (POSIX.1-2001).

This error can occur for NFS and for other filesystems.

E\_STRPIPE Streams pipe error.

E\_TIME Timer expired (POSIX.1 (XSI STREAMS option)).

(POSIX.1 says "STREAM ioctl(2) timeout".)

E\_TIMEDOUT Connection timed out (POSIX.1-2001).

E\_TOO\_MANY\_REFS Too many references: cannot splice.

E\_TXTBSY Text file busy (POSIX.1-2001).

E\_UCLEAN Structure needs cleaning.

E\_UNATCH Protocol driver not attached.

E\_USERS Too many users.

E\_WOULDBLOCK Operation would block (may be same value as EAGAIN)  
(POSIX.1-2001).

EXDEV      Improper link (POSIX.1-2001).

EXFULL     Exchange full.

## NOTES

A common mistake is to do

```
if (somecall() == -1) {
    printf("somecall() failed\n");
    if (errno == ...) { ... }
}
```

where `errno` no longer needs to have the value it had upon return from `somecall()` (i.e., it may have been changed by the `printf(3)`). If the value of `errno` should be preserved across a library call, it must be saved:

```
if (somecall() == -1) {
    int errsv = errno;
    printf("somecall() failed\n");
    if (errsv == ...) { ... }
}
```

Note that the POSIX threads APIs do not set `errno` on error. Instead, on failure they return an error number as the function result. These error numbers have the same meanings as the error numbers returned in `errno` by other APIs.

On some ancient systems, `<errno.h>` was not present or did not declare `errno`, so that it was necessary to declare `errno` manually (i.e., `extern int errno`). Do not do this. It long ago ceased to be necessary, and it will cause problems with modern versions of the C library.

## SEE ALSO

`errno(1)`, `err(3)`, `error(3)`, `perror(3)`, `strerror(3)`

## COLOPHON

This page is part of release 5.10 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/>.