NAME

strerror, strerrorname_np, strerrordesc_np, strerror_r, strerror_l − return string describing error number

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

```
#include <string.h>

char *strerror(int errnum);
const char *strerrorname_np(int errnum);
const char *strerrordesc_np(int errnum);

int strerror_r(int errnum, char buf[.buflen], size_t buflen);
        /* XSI-compliant */

char *strerror_r(int errnum, char buf[.buflen], size_t buflen);
        /* GNU-specific */

char *strerror_l(int errnum, locale_t locale);
```

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

strerrorname_np(), **strerrordesc_np**():
_GNU_SOURCE

strerror_r():
The XSI-compliant version is provided if:
(_POSIX_C_SOURCE >= 200112L) && ! _GNU_SOURCE
Otherwise, the GNU-specific version is provided.

DESCRIPTION

The **strerror**() function returns a pointer to a string that describes the error code passed in the argument *errnum*, possibly using the **LC_MESSAGES** part of the current locale to select the appropriate language. (For example, if *errnum* is **EINVAL**, the returned description will be "Invalid argument".) This string must not be modified by the application, but may be modified by a subsequent call to **strerror**() or **strerror_l**(). No other library function, including **perror**(3), will modify this string.

Like **strerror**(), the **strerrordesc_np**() function returns a pointer to a string that describes the error code passed in the argument *errnum*, with the difference that the returned string is not translated according to the current locale.

The **strerrorname_np**() function returns a pointer to a string containing the name of the error code passed in the argument *errnum*. For example, given **EPERM** as an argument, this function returns a pointer to the string "EPERM".

strerror_r()

The **strerror_r**() function is similar to **strerror**(), but is thread safe. This function is available in two versions: an XSI-compliant version specified in POSIX.1-2001 (available since glibc 2.3.4, but not POSIX-compliant until glibc 2.13), and a GNU-specific version (available since glibc 2.0). The XSI-compliant version is provided with the feature test macros settings shown in the SYNOPSIS; otherwise the GNU-specific version is provided. If no feature test macros are explicitly defined, then (since glibc 2.4) **_POSIX_C_SOURCE** is defined by default with the value 200112L, so that the XSI-compliant version of **strerror_r**() is provided by default.

The XSI-compliant **strerror_r**() is preferred for portable applications. It returns the error string in the user-supplied buffer *buf* of length *buflen*.

The GNU-specific **strerror_r**() returns a pointer to a string containing the error message. This may be either a pointer to a string that the function stores in *buf*, or a pointer to some (immutable) static string (in which case *buf* is unused). If the function stores a string in *buf*, then at most *buflen* bytes are stored (the string may be truncated if *buflen* is too small and *errnum* is unknown). The string always includes a terminating null byte ('\0').

strerror_l()

strerror_l() is like **strerror**(), but maps *errnum* to a locale-dependent error message in the locale specified by *locale*. The behavior of **strerror_l**() is undefined if *locale* is the special locale object **LC_GLOBAL_LOCALE** or is not a valid locale object handle.

RETURN VALUE

The **strerror**(), **strerror_l**(), and the GNU-specific **strerror_r**() functions return the appropriate error description string, or an "Unknown error nnn" message if the error number is unknown.

On success, **strerrorname_np**() and **strerrordesc_np**() return the appropriate error description string. If *errnum* is an invalid error number, these functions return NULL.

The XSI-compliant **strerror_r**() function returns 0 on success. On error, a (positive) error number is returned (since glibc 2.13), or −1 is returned and *errno* is set to indicate the error (before glibc 2.13).

POSIX.1-2001 and POSIX.1-2008 require that a successful call to **strerror**() or **strerror_l**() shall leave *errno* unchanged, and note that, since no function return value is reserved to indicate an error, an application that wishes to check for errors should initialize *errno* to zero before the call, and then check *errno* after the call.

ERRORS

EINVAL

The value of *errnum* is not a valid error number.

ERANGE

Insufficient storage was supplied to contain the error description string.

VERSIONS

The **strerror_l**() function first appeared in glibc 2.6.

The **strerrorname_np**() and **strerrordesc_np**() functions first appeared in glibc 2.32.

ATTRIBUTES

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

Interface	Attribute	Value
strerror()	Thread safety	MT-Unsafe race:strerror
strerrorname_np(), **strerrordesc_np**()	Thread safety	MT-Safe
strerror_r(), **strerror_l**()	Thread safety	MT-Safe

STANDARDS

strerror() is specified by POSIX.1-2001, POSIX.1-2008, and C99. **str error_r**() is specified by POSIX.1-2001 and POSIX.1-2008.

strerror_l() is specified in POSIX.1-2008.

The GNU-specific functions **strerror_r**(), **strerrorname_np**(), and **strerrordesc_np**() are nonstandard extensions.

POSIX.1-2001 permits **strerror**() to set *errno* if the call encounters an error, but does not specify what value should be returned as the function result in the event of an error. On some systems,**str error**() returns NULL if the error number is unknown. On other systems,**str error**() returns a string something like "Error nnn occurred" and sets *errno* to **EINVAL** if the error number is unknown. C99 and POSIX.1-2008 require the return value to be non-NULL.

NOTES

The GNU C Library uses a buffer of 1024 characters for **strerror**(). This buffer size therefore should be sufficient to avoid an **ERANGE** error when calling **strerror_r**().

strerrorname_np() and **strerrordesc_np**() are thread-safe and async-signal-safe.

SEE ALSO

err(3), **errno**(3), **error**(3), **perror**(3), **strsignal**(3), **locale**(7)