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

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'strerror.3p' command

$ man strerror.3p

STRERROR(3P) POSIX Programmer's Manual STRERROR(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

strerror, strerror_l, strerror_r ? get error message string

SYNOPSIS

```
#include <string.h>
char *strerror(int errnum);
char *strerror_l(int errnum, locale_t locale);
int strerror_r(int errnum, char *strerrbuf, size_t buflen);
```

DESCRIPTION

For strerror(): The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1?2017 defers to the ISO C standard.

The strerror() function shall map the error number in errnum to a lo? cale-dependent error message string and shall return a pointer to it. Typically, the values for errnum come from errno, but strerror() shall map any value of type int to a message.

The application shall not modify the string returned. The returned

string pointer might be invalidated or the string content might be overwritten by a subsequent call to strerror(), or by a subsequent call to strerror_l() in the same thread. The returned pointer and the string content might also be invalidated if the calling thread is terminated.

The string may be overwritten by a subsequent call to strerror_l() in the same thread.

The contents of the error message strings returned by strerror() should be determined by the setting of the LC_MESSAGES category in the current locale.

The implementation shall behave as if no function defined in this vol?ume of POSIX.1?2017 calls strerror().

The strerror() and strerror_l() functions shall not change the setting of errno if successful.

Since no return value is reserved to indicate an error of strerror(), an application wishing to check for error situations should set errno to 0, then call strerror(), then check errno. Similarly, since str?error_l() is required to return a string for some errors, an applica?tion wishing to check for all error situations should set errno to 0, then call strerror_l(), then check errno.

The strerror() function need not be thread-safe.

The strerror_l() function shall map the error number in errnum to a lo?cale-dependent error message string in the locale represented by locale and shall return a pointer to it.

The strerror_r() function shall map the error number in errnum to a lo?cale-dependent error message string and shall return the string in the buffer pointed to by strerrbuf, with length buflen.

If the value of errnum is a valid error number, the message string shall indicate what error occurred; if the value of errnum is zero, the message string shall either be an empty string or indicate that no er?ror occurred; otherwise, if these functions complete successfully, the message string shall indicate that an unknown error occurred.

The behavior is undefined if the locale argument to strerror_l() is the special locale object LC_GLOBAL_LOCALE or is not a valid locale object

handle.

RETURN VALUE

Upon completion, whether successful or not, strerror() shall return a pointer to the generated message string. On error errno may be set, but no return value is reserved to indicate an error.

Upon successful completion, strerror_l() shall return a pointer to the generated message string. If errnum is not a valid error number, errno may be set to [EINVAL], but a pointer to a message string shall still be returned. If any other error occurs, errno shall be set to indicate the error and a null pointer shall be returned.

Upon successful completion, strerror_r() shall return 0. Otherwise, an error number shall be returned to indicate the error.

ERRORS

These functions may fail if:

EINVAL The value of errnum is neither a valid error number nor zero.

The strerror_r() function may fail if:

ERANGE Insufficient storage was supplied via strerrbuf and buflen to contain the generated message string.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

Historically in some implementations, calls to perror() would overwrite the string that the pointer returned by strerror() points to. Such im? plementations did not conform to the ISO C standard; however, applica? tion developers should be aware of this behavior if they wish their ap? plications to be portable to such implementations.

RATIONALE

The strerror_l() function is required to be thread-safe, thereby elimi? nating the need for an equivalent to the strerror_r() function.

Earlier versions of this standard did not explicitly require that the error message strings returned by strerror() and strerror_r() provide any information about the error. This version of the standard requires

a meaningful message for any successful completion.

Since no return value is reserved to indicate a strerror() error, but all calls (whether successful or not) must return a pointer to a mes? sage string, on error strerror() can return a pointer to an empty string or a pointer to a meaningful string that can be printed.

Note that the [EINVAL] error condition is a may fail error. If an in? valid error number is supplied as the value of errnum, applications should be prepared to handle any of the following:

1. Error (with no meaningful message): errno is set to [EINVAL], the return value is a pointer to an empty string.
2. Successful completion: errno is unchanged and the return value points to a string like "unknownerror" or "errornumberxxx" (where xxx is the value of errnum).
3. Combination of #1 and #2: errno is set to [EINVAL] and the return value points to a string like "unknownerror" or "errornumberxxx" (where xxx is the value of errnum). Since applications frequently use the return value of strerror() as an argument to functions like fprintf() (without checking the return value) and since applica? tions have no way to parse an error message string to determine whether errnum represents a valid error number, implementations are encouraged to implement #3. Similarly, implementations are encour? aged to have strerror_r() return [EINVAL] and put a string like "unknownerror" or "errornumberxxx" in the buffer pointed to by str? errbuf when the value of errnum is not a valid error number.

Some applications rely on being able to set errno to 0 before calling a function with no reserved value to indicate an error, then call str? error(errno) afterwards to detect whether an error occurred (because errno changed) or to indicate success (because errno remained zero). This usage pattern requires that strerror(0) succeed with useful re? sults. Previous versions of the standard did not specify the behavior when errnum is zero.

FUTURE DIRECTIONS

None.

SEE ALSO

perror()

The Base Definitions volume of POSIX.1?2017, <string.h>

COPYRIGHT