



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 'sys_nerr.3' command

\$ man sys_nerr.3

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

NAME

 perror - print a system error message

SYNOPSIS

```
#include <stdio.h>

void perror(const char *s);

#include <errno.h>

const char * const sys_errlist[];

int sys_nerr;

int errno;     /* Not really declared this way; see errno(3) */
```

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

sys_errlist, sys_nerr:

 From glibc 2.19 to 2.31:

 _DEFAULT_SOURCE

 Glibc 2.19 and earlier:

 _BSD_SOURCE

DESCRIPTION

The `perror()` function produces a message on standard error describing the last error encountered during a call to a system or library function.

First (if `s` is not `NULL` and `*s` is not a null byte (`'\0'`)), the argument string `s` is printed, followed by a colon and a blank. Then an error message corresponding to the current value of `errno` and a new-line.

To be of most use, the argument string should include the name of the function that incurred the error.

The global error list `sys_errlist[]`, which can be indexed by `errno`, can be used to obtain the error message without the newline. The largest message number provided in the table is `sys_nerr-1`. Be careful when directly accessing this list, because new error values may not have been added to `sys_errlist[]`. The use of `sys_errlist[]` is nowadays deprecated; use `strerror(3)` instead.

When a system call fails, it usually returns `-1` and sets the variable `errno` to a value describing what went wrong. (These values can be found in `<errno.h>`.) Many library functions do likewise. The function `perror()` serves to translate this error code into human-readable form. Note that `errno` is undefined after a successful system call or library function call: this call may well change this variable, even though it succeeds, for example because it internally used some other library function that failed. Thus, if a failing call is not immediately followed by a call to `perror()`, the value of `errno` should be saved.

VERSIONS

Since glibc version 2.32, the declarations of `sys_errlist` and `sys_nerr` are no longer exposed by `<stdio.h>`.

ATTRIBUTES

For an explanation of the terms used in this section, see [attributes\(7\)](#).

??

?Interface ? Attribute ? Value ?

??

?`perror()` ? Thread safety ? MT-Safe race:stderr ?

??

CONFORMING TO

`perror()`, `errno`: POSIX.1-2001, POSIX.1-2008, C89, C99, 4.3BSD.

The externals `sys_nerr` and `sys_errlist` derive from BSD, but are not specified in POSIX.1.

NOTES

The externals `sys_nerr` and `sys_errlist` are defined by `glibc`, but in `<stdio.h>`.

SEE ALSO

`err(3)`, `errno(3)`, `error(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/>.

2020-11-01

PERROR(3)