NAME

nearbyint, nearbyintf, nearbyintl, rint, rintf, rintl − round to nearest integer

SYNOPSIS

```
#include <math.h>

double nearbyint(double x);
float nearbyintf(float x);
long double nearbyintl(long double x);

double rint(double x);
float rintf(float x);
long double rintl(long double x);
```

Link with *−lm*.

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

nearbyint(), **nearbyintf**(), **nearbyintl**():
_POSIX_C_SOURCE >= 200112L || _ISOC99_SOURCE
rint():
_ISOC99_SOURCE || _POSIX_C_SOURCE >= 200112L
|| _XOPEN_SOURCE >= 500
|| /* Since glibc 2.19: */ _DEFAULT_SOURCE
|| /* Glibc versions <= 2.19: */ _BSD_SOURCE || _SVID_SOURCE
rintf(), **rintl**():
_ISOC99_SOURCE || _POSIX_C_SOURCE >= 200112L
|| /* Since glibc 2.19: */ _DEFAULT_SOURCE
|| /* Glibc versions <= 2.19: */ _BSD_SOURCE || _SVID_SOURCE

DESCRIPTION

The **nearbyint**(), **nearbyintf**(), and **nearbyintl**() functions round their argument to an integer value in floating-point format, using the current rounding direction (see **fesetround**(3)) and without raising the *inexact* exception. When the current rounding direction is to nearest, these functions round halfway cases to the even integer in accordance with IEEE-754.

The **rint**(), **rintf**(), and **rintl**() functions do the same, but will raise the *inexact* exception (**FE_INEXACT**, checkable via **fetestexcept**(3)) when the result differs in value from the argument.

RETURN VALUE

These functions return the rounded integer value.

If *x* is integral, +0, −0, NaN, or infinite, *x* itself is returned.

ERRORS

No errors occur. POSIX.1-2001 documents a range error for overflows, but see NOTES.

ATTRIBUTES

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

Interface	Attribute	Value
nearbyint(), **nearbyintf**(), **nearbyintl**(), **rint**(), **rintf**(), **rintl**()	Thread safety	MT-Safe

CONFORMING TO

C99, POSIX.1-2001, POSIX.1-2008.

NOTES

SUSv2 and POSIX.1-2001 contain text about overflow (which might set *errno* to **ERANGE**, or raise an **FE_OVERFLOW** exception). In practice, the result cannot overflow on any current machine, so this error-handling stuff is just nonsense. (More precisely, overflow can happen only when the maximum value of the exponent is smaller than the number of mantissa bits. For the IEEE-754 standard 32-bit and 64-bit floating-

point numbers the maximum value of the exponent is 128 (respectively, 1024), and the number of mantissa bits is 24 (respectively, 53).)

If you want to store the rounded value in an integer type, you probably want to use one of the functions described in **lrint**(3) instead.

SEE ALSO

ceil(3), **floor**(3), **lrint**(3), **round**(3), **trunc**(3)

COLOPHON

This page is part of release 5.05 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/.