NAME

acos, acosf, acosl − arc cosine function

SYNOPSIS

#include <math.h>

double acos(double *x***);**
float acosf(float *x***);**
long double acosl(long double *x***);**

DESCRIPTION

The **acos()** function calculates the arc cosine of *x*; that is the value whose cosine is *x*. If *x* falls outside the range −1 to 1, **acos()** fails and *errno* is set.

RETURN VALUE

The **acos()** function returns the arc cosine in radians and the value is mathematically defined to be between 0 and PI (inclusive).

ERRORS

EDOM
x is out of range.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and long double variants are C99 requirements.

SEE ALSO

asin(3), **atan**(3), **atan2**(3), **cos**(3), **sin**(3), **tan**(3)

NAME

acosh, acoshf, acoshl − inverse hyperbolic cosine function

SYNOPSIS

#include <math.h>

double acosh(double *x***);**

float acoshf(float *x***);**

long double acoshl(long double *x***);**

DESCRIPTION

The **acosh()** function calculates the inverse hyperbolic cosine of *x*; that is the value whose hyperbolic cosine is *x*. If *x* is less than 1.0, **acosh()** returns not-a-number (NaN) and *errno* is set.

ERRORS

EDOM

x is out of range.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and long double variants are C99 requirements.

SEE ALSO

asinh(3), **atanh**(3), **cosh**(3), **sinh**(3), **tanh**(3)

NAME

asin, asinf, asinl − arc sine function

SYNOPSIS

#include <math.h>

double asin(double *x***);**

float asinf(float *x***);**

long double asinl(long double *x***);**

DESCRIPTION

The **asin()** function calculates the arc sine of *x*; that is the value whose sine is *x*. If *x* falls outside the range −1 to 1, **asin()** fails and *errno* is set.

RETURN VALUE

The **asin()** function returns the arc sine in radians and the value is mathematically defined to be between -PI/2 and PI/2 (inclusive).

ERRORS

EDOM

x is out of range.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and long double variants are C99 requirements.

SEE ALSO

acos(3), **atan**(3), **atan2**(3), **cos**(3), **sin**(3), **tan**(3)

NAME

asinh, asinhf, asinhl − inverse hyperbolic sine function

SYNOPSIS

#include <math.h>

double asinh(double *x***);**

float asinhf(float *x***);**

long double asinhl(long double *x***);**

DESCRIPTION

The **asinh()** function calculates the inverse hyperbolic sine of *x*; that is the value whose hyperbolic sine is *x*.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and long double variants are C99 requirements.

SEE ALSO

acosh(3), **atanh**(3), **cosh**(3), **sinh**(3), **tanh**(3)

NAME

atan, atanf, atanl − arc tangent function

SYNOPSIS

#include <math.h>

double atan(double *x***);**

float atanf(float *x***);**

long double atanl(long double *x***);**

DESCRIPTION

The **atan()** function calculates the arc tangent of *x*; that is the value whose tangent is *x*.

RETURN VALUE

The **atan()** function returns the arc tangent in radians and the value is mathematically defined to be between -PI/2 and PI/2 (inclusive).

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and long double variants are C99 requirements.

SEE ALSO

acos(3), **asin**(3), **atan2**(3), **cos**(3), **sin**(3), **tan**(3)

NAME

atan2, atan2f, atan2l − arc tangent function of two variables

SYNOPSIS

#include <math.h>

double atan2(double *y***, double** *x***);**

float atan2f(float *y***, float** *x***);**

long double atan2l(long double *y***, long double** *x***);**

DESCRIPTION

The **atan2()** function calculates the arc tangent of the two variables *x* and *y*. It is similar to calculating the arc tangent of *y* / *x*, except that the signs of both arguments are used to determine the quadrant of the result.

RETURN VALUE

The **atan2()** function returns the result in radians, which is between -PI and PI (inclusive).

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and long double variants are C99 requirements.

SEE ALSO

acos(3), **asin**(3), **atan**(3), **cos**(3), **sin**(3), **tan**(3)

NAME

atanh, atanhf, atanhl − inverse hyperbolic tangent function

SYNOPSIS

#include <math.h>

double atanh(double *x***);**

float atanhf(float *x***);**

long double atanhl(long double *x***);**

DESCRIPTION

The **atanh()** function calculates the inverse hyperbolic tangent of *x*; that is the value whose hyperbolic tangent is *x*. If the absolute value of *x* is greater than 1.0, **acosh()** returns not-a-number (NaN) and *errno* is set.

ERRORS

EDOM
x is out of range.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and long double variants are C99 requirements.

SEE ALSO

asinh(3), **acosh**(3), **cosh**(3), **sinh**(3), **tanh**(3)

NAME

cbrt, cbrtf, cbrtl − cube root function

SYNOPSIS

#include <math.h>

double cbrt(double *x***);**

float cbrtf(float *x***);**

long double cbrtl(long double *x***);**

DESCRIPTION

The **cbrt()** function returns the (real) cube root of *x*. This function cannot fail; every representable real value has a representable real cube root.

CONFORMING TO

cbrt() was a GNU extension. It is now a C99 requirement.

SEE ALSO

sqrt(3), **pow**(3)

NAME

ceil, ceilf, ceill − ceiling function: smallest integral value not less than argument

SYNOPSIS

#include <math.h>

double ceil(double *x***);**
float ceilf(float *x***);**
long double ceill(long double *x***);**

DESCRIPTION

These functions round *x* up to the nearest integer.

RETURN VALUE

The rounded integer value. If *x* is integral or infinite, *x* itself is returned.

ERRORS

No errors other than EDOM and ERANGE can occur. If *x* is NaN, then NaN is returned and *errno* may be set to EDOM.

NOTES

SUSv2 and POSIX 1003.1-2001 contain text about overflow (which might set *errno* to ERANGE, or raise an 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 (resp. 1024), and the number of mantissa bits is 24 (resp. 53).)

CONFORMING TO

The **ceil()** function conforms to SVID 3, POSIX, BSD 4.3, ISO 9899. The other functions are from C99.

SEE ALSO

floor(3), **lrint**(3), **nearbyint**(3), **rint**(3), **round**(3), **trunc**(3)

NAME

copysign, copysignf, copysignl − copy sign of a number

SYNOPSIS

#include <math.h>

double copysign(double *x***, double** *y***);**
float copysignf(float *x***, float** *y***);**
long double copysignl(long double *x***, long double** *y***);**

DESCRIPTION

The **copysign()** functions return a value whose absolute value matches that of *x*, but whose sign matches that of *y*. If *x* is a NaN, then a NaN with the sign of *y* is returned.

NOTES

The **copysign()** functions may treat a negative zero as positive.

CONFORMING TO

C99, BSD 4.3. This function is defined in IEC 559 (and the appendix with recommended functions in IEEE 754/IEEE 854).

SEE ALSO

signbit(3)

NAME

cos, cosf, cosl − cosine function

SYNOPSIS

#include <math.h>

double cos(double *x***);**

float cosf(float *x***);**

long double cosl(long double *x***);**

DESCRIPTION

The **cos()** function returns the cosine of *x*, where *x* is given in radians.

RETURN VALUE

The **cos()** function returns a value between −1 and 1.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and the long double variants are C99 requirements.

SEE ALSO

acos(3), **asin**(3), **atan**(3), **atan2**(3), **sin**(3), **tan**(3)

NAME

cosh, coshf, coshl − hyperbolic cosine function

SYNOPSIS

#include <math.h>

double cosh(double *x***);**

float coshf(float *x***);**

long double coshl(long double *x***);**

DESCRIPTION

The **cosh()** function returns the hyperbolic cosine of *x*, which is defined mathematically as (exp(x) + exp(-x)) / 2.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899 (C99). The float and the long double variants are C99 requirements.

SEE ALSO

acosh(3), **asinh**(3), **atanh**(3), **sinh**(3), **tanh**(3)

NAME

erf, erff, erfl, erfc, erfcf, erfcl − error function and complementary error function

SYNOPSIS

#include <math.h>

double erf(double *x***);**

float erff(float *x***);**

long double erfl(long double *x***);**

double erfc(double *x***);**

float erfcf(float *x***);**

long double erfcl(long double *x***);**

DESCRIPTION

The **erf()** function returns the error function of *x*; defined as

erf(x) = 2/sqrt(pi)* integral from 0 to x of exp(-t*t) dt

The **erfc()** function returns the complementary error function of *x*, that is 1.0 - erf(x).

CONFORMING TO

SVID 3, BSD 4.3, C99. The float and long double variants are requirements of C99.

SEE ALSO

exp(3)

NAME

exp, expf, expl − base-e exponential function

SYNOPSIS

#include <math.h>

double exp(double *x***);**

float expf(float *x***);**

long double expl(long double *x***);**

DESCRIPTION

The **exp()** function returns the value of e (the base of natural logarithms) raised to the power of *x*.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and the long double variants are C99 requirements.

SEE ALSO

sqrt(3), **cbrt**(3) **exp10**(3), **exp2**(3)

NAME

expm1, expm1f, expm1l − exponential minus 1

SYNOPSIS

#include <math.h>

double expm1(double *x***);**

float expm1f(float *x***);**

long double expm1l(long double *x***);**

DESCRIPTION

expm1(*x***)** returns a value equivalent to 'exp (*x*) - 1'. It is computed in a way that is accurate even if the value of *x* is near zero--a case where 'exp (*x*) - 1' would be inaccurate due to subtraction of two numbers that are nearly equal.

CONFORMING TO

BSD, C99. The float and the long double variants are C99 requirements.

SEE ALSO

exp(3), **log**(3), **log1p**(3)

NAME

fabs, fabsf, fabsl − absolute value of floating-point number

SYNOPSIS

#include <math.h>

double fabs(double *x***);**
float fabsf(float *x***);**
long double fabsl(long double *x***);**

DESCRIPTION

The **fabs** functions return the absolute value of the floating-point number *x*.

ERRORS

No errors can occur.

CONFORMING TO

The **fabs()** function conforms to SVID 3, POSIX, BSD 4.3, ISO 9899. The other functions are from C99.

SEE ALSO

abs(3), **ceil**(3), **floor**(3), **labs**(3), **rint**(3)

NAME

floor, floorf, floorl − largest integral value not greater than argument

SYNOPSIS

#include <math.h>

double floor(double *x***);**
float floorf(float *x***);**
long double floorl(long double *x***);**

DESCRIPTION

These functions round *x* down to the nearest integer.

RETURN VALUE

The rounded integer value. If *x* is integral or infinite, *x* itself is returned.

ERRORS

No errors other than EDOM and ERANGE can occur. If *x* is NaN, then NaN is returned and *errno* may be set to EDOM.

NOTES

SUSv2 and POSIX 1003.1-2001 contain text about overflow (which might set *errno* to ERANGE, or raise an 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 (resp. 1024), and the number of mantissa bits is 24 (resp. 53).)

CONFORMING TO

The **floor()** function conforms to SVID 3, POSIX, BSD 4.3, ISO 9899. The other functions are from C99.

SEE ALSO

ceil(3), **lrint**(3), **nearbyint**(3), **rint**(3), **round**(3), **trunc**(3)

NAME

fmod, fmodf, fmodl − floating-point remainder function

SYNOPSIS

#include <math.h>

double fmod(double *x***, double** *y***);**

float fmodf(float *x***, float** *y***);**

long double fmodl(long double *x***, long double** *y***);**

DESCRIPTION

The **fmod()** function computes the remainder of dividing *x* by *y*. The return value is *x* - *n* * *y*, where *n* is the quotient of *x* / *y*, rounded towards zero to an integer.

RETURN VALUE

The **fmod()** function returns the remainder, unless *y* is zero, when the function fails and *errno* is set.

ERRORS

EDOM
The denominator *y* is zero.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and the long double variants are C99 requirements.

SEE ALSO

remainder(3)

NAME

frexp, frexpf, frexpl − convert floating-point number to fractional and integral components

SYNOPSIS

#include <math.h>

double frexp(double *x***, int ****exp***);**

float frexpf(float *x***, int ****exp***);**

long double frexpl(long double *x***, int ****exp***);**

DESCRIPTION

The **frexp()** function is used to split the number *x* into a normalized fraction and an exponent which is stored in *exp*.

RETURN VALUE

The **frexp()** function returns the normalized fraction. If the argument *x* is not zero, the normalized fraction is *x* times a power of two, and is always in the range 1/2 (inclusive) to 1 (exclusive). If *x* is zero, then the normalized fraction is zero and zero is stored in *exp*.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and the long double variants are C99 requirements.

EXAMPLE

```
#include <stdio.h>
#include <math.h>
#include <float.h>
int main () {
    double d = 2560;
    int e;
    double f = frexp(d, &e);
    printf("frexp(%g, &e) = %g: %g * %d^%d = %g\n",
        d, f, f, FLT_RADIX, e, d);
    return 0;
}
```

This program prints

```
frexp(2560, &e) = 0.625: 0.625 * 2^12 = 2560
```

SEE ALSO

ldexp(3), **modf**(3)

NAME

gamma, gammaf, gammal − (logarithm of the) gamma function

SYNOPSIS

#include <math.h>

double gamma(double *x***);**

float gammaf(float *x***);**

long double gammal(long double *x***);**

DESCRIPTION

For the definition of the Gamma function, see **tgamma**(3).

*BSD version

4.4BSD and FreeBSD libm have a **gamma()** function that computes the Gamma function, as one would expect.

glibc version

Glibc has a **gamma()** function that is equivalent to **lgamma()** and computes the natural logarithm of the Gamma function. (This is for compatibility reasons only. Don’t use this function.)

HISTORY

4.2BSD had a **gamma()** that computed ln(|Gamma(|*x*|)|), leaving the sign of Gamma(|*x*|) in the external integer *signgam*. In 4.3BSD the name was changed to **lgamma()**, and the man page promises

> "At some time in the future the name gamma will be rehabilitated and used for the Gamma function"

This did indeed happen in 4.4BSD, where **gamma()** computes the Gamma function (with no effect on *signgam*). However, this came too late, and we now have **tgamma()**, the "true gamma" function.

CONFORMING TO

4.2BSD. Compatible with previous mistakes.

SEE ALSO

lgamma(3), **signgam**(3), **tgamma**(3)

NAME

hypot, hypotf, hypotl − Euclidean distance function

SYNOPSIS

#include <math.h>

double hypot(double *x*, **double** *y***);**

float hypotf(float *x*, **float** *y***);**

long double hypotl (long double *x*, **long double** *y***);**

DESCRIPTION

The **hypot()** function returns sqrt(*x***x*+*y***y*). This is the length of the hypotenuse of a right-angle triangle with sides of length *x* and *y*, or the distance of the point (*x*,*y*) from the origin.

CONFORMING TO

SVID 3, BSD 4.3, C99. The float and the long double variants are C99 requirements.

SEE ALSO

sqrt(3), **cabs**(3)

NAME

isinf, isnan, finite − test for infinity or not-a-number (NaN)

SYNOPSIS

#include <math.h>

int isinf(double *value***);**

int isnan(double *value***);**

int finite(double *value***);**

DESCRIPTION

The **isinf()** function returns −1 if *value* represents negative infinity, 1 if *value* represents positive infinity, and 0 otherwise.

The **isnan()** function returns a non-zero value if *value* is "not-a-number" (NaN), and 0 otherwise.

The **finite()** function returns a non-zero value if *value* is neither infinite nor a "not-a-number" (NaN) value, and 0 otherwise.

NOTE

C99 provides additional macros, such as the type-independent **fpclassify()**, **isinf()** and **isnan()**.

CONFORMING TO

BSD 4.3

SEE ALSO

fpclassify(3)

NAME

j0, j0f, j0l, j1, j1f, j1l, jn, jnf, jnl, y0, y0f, y0l, y1, y1f, y1l, yn, ynf, ynl − Bessel functions

SYNOPSIS

```
#include <math.h>

double j0(double x);
double j1(double x);
double jn(int n, double x);
double y0(double x);
double y1(double x);
double yn(int n, double x);

float j0f(float x);
float j1f(float x);
float jnf(int n, float x);
float y0f(float x);
float y1f(float x);
float ynf(int n, float x);

long double j0l(long double x);
long double j1l(long double x);
long double jnl(int n, long double x);
long double y0l(long double x);
long double y1l(long double x);
long double ynl(int n, long double x);
```

DESCRIPTION

The **j0()** and **j1()** functions return Bessel functions of *x* of the first kind of orders 0 and 1, respectively. The **jn()** function returns the Bessel function of *x* of the first kind of order *n*.

The **y0()** and **y1()** functions return Bessel functions of *x* of the second kind of orders 0 and 1, respectively. The **yn()** function returns the Bessel function of *x* of the second kind of order *n*.

For the functions **y0()**, **y1()** and **yn()**, the value of *x* must be positive. For negative values of *x*, these functions return −HUGE_VAL.

The **j0f()** etc. and **j0l()** etc. functions are versions that take and return float and long double values, respectively.

CONFORMING TO

The functions returning double conform to SVID 3, BSD 4.3, XPG4, POSIX 1003.1-2001. The other functions are C99 requirements.

BUGS

There are errors of up to 2e−16 in the values returned by **j0()**, **j1()** and **jn()** for values of *x* between −8 and 8.

NAME

ldexp, ldexpf, ldexpl − multiply floating-point number by integral power of 2

SYNOPSIS

#include <math.h>

double ldexp(double *x***, int** *exp***);**

float ldexp(float *x***, int** *exp***);**

long double ldexp(long double *x***, int** *exp***);**

DESCRIPTION

The **ldexp()** function returns the result of multiplying the floating-point number *x* by 2 raised to the power *exp*.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and the long double variants are C99 requirements.

SEE ALSO

frexp(3), **modf**(3)

NAME

lgamma, lgammaf, lgammal, lgamma_r, lgammaf_r, lgammal_r − log gamma function

SYNOPSIS

#include <math.h>

double lgamma(double *x***);**
float lgammaf(float *x***);**
long double lgammal(long double *x***);**

double lgamma_r(double *x***, int ***signp***);**
float lgammaf_r(float *x***, int ***signp***);**
long double lgammal_r(long double *x***, int ***signp***);**

DESCRIPTION

For the definition of the Gamma function, see **tgamma**(3).

The **lgamma()** function returns the natural logarithm of the absolute value of the Gamma function. The sign of the Gamma function is returned in the external integer *signgam* declared in <*math.h*>. It is 1 when the Gamma function is positive or zero, −1 when it is negative.

Since using a constant location *signgam* is not thread-safe, the functions **lgamma_r()** etc. have been introduced; they return this sign via the parameter *signp*.

For nonpositive integer values of *x*, **lgamma()** returns HUGE_VAL, sets *errno* to ERANGE and raises the zero divide exception. (Similarly, **lgammaf()** returns HUGE_VALF and **lgammal()** returns HUGE_VALL.)

ERRORS

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept(FE_ALL_EXCEPT)* before calling these functions. On return, if *errno* is non-zero or *fetestexcept(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW)* is non-zero, an error has occurred.

A range error occurs if x is too large. A pole error occurs if x is a negative integer or zero.

CONFORMING TO

C99, SVID 3, BSD 4.3

SEE ALSO

tgamma(3)

NAME

log, logf, logl − natural logarithmic function

SYNOPSIS

#include <math.h>

double log(double *x***);**

float logf(float *x***);**

long double logl(long double *x***);**

DESCRIPTION

The **log()** function returns the natural logarithm of *x*.

ERRORS

The **log()** function can return the following errors:

EDOM

The argument *x* is negative.

ERANGE

The argument *x* is zero. The log of zero is not defined (minus infinity).

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and the long double variants are C99 requirements.

SEE ALSO

sqrt(3), **cbrt**(3)

NAME

log10, log10f, log10l − base-10 logarithmic function

SYNOPSIS

#include <math.h>

double log10(double *x***);**

float log10f(float *x***);**

long double log10l(long double *x***);**

DESCRIPTION

The **log10()** function returns the base 10 logarithm of *x*.

ERRORS

The **log10()** function can return the following errors:

EDOM
The argument *x* is negative.

ERANGE
The argument *x* is zero. The log of zero is not defined.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and the long double variants are C99 requirements.

SEE ALSO

sqrt(3), **cbrt**(3)

NAME

log1p − logarithm of 1 plus argument

SYNOPSIS

#include <math.h>

double log1p(double *x***);**

float log1pf(float *x***);**

long double log1pl(long double *x***);**

DESCRIPTION

log1p(*x***)** returns a value equivalent to ‘log (1 + *x*)’. It is computed in a way that is accurate even if the value of *x* is near zero.

CONFORMING TO

BSD, C99. The float and the long double variants are C99 requirements.

SEE ALSO

exp(3), **log**(3), **expm1**(3)

NAME

modf, modff, modfl − extract signed integral and fractional values from floating-point number

SYNOPSIS

#include <math.h>

double modf(double *x***, double ****iptr***);**

float modff(float *x***, float ****iptr***);**

long double modfl(long double *x***, long double ****iptr***);**

DESCRIPTION

The **modf()** function breaks the argument *x* into an integral part and a fractional part, each of which has the same sign as *x*. The integral part is stored in *iptr*.

RETURN VALUE

The **modf()** function returns the fractional part of *x*.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and the long double variants are C99 requirements.

SEE ALSO

frexp(3), **ldexp**(3)

NAME

nextafter, nextafterf, nextafterl, nexttoward, nexttowardf, nexttowardl − floating point number manipulation

SYNOPSIS

#include <math.h>

double nextafter(double *x*, **double** *y*);
float nextafterf(float *x*, **float** *y*);
long double nextafterl(long double *x*, **long double** *y*);

double nexttoward(double *x*, **long double** *y*);
float nexttowardf(float *x*, **long double** *y*);
long double nexttowardl(long double *x*, **long double** *y*);

DESCRIPTION

The **nextafter()** functions return the next representable neighbor of *x* in the direction towards *y*. The size of the step between *x* and the result depends on the type of the result. If *x* = *y* the function simply returns *y*. If either value is *NaN*, then *NaN* is returned. Otherwise a value corresponding to the value of the least significant bit in the mantissa is added or subtracted, depending on the direction.

The **nexttoward()** functions do the same as the **nextafter()** functions, except that they have a long double second argument.

These functions will signal overflow or underflow if the result goes outside of the range of normalized numbers.

CONFORMING TO

C99. This function is defined in IEC 559 (and the appendix with recommended functions in IEEE 754/IEEE 854).

SEE ALSO

nearbyint(3)

NAME

pow, powf, powl − power functions

SYNOPSIS

#include <math.h>

double pow(double *x***, double** *y***);**

float powf(float *x***, float** *y***);**

long double powl(long double *x***, long double** *y***);**

DESCRIPTION

The **pow()** function returns the value of *x* raised to the power of *y*.

ERRORS

The **pow()** function can return the following error:

EDOM

The argument *x* is negative and *y* is not an integral value. This would result in a complex number.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and the long double variants are C99 requirements.

SEE ALSO

sqrt(3), **cbrt**(3)

NAME

drem, dremf, dreml, remainder, remainderf, remainderl − floating-point remainder function

SYNOPSIS

#include <math.h>

/* The C99 versions */
double remainder(double *x***, double** *y***);**
float remainderf(float *x***, float** *y***);**
long double remainderl(long double *x***, long double** *y***);**

/* Obsolete synonyms */
double drem(double *x***, double** *y***);**
float dremf(float *x***, float** *y***);**
long double dreml(long double *x***, long double** *y***);**

Link with −lm.

DESCRIPTION

The **remainder()** function computes the remainder of dividing *x* by *y*. The return value is *x* - *n* * *y*, where *n* is the value *x* / *y*, rounded to the nearest integer. If this quotient is 1/2, it is rounded to the nearest even number (independent of the current rounding mode). If the return value is 0, it has the sign of *x*.

The **drem()** function does precisely the same thing.

RETURN VALUE

The **remainder()** function returns the remainder, unless *y* is zero, when the function fails and *errno* is set.

ERRORS

EDOM
The denominator *y* is zero.

CONFORMING TO

IEC 60559. The three **remainder*()** functions are from C99. The function **drem()** is from BSD 4.3. The float and long double variants **dremf()** and **dreml()** exist on some systems, such as Tru64 and glibc2.

EXAMPLE

The call "remainder(29.0, 3.0)" returns −1.

SEE ALSO

fmod(3)

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***);**

DESCRIPTION

The **nearbyint** functions round their argument to an integer value in floating point format, using the current rounding direction and without raising the *inexact* exception.

The **rint** functions do the same, but will raise the *inexact* exception when the result differs in value from the argument.

RETURN VALUE

The rounded integer value. If *x* is integral or infinite, *x* itself is returned.

ERRORS

No errors other than EDOM and ERANGE can occur. If *x* is NaN, then NaN is returned and *errno* may be set to EDOM.

NOTES

SUSv2 and POSIX 1003.1-2001 contain text about overflow (which might set *errno* to ERANGE, or raise an 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 (resp. 1024), and the number of mantissa bits is 24 (resp. 53).)

CONFORMING TO

The **rint()** function conforms to BSD 4.3. The other functions are from C99.

SEE ALSO

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

NAME

sin, sinf, sinl − sine function

SYNOPSIS

#include <math.h>

double sin(double *x***);**

float sinf(float *x***);**

long double sinl(long double *x***);**

DESCRIPTION

The **sin()** function returns the sine of *x*, where *x* is given in radians.

RETURN VALUE

The **sin()** function returns a value between −1 and 1.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and the long double variants are C99 requirements.

SEE ALSO

acos(3), **asin**(3), **atan**(3), **atan2**(3), **cos**(3), **tan**(3)

NAME

sinh, sinhf, sinhl − hyperbolic sine function

SYNOPSIS

#include <math.h>

double sinh(double *x***);**

float sinhf(float *x***);**

long double sinhl(long double *x***);**

DESCRIPTION

The **sinh()** function returns the hyperbolic sine of *x*, which is defined mathematically as (exp(x) - exp(-x)) / 2.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899 (C99). The float and the long double variants are C99 requirements.

SEE ALSO

acosh(3), **asinh**(3), **atanh**(3), **cosh**(3), **tanh**(3)

NAME

sqrt, sqrtf, sqrtl − square root function

SYNOPSIS

#include <math.h>

double sqrt(double *x***);**

float sqrtf(float *x***);**

long double sqrtl(long double *x***);**

DESCRIPTION

The **sqrt()** function returns the non-negative square root of *x*. It fails and sets *errno* to EDOM, if *x* is negative.

ERRORS

EDOM

x is negative.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and the long double variants are C99 requirements.

SEE ALSO

hypot(3)

NAME

tan, tanf, tanl − tangent function

SYNOPSIS

#include <math.h>

double tan(double *x***);**

float tanf(float *x***);**

long double tanl(long double *x***);**

DESCRIPTION

The **tan()** function returns the tangent of *x*, where *x* is given in radians.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899. The float and the long double variants are C99 requirements.

SEE ALSO

acos(3), **asin**(3), **atan**(3), **atan2**(3), **cos**(3), **sin**(3)

NAME

tanh, tanhf, tanhl − hyperbolic tangent function

SYNOPSIS

#include <math.h>

double tanh(double *x***);**

float tanhf(float *x***);**

long double tanhl(long double *x***);**

DESCRIPTION

The **tanh()** function returns the hyperbolic tangent of *x*, which is defined mathematically as sinh(x) / cosh(x).

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899 (C99). The float and the long double variants are C99 requirements.

SEE ALSO

acosh(3), **asinh**(3), **atanh**(3), **cosh**(3), **sinh**(3)

NAME

acos, acosf, acosl - arc cosine functions

SYNOPSIS

#include <math.h>

double acos(double *x***);**
float acosf(float *x***);**
long double acosl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the principal value of the arc cosine of their argument *x*. The value of *x* should be in the range [-1,1].

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the arc cosine of *x*, in the range [0,] radians.

For finite values of *x* not in the range [-1,1], a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

If *x* is NaN, a NaN shall be returned.

If *x* is +1, +0 shall be returned.

If *x* is ±Inf, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

ERRORS

These functions shall fail if:

Domain Error
The *x* argument is finite and is not in the range [-1,1], <img src="../images/opt-start.gif" alt="[Option Start]" border="0"> or is ±Inf.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

cos() , *feclearexcept*() , *fetestexcept*() , *isnan*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

acosh, acoshf, acoshl - inverse hyperbolic cosine functions

SYNOPSIS

#include <math.h>

double acosh(double *x***);**
float acoshf(float *x***);**
long double acoshl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the inverse hyperbolic cosine of their argument *x*.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the inverse hyperbolic cosine of their argument.

For finite values of $x < 1$, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

If *x* is NaN, a NaN shall be returned.

If *x* is +1, +0 shall be returned.

If *x* is +Inf, +Inf shall be returned.

If *x* is -Inf, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

ERRORS

These functions shall fail if:

Domain Error
The *x* argument is finite and less than +1.0, or is -Inf.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

cosh() , *feclearexcept*() , *fetestexcept*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

asin, asinf, asinl - arc sine function

SYNOPSIS

#include <math.h>

double asin(double *x***);**
float asinf(float *x***);**
long double asinl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the principal value of the arc sine of their argument *x*. The value of *x* should be in the range [-1,1].

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the arc sine of *x*, in the range [-/2,/2] radians.

For finite values of *x* not in the range [-1,1], a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0, *x* shall be returned.

If *x* is ±Inf, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

If *x* is subnormal, a range error may occur and *x* should be returned.

ERRORS

These functions shall fail if:

Domain Error
    The *x* argument is finite and is not in the range [-1,1], <img src="../images/opt-start.gif" alt="[Option Start]" border="0"> or is ±Inf.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

These functions may fail if:

Range Error
    The value of *x* is subnormal.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *isnan*() , *sin*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, <*math.h*>

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

asinh, asinhf, asinhl - inverse hyperbolic sine functions

SYNOPSIS

#include <math.h>

double asinh(double *x***);**
float asinhf(float *x***);**
long double asinhl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the inverse hyperbolic sine of their argument *x*.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the inverse hyperbolic sine of their argument.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0, or ±Inf, *x* shall be returned.

If *x* is subnormal, a range error may occur and *x* should be returned.

ERRORS

These functions may fail if:

Range Error
    The value of *x* is subnormal.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *sinh*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, <*math.h*>

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the

referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

atan, atanf, atanl - arc tangent function

SYNOPSIS

#include <math.h>

double atan(double *x***);**
float atanf(float *x***);**
long double atanl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the principal value of the arc tangent of their argument *x*.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the arc tangent of *x* in the range [-/2,/2] radians.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0, *x* shall be returned.

If *x* is ±Inf, ±/2 shall be returned.

If *x* is subnormal, a range error may occur and *x* should be returned.

ERRORS

These functions may fail if:

Range Error
    The value of *x* is subnormal.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

atan2() , *feclearexcept*() , *fetestexcept*() , *isnan*() , *tan*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open

Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

atan2, atan2f, atan2l - arc tangent functions

SYNOPSIS

#include <math.h>

double atan2(double *y*, **double** *x***);**
float atan2f(float *y*, **float** *x***);**
long double atan2l(long double *y*, **long double** *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the principal value of the arc tangent of *y*/ *x*, using the signs of both arguments to determine the quadrant of the return value.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the arc tangent of *y*/ *x* in the range [-,] radians.

If *y* is ±0 and *x* is < 0, ± shall be returned.

If *y* is ±0 and *x* is > 0, ±0 shall be returned.

If *y* is < 0 and *x* is ±0, -/2 shall be returned.

If *y* is > 0 and *x* is ±0, /2 shall be returned.

If *x* is 0, a pole error shall not occur.

If either *x* or *y* is NaN, a NaN shall be returned.

If the result underflows, a range error may occur and *y*/ *x* should be returned.

If *y* is ±0 and *x* is -0, ± shall be returned.

If *y* is ±0 and *x* is +0, ±0 shall be returned.

For finite values of ± *y* > 0, if *x* is -Inf, ± shall be returned.

For finite values of ± *y* > 0, if *x* is +Inf, ±0 shall be returned.

For finite values of *x*, if *y* is ±Inf, ±/2 shall be returned.

If *y* is ±Inf and *x* is -Inf, ±3/4 shall be returned.

If *y* is ±Inf and *x* is +Inf, ±/4 shall be returned.

If both arguments are 0, a domain error shall not occur.

ERRORS

These functions may fail if:

Range Error
The result underflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

atan() , *feclearexcept*() , *fetestexcept*() , *isnan*() , *tan*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

atanh, atanhf, atanhl - inverse hyperbolic tangent functions

SYNOPSIS

#include <math.h>

double atanh(double *x***);**
float atanhf(float *x***);**
long double atanhl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the inverse hyperbolic tangent of their argument *x*.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the inverse hyperbolic tangent of their argument.

If *x* is ±1, a pole error shall occur, and *atanh*(), *atanhf*(), and *atanhl*() shall return the value of the macro HUGE_VAL, HUGE_VALF, and HUGE_VALL, respectively, with the same sign as the correct value of the function.

For finite |*x*|>1, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0, *x* shall be returned.

If *x* is ±Inf, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

If *x* is subnormal, a range error may occur and *x* should be returned.

ERRORS

These functions shall fail if:

Domain Error
The *x* argument is finite and not in the range [-1,1], or is ±Inf.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

Pole Error
The *x* argument is ±1.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the divide-by-zero floating-point exception shall be raised.

These functions may fail if:

Range Error
The value of *x* is subnormal.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *tanh*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, <*math.h*>

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

cbrt, cbrtf, cbrtl - cube root functions

SYNOPSIS

#include <math.h>

double cbrt(double *x***);**
float cbrtf(float *x***);**
long double cbrtl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the real cube root of their argument *x*.

RETURN VALUE

Upon successful completion, these functions shall return the cube root of *x*.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0 or ±Inf, *x* shall be returned.

ERRORS

No errors are defined.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

For some applications, a true cube root function, which returns negative results for negative arguments, is more appropriate than *pow*(*x*, 1.0/3.0), which returns a NaN for *x* less than 0.

FUTURE DIRECTIONS

None.

SEE ALSO

The Base Definitions volume of IEEE Std 1003.1-2001, <*math.h*>

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

ceil, ceilf, ceill - ceiling value function

SYNOPSIS

#include <math.h>

double ceil(double *x***);**
float ceilf(float *x***);**
long double ceill(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the smallest integral value not less than *x*.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, *ceil*(), *ceilf*(), and *ceill*() shall return the smallest integral value not less than *x*, expressed as a type **double**, **float**, or **long double**, respectively.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0 or ±Inf, *x* shall be returned.

If the correct value would cause overflow, a range error shall occur and *ceil*(), *ceilf*(), and *ceill*() shall return the value of the macro HUGE_VAL, HUGE_VALF, and HUGE_VALL, respectively.

ERRORS

These functions shall fail if:

Range Error
The result overflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

The integral value returned by these functions need not be expressible as an **int** or **long**. The return value should be tested before assigning it to an integer type to avoid the undefined results of an integer overflow.

The *ceil*() function can only overflow when the floating-point representation has DBL_MANT_DIG > DBL_MAX_EXP.

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *floor*() , *isnan*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

copysign, copysignf, copysignl - number manipulation function

SYNOPSIS

```
#include <math.h>

double copysign(double x, double y);
float copysignf(float x, float y);
long double copysignl(long double x, long double y);
```

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall produce a value with the magnitude of *x* and the sign of *y*. On implementations that represent a signed zero but do not treat negative zero consistently in arithmetic operations, these functions regard the sign of zero as positive.

RETURN VALUE

Upon successful completion, these functions shall return a value with the magnitude of *x* and the sign of *y*.

ERRORS

No errors are defined.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

signbit() , the Base Definitions volume of IEEE Std 1003.1-2001, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

cos, cosf, cosl - cosine function

SYNOPSIS

#include <math.h>

double cos(double *x***);**
float cosf(float *x***);**
long double cosl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the cosine of their argument *x*, measured in radians.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the cosine of *x*.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0, the value 1.0 shall be returned.

If *x* is ±Inf, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

ERRORS

These functions shall fail if:

Domain Error
The *x* argument is ±Inf.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

Taking the Cosine of a 45-Degree Angle

```
#include <math.h>
...
double radians = 45 * M_PI / 180;
double result;
...
result = cos(radians);
```

APPLICATION USAGE

These functions may lose accuracy when their argument is near an odd multiple of /2 or is far from 0.

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

acos() , *feclearexcept*() , *fetestexcept*() , *isnan*() , *sin*() , *tan*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

cosh, coshf, coshl - hyperbolic cosine functions

SYNOPSIS

#include <math.h>

double cosh(double *x***);**
float coshf(float *x***);**
long double coshl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the hyperbolic cosine of their argument *x*.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the hyperbolic cosine of *x*.

If the correct value would cause overflow, a range error shall occur and *cosh*(), *coshf*(), and *coshl*() shall return the value of the macro HUGE_VAL, HUGE_VALF, and HUGE_VALL, respectively.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0, the value 1.0 shall be returned.

If *x* is ±Inf, +Inf shall be returned.

ERRORS

These functions shall fail if:

Range Error
    The result would cause an overflow.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

For IEEE Std 754-1985 **double**, 710.5 < |*x*| implies that *cosh*(*x*) has overflowed.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

acosh() , *feclearexcept*() , *fetestexcept*() , *isnan*() , *sinh*() , *tanh*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

erf, erff, erfl - error functions

SYNOPSIS

#include <math.h>

double erf(double *x***);**
float erff(float *x***);**
long double erfl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the error function of their argument *x*, defined as:

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the value of the error function.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0, ±0 shall be returned.

If *x* is ±Inf, ±1 shall be returned.

If *x* is subnormal, a range error may occur, and 2 * *x*/ *sqrt*() should be returned.

ERRORS

These functions may fail if:

Range Error
The result underflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

Underflow occurs when |*x*| < DBL_MIN * (*sqrt*()/2).

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

erfc() , *feclearexcept*() , *fetestexcept*() , *isnan*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

exp, expf, expl - exponential function

SYNOPSIS

#include <math.h>

double exp(double *x***);**
float expf(float *x***);**
long double expl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the base- *e* exponential of *x*.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the exponential value of *x*.

If the correct value would cause overflow, a range error shall occur and *exp*(), *expf*(), and *expl*() shall return the value of the macro HUGE_VAL, HUGE_VALF, and HUGE_VALL, respectively.

If the correct value would cause underflow, and is not representable, a range error may occur, and either 0.0 (if supported), or an implementation-defined value shall be returned.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0, 1 shall be returned.

If *x* is -Inf, +0 shall be returned.

If *x* is +Inf, *x* shall be returned.

If the correct value would cause underflow, and is representable, a range error may occur and the correct value shall be returned.

ERRORS

These functions shall fail if:

Range Error
The result overflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

These functions may fail if:

Range Error
The result underflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

Note that for IEEE Std 754-1985 **double**, $709.8 < x$ implies *exp*(*x*) has overflowed. The value $x < -708.4$ implies *exp*(*x*) has underflowed.

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *isnan*() , *log*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, <*math.h*>

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

expm1, expm1f, expm1l - compute exponential functions

SYNOPSIS

#include <math.h>

double expm1(double *x***);**
float expm1f(float *x***);**
long double expm1l(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute *e*-1.0.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions return *e****x*-1.0.

If the correct value would cause overflow, a range error shall occur and *expm1*(), *expm1f*(), and *expm1l*() shall return the value of the macro HUGE_VAL, HUGE_VALF, and HUGE_VALL, respectively.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0, ±0 shall be returned.

If *x* is -Inf, -1 shall be returned.

If *x* is +Inf, *x* shall be returned.

If *x* is subnormal, a range error may occur and *x* should be returned.

ERRORS

These functions shall fail if:

Range Error
    The result overflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

These functions may fail if:

Range Error
    The value of *x* is subnormal.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

The value of *expm1*(*x*) may be more accurate than *exp*(*x*)-1.0 for small values of *x*.

The *expm1*() and *log1p*() functions are useful for financial calculations of ((1+*x*)-1)/*x*, namely:

expm1(*n* *** log1p(***x***))/***x*

when *x* is very small (for example, when calculating small daily interest rates). These functions also simplify writing accurate inverse hyperbolic functions.

For IEEE Std 754-1985 **double**, 709.8 < *x* implies *expm1*(*x*) has overflowed.

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

exp() , *feclearexcept*() , *fetestexcept*() , *ilogb*() , *log1p*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

fabs, fabsf, fabsl - absolute value function

SYNOPSIS

#include <math.h>

double fabs(double *x***);**
float fabsf(float *x***);**
long double fabsl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the absolute value of their argument *x*,| *x*|.

RETURN VALUE

Upon successful completion, these functions shall return the absolute value of *x*.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0, +0 shall be returned.

If *x* is ±Inf, +Inf shall be returned.

ERRORS

No errors are defined.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

isnan() , the Base Definitions volume of IEEE Std 1003.1-2001, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

floor, floorf, floorl - floor function

SYNOPSIS

#include <math.h>

double floor(double *x***);**
float floorf(float *x***);**
long double floorl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the largest integral value not greater than *x*.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the largest integral value not greater than *x*, expressed as a **double**, **float**, or **long double**, as appropriate for the return type of the function.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0 or ±Inf, *x* shall be returned.

If the correct value would cause overflow, a range error shall occur and *floor*(), *floorf*(), and *floorl*() shall return the value of the macro -HUGE_VAL, -HUGE_VALF, and -HUGE_VALL, respectively.

ERRORS

These functions shall fail if:

Range Error
The result would cause an overflow.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

The integral value returned by these functions might not be expressible as an **int** or **long**. The return value should be tested before assigning it to an integer type to avoid the undefined results of an integer overflow.

The *floor*() function can only overflow when the floating-point representation has DBL_MANT_DIG > DBL_MAX_EXP.

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

ceil() , *feclearexcept*() , *fetestexcept*() , *isnan*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

fmod, fmodf, fmodl - floating-point remainder value function

SYNOPSIS

#include <math.h>

double fmod(double *x*, **double** *y*);
float fmodf(float *x*, **float** *y*);
long double fmodl(long double *x*, **long double** *y*);

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall return the floating-point remainder of the division of *x* by *y*.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

These functions shall return the value *x*- *i** *y*, for some integer *i* such that, if *y* is non-zero, the result has the same sign as *x* and magnitude less than the magnitude of *y*.

If the correct value would cause underflow, and is not representable, a range error may occur, and either 0.0 (if supported), or an implementation-defined value shall be returned.

If *x* or *y* is NaN, a NaN shall be returned.

If *y* is zero, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

If *x* is infinite, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

If *x* is ±0 and *y* is not zero, ±0 shall be returned.

If *x* is not infinite and *y* is ±Inf, *x* shall be returned.

If the correct value would cause underflow, and is representable, a range error may occur and the correct value shall be returned.

ERRORS

These functions shall fail if:

Domain Error
    The *x* argument is infinite or *y* is zero.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

These functions may fail if:

Range Error
    The result underflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *isnan*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, <*math.h*>

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

frexp, frexpf, frexpl - extract mantissa and exponent from a double precision number

SYNOPSIS

#include <math.h>

double frexp(double *num***, int ***exp***);**
float frexpf(float *num***, int ***exp***);**
long double frexpl(long double *num***, int ***exp***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall break a floating-point number *num* into a normalized fraction and an integral power of 2. The integer exponent shall be stored in the **int** object pointed to by *exp*.

RETURN VALUE

For finite arguments, these functions shall return the value *x*, such that *x* has a magnitude in the interval [0.5,1) or 0, and *num* equals *x* times 2 raised to the power **exp*.

If *num* is NaN, a NaN shall be returned, and the value of **exp* is unspecified.

If *num* is ±0, ±0 shall be returned, and the value of **exp* shall be 0.

If *num* is ±Inf, *num* shall be returned, and the value of **exp* is unspecified.

ERRORS

No errors are defined.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

isnan() , *ldexp*() , *modf*() , the Base Definitions volume of IEEE Std 1003.1-2001, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

hypot, hypotf, hypotl - Euclidean distance function

SYNOPSIS

#include <math.h>

double hypot(double *x***, double** *y***);**
float hypotf(float *x***, float** *y***);**
long double hypotl(long double *x***, long double** *y***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the value of the square root of *x***2+ *y***2 without undue overflow or underflow.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the length of the hypotenuse of a right-angled triangle with sides of length *x* and *y*.

If the correct value would cause overflow, a range error shall occur and *hypot*(), *hypotf*(), and *hypotl*() shall return the value of the macro HUGE_VAL, HUGE_VALF, and HUGE_VALL, respectively.

If *x* or *y* is ±Inf, +Inf shall be returned (even if one of *x* or *y* is NaN).

If *x* or *y* is NaN, and the other is not ±Inf, a NaN shall be returned.

If both arguments are subnormal and the correct result is subnormal, a range error may occur and the correct result is returned.

ERRORS

These functions shall fail if:

Range Error
The result overflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

These functions may fail if:

Range Error
The result underflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

hypot(*x*,*y*), *hypot*(*y*,*x*), and *hypot*(*x*, -*y*) are equivalent.

hypot(*x*, ±0) is equivalent to *fabs*(*x*).

Underflow only happens when both *x* and *y* are subnormal and the (inexact) result is also subnormal.

These functions take precautions against overflow during intermediate steps of the computation.

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *isnan*() , *sqrt*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

ilogb, ilogbf, ilogbl - return an unbiased exponent

SYNOPSIS

#include <math.h>

int ilogb(double *x***);**
int ilogbf(float *x***);**
int ilogbl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall return the exponent part of their argument *x*. Formally, the return value is the integral part of log_r|x| as a signed integral value, for non-zero *x*, where *r* is the radix of the machine's floating-point arithmetic, which is the value of FLT_RADIX defined in <*float.h*>.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the exponent part of *x* as a signed integer value. They are equivalent to calling the corresponding *logb*() function and casting the returned value to type **int**.

If *x* is 0, a domain error shall occur, and the value FP_ILOGB0 shall be returned.

If *x* is ±Inf, a domain error shall occur, and the value {INT_MAX} shall be returned.

If *x* is a NaN, a domain error shall occur, and the value FP_ILOGBNAN shall be returned.

If the correct value is greater than {INT_MAX}, {INT_MAX} shall be returned and a domain error shall occur.

If the correct value is less than {INT_MIN}, {INT_MIN} shall be returned and a domain error shall occur.

ERRORS

These functions shall fail if:

Domain Error
The *x* argument is zero, NaN, or ±Inf, or the correct value is not representable as an integer.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

The errors come from taking the expected floating-point value and converting it to **int**, which is an invalid operation in IEEE Std 754-1985 (since overflow, infinity, and NaN are not representable in a type **int**), so should be a domain error.

There are no known implementations that overflow. For overflow to happen, {INT_MAX} must be less than LDBL_MAX_EXP**log2*(FLT_RADIX) or {INT_MIN} must be greater than LDBL_MIN_EXP**log2*(FLT_RADIX) if subnormals are not supported, or {INT_MIN} must be greater than (LDBL_MIN_EXP-LDBL_MANT_DIG)**log2*(FLT_RADIX) if subnormals are supported.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *logb*() , *scalb*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, <*float.h*>, <*math.h*>

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

isnan - test for a NaN

SYNOPSIS

#include <math.h>

int isnan(real-floating *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

The *isnan*() macro shall determine whether its argument value is a NaN. First, an argument represented in a format wider than its semantic type is converted to its semantic type. Then determination is based on the type of the argument.

RETURN VALUE

The *isnan*() macro shall return a non-zero value if and only if its argument has a NaN value.

ERRORS

No errors are defined.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

fpclassify() , *isfinite*() , *isinf*() , *isnormal*() , *signbit*() , the Base Definitions volume of IEEE Std 1003.1-2001, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

j0, j1, jn - Bessel functions of the first kind

SYNOPSIS

#include <math.h>

double j0(double *x***);**
double j1(double *x***);**
double jn(int *n***, double** *x***);**

DESCRIPTION

The *j0*(), *j1*(), and *jn*() functions shall compute Bessel functions of *x* of the first kind of orders 0, 1, and *n*, respectively.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the relevant Bessel value of *x* of the first kind.

If the *x* argument is too large in magnitude, or the correct result would cause underflow, 0 shall be returned and a range error may occur.

If *x* is NaN, a NaN shall be returned.

ERRORS

These functions may fail if:

Range Error
: The value of *x* was too large in magnitude, or an underflow occurred.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

No other errors shall occur.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *isnan*() , *y0*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the

referee document. The original Standard can be obtained online at http://www.open-group.org/unix/online.html .

NAME

j0, j1, jn - Bessel functions of the first kind

SYNOPSIS

#include <math.h>

double j0(double *x***);**
double j1(double *x***);**
double jn(int *n***, double** *x***);**

DESCRIPTION

The *j0*(), *j1*(), and *jn*() functions shall compute Bessel functions of *x* of the first kind of orders 0, 1, and *n*, respectively.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the relevant Bessel value of *x* of the first kind.

If the *x* argument is too large in magnitude, or the correct result would cause underflow, 0 shall be returned and a range error may occur.

If *x* is NaN, a NaN shall be returned.

ERRORS

These functions may fail if:

Range Error
: The value of *x* was too large in magnitude, or an underflow occurred.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

No other errors shall occur.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *isnan*() , *y0*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, <*math.h*>

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the

referee document. The original Standard can be obtained online at http://www.open-group.org/unix/online.html .

NAME

j0, j1, jn - Bessel functions of the first kind

SYNOPSIS

#include <math.h>

double j0(double *x***);**
double j1(double *x***);**
double jn(int *n***, double** *x***);**

DESCRIPTION

The *j0*(), *j1*(), and *jn*() functions shall compute Bessel functions of *x* of the first kind of orders 0, 1, and *n*, respectively.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the relevant Bessel value of *x* of the first kind.

If the *x* argument is too large in magnitude, or the correct result would cause underflow, 0 shall be returned and a range error may occur.

If *x* is NaN, a NaN shall be returned.

ERRORS

These functions may fail if:

Range Error
    The value of *x* was too large in magnitude, or an underflow occurred.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

No other errors shall occur.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *isnan*() , *y0*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, <*math.h*>

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the

referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

ldexp, ldexpf, ldexpl - load exponent of a floating-point number

SYNOPSIS

#include <math.h>

double ldexp(double *x***, int** *exp***);**
float ldexpf(float *x***, int** *exp***);**
long double ldexpl(long double *x***, int** *exp***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the quantity *x* * 2.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return *x* multiplied by 2, raised to the power *exp*.

If these functions would cause overflow, a range error shall occur and *ldexp*(), *ldexpf*(), and *ldexpl*() shall return ±HUGE_VAL, ±HUGE_VALF, and ±HUGE_VALL (according to the sign of *x*), respectively.

If the correct value would cause underflow, and is not representable, a range error may occur, and either 0.0 (if supported), or an implementation-defined value shall be returned.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0 or ±Inf, *x* shall be returned.

If *exp* is 0, *x* shall be returned.

If the correct value would cause underflow, and is representable, a range error may occur and the correct value shall be returned.

ERRORS

These functions shall fail if:

Range Error
The result overflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

These functions may fail if:

Range Error
The result underflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *frexp*() , *isnan*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

lgamma, lgammaf, lgammal - log gamma function

SYNOPSIS

#include <math.h>

double lgamma(double *x***);**
float lgammaf(float *x***);**
long double lgammal(long double *x***);**

extern int signgam;

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute

The argument *x* need not be a non-positive integer (is defined over the reals, except the non-positive integers).

The sign of is returned in the external integer *signgam*.

These functions need not be reentrant. A function that is not required to be reentrant is not required to be thread-safe.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the logarithmic gamma of *x*.

If *x* is a non-positive integer, a pole error shall occur and *lgamma*(), *lgammaf*(), and *lgammal*() shall return +HUGE_VAL, +HUGE_VALF, and +HUGE_VALL, respectively.

If the correct value would cause overflow, a range error shall occur and *lgamma*(), *lgammaf*(), and *lgammal*() shall return ±HUGE_VAL, ±HUGE_VALF, and ±HUGE_VALL (having the same sign as the correct value), respectively.

If *x* is NaN, a NaN shall be returned.

If *x* is 1 or 2, +0 shall be returned.

If *x* is ±Inf, +Inf shall be returned.

ERRORS

These functions shall fail if:

Pole Error

The *x* argument is a negative integer or zero.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the divide-by-zero floating-point exception shall be raised.

Range Error

The result overflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

exp() , *feclearexcept*() , *fetestexcept*() , *isnan*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

log, logf, logl - natural logarithm function

SYNOPSIS

#include <math.h>

double log(double *x***);**
float logf(float *x***);**
long double logl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the natural logarithm of their argument *x*, log_*e*(*x*).

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the natural logarithm of *x*.

If *x* is ±0, a pole error shall occur and *log*(), *logf*(), and *logl*() shall return -HUGE_VAL, -HUGE_VALF, and -HUGE_VALL, respectively.

For finite values of *x* that are less than 0, or if *x* is -Inf, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

If *x* is NaN, a NaN shall be returned.

If *x* is 1, +0 shall be returned.

If *x* is +Inf, *x* shall be returned.

ERRORS

These functions shall fail if:

Domain Error
: The finite value of *x* is negative, or *x* is -Inf.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

Pole Error
: The value of *x* is zero.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the divide-by-zero floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

exp() , *feclearexcept*() , *fetestexcept*() , *isnan*() , *log10*() , *log1p*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, <*math.h*>

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

log10, log10f, log10l - base 10 logarithm function

SYNOPSIS

#include <math.h>

double log10(double *x***);**
float log10f(float *x***);**
long double log10l(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the base 10 logarithm of their argument *x*, log_10(*x*).

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the base 10 logarithm of *x*.

If *x* is ±0, a pole error shall occur and *log10*(), *log10f*(), and *log10l*() shall return -HUGE_VAL, -HUGE_VALF, and -HUGE_VALL, respectively.

For finite values of *x* that are less than 0, or if *x* is -Inf, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

If *x* is NaN, a NaN shall be returned.

If *x* is 1, +0 shall be returned.

If *x* is +Inf, +Inf shall be returned.

ERRORS

These functions shall fail if:

Domain Error
The finite value of *x* is negative, or *x* is -Inf.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

Pole Error
The value of *x* is zero.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the divide-by-zero floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *isnan*() , *log*() , *pow*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, <*math.h*>

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

log1p, log1pf, log1pl - compute a natural logarithm

SYNOPSIS

#include <math.h>

double log1p(double *x***);**
float log1pf(float *x***);**
long double log1pl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute log_e(1.0 + *x*).

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the natural logarithm of 1.0 + *x*.

If *x* is -1, a pole error shall occur and *log1p*(), *log1pf*(), and *log1pl*() shall return -HUGE_VAL, -HUGE_VALF, and -HUGE_VALL, respectively.

For finite values of *x* that are less than -1, or if *x* is -Inf, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0, or +Inf, *x* shall be returned.

If *x* is subnormal, a range error may occur and *x* should be returned.

ERRORS

These functions shall fail if:

Domain Error
The finite value of *x* is less than -1, or *x* is -Inf.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

Pole Error
The value of *x* is -1.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the divide-by-zero floating-point exception shall be raised.

These functions may fail if:

Range Error
The value of *x* is subnormal.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *log*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, <*math.h*>

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

logb, logbf, logbl - radix-independent exponent

SYNOPSIS

#include <math.h>

double logb(double *x***);**
float logbf(float *x***);**
long double logbl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the exponent of *x*, which is the integral part of log_*r* |*x*|, as a signed floating-point value, for non-zero *x*, where *r* is the radix of the machine's floating-point arithmetic, which is the value of FLT_RADIX defined in the <*float.h*> header.

If *x* is subnormal it is treated as though it were normalized; thus for finite positive *x*:

1 <= *x* *** FLT_RADIX**-logb(x) < FLT_RADIX**

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the exponent of *x*.

If *x* is ±0, a pole error shall occur and *logb*(), *logbf*(), and *logbl*() shall return -HUGE_VAL, -HUGE_VALF, and -HUGE_VALL, respectively.

If *x* is NaN, a NaN shall be returned.

If *x* is ±Inf, +Inf shall be returned.

ERRORS

These functions shall fail if:

Pole Error
    The value of *x* is ±0.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the divide-by-zero floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *ilogb*() , *scalb*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<float.h>*, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

modf, modff, modfl - decompose a floating-point number

SYNOPSIS

#include <math.h>

double modf(double *x*, **double ****iptr***);**
float modff(float *value*, **float ****iptr***);**
long double modfl(long double *value*, **long double ****iptr***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall break the argument *x* into integral and fractional parts, each of which has the same sign as the argument. It stores the integral part as a **double** (for the *modf*() function), a **float** (for the *modff*() function), or a **long double** (for the *modfl*() function), in the object pointed to by *iptr*.

RETURN VALUE

Upon successful completion, these functions shall return the signed fractional part of *x*.

If *x* is NaN, a NaN shall be returned, and **iptr* shall be set to a NaN.

If *x* is ±Inf, ±0 shall be returned, and **iptr* shall be set to ±Inf.

ERRORS

No errors are defined.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

The *modf*() function computes the function result and **iptr* such that:

```
a = modf(x, iptr) ;
x == a+*iptr ;
```

allowing for the usual floating-point inaccuracies.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

frexp() , *isnan*() , *ldexp*() , the Base Definitions volume of IEEE Std 1003.1-2001, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

nextafter, nextafterf, nextafterl, nexttoward, nexttowardf, nexttowardl - next representable floating-point number

SYNOPSIS

#include <math.h>

double nextafter(double *x*, **double** *y***);**
float nextafterf(float *x*, **float** *y***);**
long double nextafterl(long double *x*, **long double** *y***);**
double nexttoward(double *x*, **long double** *y***);**
float nexttowardf(float *x*, **long double** *y***);**
long double nexttowardl(long double *x*, **long double** *y***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

The *nextafter*(), *nextafterf*(), and *nextafterl*() functions shall compute the next representable floating-point value following *x* in the direction of *y*. Thus, if *y* is less than *x*, *nextafter*() shall return the largest representable floating-point number less than *x*. The *nextafter*(), *nextafterf*(), and *nextafterl*() functions shall return *y* if *x* equals *y*.

The *nexttoward*(), *nexttowardf*(), and *nexttowardl*() functions shall be equivalent to the corresponding *nextafter*() functions, except that the second parameter shall have type **long double** and the functions shall return *y* converted to the type of the function if *x* equals *y*.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the next representable floating-point value following *x* in the direction of *y*.

If *x*== *y*, *y* (of the type *x*) shall be returned.

If *x* is finite and the correct function value would overflow, a range error shall occur and ±HUGE_VAL, ±HUGE_VALF, and ±HUGE_VALL (with the same sign as *x*) shall be returned as appropriate for the return type of the function.

If *x* or *y* is NaN, a NaN shall be returned.

If *x*!= *y* and the correct function value is subnormal, zero, or underflows, a range error shall occur, and either the correct function value (if representable) or 0.0 shall be returned.

ERRORS

These functions shall fail if:

Range Error
The correct value overflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

Range Error
The correct value is subnormal or underflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, <*math.h*>

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

pow, powf, powl - power function

SYNOPSIS

#include <math.h>

double pow(double *x***, double** *y***);**
float powf(float *x***, float** *y***);**
long double powl(long double *x***, long double** *y***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the value of x raised to the power y, x. If x is negative, the application shall ensure that y is an integer value.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the value of x raised to the power y.

For finite values of $x < 0$, and finite non-integer values of y, a domain error shall occur and either a NaN (if representable), or an implementation-defined value shall be returned.

If the correct value would cause overflow, a range error shall occur and *pow*(), *powf*(), and *powl*() shall return ±HUGE_VAL, ±HUGE_VALF, and ±HUGE_VALL, respectively, with the same sign as the correct value of the function.

If the correct value would cause underflow, and is not representable, a range error may occur, and either 0.0 (if supported), or an implementation-defined value shall be returned.

If x or y is a NaN, a NaN shall be returned (unless specified elsewhere in this description).

For any value of y (including NaN), if x is +1, 1.0 shall be returned.

For any value of x (including NaN), if y is ±0, 1.0 shall be returned.

For any odd integer value of $y > 0$, if x is ±0, ±0 shall be returned.

For $y > 0$ and not an odd integer, if x is ±0, +0 shall be returned.

If x is -1, and y is ±Inf, 1.0 shall be returned.

For $|x| < 1$, if y is -Inf, +Inf shall be returned.

For $|x| > 1$, if y is -Inf, +0 shall be returned.

For $|x| < 1$, if y is +Inf, +0 shall be returned.

For $|x| > 1$, if y is +Inf, +Inf shall be returned.

For y an odd integer < 0, if x is -Inf, -0 shall be returned.

For $y < 0$ and not an odd integer, if x is -Inf, +0 shall be returned.

For y an odd integer > 0, if x is -Inf, -Inf shall be returned.

For $y > 0$ and not an odd integer, if x is -Inf, +Inf shall be returned.

For $y < 0$, if x is +Inf, +0 shall be returned.

For $y > 0$, if x is +Inf, +Inf shall be returned.

For y an odd integer < 0, if x is ±0, a pole error shall occur and ±HUGE_VAL, ±HUGE_VALF, and ±HUGE_VALL shall be returned for *pow*(), *powf*(), and *powl*(), respectively.

For $y < 0$ and not an odd integer, if x is ±0, a pole error shall occur and HUGE_VAL, HUGE_VALF, and HUGE_VALL shall be returned for *pow*(), *powf*(), and *powl*(), respectively.

If the correct value would cause underflow, and is representable, a range error may occur and the correct value shall be returned.

ERRORS

These functions shall fail if:

Domain Error
: The value of *x* is negative and *y* is a finite non-integer.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

Pole Error
: The value of *x* is zero and *y* is negative.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the divide-by-zero floating-point exception shall be raised.

Range Error
: The result overflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

These functions may fail if:

Range Error
: The result underflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

exp() , *feclearexcept*() , *fetestexcept*() , *isnan*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

remainder, remainderf, remainderl - remainder function

SYNOPSIS

#include <math.h>

double remainder(double *x*, **double** *y***);**
float remainderf(float *x*, **float** *y***);**
long double remainderl(long double *x*, **long double** *y***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall return the floating-point remainder *r*= *x*- *ny* when *y* is non-zero. The value *n* is the integral value nearest the exact value *x*/ *y*. When |*n*-*x*/*y*|=0.5, the value *n* is chosen to be even.

The behavior of *remainder*() shall be independent of the rounding mode.

RETURN VALUE

Upon successful completion, these functions shall return the floating-point remainder *r*= *x*- *ny* when *y* is non-zero.

If *x* or *y* is NaN, a NaN shall be returned.

If *x* is infinite or *y* is 0 and the other is non-NaN, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

ERRORS

These functions shall fail if:

Domain Error
The *x* argument is ±Inf, or the *y* argument is ±0 and the other argument is non-NaN.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

abs() , *div*() , *feclearexcept*() , *fetestexcept*() , *ldiv*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the

referee document. The original Standard can be obtained online at http://www.open-group.org/unix/online.html .

NAME

rint, rintf, rintl - round-to-nearest integral value

SYNOPSIS

#include <math.h>

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

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall return the integral value (represented as a **double**) nearest *x* in the direction of the current rounding mode. The current rounding mode is implementation-defined.

If the current rounding mode rounds toward negative infinity, then *rint*() shall be equivalent to *floor*() . If the current rounding mode rounds toward positive infinity, then *rint*() shall be equivalent to *ceil*() .

These functions differ from the *nearbyint*(), *nearbyintf*(), and *nearbyintl*() functions only in that they may raise the inexact floating-point exception if the result differs in value from the argument.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the integer (represented as a double precision number) nearest *x* in the direction of the current rounding mode.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0 or ±Inf, *x* shall be returned.

If the correct value would cause overflow, a range error shall occur and *rint*(), *rintf*(), and *rintl*() shall return the value of the macro ±HUGE_VAL, ±HUGE_VALF, and ±HUGE_VALL (with the same sign as *x*), respectively.

ERRORS

These functions shall fail if:

Range Error
    The result would cause an overflow.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

abs() , *ceil*() , *feclearexcept*() , *fetestexcept*() , *floor*() , *isnan*() , *nearbyint*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, <*math.h*>

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

scalb - load exponent of a radix-independent floating-point number

SYNOPSIS

#include <math.h>

double scalb(double *x***, double** *n***);**

DESCRIPTION

The *scalb*() function shall compute *x***r*, where *r* is the radix of the machine's floating-point arithmetic. When *r* is 2, *scalb*() shall be equivalent to *ldexp*() . The value of *r* is FLT_RADIX which is defined in *<float.h>*.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, the *scalb*() function shall return *x***r*.

If *x* or *n* is NaN, a NaN shall be returned.

If *n* is zero, *x* shall be returned.

If *x* is ±Inf and *n* is not -Inf, *x* shall be returned.

If *x* is ±0 and *n* is not +Inf, *x* shall be returned.

If *x* is ±0 and *n* is +Inf, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

If *x* is ±Inf and *n* is -Inf, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

If the result would cause an overflow, a range error shall occur and ±HUGE_VAL (according to the sign of *x*) shall be returned.

If the correct value would cause underflow, and is representable, a range error may occur and the correct value shall be returned.

If the correct value would cause underflow, and is not representable, a range error may occur, and 0.0 shall be returned.

ERRORS

The *scalb*() function shall fail if:

Domain Error

If *x* is zero and *n* is +Inf, or *x* is Inf and *n* is -Inf.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

Range Error

The result would overflow.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

The *scalb*() function may fail if:

Range Error

The result underflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to

[ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

Applications should use either *scalbln*(), *scalblnf*(), or *scalblnl*() in preference to this function.

IEEE Std 1003.1-2001 only defines the behavior for the *scalb*() function when the *n* argument is an integer, a NaN, or Inf. The behavior of other values for the *n* argument is unspecified.

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *ilogb*() , *ldexp*() , *logb*() , *scalbln*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<float.h>*, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

scalbln, scalblnf, scalblnl, scalbn, scalbnf, scalbnl - compute exponent using FLT_RADIX

SYNOPSIS

#include <math.h>

double scalbln(double *x*, **long** *n***);**
float scalblnf(float *x*, **long** *n***);**
long double scalblnl(long double *x*, **long** *n***);**
double scalbn(double *x*, **int** *n***);**
float scalbnf(float *x*, **int** *n***);**
long double scalbnl(long double *x*, **int** *n***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute *x* * FLT_RADIX***n* efficiently, not normally by computing FLT_RADIX***n* explicitly.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return *x* * FLT_RADIX***n*.

If the result would cause overflow, a range error shall occur and these functions shall return ±HUGE_VAL, ±HUGE_VALF, and ±HUGE_VALL (according to the sign of *x*) as appropriate for the return type of the function.

If the correct value would cause underflow, and is not representable, a range error may occur, and either 0.0 (if supported), or an implementation-defined value shall be returned.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0 or ±Inf, *x* shall be returned.

If *n* is 0, *x* shall be returned.

If the correct value would cause underflow, and is representable, a range error may occur and the correct value shall be returned.

ERRORS

These functions shall fail if:

Range Error
The result overflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

These functions may fail if:

Range Error
The result underflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

These functions are named so as to avoid conflicting with the historical definition of the *scalb*() function from the Single UNIX Specification. The difference is that the *scalb*() function has a second argument of **double** instead of **int**. The *scalb*() function is not part of the ISO C standard. The three functions whose second type is **long** are provided because the factor required to scale from the smallest positive floating-point value to the largest finite one, on many implementations, is too large to represent in the minimum-width **int** format.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *scalb*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, <*math.h*>

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

lgamma, lgammaf, lgammal - log gamma function

SYNOPSIS

#include <math.h>

double lgamma(double *x***);**
float lgammaf(float *x***);**
long double lgammal(long double *x***);**

extern int signgam;

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute

The argument *x* need not be a non-positive integer (is defined over the reals, except the non-positive integers).

The sign of is returned in the external integer *signgam*.

These functions need not be reentrant. A function that is not required to be reentrant is not required to be thread-safe.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the logarithmic gamma of *x*.

If *x* is a non-positive integer, a pole error shall occur and *lgamma*(), *lgammaf*(), and *lgammal*() shall return +HUGE_VAL, +HUGE_VALF, and +HUGE_VALL, respectively.

If the correct value would cause overflow, a range error shall occur and *lgamma*(), *lgammaf*(), and *lgammal*() shall return ±HUGE_VAL, ±HUGE_VALF, and ±HUGE_VALL (having the same sign as the correct value), respectively.

If *x* is NaN, a NaN shall be returned.

If *x* is 1 or 2, +0 shall be returned.

If *x* is ±Inf, +Inf shall be returned.

ERRORS

These functions shall fail if:

Pole Error
    The *x* argument is a negative integer or zero.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the divide-by-zero floating-point exception shall be raised.

Range Error
    The result overflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

exp() , *feclearexcept*() , *fetestexcept*() , *isnan*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

sin, sinf, sinl - sine function

SYNOPSIS

```
#include <math.h>

double sin(double x);
float sinf(float x);
long double sinl(long double x);
```

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the sine of their argument *x*, measured in radians.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the sine of *x*.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0, *x* shall be returned.

If *x* is subnormal, a range error may occur and *x* should be returned.

If *x* is ±Inf, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

ERRORS

These functions shall fail if:

Domain Error
: The *x* argument is ±Inf.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

These functions may fail if:

Range Error
: The value of *x* is subnormal

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

Taking the Sine of a 45-Degree Angle

```
#include <math.h>
...
double radians = 45.0 * M_PI / 180;
double result;
...
result = sin(radians);
```

APPLICATION USAGE

These functions may lose accuracy when their argument is near a multiple of or is far from 0.0.

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

asin() , *feclearexcept*() , *fetestexcept*() , *isnan*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

sinh, sinhf, sinhl - hyperbolic sine functions

SYNOPSIS

#include <math.h>

double sinh(double *x***);**
float sinhf(float *x***);**
long double sinhl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the hyperbolic sine of their argument *x*.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the hyperbolic sine of *x*.

If the result would cause an overflow, a range error shall occur and ±HUGE_VAL, ±HUGE_VALF, and ±HUGE_VALL (with the same sign as *x*) shall be returned as appropriate for the type of the function.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0 or ±Inf, *x* shall be returned.

If *x* is subnormal, a range error may occur and *x* should be returned.

ERRORS

These functions shall fail if:

Range Error
The result would cause an overflow.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

These functions may fail if:

Range Error
The value *x* is subnormal.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

asinh() , *cosh*() , *feclearexcept*() , *fetestexcept*() , *isnan*() , *tanh*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, <*math.h*>

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

sqrt, sqrtf, sqrtl - square root function

SYNOPSIS

#include <math.h>

double sqrt(double *x***);**
float sqrtf(float *x***);**
long double sqrtl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the square root of their argument *x*,

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the square root of *x*.

For finite values of *x* < -0, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0 or +Inf, *x* shall be returned.

If *x* is -Inf, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

ERRORS

These functions shall fail if:

Domain Error
The finite value of *x* is < -0, or *x* is -Inf.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

Taking the Square Root of 9.0

```
#include <math.h>
...
double x = 9.0;
double result;
...
result = sqrt(x);
```

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept() , *fetestexcept*() , *isnan*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*, *<stdio.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

tan, tanf, tanl - tangent function

SYNOPSIS

#include <math.h>

double tan(double *x***);**
float tanf(float *x***);**
long double tanl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the tangent of their argument *x*, measured in radians.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the tangent of *x*.

If the correct value would cause underflow, and is not representable, a range error may occur, and either 0.0 (if supported), or an implementation-defined value shall be returned.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0, *x* shall be returned.

If *x* is subnormal, a range error may occur and *x* should be returned.

If *x* is ±Inf, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

If the correct value would cause underflow, and is representable, a range error may occur and the correct value shall be returned.

If the correct value would cause overflow, a range error shall occur and *tan*(), *tanf*(), and *tanl*() shall return ±HUGE_VAL, ±HUGE_VALF, and ±HUGE_VALL, respectively, with the same sign as the correct value of the function.

ERRORS

These functions shall fail if:

Domain Error
The value of *x* is ±Inf.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

Range Error
The result overflows

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.

These functions may fail if:

Range Error
The result underflows, or the value of *x* is subnormal.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to

[ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

Taking the Tangent of a 45-Degree Angle

```
#include <math.h>
...
double radians = 45.0 * M_PI / 180;
double result;
...
result = tan (radians);
```

APPLICATION USAGE

There are no known floating-point representations such that for a normal argument, *tan*(*x*) is either overflow or underflow.

These functions may lose accuracy when their argument is near a multiple of /2 or is far from 0.0.

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

atan() , *feclearexcept*() , *fetestexcept*() , *isnan*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

tanh, tanhf, tanhl - hyperbolic tangent functions

SYNOPSIS

#include <math.h>

double tanh(double *x***);**
float tanhf(float *x***);**
long double tanhl(long double *x***);**

DESCRIPTION

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 IEEE Std 1003.1-2001 defers to the ISO C standard.

These functions shall compute the hyperbolic tangent of their argument *x*.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE_ALL_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the hyperbolic tangent of *x*.

If *x* is NaN, a NaN shall be returned.

If *x* is ±0, *x* shall be returned.

If *x* is ±Inf, ±1 shall be returned.

If *x* is subnormal, a range error may occur and *x* should be returned.

ERRORS

These functions may fail if:

Range Error
The value of *x* is subnormal.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_errhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

atanh() , *feclearexcept*() , *fetestexcept*() , *isnan*() , *tan*() , the Base Definitions volume of IEEE Std 1003.1-2001, Section 4.18, Treatment of Error Conditions for Mathematical Functions, *<math.h>*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and

Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .