NAME

frexp, ldexp, logb, modf, modff, nextafter, scalb – manipulate parts of floating-point numbers

SYNOPSIS

cc [*flag* ...] *file* ... –**lm** [*library* ...]

#include <math.h>

double frexp(double *value*, **int** **eptr*);

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

double logb(double *value*);

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

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

double nextafter(double *value1*, **double** *value2*);

double scalb(double *value*, **double** *exp*);

MT-LEVEL

MT-Safe

DESCRIPTION

Every non-zero number can be written uniquely as $x * 2^n$, where the ''mantissa'' (fraction) x is in the range $0.5 \leq | x | < 1.0$, and the ''exponent'' n is an integer. **frexp( )** returns the mantissa of a double *value*, and stores the exponent indirectly in the location pointed to by *eptr*. If *value* is zero, both results returned by **frexp( )** are zero.

ldexp( ) and **scalb( )** return the quantity $value * 2^{exp}$. The only difference between the two is that **scalb( )** of a signaling NaN will result in the invalid operation exception being raised.

logb( ) returns the unbiased exponent of its floating-point argument as a double-precision floating-point value.

modf( ) and **modff( )** (single-precision version) return the signed fractional part of *value* and store the integral part indirectly in the location pointed to by *iptr*.

nextafter( ) returns the next representable double-precision floating-point value following *value1* in the direction of *value2*. Thus, if *value2* is less than *value1*, **nextafter( )** returns the largest representable floating-point number less than *value1*.

RETURN VALUES

If **ldexp( )** would cause overflow, **±HUGE** (defined in **<math.h>**) is returned (according to the sign of *value*), and **errno** is set to **ERANGE**. If **ldexp( )** would cause underflow, zero is returned and **errno** is set to **ERANGE**. If the input *value* to **ldexp( )** is NaN or infinity, that input is returned and **errno** is set to **EDOM**. The same error conditions apply to **scalb( )** except that a signaling NaN as input will result in the raising of the invalid operation exception.

logb( ) of NaN returns that NaN, **logb( )** of infinity returns positive infinity, and **logb( )** of zero returns negative infinity and results in the raising of the divide by zero exception. In each of these conditions **errno** is set to **EDOM**.

If input *value1* to **nextafter( )** is positive or negative infinity, that input is returned and **errno** is set to **EDOM**. The overflow and inexact exceptions are signalled when input *value1* is finite, but **nextafter(***value1***,** *value2***)** is not. The underflow and inexact exceptions are signalled when **nextafter(***value1***,** *value2***)** lies strictly between $+2^{-1022}$ and -2^{-1022}. In both cases **errno** is set to **ERANGE**.