NAME

floating_to_decimal, single_to_decimal, double_to_decimal, extended_to_decimal, quadruple_to_decimal – convert floating-point value to decimal record

SYNOPSIS

#include <floatingpoint.h>

void single_to_decimal(single **px*, **decimal_mode** **pm*, **decimal_record** **pd*,
    fp_exception_field_type **ps***);**

void double_to_decimal(double **px*, **decimal_mode** **pm*, **decimal_record** **pd*,
    fp_exception_field_type **ps***);**

void extended_to_decimal(extended **px*, **decimal_mode** **pm*, **decimal_record** **pd*,
    fp_exception_field_type **ps***);**

void quadruple_to_decimal(quadruple **px*, **decimal_mode** **pm*, **decimal_record** **pd*,
    fp_exception_field_type **ps***);**

MT-LEVEL

MT-Safe

DESCRIPTION

The **floating_to_decimal( )** functions convert the floating-point value at **px* into a decimal record at **pd*, observing the modes specified in **pm* and setting exceptions in **ps*. If there are no IEEE exceptions, **ps* will be zero.

If **px* is zero, infinity, or NaN, then only *pd->sign* and *pd->fpclass* are set. Otherwise *pd->exponent* and *pd->ds* are also set so that

(pd->sign)*(pd->ds)*10(pd->exponent)**

is a correctly rounded approximation to **px. pd->ds* has at least one and no more than **DECIMAL_STRING_LENGTH–1** significant digits because one character is used to terminate the string with a NULL.

pd->ds is correctly rounded according to the IEEE rounding modes in *pm->rd*. **ps* has *fp_inexact* set if the result was inexact, and has *fp_overflow* set if the string result does not fit in *pd->ds* because of the limitation **DECIMAL_STRING_LENGTH**.

If *pm->df == floating_form*, then *pd->ds* always contains *pm->ndigits* significant digits. Thus if **px* == 12.34 and *pm->ndigits* == 8, then *pd->ds* will contain 12340000 and *pd->exponent* will contain –6.

If *pm->df == fixed_form* and *pm->ndigits* >= 0, then *pd->ds* always contains *pm->ndigits* after the point and as many digits as necessary before the point. Since the latter is not known in advance, the total number of digits required is returned in *pd->ndigits*; if that number >= **DECIMAL_STRING_LENGTH**, then *ds* is undefined. *pd->exponent* always gets –*pm->ndigits*. Thus if **px* == 12.34 and *pm->ndigits* == 1, then *pd->ds* gets 123, *pd->exponent* gets –1, and *pd->ndigits* gets 3.

If *pm->df == fixed_form* and *pm->ndigits* < 0, then *pd->ds* always contains –*pm->ndigits* trailing zeros; in other words, rounding occurs –*pm->ndigits* to the left of the decimal point, but the digits rounded away are retained as zeros. The total number of digits required is in *pd->ndigits*. *pd->exponent* always gets 0. Thus if **px* == 12.34 and *pm*->*ndigits* == –1, then *pd*->*ds* gets 10, *pd*->*exponent* gets 0, and *pd*->*ndigits* gets 2.

pd->*more* is not used.

econvert(3), **fconvert**(3), **gconvert**(3), **printf**(3S), and **sprintf**(3S) all use **double_to_decimal( )**.

SEE ALSO

econvert(3), **fconvert**(3), **gconvert**(3), **printf**(3S), **sprintf**(3S)