NAME

string_to_decimal, file_to_decimal, func_to_decimal – parse characters into decimal record

SYNOPSIS

#include <floatingpoint.h>

void string_to_decimal(char ***pc*, **int** *nmax*, **int** *fortran_conventions*, **decimal_record** **pd*, **enum decimal_string_form** **pform*, **char** ***pechar*);

void func_to_decimal(char ***pc*, **int** *nmax*, **int** *fortran_conventions*, **decimal_record** **pd*, **enum decimal_string_form** **pform*, **char** ***pechar*, **int (****pget***)(void), int** **pnread*, **int (****punget***)(int** *c***));**

#include <stdio.h>

void file_to_decimal(char ***pc*, **int** *nmax*, **int** *fortran_conventions*, **decimal_record** **pd*, **enum decimal_string_form** **pform*, **char** ***pechar*, **FILE** **pf*, **int** **pnread*);

MT-LEVEL

MT-Safe

DESCRIPTION

The **char_to_decimal** functions parse a numeric token from at most *nmax* characters in a string ***pc* or file **pf* or function *(*pget)***( )** into a decimal record **pd*, classifying the form of the string in **pform* and **pechar*. The accepted syntax is intended to be sufficiently flexible to accomodate many languages:

whitespace value

or

whitespace sign value

where *whitespace* is any number of characters defined by *isspace* in **<ctype.h>**, *sign* is either of [+–], and *value* can be *number*, *nan*, or *inf*. *inf* can be INF (*inf_form*) or INFINITY (*infinity_form*) without regard to case. *nan* can be NAN (*nan_form*) or NAN(*nstring*) (*nanstring_form*) without regard to case; *nstring* is any string of characters not containing **')'** or NULL; *nstring* is copied to *pd*–**>ds** and, currently, not used subsequently. *number* consists of

significand

or

significand efield

where *significand* must contain one or more digits and may contain one point; possible forms are

digits	*(int_form)*
digits.	*(intdot_form)*
.digits	*(dotfrac_form)*
digits.digits	*(intdotfrac_form)*

efield consists of

echar digits

or

echar sign digits

where *echar* is one of [Ee], and *digits* contains one or more digits.

When *fortran_conventions* is nonzero, additional input forms are accepted according to various Fortran conventions:

0 no Fortran conventions
1 Fortran list-directed input conventions
2 Fortran formatted input conventions, ignore blanks **(BN)**
3 Fortran formatted input conventions, blanks are zeros **(BZ)**

When *fortran_conventions* is nonzero, *echar* may also be one of [DdQq], and *efield* may also have the form

sign digits.

When *fortran_conventions* >= 2, blanks may appear in the *digits* strings for the integer, fraction, and exponent fields and may appear between *echar* and the exponent sign and after the infinity and NaN forms. If *fortran_conventions* == 2, the blanks are ignored. When *fortran_conventions* == 3, the blanks that appear in *digits* strings are interpreted as zeros, and other blanks are ignored.

When *fortran_conventions* is zero, the current locale's decimal point character is used as the decimal point; when *fortran_conventions* is nonzero, the period is used as the decimal point.

The form of the accepted decimal string is placed in **peform*. If an *efield* is recognized, **pechar* is set to point to the *echar*.

On input, **pc* points to the beginning of a character string buffer of length **>=** *nmax*. On output, **pc* points to a character in that buffer, one past the last accepted character. **string_to_decimal( )** gets its characters from the buffer; **file_to_decimal( )** gets its characters from **pf* and records them in the buffer, and places a null after the last character read. **func_to_decimal( )** gets its characters from an int function *(*pget)***( )**.

The scan continues until no more characters could possibly fit the acceptable syntax or until *nmax* characters have been scanned. If the *nmax* limit is not reached then at least one extra character will usually be scanned that is not part of the accepted syntax. **file_to_decimal( )** and **func_to_decimal( )** set **pnread* to the number of characters read from the file; if greater than *nmax*, some characters were lost. If no characters were lost, **file_to_decimal( )** and **func_to_decimal( )** attempt to push back, with **ungetc**(3S) or *(*punget)***( )**, as many as possible of the excess characters read, adjusting **pnread* accordingly. If all unget calls are successful, then ***pc* will be NULL. No push back will be attempted if *(*punget)***( )** is NULL.

Typical declarations for **pget***( )** and **punget***( )** are:

```
int xget(void)
{ . . . }
int (*pget)(void) = xget;
int xungetc(int c)
{ . . . }
int (*punget)(int) = xunget;
```

If no valid number was detected, *pd*−**>fpclass** is set to **fp_signaling**, **pc* is unchanged, and **pform* is set to **invalid_form**.

atof(3C) and **strtod**(3C) use **string_to_decimal( )**. **scanf**(3S) uses **file_to_decimal( )**.

SEE ALSO **ctype**(3C), **localeconv**(3C), **scanf**(3S), **setlocale**(3C), **strtod**(3C), **ungetc**(3S)