

NAME

strtod, strtodf, strtold – convert ASCII string to floating-point number

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <stdlib.h>
```

```
double strtod(const char *restrict nptr, char **restrict endptr);
```

```
float strtodf(const char *restrict nptr, char **restrict endptr);
```

```
long double strtold(const char *restrict nptr, char **restrict endptr);
```

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

```
strtod(), strtold():
```

```
_ISOC99_SOURCE || _POSIX_C_SOURCE >= 200112L
```

DESCRIPTION

The **strtod()**, **strtodf()**, and **strtold()** functions convert the initial portion of the string pointed to by *nptr* to *double*, *float*, and *long double* representation, respectively.

The expected form of the (initial portion of the) string is optional leading white space as recognized by **isspace(3)**, an optional plus ('+') or minus sign ('-') and then either (i) a decimal number, or (ii) a hexadecimal number, or (iii) an infinity, or (iv) a NAN (not-a-number).

A *decimal number* consists of a nonempty sequence of decimal digits possibly containing a radix character (decimal point, locale-dependent, usually '.'), optionally followed by a decimal exponent. A decimal exponent consists of an 'E' or 'e', followed by an optional plus or minus sign, followed by a nonempty sequence of decimal digits, and indicates multiplication by a power of 10.

A *hexadecimal number* consists of a "0x" or "0X" followed by a nonempty sequence of hexadecimal digits possibly containing a radix character, optionally followed by a binary exponent. A binary exponent consists of a 'P' or 'p', followed by an optional plus or minus sign, followed by a nonempty sequence of decimal digits, and indicates multiplication by a power of 2. At least one of radix character and binary exponent must be present.

An *infinity* is either "INF" or "INFINITY", disregarding case.

A *NAN* is "NAN" (disregarding case) optionally followed by a string, (*n-char-sequence*), where *n-char-sequence* specifies in an implementation-dependent way the type of NAN (see NOTES).

RETURN VALUE

These functions return the converted value, if any.

If *endptr* is not NULL, a pointer to the character after the last character used in the conversion is stored in the location referenced by *endptr*.

If no conversion is performed, zero is returned and (unless *endptr* is null) the value of *nptr* is stored in the location referenced by *endptr*.

If the correct value would cause overflow, plus or minus **HUGE_VAL**, **HUGE_VALF**, or **HUGE_VALL** is returned (according to the return type and sign of the value), and **ERANGE** is stored in *errno*.

If the correct value would cause underflow, a value with magnitude no larger than **DBL_MIN**, **FLT_MIN**, or **LDBL_MIN** is returned and **ERANGE** is stored in *errno*.

ERRORS**ERANGE**

Overflow or underflow occurred.

ATTRIBUTES

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

Interface	Attribute	Value
strtod() , strtof() , strtold()	Thread safety	MT-Safe locale

STANDARDS

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

NOTES

Since 0 can legitimately be returned on both success and failure, the calling program should set *errno* to 0 before the call, and then determine if an error occurred by checking whether *errno* has a nonzero value after the call.

In the glibc implementation, the *n-char-sequence* that optionally follows "NAN" is interpreted as an integer number (with an optional '0' or '0x' prefix to select base 8 or 16) that is to be placed in the mantissa component of the returned value.

EXAMPLES

See the example on the **strtol**(3) manual page; the use of the functions described in this manual page is similar.

SEE ALSO

atof(3), **atoi**(3), **atol**(3), **nan**(3), **nanf**(3), **nanl**(3), **strfromd**(3), **strtol**(3), **strtoul**(3)