

**NAME**

**strtod**, **strtof**, **strtold** - convert ASCII string to floating point

**LIBRARY**

Standard C Library (libc, -lc)

**SYNOPSIS**

**#include <stdlib.h>**

*double*

**strtod**(*const char \* restrict nptr, char \*\* restrict endptr*);

*float*

**strtof**(*const char \* restrict nptr, char \*\* restrict endptr*);

*long double*

**strtold**(*const char \* restrict nptr, char \*\* restrict endptr*);

**DESCRIPTION**

These conversion 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 string is an optional plus (“+”) or minus sign (“-”) followed by either:

- a decimal significand consisting of a sequence of decimal digits optionally containing a decimal-point character, or
- a hexadecimal significand consisting of a “0X” or “0x” followed by a sequence of hexadecimal digits optionally containing a decimal-point character.

In both cases, the significand may be optionally followed by an exponent. An exponent consists of an “E” or “e” (for decimal constants) or a “P” or “p” (for hexadecimal constants), followed by an optional plus or minus sign, followed by a sequence of decimal digits. For decimal constants, the exponent indicates the power of 10 by which the significand should be scaled. For hexadecimal constants, the scaling is instead done by powers of 2.

Alternatively, if the portion of the string following the optional plus or minus sign begins with "INFINITY" or "NaN", ignoring case, it is interpreted as an infinity or a quiet NaN, respectively. The syntax "NaN(*s*)", where *s* is an alphanumeric string, produces the same value as the call **nan**("s") (respectively, **nanf**("s") and **nanl**("s").)

In any of the above cases, leading white-space characters in the string (as defined by the `isspace(3)` function) are skipped. The decimal point character is defined in the program's locale (category `LC_NUMERIC`).

## RETURN VALUES

The **`strtod()`**, **`strtof()`**, and **`strtold()`** 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 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 sign and type of the return value), and `ERANGE` is stored in *errno*. If the correct value would cause underflow, zero is returned and `ERANGE` is stored in *errno*.

## ERRORS

[`ERANGE`]            Overflow or underflow occurred.

## SEE ALSO

`atof(3)`, `atoi(3)`, `atol(3)`, `nan(3)`, `strtol(3)`, `strtoul(3)`, `wcstod(3)`

## STANDARDS

The **`strtod()`** function conforms to ISO/IEC 9899:1999 ("ISO C99").

## AUTHORS

The author of this software is David M. Gay.

Copyright (c) 1998 by Lucent Technologies  
All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that the copyright notice and this permission notice and warranty disclaimer appear in supporting documentation, and that the name of Lucent or any of its entities not be used in advertising or publicity pertaining to distribution of the software without specific, written prior

permission.

LUCENT DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL LUCENT OR ANY OF ITS ENTITIES BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.