```
**********************************************************
   13269 Mon Aug 26 04:09:40 2019
new/usr/src/man/man3c/strtod.3c
11620 strtod man page typo equence
**********************************************************
```

```
.\"
.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for
.\" permission to reproduce portions of its copyrighted documentation.
.\" Original documentation from The Open Group can be obtained online at
.\" http://www.opengroup.org/bookstore/.
.\"
.\" The Institute of Electrical and Electronics Engineers and The Open
.\" Group, have given us permission to reprint portions of their
.\" documentation.
.\"
.\" In the following statement, the phrase ‘‘this text’’ refers to portions
.\" of the system documentation.
.\"
.\" Portions of this text are reprinted and reproduced in electronic form
.\" in the SunOS Reference Manual, from IEEE Std 1003.1, 2004 Edition,
.\" Standard for Information Technology -- Portable Operating System
.\" Interface (POSIX), The Open Group Base Specifications Issue 6,
.\" Copyright (C) 2001-2004 by the Institute of Electrical and Electronics
.\" Engineers, Inc and The Open Group.  In the event of any discrepancy
.\" between these versions and the original IEEE and The Open Group
.\" Standard, the original IEEE and The Open Group Standard is the referee
.\" document.  The original Standard can be obtained online at
.\" http://www.opengroup.org/unix/online.html.
.\"
.\" This notice shall appear on any product containing this material.
.\"
.\" The contents of this file are subject to the terms of the
.\" Common Development and Distribution License (the "License").
.\" You may not use this file except in compliance with the License.
.\"
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
.\" or http://www.opensolaris.org/os/licensing.
.\" See the License for the specific language governing permissions
.\" and limitations under the License.
.\"
.\" When distributing Covered Code, include this CDDL HEADER in each
.\" file and include the License file at usr/src/OPENSOLARIS.LICENSE.
.\" If applicable, add the following below this CDDL HEADER, with the
.\" fields enclosed by brackets "[]" replaced with your own identifying
.\" information: Portions Copyright [yyyy] [name of copyright owner]
.\"
.\"
.\" Copyright 1989 AT&T
.\" Copyright (c) 1992, X/Open Company Limited.  All Rights Reserved.
.\" Portions Copyright (c) 2006, Sun Microsystems, Inc.  All Rights Reserved.
.\"
.TH STRTOD 3C "Aug 25, 2019"
.TH STRTOD 3C "Nov 1, 2003"
.SH NAME
strtod, strtof, strtold, atof \- convert string to floating-point number
.SH SYNOPSIS
.LP
.nf
#include <stdlib.h>

\fBdouble\fR \fBstrtod\fR(\fBconst char *restrict\fR \fInptr\fR, \fBchar **restr
.fi

.LP
.nf
\fBfloat\fR \fBstrtof\fR(\fBconst char *restrict\fR \fInptr\fR, \fBchar **restri
```



```
.fi

.LP
.nf
\fBlong double\fR \fBstrtold\fR(\fBconst char *restrict\fR \fInptr\fR, \fBchar *
.fi

.LP
.nf
\fBdouble\fR \fBatof\fR(\fBconst char *\fR\fIstr\fR);
.fi

.SH DESCRIPTION
.sp
.LP
The \fBstrtod()\fR, \fBstrtof()\fR, and \fBstrtold()\fR functions convert the
initial portion of the string pointed to by \fInptr\fR to \fBdouble\fR,
\fBfloat\fR, and \fBlong double\fR representation, respectively. First they
decompose the input string into three parts:
.RS +4
.TP
1.
An initial, possibly empty, sequence of white-space characters (as specified
by \fBisspace\fR(3C))
.RE
.RS +4
.TP
2.
A subject sequence interpreted as a floating-point constant or representing
infinity or NaN
.RE
.RS +4
.TP
3.
A final string of one or more unrecognized characters, including the
terminating null byte of the input string.
.RE
.sp
.LP
Then they attempt to convert the subject sequence to a floating-point number,
and return the result.
.sp
.LP
The expected form of the subject sequence is an optional plus or minus sign,
then one of the following:
.RS +4
.TP
.ie t \(bu
.el o
A non-empty sequence of digits optionally containing a radix character, then an
optional exponent part
.RE
.RS +4
.TP
.ie t \(bu
.el o
A 0x or 0X, then a non-empty sequence of hexadecimal digits optionally
containing a radix character, then an optional binary exponent part
.RE
.RS +4
.TP
.ie t \(bu
.el o
One of INF or INFINITY, ignoring case
.RE
.RS +4
```

```
.TP
.ie t \(bu
.el o
One of NAN or NAN(\fIn-char-sequence\fR(\fIopt\fR)), ignoring case in the NAN
part, where:
.sp
.in +2
.nf
n-char-sequence:
    digit
    nondigit
    n-char-sequence digit
    n-char-sequence nondigit
.fi
.in -2

.RE
.sp
.LP
In default mode for \fBstrtod()\fR, only decimal, INF/INFINITY, and
NAN/NAN(\fIn-char-sequence\fR) forms are recognized. In C99/SUSv3 mode,
hexadecimal strings are also recognized.
.sp
.LP
In default mode for \fBstrtod()\fR, the \fIn-char-sequence\fR in the
NAN(\fIn-char-sequence\fR) form can contain any character except ')' (right
NAN(\fIn-char-equence\fR) form can contain any character except ')' (right
parenthesis) or '\e0' (null).  In C99/SUSv3 mode, the \fIn-char-sequence\fR can
contain only upper and lower case letters, digits, and '_' (underscore).
.sp
.LP
The \fBstrtof()\fR and \fBstrtold()\fR functions always function in
C99/SUSv3-conformant mode.
.sp
.LP
The subject sequence is defined as the longest initial subsequence of the input
string, starting with the first non-white-space character, that is of the
expected form. The subject sequence contains no characters if the input string
is not of the expected form.
.sp
.LP
If the subject sequence has the expected form for a floating-point number, the
sequence of characters starting with the first digit or the decimal-point
character (whichever occurs first) is interpreted as a floating constant of the
C language, except that the radix character is used in place of a period, and
that if neither an exponent part nor a radix character appears in a decimal
floating-point number, or if a binary exponent part does not appear in a
hexadecimal floating-point number, an exponent part of the appropriate type
with value zero is assumed to follow the last digit in the string. If the
subject sequence begins with a minus sign, the sequence is interpreted as
negated. A character sequence INF or INFINITY is interpreted as an infinity. A
character sequence NAN or NAN(\fIn-char-sequence\fR(\fIopt\fR)) is interpreted
as a quiet NaN. A pointer to the final string is stored in the object pointed
to by \fIendptr\fR, provided that \fIendptr\fR is not a null pointer.
.sp
.LP
If the subject sequence has either the decimal or hexadecimal form, the value
resulting from the conversion is rounded correctly according to the prevailing
floating point rounding direction mode. The conversion also raises floating
point inexact, underflow, or overflow exceptions as appropriate.
.sp
.LP
The radix character is defined in the program's locale (category
\fBLC_NUMERIC\fR). In the POSIX locale, or in a locale where the radix
character is not defined, the radix character defaults to a period ('.').
.sp
```



```
.LP
If the subject sequence is empty or does not have the expected form, no
conversion is performed; the value of \fInptr\fR is stored in the object
pointed to by \fIendptr\fR, provided that \fIendptr\fR is not a null pointer.
.sp
.LP
The \fBstrtod()\fR function does not change the setting of \fBerrno\fR if
successful.
.sp
.LP
The \fBatof(\fR\fIstr\fR\fB)\fR function call is equivalent to
\fBstrtod(\fR\fInptr\fR\fB, (char **)NULL)\fR.
.SH RETURN VALUES
.sp
.LP
Upon successful completion, these functions return the converted value. If no
conversion could be performed, \fB0\fR is returned.
.sp
.LP
If the correct value is outside the range of representable values,
\fB\(+-HUGE_VAL\fR, \fB\(+-HUGE_VALF\fR, or \fB\(+-HUGE_VALL\fR is returned
(according to the sign of the value), a floating point overflow exception is
raised, and \fBerrno\fR is set to \fBERANGE\fR.
.sp
.LP
If the correct value would cause an underflow, the correctly rounded result
(which may be normal, subnormal, or zero) is returned, a floating point
underflow exception is raised, and \fBerrno\fR is set to \fBERANGE\fR.
.SH ERRORS
.sp
.LP
These functions will fail if:
.sp
.ne 2
.na
\fBERANGE\fR
\fB\fBERANGE\fR\fR
.ad
.RS 10n
The value to be returned would cause overflow or underflow
.RE

.sp
.LP
These functions may fail if:
.sp
.ne 2
.na
\fBEINVAL\fR
\fB\fBEINVAL\fR\fR
.ad
.RS 10n
No conversion could be performed.
.RE

.SH USAGE
.sp
.LP
Since 0 is returned on error and is also a valid return on success, an
application wishing to check for error situations should set \fBerrno\fR to 0,
then call \fBstrtod()\fR, \fBstrtof()\fR, or \fBstrtold()\fR, then check
\fBerrno\fR.
.sp
.LP
The changes to \fBstrtod()\fR introduced by the ISO/IEC 9899: 1999 standard can
alter the behavior of well-formed applications complying with the ISO/IEC 9899:
```

```
247 1990 standard and thus earlier versions of IEEE Std 1003.1-200x. One such
248 example would be:
249 .sp
250 .in +2
251 .nf
252 int
253 what_kind_of_number (char *s)
254 {
255      char *endp;
256      double d;
257      long l;
258      d = strtod(s, &endp);
259      if (s != endp && *endp == '\e0')
260          printf("It's a float with value %g\en", d);
261      else
262      {
263          l = strtol(s, &endp, 0);
264          if (s != endp && *endp == '\e0')
265              printf("It's an integer with value %ld\en", 1);
266          else
267              return 1;
268      }
269      return 0;
270 }
271 .fi
272 .in -2

274 .sp
275 .LP
276 If the function is called with:
277 .sp
278 .in +2
279 .nf
280 what_kind_of_number ("0x10")
281 .fi
282 .in -2

284 .sp
285 .LP
286 an ISO/IEC 9899: 1990 standard-compliant library will result in the function
287 printing:
288 .sp
289 .in +2
290 .nf
291 It's an integer with value 16
292 .fi
293 .in -2

295 .sp
296 .LP
297 With the ISO/IEC 9899: 1999 standard, the result is:
298 .sp
299 .in +2
300 .nf
301 It's a float with value 16
302 .fi
303 .in -2

305 .sp
306 .LP
307 The change in behavior is due to the inclusion of floating-point numbers in
308 hexadecimal notation without requiring that either a decimal point or the
309 binary exponent be present.
310 .SH ATTRIBUTES
320 .sp
321 .LP
```



```
311 See \fBattributes\fR(5) for descriptions of the following attributes:
312 .sp

314 .sp
315 .TS
316 box;
317 c | c
318 l | l .
319 ATTRIBUTE TYPE  ATTRIBUTE VALUE
320 _
321 CSI     Enabled
322 _
323 Interface Stability     Standard
324 _
325 MT-Level        MT-Safe with exceptions
326 .TE

328 .SH SEE ALSO
340 .sp
341 .LP
329 \fBisspace\fR(3C), \fBlocaleconv\fR(3C), \fBscanf\fR(3C), \fBsetlocale\fR(3C),
330 \fBstrtol\fR(3C), \fBattributes\fR(5), \fBstandards\fR(5)
331 .SH NOTES
345 .sp
346 .LP
332 The \fBstrtod()\fR and \fBatof()\fR functions can be used safely in
333 multithreaded applications, as long as \fBsetlocale\fR(3C) is not called to
334 change the locale.
335 .sp
336 .LP
337 The DESCRIPTION and RETURN VALUES sections above are very similar to the
338 wording used by the Single UNIX Specification version 2 (SUSv2) and the 1989 C
339 Standard to describe the behavior of the \fBstrtod()\fR function.  Since some
340 users have reported that they find the description confusing, the following
341 notes might be helpful.
342 .RS +4
343 .TP
344 1.
345 The \fBstrtod()\fR function does not modify the string pointed to by
346 \fIstr\fR and does not \fBmalloc()\fR space to hold the decomposed portions of
347 the input string.
348 .RE
349 .RS +4
350 .TP
351 2.
352 If \fIendptr\fR is not \fB(char **)NULL\fR, \fBstrtod()\fR will set the
353 pointer pointed to by \fIendptr\fR to the first byte of the "final string of
354 unrecognized characters".  (If all input characters were processed, the pointer
355 pointed to by \fIendptr\fR will be set to point to the null character at the
356 end of the input string.)
357 .RE
358 .RS +4
359 .TP
360 3.
361 If \fBstrtod()\fR returns 0.0, one of the following occurred:
362 .RS +4
363 .TP
364 a.
365 The "subject sequence" was not an empty string, but evaluated to 0.0.  (In
366 this case, \fBerrno\fR will be left unchanged.)
367 .RE
368 .RS +4
369 .TP
370 b.
371 The "subject sequence" was an empty string . In this case, \fBerrno\fR will
372 be left unchanged. (The Single UNIX Specification version 2 allows \fBerrno\fR
```

```
373 to be set to \fBEINVAL\fR or to be left unchanged. The C Standard does not
374 specify any specific behavior in this case.)
375 .RE
376 .RS +4
377 .TP
378 c.
379 The "subject sequence" specified a numeric value whose conversion resulted
380 in a floating point underflow.  In this case, an underflow exception is raised
381 and \fBerrno\fR is set to \fBERANGE\fR.
382 .RE
383 Note that the standards do not require that implementations distinguish between
384 these three cases.  An application can determine case (b) by making sure that
385 there are no leading white-space characters in the string pointed to by
386 \fIstr\fR and giving \fBstrtod()\fR an \fIendptr\fR that is not \fB(char
387 **)NULL\fR.  If \fIendptr\fR points to the first character of \fIstr\fR when
388 \fBstrtod()\fR returns, you have detected case (b).  Case (c) can be detected
389 by examining the underflow flag or by looking for a non-zero digit before the
390 exponent part of the "subject sequence".  Note, however, that the decimal-point
391 character is locale-dependent.
392 .RE
393 .RS +4
394 .TP
395 4.
396 If \fBstrtod()\fR returns \fB+HUGE_VAL\fR or \fB\(miHUGE_VAL\fR, one of the
397 following occurred:
398 .RS +4
399 .TP
400 a.
401 If \fB+HUGE_VAL\fR is returned and \fBerrno\fR is set to \fBERANGE\fR, a
402 floating point overflow occurred while processing a positive value, causing a
403 floating point overflow exception to be raised.
404 .RE
405 .RS +4
406 .TP
407 b.
408 If \fB\(miHUGE_VAL\fR is returned and \fBerrno\fR is set to \fBERANGE\fR, a
409 floating point overflow occurred while processing a negative value, causing a
410 floating point overflow exception to be raised.
411 .RE
412 .RS +4
413 .TP
414 c.
415 If \fBstrtod()\fR does not set \fBerrno\fR to \fBERANGE\fR, the value
416 specified by the "subject string" converted to \fB+HUGE_VAL\fR or
417 \fB\(miHUGE_VAL\fR, respectively.
418 .RE
419 Note that if \fBerrno\fR is set to \fBERANGE\fR when \fBstrtod()\fR is called,
420 case (c) can be distinguished from cases (a) and (b) by examining either
421 \fBERANGE\fR or the overflow flag.
422 .RE
```

```
**********************************************************
    9260 Mon Aug 26 04:09:40 2019
new/usr/src/man/man3c/wcstod.3c
11620 strtod man page typo equence
**********************************************************
.\"
.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for
.\" permission to reproduce portions of its copyrighted documentation.
.\" Original documentation from The Open Group can be obtained online at
.\" http://www.opengroup.org/bookstore/.
.\"
.\" The Institute of Electrical and Electronics Engineers and The Open
.\" Group, have given us permission to reprint portions of their
.\" documentation.
.\"
.\" In the following statement, the phrase ‘‘this text’’ refers to portions
.\" of the system documentation.
.\"
.\" Portions of this text are reprinted and reproduced in electronic form
.\" in the SunOS Reference Manual, from IEEE Std 1003.1, 2004 Edition,
.\" Standard for Information Technology -- Portable Operating System
.\" Interface (POSIX), The Open Group Base Specifications Issue 6,
.\" Copyright (C) 2001-2004 by the Institute of Electrical and Electronics
.\" Engineers, Inc and The Open Group.  In the event of any discrepancy
.\" between these versions and the original IEEE and The Open Group
.\" Standard, the original IEEE and The Open Group Standard is the referee
.\" document.  The original Standard can be obtained online at
.\" http://www.opengroup.org/unix/online.html.
.\"
.\" This notice shall appear on any product containing this material.
.\"
.\" The contents of this file are subject to the terms of the
.\" Common Development and Distribution License (the "License").
.\" You may not use this file except in compliance with the License.
.\"
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
.\" or http://www.opensolaris.org/os/licensing.
.\" See the License for the specific language governing permissions
.\" and limitations under the License.
.\"
.\" When distributing Covered Code, include this CDDL HEADER in each
.\" file and include the License file at usr/src/OPENSOLARIS.LICENSE.
.\" If applicable, add the following below this CDDL HEADER, with the
.\" fields enclosed by brackets "[]" replaced with your own identifying
.\" information: Portions Copyright [yyyy] [name of copyright owner]
.\"
.\"
.\" Copyright (c) 1992, X/Open Company Limited.  All Rights Reserved.
.\" Portions Copyright (c) 2003, Sun Microsystems, Inc.  All Rights Reserved
.\"
.TH WCSTOD 3C "Aug 25, 2019"
.TH WCSTOD 3C "Mar 31, 2003"
.SH NAME
wcstod, wcstof, wcstold, wstod, watof \- convert wide character string to
floating-point number
.SH SYNOPSIS
.LP
.nf
#include <wchar.h>

\fBdouble\fR \fBwcstod\fR(\fBconst wchar_t *restrict\fR \fInptr\fR,
     \fBwchar_t **restrict\fR \fIendptr\fR);
.fi

.LP
.nf
```



```
\fBfloat\fR \fBwcstof\fR(\fBconst wchar_t *restrict\fR \fInptr\fR,
     \fBwchar_t **restrict\fR \fIendptr\fR);
.fi

.LP
.nf
\fBlong double\fR \fBwcstold\fR(\fBconst wchar_t *restrict\fR \fInptr\fR,
     \fBwchar_t **restrict\fR \fIendptr\fR);
.fi

.LP
.nf
\fBdouble\fR \fBwstod\fR(\fBconst wchar_t *\fR\fInptr\fR, \fBwchar_t **\fR\fIend
.fi

.LP
.nf
\fBdouble\fR \fBwatof\fR(\fBwchar_t *\fR\fInptr\fR);
.fi

.SH DESCRIPTION
.sp
.LP
The \fBwcstod()\fR, \fBwcstof()\fR, and \fBwcstold()\fR functions convert the
initial portion of the wide-character string pointed to by \fInptr\fR to
\fBdouble\fR, \fBfloat\fR, and \fBlong double\fR representation, respectively.
They first decompose the input wide-character string into three parts:
.RS +4
.TP
1.
An initial, possibly empty, sequence of white-space wide-character codes (as
specified by \fBiswspace\fR(3C))
.RE
.RS +4
.TP
2.
A subject sequence interpreted as a floating-point constant or representing
infinity or NaN
.RE
.RS +4
.TP
3.
A final wide-character string of one or more unrecognized wide-character
codes, including the terminating null wide-character code of the input
wide-character string.
.RE
.sp
.LP
Then they attempt to convert the subject sequence to a floating-point number,
and return the result.
.sp
.LP
The expected form of the subject sequence is an optional plus or minus sign,
then one of the following:
.RS +4
.TP
.ie t \(bu
.el o
A non-empty sequence of decimal digits optionally containing a radix character,
then an optional exponent part
.RE
.RS +4
.TP
.ie t \(bu
.el o
A 0x or 0X, then a non-empty sequence of hexadecimal digits optionally
```

```
124 containing a radix character, then an optional binary exponent part
125 .RE
126 .RS +4
127 .TP
128 .ie t \(bu
129 .el o
130 One of INF or INFINITY, or any other wide string equivalent except for case
131 .RE
132 .RS +4
133 .TP
134 .ie t \(bu
135 .el o
136 One of NAN or NAN(\fIn-wchar-sequence\fR(\fIopt\fR)), or any other wide string
137 ignoring case in the NAN part, where:
138 .sp
139 .in +2
140 .nf
141 n-wchar-sequence:
142     digit
143     nondigit
144     n-wchar-sequence digit
145     n-wchar-sequence nondigit
146 .fi
147 .in -2

149 .RE
150 .sp
151 .LP
152 In default mode for \fBwcstod()\fR, only decimal, INF/INFINITY, and
153 NAN/NAN(\fIn-wchar-sequence\fR) forms are recognized. In C99/SUSv3 mode,
156 NAN/NAN(\fIn-char-sequence\fR) forms are recognized. In C99/SUSv3 mode,
154 hexadecimal strings are also recognized.
155 .sp
156 .LP
157 In default mode for \fBwcstod()\fR, the \fIn-wchar-sequence\fR in the
158 NAN(\fIn-wchar-sequence\fR) form can contain any character except ')' (right
159 parenthesis) or '\e0' (null).  In C99/SUSv3 mode, the \fIn-wchar-sequence\fR can
160 In default mode for \fBwcstod()\fR, the \fIn-char-sequence\fR in the
161 NAN(\fIn-char-equence\fR) form can contain any character except ')' (right
162 parenthesis) or '\e0' (null).  In C99/SUSv3 mode, the \fIn-char-sequence\fR can
160 contain only upper and lower case letters, digits, and '_' (underscore).
161 .sp
162 .LP
163 The \fBwcstof()\fR and \fBwcstold()\fR functions always function in
164 C99/SUSv3-conformant mode.
165 .sp
166 .LP
167 The subject sequence is defined as the longest initial subsequence of the input
168 wide string, starting with the first non-white-space wide character, that is of
169 the expected form. The subject sequence contains no wide characters if the
170 input wide string is not of the expected form.
171 .sp
172 .LP
173 If the subject sequence has the expected form for a floating-point number, the
174 sequence of wide characters starting with the first digit or the radix
175 character (whichever occurs first) is interpreted as a floating constant
176 according to the rules of the C language, except that the radix character is
177 used in place of a period, and that if neither an exponent part nor a radix
178 character appears in a decimal floating-point number, or if a binary exponent
179 part does not appear in a hexadecimal floating-point number, an exponent part
180 of the appropriate type with value zero is assumed to follow the last digit in
181 the string. If the subject sequence begins with a minus sign, the sequence is
182 interpreted as negated. A wide-character sequence INF or INFINITY is
183 interpreted as an infinity. A wide-character sequence NAN or
184 NAN(\fIn-wchar-sequence\fR(\fIopt\fR)) is interpreted as a quiet NaN. A pointer
185 to the final wide string is stored in the object pointed to by \fIendptr\fR,
```



```
186 provided that \fIendptr\fR is not a null pointer.
187 .sp
188 .LP
189 If the subject sequence has either the decimal or hexadecimal form, the value
190 resulting from the conversion is rounded correctly according to the prevailing
191 floating point rounding direction mode. The conversion also raises floating
192 point inexact, underflow, or overflow exceptions as appropriate.
193 .sp
194 .LP
195 The radix character is defined in the program's locale (category
196 \fBLC_NUMERIC\fR). In the POSIX locale, or in a locale where the radix
197 character is not defined, the radix character defaults to a period ('.').
198 .sp
199 .LP
200 If the subject sequence is empty or does not have the expected form, no
201 conversion is performed; the value of  \fInptr\fR is stored in the object
202 pointed to by \fIendptr\fR, provided that  \fIendptr\fR is not a null pointer.
203 .sp
204 .LP
205 The \fBwcstod()\fR function does not change the setting of \fBerrno\fR if
206 successful.
207 .sp
208 .LP
209 The \fBwstod()\fR function is identical to \fBwcstod()\fR.
210 .sp
211 .LP
212 The \fBwatof\fR(\fIstr\fR) function is equivalent to \fBwstod(\fR\fInptr\fR\fB,
213 (wchar_t **)NULL)\fR.
214 .SH RETURN VALUES
218 .sp
219 .LP
215 Upon successful completion, these functions return the converted value. If no
216 conversion could be performed, \fB0\fR is returned.
217 .sp
218 .LP
219 If the correct value is outside the range of representable values,
220 \fB\(+-HUGE_VAL\fR, \fB\(+-HUGE_VALF\fR, or \fB\(+-HUGE_VALL\fR is returned
221 (according to the sign of the value), a floating point overflow exception is
222 raised, and \fBerrno\fR is set to \fBERANGE\fR.
223 .sp
224 .LP
225 If the correct value would cause an underflow, the correctly rounded result
226 (which may be normal, subnormal, or zero) is returned, a floating point
227 underflow exception is raised, and \fBerrno\fR is set to \fBERANGE\fR.
228 .SH ERRORS
234 .sp
235 .LP
229 The \fBwcstod()\fR and \fBwstod()\fR functions will fail if:
230 .sp
231 .ne 2
232 .na
233 \fBERANGE\fR
240 \fB\fBERANGE\fR\fR
234 .ad
235 .RS 10n
236 The value to be returned would cause overflow or underflow.
237 .RE

239 .sp
240 .LP
241 The \fBwcstod()\fR and \fBwcstod()\fR functions may fail if:
242 .sp
243 .ne 2
244 .na
245 \fBEINVAL\fR
252 \fB\fBEINVAL\fR\fR
```

```
.ad
.RS 10n
No conversion could be performed.
.RE

.SH USAGE
.sp
.LP
Because 0 is returned on error and is also a valid return on success, an
application wishing to check for error situations should set \fBerrno\fR to 0
call \fBwcstod()\fR, \fBwcstof()\fR, \fBwcstold()\fR, or \fBwstod()\fR, then
check  \fBerrno\fR and if it is non-zero, assume an error has occurred.
.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
l | l
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
Interface Stability     T{
\fBwcstod()\fR, \fBwcstof()\fR, and \fBwcstold()\fR are Standard.
T}
_
MT-Level        MT-Safe
.TE

.SH SEE ALSO
.sp
.LP
\fBiswspace\fR(3C), \fBlocaleconv\fR(3C), \fBscanf\fR(3C), \fBsetlocale\fR(3C),
\fBwcstol\fR(3C), \fBattributes\fR(5), \fBstandards\fR(5)
```