



Red Hat Enterprise Linux Release 9.2 Manual Pages on 'fwscanf.3p' command

\$ man fwscanf.3p

FWSCANF(3P) POSIX Programmer's Manual FWSCANF(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

fwscanf, swscanf, wscanf ? convert formatted wide-character input

SYNOPSIS

```
#include <stdio.h>
#include <wchar.h>

int fwscanf(FILE *restrict stream, const wchar_t *restrict format, ...);
int swscanf(const wchar_t *restrict ws,
            const wchar_t *restrict format, ...);
int wscanf(const wchar_t *restrict format, ...);
```

DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1?2017 defers to the ISO C standard.

The fwscanf() function shall read from the named input stream. The swscanf() function shall read from the standard input stream stdin. The wscanf() function shall read from the wide-character string ws. Each

function reads wide characters, interprets them according to a format, and stores the results in its arguments. Each expects, as arguments, a control wide-character string format described below, and a set of pointer arguments indicating where the converted input should be stored. The result is undefined if there are insufficient arguments for the format. If the format is exhausted while arguments remain, the excess arguments are evaluated but are otherwise ignored.

Conversions can be applied to the *n*th argument after the format in the argument list, rather than to the next unused argument. In this case, the conversion specifier wide character % (see below) is replaced by the sequence "%n\$", where *n* is a decimal integer in the range [1, {NL_ARGMAX}]. This feature provides for the definition of format wide-character strings that select arguments in an order appropriate to specific languages. In format wide-character strings containing the "%n\$" form of conversion specifications, it is unspecified whether numbered arguments in the argument list can be referenced from the format wide-character string more than once.

The format can contain either form of a conversion specification—that is, % or "%n\$"—but the two forms cannot normally be mixed within a single format wide-character string. The only exception to this is that %% or %* can be mixed with the "%n\$" form. When numbered argument specifications are used, specifying the *N*th argument requires that all the leading arguments, from the first to the (*N*-1)th, are pointers.

The `fwscanf()` function in all its forms allows for detection of a language-dependent radix character in the input string, encoded as a wide-character value. The radix character is defined in the current locale (category `LC_NUMERIC`). In the POSIX locale, or in a locale where the radix character is not defined, the radix character shall default to a <period> ('.').

The format is a wide-character string composed of zero or more directives. Each directive is composed of one of the following: one or more white-space wide characters (<space>, <tab>, <newline>, <vertical-tab>, or <form-feed>); an ordinary wide character (neither '%' nor a white-

space character); or a conversion specification. It is unspecified whether an encoding error occurs if the format string contains `wchar_t` values that do not correspond to members of the character set of the current locale and the specified semantics do not require that value to be processed by `wcrtomb()`.

Each conversion specification is introduced by the '%' or by the character sequence "%n\$", after which the following appear in sequence:

- * An optional assignment-suppressing character '*'.
- * An optional non-zero decimal integer that specifies the maximum field width.
- * An optional assignment-allocation character 'm'.
- * An optional length modifier that specifies the size of the receiving object.
- * A conversion specifier wide character that specifies the type of conversion to be applied. The valid conversion specifiers are described below.

The `fwscanf()` functions shall execute each directive of the format in turn. If a directive fails, as detailed below, the function shall re-

turn. Failures are described as input failures (due to the unavailability of input bytes) or matching failures (due to inappropriate input).

A directive composed of one or more white-space wide characters is executed by reading input until no more valid input can be read, or up to the first wide character which is not a white-space wide character, which remains unread.

A directive that is an ordinary wide character shall be executed as follows. The next wide character is read from the input and compared with the wide character that comprises the directive; if the comparison shows that they are not equivalent, the directive shall fail, and the differing and subsequent wide characters remain unread. Similarly, if end-of-file, an encoding error, or a read error prevents a wide character from being read, the directive shall fail.

A directive that is a conversion specification defines a set of matching input sequences, as described below for each conversion wide character.

acter. A conversion specification is executed in the following steps.

Input white-space wide characters (as specified by `iswspace()`) shall be skipped, unless the conversion specification includes a `l`, `c`, or `n` conversion specifier.

An item shall be read from the input, unless the conversion specification includes an `n` conversion specifier wide character. An input item is defined as the longest sequence of input wide characters, not exceeding any specified field width, which is an initial subsequence of a matching sequence. The first wide character, if any, after the input item shall remain unread. If the length of the input item is zero, the execution of the conversion specification shall fail; this condition is a matching failure, unless end-of-file, an encoding error, or a read error prevented input from the stream, in which case it is an input failure.

Except in the case of a `%` conversion specifier, the input item (or, in the case of a `%n` conversion specification, the count of input wide characters) shall be converted to a type appropriate to the conversion wide character. If the input item is not a matching sequence, the execution of the conversion specification shall fail; this condition is a matching failure. Unless assignment suppression was indicated by a `*`, the result of the conversion shall be placed in the object pointed to by the first argument following the format argument that has not already received a conversion result if the conversion specification is introduced by `%`, or in the `n`th argument if introduced by the wide-character sequence `"%n$"`. If this object does not have an appropriate type, or if the result of the conversion cannot be represented in the space provided, the behavior is undefined.

The `%c`, `%s`, and `%[` conversion specifiers shall accept an optional assignment-allocation character `'m'`, which shall cause a memory buffer to be allocated to hold the wide-character string converted including a terminating null wide character. In such a case, the argument corresponding to the conversion specifier should be a reference to a pointer value that will receive a pointer to the allocated buffer. The system

shall allocate a buffer as if `malloc()` had been called. The application shall be responsible for freeing the memory after usage. If there is insufficient memory to allocate a buffer, the function shall set `errno` to `[ENOMEM]` and a conversion error shall result. If the function returns `EOF`, any memory successfully allocated for parameters using assignment-allocation character 'm' by this call shall be freed before the function returns.

The length modifiers and their meanings are:

- hh Specifies that a following `d`, `i`, `o`, `u`, `x`, `X`, or `n` conversion specifier applies to an argument with type pointer to signed char or unsigned char.
- h Specifies that a following `d`, `i`, `o`, `u`, `x`, `X`, or `n` conversion specifier applies to an argument with type pointer to short or unsigned short.
- l (ell) Specifies that a following `d`, `i`, `o`, `u`, `x`, `X`, or `n` conversion specifier applies to an argument with type pointer to long or unsigned long; that a following `a`, `A`, `e`, `E`, `f`, `F`, `g`, or `G` conversion specifier applies to an argument with type pointer to double; or that a following `c`, `s`, or `[` conversion specifier applies to an argument with type pointer to `wchar_t`. If the 'm' assignment-allocation character is specified, the conversion applies to an argument with the type pointer to a pointer to `wchar_t`.
- ll (ell-ell) Specifies that a following `d`, `i`, `o`, `u`, `x`, `X`, or `n` conversion specifier applies to an argument with type pointer to long long or unsigned long long.
- j Specifies that a following `d`, `i`, `o`, `u`, `x`, `X`, or `n` conversion specifier applies to an argument with type pointer to `intmax_t` or `uintmax_t`.
- z Specifies that a following `d`, `i`, `o`, `u`, `x`, `X`, or `n` conversion specifier applies to an argument with type pointer to `size_t` or the corresponding signed integer type.

t Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to ptrdiff_t or the corresponding unsigned type.

L Specifies that a following a, A, e, E, f, F, g, or G conversion specifier applies to an argument with type pointer to long double.

If a length modifier appears with any conversion specifier other than as specified above, the behavior is undefined.

The following conversion specifier wide characters are valid:

d Matches an optionally signed decimal integer, whose format is the same as expected for the subject sequence of wcstol() with the value 10 for the base argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to int.

i Matches an optionally signed integer, whose format is the same as expected for the subject sequence of wcstol() with 0 for the base argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to int.

o Matches an optionally signed octal integer, whose format is the same as expected for the subject sequence of wcstoul() with the value 8 for the base argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to unsigned.

u Matches an optionally signed decimal integer, whose format is the same as expected for the subject sequence of wcstoul() with the value 10 for the base argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to unsigned.

x Matches an optionally signed hexadecimal integer, whose format is the same as expected for the subject sequence of wcstoul() with the value 16 for the base argument. In the absence of a size modifier, the application shall ensure that the corre?

sponding argument is a pointer to unsigned.

a, e, f, g

Matches an optionally signed floating-point number, infinity, or NaN whose format is the same as expected for the subject sequence of `wcstod()`. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to float.

If the `fwprintf()` family of functions generates character string representations for infinity and NaN (a symbolic entity encoded in floating-point format) to support IEEE Std 754-1985, the `fwscanf()` family of functions shall recognize them as input.

s Matches a sequence of non-white-space wide characters. If no `l` (`ell`) qualifier is present, characters from the input field shall be converted as if by repeated calls to the `wcrtomb()` function, with the conversion state described by an `mbstate_t` object initialized to zero before the first wide character is converted. If the `'m'` assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to a character array large enough to accept the sequence and the terminating null character, which shall be added automatically. Otherwise, the application shall ensure that the corresponding argument is a pointer to a pointer to a `wchar_t`.

If the `l` (`ell`) qualifier is present and the `'m'` assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to an array of `wchar_t` large enough to accept the sequence and the terminating null wide character, which shall be added automatically.

If the `l` (`ell`) qualifier is present and the `'m'` assignment-allocation character is present, the application shall ensure that the corresponding argument is a pointer to a pointer to a `wchar_t`.

[Matches a non-empty sequence of wide characters from a set of expected wide characters (the scanset). If no l (ell) qualifier is present, wide characters from the input field shall be converted as if by repeated calls to the wctomb() function, with the conversion state described by an mbstate_t object initialized to zero before the first wide character is converted. If the 'm' assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to a character array large enough to accept the sequence and the terminating null character, which shall be added automatically. Otherwise, the application shall ensure that the corresponding argument is a pointer to a pointer to a wchar_t.

If an l (ell) qualifier is present and the 'm' assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to an array of wchar_t large enough to accept the sequence and the terminating null wide character. If an l (ell) qualifier is present and the 'm' assignment-allocation character is specified, the application shall ensure that the corresponding argument is a pointer to a pointer to a wchar_t.

The conversion specification includes all subsequent wide characters in the format string up to and including the matching <right-square-bracket> (']'). The wide characters between the square brackets (the scanlist) comprise the scanset, unless the wide character after the <left-square-bracket> is a <circumflex> ('^'), in which case the scanset contains all wide characters that do not appear in the scanlist between the <circumflex> and the <right-square-bracket>. If the conversion specification begins with "[^]" or "[^]", the <right-square-bracket> is included in the scanlist and the next <right-square-bracket> is the matching <right-square-bracket> that ends the conversion specification; otherwise, the first <right-square-bracket> is

the one that ends the conversion specification. If a '-' is in the scanlist and is not the first wide character, nor the second where the first wide character is a '^', nor the last wide character, the behavior is implementation-defined.

- c Matches a sequence of wide characters of exactly the number specified by the field width (1 if no field width is present in the conversion specification).

If no l (ell) length modifier is present, characters from the input field shall be converted as if by repeated calls to the wctomb() function, with the conversion state described by an mbstate_t object initialized to zero before the first wide character is converted. No null character is added. If the 'm' assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to the initial element of a character array large enough to accept the sequence. Otherwise, the application shall ensure that the corresponding argument is a pointer to a pointer to a char.

No null wide character is added. If an l (ell) length modifier is present and the 'm' assignment-allocation character is not specified, the application shall ensure that the corresponding argument shall be a pointer to the initial element of an array of wchar_t large enough to accept the sequence. If an l (ell) qualifier is present and the 'm' assignment-allocation character is specified, the application shall ensure that the corresponding argument is a pointer to a pointer to a wchar_t.

- p Matches an implementation-defined set of sequences, which shall be the same as the set of sequences that is produced by the %p conversion specification of the corresponding fprintf() functions. The application shall ensure that the corresponding argument is a pointer to a pointer to void. The interpretation of the input item is implementation-defined. If the input item is a value converted earlier during the same program execution,

the pointer that results shall compare equal to that value; otherwise, the behavior of the %p conversion is undefined.

n No input is consumed. The application shall ensure that the corresponding argument is a pointer to the integer into which is to be written the number of wide characters read from the input so far by this call to the fwscanf() functions. Execution of a %n conversion specification shall not increment the assignment count returned at the completion of execution of the function. No argument shall be converted, but one shall be consumed. If the conversion specification includes an assignment-suppressing wide character or a field width, the behavior is undefined.

C Equivalent to lc.

S Equivalent to ls.

% Matches a single '%' wide character; no conversion or assignment shall occur. The complete conversion specification shall be %%.

If a conversion specification is invalid, the behavior is undefined.

The conversion specifiers A, E, F, G, and X are also valid and shall be equivalent to, respectively, a, e, f, g, and x.

If end-of-file is encountered during input, conversion is terminated.

If end-of-file occurs before any wide characters matching the current conversion specification (except for %n) have been read (other than leading white-space, where permitted), execution of the current conversion specification shall terminate with an input failure. Otherwise, unless execution of the current conversion specification is terminated with a matching failure, execution of the following conversion specification (if any) shall be terminated with an input failure.

Reaching the end of the string in swscanf() shall be equivalent to encountering end-of-file for fwscanf().

If conversion terminates on a conflicting input, the offending input shall be left unread in the input. Any trailing white space (including <newline>) shall be left unread unless matched by a conversion specification.

cation. The success of literal matches and suppressed assignments is only directly determinable via the %n conversion specification.

The fwscanf() and wscanf() functions may mark the last data access timestamp of the file associated with stream for update. The last data access timestamp shall be marked for update by the first successful execution of fgetwc(), fgetws(), fwscanf(), getwc(), getwchar(), vfwscanf(), vwscanf(), or wscanf() using stream that returns data not supplied by a prior call to ungetwc().

RETURN VALUE

Upon successful completion, these functions shall return the number of successfully matched and assigned input items; this number can be zero in the event of an early matching failure. If the input ends before the first conversion (if any) has completed, and without a matching failure having occurred, EOF shall be returned. If an error occurs before the first conversion (if any) has completed, and without a matching failure having occurred, EOF shall be returned and errno shall be set to indicate the error. If a read error occurs, the error indicator for the stream shall be set.

ERRORS

For the conditions under which the fwscanf() functions shall fail and may fail, refer to fgetwc().

In addition, the fwscanf() function shall fail if:

EILSEQ Input byte sequence does not form a valid character.

ENOMEM Insufficient storage space is available.

In addition, the fwscanf() function may fail if:

EINVAL There are insufficient arguments.

The following sections are informative.

EXAMPLES

The call:

```
int i, n; float x; char name[50];  
n = wscanf(L"%d%f%s", &i, &x, name);
```

with the input line:

```
25 54.32E-1 Hamster
```

assigns to `n` the value 3, to `i` the value 25, to `x` the value 5.432, and `name` contains the string "Hamster".

The call:

```
int i; float x; char name[50];  
(void) wscanf(L"%2d%f%d %[0123456789]", &i, &x, name);
```

with input:

```
56789 0123 56a72
```

assigns 56 to `i`, 789.0 to `x`, skips 0123, and places the string "56\0" in `name`. The next call to `getchar()` shall return the character 'a'.

APPLICATION USAGE

In format strings containing the '%' form of conversion specifications, each argument in the argument list is used exactly once.

For functions that allocate memory as if by `malloc()`, the application should release such memory when it is no longer required by a call to `free()`. For `fwscanf()`, this is memory allocated via use of the 'm' as? signment-allocation character.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.5, Standard I/O Streams, `getwc()`, `fwprintf()`, `setlocale()`, `wcstod()`, `wcstol()`, `wcstoul()`, `wcrtomb()`

The Base Definitions volume of POSIX.1-2017, Chapter 7, Locale, `<inttypes.h>`, `<stdio.h>`, `<wchar.h>`

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1-2017, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 7, 2018 Edition, Copyright (C) 2018 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version 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> .

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.kernel.org/doc/man-pages/reporting_bugs.html .

IEEE/The Open Group

2017

FWSCANF(3P)