NAME

wscanf, **fwscanf**, **swscanf**, **vwscanf**, **vswscanf**, **vfwscanf** - wide character input format conversion

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

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

int
wscanf(*const wchar_t * restrict format, ...*);

int
fwscanf(*FILE * restrict stream, const wchar_t * restrict format, ...*);

int
swscanf(*const wchar_t * restrict str, const wchar_t * restrict format, ...*);

#include <stdarg.h>

int
vwscanf(*const wchar_t * restrict format, va_list ap*);

int
vswscanf(*const wchar_t * restrict str, const wchar_t * restrict format, va_list ap*);

int
vfwscanf(*FILE * restrict stream, const wchar_t * restrict format, va_list ap*);

DESCRIPTION

The **wscanf**() family of functions scans input according to a *format* as described below. This format may contain *conversion specifiers*; the results from such conversions, if any, are stored through the *pointer* arguments. The **wscanf**() function reads input from the standard input stream stdin, **fwscanf**() reads input from the stream pointer *stream*, and **swscanf**() reads its input from the wide character string pointed to by *str*. The **vfwscanf**() function is analogous to vfwprintf(3) and reads input from the stream pointer *stream* using a variable argument list of pointers (see stdarg(3)). The **vwscanf**() function scans a variable argument list from the standard input and the **vswscanf**() function scans it from a wide character string; these are analogous to the **vwprintf**() and **vswprintf**() functions respectively. Each successive *pointer* argument must correspond properly with each successive conversion specifier (but see the *****

conversion below). All conversions are introduced by the **%** (percent sign) character. The *format* string may also contain other characters. White space (such as blanks, tabs, or newlines) in the *format* string match any amount of white space, including none, in the input. Everything else matches only itself. Scanning stops when an input character does not match such a format character. Scanning also stops when an input conversion cannot be made (see below).

CONVERSIONS

Following the **%** character introducing a conversion there may be a number of *flag* characters, as follows:

***** Suppresses assignment. The conversion that follows occurs as usual, but no pointer is used; the result of the conversion is simply discarded.

hh Indicates that the conversion will be one of **dioux** or **n** and the next pointer is a pointer to a *char* (rather than *int*).

h Indicates that the conversion will be one of **dioux** or **n** and the next pointer is a pointer to a *short int* (rather than *int*).

l (ell) Indicates that the conversion will be one of **dioux** or **n** and the next pointer is a pointer to a *long int* (rather than *int*), that the conversion will be one of **a**, **e**, **f**, or **g** and the next pointer is a pointer to *double* (rather than *float*), or that the conversion will be one of **c** or **s** and the next pointer is a pointer to an array of *wchar_t* (rather than *char*).

ll (ell ell)
Indicates that the conversion will be one of **dioux** or **n** and the next pointer is a pointer to a *long long int* (rather than *int*).

L Indicates that the conversion will be one of **a**, **e**, **f**, or **g** and the next pointer is a pointer to *long double*.

j Indicates that the conversion will be one of **dioux** or **n** and the next pointer is a pointer to a *intmax_t* (rather than *int*).

t Indicates that the conversion will be one of **dioux** or **n** and the next pointer is a pointer to a *ptrdiff_t* (rather than *int*).

z Indicates that the conversion will be one of **dioux** or **n** and the next pointer is a pointer to a *size_t* (rather than *int*).

q (deprecated.) Indicates that the conversion will be one of **dioux** or **n** and the next pointer is a pointer to a *long long int* (rather than *int*).

In addition to these flags, there may be an optional maximum field width, expressed as a decimal integer, between the **%** and the conversion. If no width is given, a default of "infinity" is used (with one exception, below); otherwise at most this many characters are scanned in processing the conversion. Before conversion begins, most conversions skip white space; this white space is not counted against the field width.

The following conversions are available:

% Matches a literal '%'. That is, "%%" in the format string matches a single input '%' character. No conversion is done, and assignment does not occur.

d Matches an optionally signed decimal integer; the next pointer must be a pointer to *int*.

i Matches an optionally signed integer; the next pointer must be a pointer to *int*. The integer is read in base 16 if it begins with '0x' or '0X', in base 8 if it begins with '0', and in base 10 otherwise. Only characters that correspond to the base are used.

o Matches an octal integer; the next pointer must be a pointer to *unsigned int*.

u Matches an optionally signed decimal integer; the next pointer must be a pointer to *unsigned int*.

x, **X** Matches an optionally signed hexadecimal integer; the next pointer must be a pointer to *unsigned int*.

a, **A**, **e**, **E**, **f**, **F**, **g**, **G**
Matches a floating-point number in the style of wcstod(3). The next pointer must be a pointer to *float* (unless **l** or **L** is specified.)

s Matches a sequence of non-white-space wide characters; the next pointer must be a pointer to *char*, and the array must be large enough to accept the multibyte representation of all the sequence and the terminating NUL character. The input string stops at white space or at the maximum field width, whichever occurs first.

If an **l** qualifier is present, the next pointer must be a pointer to *wchar_t*, into which the input will be placed.

S The same as **ls**.

c Matches a sequence of *width* count wide characters (default 1); the next pointer must be a pointer to *char*, and there must be enough room for the multibyte representation of all the characters (no terminating NUL is added). The usual skip of leading white space is suppressed. To skip white space first, use an explicit space in the format.

If an **l** qualifier is present, the next pointer must be a pointer to *wchar_t*, into which the input will be placed.

C The same as **lc**.

[Matches a nonempty sequence of characters from the specified set of accepted characters; the next pointer must be a pointer to *char*, and there must be enough room for the multibyte representation of all the characters in the string, plus a terminating NUL character. The usual skip of leading white space is suppressed. The string is to be made up of characters in (or not in) a particular set; the set is defined by the characters between the open bracket **[** character and a close bracket **]** character. The set *excludes* those characters if the first character after the open bracket is a circumflex ^. To include a close bracket in the set, make it the first character after the open bracket or the circumflex; any other position will end the set. To include a hyphen in the set, make it the last character before the final close bracket; some implementations of **wscanf**() use "A-Z" to represent the range of characters between ‘A’ and ‘Z’. The string ends with the appearance of a character not in the (or, with a circumflex, in) set or when the field width runs out.

If an **l** qualifier is present, the next pointer must be a pointer to *wchar_t*, into which the input will be placed.

p Matches a pointer value (as printed by ‘%p’ in wprintf(3)); the next pointer must be a pointer to *void*.

n Nothing is expected; instead, the number of characters consumed thus far from the input is stored through the next pointer, which must be a pointer to *int*. This is *not* a conversion, although it can be suppressed with the ***** flag.

The decimal point character is defined in the program’s locale (category LC_NUMERIC).

For backwards compatibility, a "conversion" of ‘%\0’ causes an immediate return of EOF.

RETURN VALUES

These functions return the number of input items assigned, which can be fewer than provided for, or even zero, in the event of a matching failure. Zero indicates that, while there was input available, no

conversions were assigned; typically this is due to an invalid input character, such as an alphabetic character for a ‘%d’ conversion. The value EOF is returned if an input failure occurs before any conversion such as an end-of-file occurs. If an error or end-of-file occurs after conversion has begun, the number of conversions which were successfully completed is returned.

SEE ALSO

fgetwc(3), scanf(3), wcrtomb(3), wcstod(3), wcstol(3), wcstoul(3), wprintf(3)

STANDARDS

The **fwscanf**(), **wscanf**(), **swscanf**(), **vfwscanf**(), **vwscanf**() and **vswscanf**() functions conform to ISO/IEC 9899:1999 ("ISO C99").

BUGS

In addition to the bugs documented in scanf(3), **wscanf**() does not support the "A-Z" notation for specifying character ranges with the character class conversion (‘**%[**’).