## NAME

**strfmon**, **strfmon_l** - convert monetary value to string

## LIBRARY

Standard C Library (libc, -lc)

## SYNOPSIS

**#include <monetary.h>**

*ssize_t*
**strfmon**(*char * restrict s, size_t maxsize, const char * restrict format, ...*);

**#include <monetary.h>**
**#include <xlocale.h>**

*ssize_t*
**strfmon_l**(*char * restrict s, size_t maxsize, locale_t loc, const char * restrict format, ...*);

## DESCRIPTION

The **strfmon**() function places characters into the array pointed to by *s*, as controlled by the string pointed to by *format*. No more than *maxsize* bytes are placed into the array.

The **strfmon_l**() function takes an explicit locale argument, whereas the **strfmon**() function uses the current global or per-thread locale.

The format string is composed of zero or more directives: ordinary characters (not **%**), which are copied unchanged to the output stream; and conversion specifications, each of which results in fetching zero or more subsequent arguments. Each conversion specification is introduced by the **%** character. After the **%**, the following appear in sequence:

- Zero or more of the following flags:

  **=***f* A '=' character followed by another character *f* which is used as the numeric fill character.

  **^** Do not use grouping characters, regardless of the current locale default.

  **+** Represent positive values by prefixing them with a positive sign, and negative values by prefixing them with a negative sign. This is the default.

  **(** Enclose negative values in parentheses.

**!** Do not include a currency symbol in the output.

**-** Left justify the result. Only valid when a field width is specified.

- An optional minimum field width as a decimal number. By default, there is no minimum width.

- A ‘#’ sign followed by a decimal number specifying the maximum expected number of digits before the radix character. When this option is used, values that do not exceed the specified number of digits are formatted so they will be correctly aligned with other values printed using the same format. This includes always leaving space for a possible sign indicator, even if none is needed for a particular value.

- A ‘**.**’ character followed by a decimal number specifying the number of digits after the radix character.

- One of the following conversion specifiers:

**i** The *double* argument is formatted as an international monetary amount.

**n** The *double* argument is formatted as a national monetary amount.

**%** A ‘%’ character is written.

## RETURN VALUES

If the total number of resulting bytes, including the terminating NUL byte, is not more than *maxsize*, **strfmon**() and **strfmon_l**() return the number of bytes placed into the array pointed to by *s*, not including the terminating NUL byte. Otherwise, -1 is returned, the contents of the array are indeterminate, and *errno* is set to indicate the error.

## EXAMPLES

The following example will format the value "1234567.89" to the string "$1,234,567.89":

```
#include <stdio.h>
#include <monetary.h>
#include <xlocale.h>

int
main()
{
        char string[100];
```

```
        double money = 1234567.89;

        if (setlocale(LC_MONETARY, "en_US.UTF-8") == NULL) {
                fprintf(stderr, "Unable to setlocale().\n");
                return (1);
        }

        strfmon(string, sizeof(string) - 1, "%n", money);
        printf("%s\n", string);
}
```

**ERRORS**

The **strfmon**() function will fail if:

[E2BIG] Conversion stopped due to lack of space in the buffer.

[EINVAL] The format string is invalid.

[ENOMEM] Not enough memory for temporary buffers.

**SEE ALSO**

localeconv(3), xlocale(3)

**STANDARDS**

The **strfmon**() function conforms to IEEE Std 1003.1-2001 ("POSIX.1"). The **strfmon_l**() function conforms to IEEE Std 1003.1-2008 ("POSIX.1").

**AUTHORS**

The **strfmon**() function was implemented by Alexey Zelkin <*phantom@FreeBSD.org*>.

This manual page was written by Jeroen Ruigrok van der Werven <*asmodai@FreeBSD.org*> based on the standards' text.

**BUGS**

The **strfmon**() function does not correctly handle multibyte characters in the *format* argument.