Full credit is given to the above companies including the OS that this PDF file was generated!

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

$ man strftime.3p

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

strftime, strftime_l ? convert date and time to a string

SYNOPSIS

```
#include <time.h>
size_t strftime(char *restrict s, size_t maxsize,
    const char *restrict format, const struct tm *restrict timeptr);
size_t strftime_l(char *restrict s, size_t maxsize,
    const char *restrict format, const struct tm *restrict timeptr,
    locale_t locale);
```

DESCRIPTION

For strftime(): 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 strftime() function shall place bytes into the array pointed to by s as controlled by the string pointed to by format. The format is a character string, beginning and ending in its initial shift state, if

any. The format string consists of zero or more conversion specifica? tions and ordinary characters.

Each conversion specification is introduced by the '%' character after which the following appear in sequence:

* An optional flag:

  0 The zero character ('0'), which specifies that the character used as the padding character is '0',

  + The <plus-sign> character ('+'), which specifies that the character used as the padding character is '0', and that if and only if the field being produced consumes more than four bytes to represent a year (for %F, %G, or %Y) or more than two bytes to represent the year divided by 100 (for %C) then a leading <plus-sign> character shall be included if the year being processed is greater than or equal to zero or a leading <hyphen-minus> character ('-') shall be included if the year is less than zero.

  The default padding character is unspecified.

* An optional minimum field width. If the converted value, including any leading '+' or '-' sign, has fewer bytes than the minimum field width and the padding character is not the NUL character, the out? put shall be padded on the left (after any leading '+' or '-' sign) with the padding character.

* An optional E or O modifier.

* A terminating conversion specifier character that indicates the type of conversion to be applied.

The results are unspecified if more than one flag character is speci? fied, a flag character is specified without a minimum field width; a minimum field width is specified without a flag character; a modifier is specified with a flag or with a minimum field width; or if a minimum field width is specified for any conversion specifier other than C, F, G, or Y.

All ordinary characters (including the terminating NUL character) are copied unchanged into the array. If copying takes place between objects

that overlap, the behavior is undefined. No more than maxsize bytes are placed into the array. Each conversion specifier is replaced by appro? priate characters as described in the following list. The appropriate characters are determined using the LC_TIME category of the current lo? cale and by the values of zero or more members of the broken-down time structure pointed to by timeptr, as specified in brackets in the de? scription. If any of the specified values are outside the normal range, the characters stored are unspecified.

The strftime_l() function shall be equivalent to the strftime() func? tion, except that the locale data used is from the locale represented by locale.

Local timezone information is used as though strftime() called tzset().

The following conversion specifiers shall be supported:

a Replaced by the locale's abbreviated weekday name. [tm_wday]

A Replaced by the locale's full weekday name. [tm_wday]

b Replaced by the locale's abbreviated month name. [tm_mon]

B Replaced by the locale's full month name. [tm_mon]

c Replaced by the locale's appropriate date and time representa? tion. (See the Base Definitions volume of POSIX.1?2017, <time.h>.)

C Replaced by the year divided by 100 and truncated to an inte? ger, as a decimal number. [tm_year]

If a minimum field width is not specified, the number of char? acters placed into the array pointed to by s will be the number of digits in the year divided by 100 or two, whichever is greater. If a minimum field width is specified, the number of characters placed into the array pointed to by s will be the number of digits in the year divided by 100 or the minimum field width, whichever is greater.

d Replaced by the day of the month as a decimal number [01,31]. [tm_mday]

D Equivalent to %m/%d/%y. [tm_mon, tm_mday, tm_year]

e Replaced by the day of the month as a decimal number [1,31]; a

single digit is preceded by a space. [tm_mday]

F Equivalent to %+4Y-%m-%d if no flag and no minimum field width are specified. [tm_year, tm_mon, tm_mday]

If a minimum field width of x is specified, the year shall be output as if by the Y specifier (described below) with whatever flag was given and a minimum field width of x-6. If x is less than 6, the behavior shall be as if x equalled 6.

If the minimum field width is specified to be 10, and the year is four digits long, then the output string produced will match the ISO 8601:2004 standard subclause 4.1.2.2 complete represen? tation, extended format date representation of a specific day.

If a + flag is specified, a minimum field width of x is speci? fied, and x-7 bytes are sufficient to hold the digits of the year (not including any needed sign character), then the output will match the ISO 8601:2004 standard subclause 4.1.2.4 com? plete representation, expanded format date representation of a specific day.

g Replaced by the last 2 digits of the week-based year (see be? low) as a decimal number [00,99]. [tm_year, tm_wday, tm_yday]

G Replaced by the week-based year (see below) as a decimal number (for example, 1977). [tm_year, tm_wday, tm_yday]

If a minimum field width is specified, the number of characters placed into the array pointed to by s will be the number of digits and leading sign characters (if any) in the year, or the minimum field width, whichever is greater.

h Equivalent to %b. [tm_mon]

H Replaced by the hour (24-hour clock) as a decimal number [00,23]. [tm_hour]

I Replaced by the hour (12-hour clock) as a decimal number [01,12]. [tm_hour]

j Replaced by the day of the year as a decimal number [001,366]. [tm_yday]

m Replaced by the month as a decimal number [01,12]. [tm_mon]

M Replaced by the minute as a decimal number [00,59]. [tm_min]

n Replaced by a <newline>.

p Replaced by the locale's equivalent of either a.m. or p.m. [tm_hour]

r Replaced by the time in a.m. and p.m. notation; in the POSIX locale this shall be equivalent to %I:%M:%S %p. [tm_hour, tm_min, tm_sec]

R Replaced by the time in 24-hour notation (%H:%M). [tm_hour, tm_min]

S Replaced by the second as a decimal number [00,60]. [tm_sec]

t Replaced by a <tab>.

T Replaced by the time (%H:%M:%S). [tm_hour, tm_min, tm_sec]

u Replaced by the weekday as a decimal number [1,7], with 1 rep? resenting Monday. [tm_wday]

U Replaced by the week number of the year as a decimal number [00,53]. The first Sunday of January is the first day of week 1; days in the new year before this are in week 0. [tm_year, tm_wday, tm_yday]

V Replaced by the week number of the year (Monday as the first day of the week) as a decimal number [01,53]. If the week con? taining 1 January has four or more days in the new year, then it is considered week 1. Otherwise, it is the last week of the previous year, and the next week is week 1. Both January 4th and the first Thursday of January are always in week 1. [tm_year, tm_wday, tm_yday]

w Replaced by the weekday as a decimal number [0,6], with 0 rep? resenting Sunday. [tm_wday]

W Replaced by the week number of the year as a decimal number [00,53]. The first Monday of January is the first day of week 1; days in the new year before this are in week 0. [tm_year, tm_wday, tm_yday]

x Replaced by the locale's appropriate date representation. (See the Base Definitions volume of POSIX.1?2017, <time.h>.)

X Replaced by the locale's appropriate time representation. (See the Base Definitions volume of POSIX.1?2017, <time.h>.)

y Replaced by the last two digits of the year as a decimal number [00,99]. [tm_year]

Y Replaced by the year as a decimal number (for example, 1997). [tm_year]

If a minimum field width is specified, the number of characters placed into the array pointed to by s will be the number of digits and leading sign characters (if any) in the year, or the minimum field width, whichever is greater.

z Replaced by the offset from UTC in the ISO 8601:2004 standard format (+hhmm or -hhmm), or by no characters if no timezone is determinable. For example, "-0430" means 4 hours 30 minutes be? hind UTC (west of Greenwich). If tm_isdst is zero, the stan? dard time offset is used. If tm_isdst is greater than zero, the daylight savings time offset is used. If tm_isdst is negative, no characters are returned. [tm_isdst]

Z Replaced by the timezone name or abbreviation, or by no bytes if no timezone information exists. [tm_isdst]

% Replaced by %.

If a conversion specification does not correspond to any of the above, the behavior is undefined.

If a struct tm broken-down time structure is created by localtime() or localtime_r(), or modified by mktime(), and the value of TZ is subse? quently modified, the results of the %Z and %z strftime() conversion specifiers are undefined, when strftime() is called with such a broken-down time structure.

If a struct tm broken-down time structure is created or modified by gm? time() or gmtime_r(), it is unspecified whether the result of the %Z and %z conversion specifiers shall refer to UTC or the current local timezone, when strftime() is called with such a broken-down time struc? ture.

Some conversion specifiers can be modified by the E or O modifier char?
acters to indicate that an alternative format or specification should
be used rather than the one normally used by the unmodified conversion
specifier. If the alternative format or specification does not exist
for the current locale (see ERA in the Base Definitions volume of
POSIX.1?2017, Section 7.3.5, LC_TIME), the behavior shall be as if the
unmodified conversion specification were used.

%Ec Replaced by the locale's alternative appropriate date and time representation.

%EC Replaced by the name of the base year (period) in the locale's alternative representation.

%Ex Replaced by the locale's alternative date representation.

%EX Replaced by the locale's alternative time representation.

%Ey Replaced by the offset from %EC (year only) in the locale's al? ternative representation.

%EY Replaced by the full alternative year representation.

%Od Replaced by the day of the month, using the locale's alterna? tive numeric symbols, filled as needed with leading zeros if there is any alternative symbol for zero; otherwise, with lead? ing <space> characters.

%Oe Replaced by the day of the month, using the locale's alterna? tive numeric symbols, filled as needed with leading <space> characters.

%OH Replaced by the hour (24-hour clock) using the locale's alter? native numeric symbols.

%OI Replaced by the hour (12-hour clock) using the locale's alter? native numeric symbols.

%Om Replaced by the month using the locale's alternative numeric symbols.

%OM Replaced by the minutes using the locale's alternative numeric symbols.

%OS Replaced by the seconds using the locale's alternative numeric symbols.

%Ou Replaced by the weekday as a number in the locale's alternative representation (Monday=1).

%OU Replaced by the week number of the year (Sunday as the first day of the week, rules corresponding to %U) using the locale's alternative numeric symbols.

%OV Replaced by the week number of the year (Monday as the first day of the week, rules corresponding to %V) using the locale's alternative numeric symbols.

%Ow Replaced by the number of the weekday (Sunday=0) using the lo? cale's alternative numeric symbols.

%OW Replaced by the week number of the year (Monday as the first day of the week) using the locale's alternative numeric sym? bols.

%Oy Replaced by the year (offset from %C) using the locale's alter? native numeric symbols.

%g, %G, and %V give values according to the ISO 8601:2004 standard week-based year. In this system, weeks begin on a Monday and week 1 of the year is the week that includes January 4th, which is also the week that includes the first Thursday of the year, and is also the first week that contains at least four days in the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year; thus, for Saturday 2nd January 1999, %G is replaced by 1998 and %V is replaced by 53. If December 29th, 30th, or 31st is a Monday, it and any following days are part of week 1 of the following year. Thus, for Tuesday 30th December 1997, %G is re? placed by 1998 and %V is replaced by 01.

If a conversion specifier is not one of the above, the behavior is un? defined.

The behavior is undefined if the locale argument to strftime_l() is the special locale object LC_GLOBAL_LOCALE or is not a valid locale object handle.

RETURN VALUE

If the total number of resulting bytes including the terminating null

byte is not more than maxsize, these functions shall return the number of bytes placed into the array pointed to by s, not including the ter? minating NUL character. Otherwise, 0 shall be returned and the contents of the array are unspecified.

ERRORS

No errors are defined.

The following sections are informative.

EXAMPLES

Getting a Localized Date String

The following example first sets the locale to the user's default. The locale information will be used in the nl_langinfo() and strftime() functions. The nl_langinfo() function returns the localized date string which specifies how the date is laid out. The strftime() function takes this information and, using the tm structure for values, places the date and time information into datestring.

```
#include <time.h>
#include <locale.h>
#include <langinfo.h>
...
struct tm *tm;
char datestring[256];
...
setlocale (LC_ALL, "");
...
strftime (datestring, sizeof(datestring), nl_langinfo (D_T_FMT), tm);
...
```

APPLICATION USAGE

The range of values for %S is [00,60] rather than [00,59] to allow for the occasional leap second.

Some of the conversion specifications are duplicates of others. They are included for compatibility with nl_cxtime() and nl_ascxtime(), which were published in Issue 2.

The %C, %F, %G, and %Y format specifiers in strftime() always print

full values, but the strptime() %C, %F, and %Y format specifiers only scan two digits (assumed to be the first two digits of a four-digit year) for %C and four digits (assumed to be the entire (four-digit) year) for %F and %Y. This mimics the behavior of printf() and scanf(); that is:

```
printf("%2d", x = 1000);
```

prints "1000", but:

```
scanf(%2d", &x);
```

when given "1000" as input will only store 10 in x). Applications us? ing extended ranges of years must be sure that the number of digits specified for scanning years with strptime() matches the number of dig? its that will actually be present in the input stream. Historic imple? mentations of the %Y conversion specification (with no flags and no minimum field width) produced different output formats. Some always produced at least four digits (with 0 fill for years from 0 through 999) while others only produced the number of digits present in the year (with no fill and no padding). These two forms can be produced with the '0' flag and a minimum field width options using the conver? sions specifications %04Y and %01Y, respectively.

In the past, the C and POSIX standards specified that %F produced an ISO 8601:2004 standard date format, but didn't specify which one. For years in the range [0001,9999], POSIX.1?2008 requires that the output produced match the ISO 8601:2004 standard complete representation ex? tended format (YYYY-MM-DD) and for years outside of this range produce output that matches the ISO 8601:2004 standard expanded representation extended format (<+/-><Underline>Y</Underline>YYYY-MM-DD). To fully meet ISO 8601:2004 standard requirements, the producer and consumer must agree on a date format that has a specific number of bytes re? served to hold the characters used to represent the years that is suf? ficiently large to hold all values that will be shared. For example, the %+13F conversion specification will produce output matching the format "<+/->YYYYYY-MM-DD" (a leading '+' or '-' sign; a six-digit, 0-filled year; a '-'; a two-digit, leading 0-filled month; another '-';

and the two-digit, leading 0-filled day within the month).

Note that if the year being printed is greater than 9999, the resulting string from the unadorned %F conversion specifications will not conform to the ISO 8601:2004 standard extended format, complete representation for a date and will instead be an extended format, expanded representa? tion (presumably without the required agreement between the date's pro? ducer and consumer).

In the C or POSIX locale, the E and O modifiers are ignored and the re? placement strings for the following specifiers are:

%a The first three characters of %A.

%A One of Sunday, Monday, ..., Saturday.

%b The first three characters of %B.

%B One of January, February, ..., December.

%c Equivalent to %a %b %e %T %Y.

%p One of AM or PM.

%r Equivalent to %I:%M:%S %p.

%x Equivalent to %m/%d/%y.

%X Equivalent to %T.

%Z Implementation-defined.

RATIONALE

The %Y conversion specification to strftime() was frequently assumed to be a four-digit year, but the ISO C standard does not specify that %Y is restricted to any subset of allowed values from the tm_year field. Similarly, the %C conversion specification was assumed to be a two-digit field and the first part of the output from the %F conversion specification was assumed to be a four-digit field. With tm_year being a signed 32 or more-bit int and with many current implementations sup? porting 64-bit time_t types in one or more programming environments, these assumptions are clearly wrong.

POSIX.1?2008 now allows the format specifications %0xC, %0xF, %0xG, and %0xY (where 'x' is a string of decimal digits used to specify printing and scanning of a string of x decimal digits) with leading zero fill characters. Allowing applications to set the field width enables them

to agree on the number of digits to be printed and scanned in the ISO 8601:2004 standard expanded representation of a year (for %F, %G, and %Y) or all but the last two digits of the year (for %C). This is based on a feature in some versions of GNU libc's strftime(). The GNU version allows specifying space, zero, or no-fill characters in strf? time() format strings, but does not allow any flags to be specified in strptime() format strings. These implementations also allow these flags to be specified for any numeric field. POSIX.1?2008 only requires the zero fill flag ('0') and only requires that it be recognized when pro? cessing %C, %F, %G, and %Y specifications when a minimum field width is also specified. The '0' flag is the only flag needed to produce and scan the ISO 8601:2004 standard year fields using the extended format forms. POSIX.1?2008 also allows applications to specify the same flag and field width specifiers to be used in both strftime() and strptime() format strings for symmetry. Systems may provide other flag characters and may accept flags in conjunction with conversion specifiers other than %C, %F, %G, and %Y; but portable applications cannot depend on such extensions.

POSIX.1?2008 now also allows the format specifications %+xC, %+xF, %+xG, and %+xY (where 'x' is a string of decimal digits used to specify printing and scanning of a string of 'x' decimal digits) with leading zero fill characters and a leading '+' sign character if the year being converted is more than four digits or a minimum field width is speci? fied that allows room for more than four digits for the year. This al? lows date providers and consumers to agree on a specific number of dig? its to represent a year as required by the ISO 8601:2004 standard ex? panded representation formats. The expanded representation formats all require the year to begin with a leading '+' or '-' sign. (All of these specifiers can also provide a leading '-' sign for negative years. Since negative years and the year 0 don't fit well with the Gre? gorian or Julian calendars, the normal ranges of dates start with year 1. The ISO C standard allows tm_year to assume values corresponding to years before year 1, but the use of such years provided unspecified re?

sults.)

Some earlier version of this standard specified that applications want? ing to use strptime() to scan dates and times printed by strftime() should provide non-digit characters between fields to separate years from months and days. It also supported %F to print and scan the ISO 8601:2004 standard extended format, complete representation date for years 1 through 9999 (i.e., YYYY-MM-DD). However, many applications were written to print (using strftime()) and scan (using strptime()) dates written using the basic format complete representation (four-digit years) and truncated representation (two-digit years) specified by the ISO 8601:2004 standard representation of dates and times which do not have any separation characters between fields. The ISO 8601:2004 standard also specifies basic format expanded representation where the creator and consumer of these fields agree beforehand to represent years as leading zero-filled strings of an agreed length of more than four digits to represent a year (again with no separation characters when year, month, and day are all displayed). Applications producing and consuming expanded representations are encouraged to use the '+' flag and an appropriate maximum field width to scan the year including the leading sign. Note that even without the '+' flag, years less than zero may be represented with a leading <hyphen-minus> for %F, %G, and %Y conversion specifications. Using negative years results in unspeci? fied behavior.

If a format specification %+xF with the field width x greater than 11 is specified and the width is large enough to display the full year, the output string produced will match the ISO 8601:2004 standard sub? clause 4.1.2.4 expanded representation, extended format date represen? tation for a specific day. (For years in the range [1,99999], %+12F is sufficient for an agreed five-digit year with a leading sign using the ISO 8601:2004 standard expanded representation, extended format for a specific day "<+/->YYYYY-MM-DD".) Note also that years less than 0 may produce a leading <hyphen-minus> character ('-') when using %Y or %C whether or not the '0' or '+' flags are used.

The difference between the '0' flag and the '+' flag is whether the leading '+' character will be provided for years >9999 as required for the ISO 8601:2004 standard extended representation format containing a year. For example:

Year	Conversion Specification	Output	strftime()	strptime()
1970	%Y	1970	1970	
1970	%+4Y	1970	1970	
27	%Y	27 or 0027	27	
270	%Y	270 or 0270	270	
270	%+4Y	0270	270	
17	%C%y	0017	17	
270	%C%y	0270	270	
12345	%Y	12345	1234*	
12345	%+4Y	+12345	123*	
12345	%05Y	12345	12345	
270	%+5Y or %+3C%y	+0270	270	
12345	%+5Y or %+3C%y	+12345	1234*	
12345	%06Y or %04C%y	012345	12345	

??

?12345 ? %+6Y or %+4C%y ? +12345 ? 12345 ?

??

?123456 ? %08Y or %06C%y ? 00123456 ? 123456 ?

??

?123456 ? %+8Y or %+6C%y ? +0123456 ? 123456 ?

??

In the cases above marked with a * in the strptime() scan back field, the implied or specified number of characters scanned by strptime() was less than the number of characters output by strftime() using the same format; so the remaining digits of the year were dropped when the out? put date produced by strftime() was scanned back in by strptime().

FUTURE DIRECTIONS

None.

SEE ALSO

asctime(), clock(), ctime(), difftime(), getdate(), gmtime(), local? time(), mktime(), strptime(), time(), tzset(), uselocale(), utime()

The Base Definitions volume of POSIX.1?2017, Section 7.3.5, LC_TIME, <time.h>

COPYRIGHT