Rocky Enterprise Linux 9.2 Manual Pages on command 'strftime.3'

$ man strftime.3

STRFTIME(3) Linux Programmer's Manual STRFTIME(3)

NAME

strftime - format date and time

SYNOPSIS

#include <time.h>

size_t strftime(char *s, size_t max, const char *format,

const struct tm *tm);

DESCRIPTION

The strftime() function formats the broken-down time tm according to the format specification format and places the result in the character array s of size max. The broken-down time structure tm is defined in <time.h>. See also ctime(3).

The format specification is a null-terminated string and may contain special character sequences called conversion specifications, each of which is introduced by a '%' character and terminated by some other character known as a conversion specifier character. All other charac? ter sequences are ordinary character sequences.

The characters of ordinary character sequences (including the null

byte) are copied verbatim from format to s. However, the characters of conversion specifications are replaced as shown in the list below. In this list, the field(s) employed from the tm structure are also shown.

%a The abbreviated name of the day of the week according to the current locale. (Calculated from tm_wday.) (The specific names used in the current locale can be obtained by calling nl_lang? info(3) with ABDAY_{1?7} as an argument.)

%A The full name of the day of the week according to the current locale. (Calculated from tm_wday.) (The specific names used in the current locale can be obtained by calling nl_langinfo(3) with DAY_{1?7} as an argument.)

%b The abbreviated month name according to the current locale. (Calculated from tm_mon.) (The specific names used in the cur? rent locale can be obtained by calling nl_langinfo(3) with AB? MON_{1?12} as an argument.)

%B The full month name according to the current locale. (Calcu? lated from tm_mon.) (The specific names used in the current lo? cale can be obtained by calling nl_langinfo(3) with MON_{1?12} as an argument.)

%c The preferred date and time representation for the current lo? cale. (The specific format used in the current locale can be obtained by calling nl_langinfo(3) with D_T_FMT as an argument for the %c conversion specification, and with ERA_D_T_FMT for the %Ec conversion specification.) (In the POSIX locale this is equivalent to %a %b %e %H:%M:%S %Y.)

%C The century number (year/100) as a 2-digit integer. (SU) (The %EC conversion specification corresponds to the name of the era.) (Calculated from tm_year.)

%d The day of the month as a decimal number (range 01 to 31). (Calculated from tm_mday.)

%D Equivalent to %m/%d/%y. (Yecch?for Americans only. Americans should note that in other countries %d/%m/%y is rather common. This means that in international context this format is ambigu?

ous and should not be used.) (SU)

%e Like %d, the day of the month as a decimal number, but a leading zero is replaced by a space. (SU) (Calculated from tm_mday.)

%E Modifier: use alternative ("era-based") format, see below. (SU)

%F Equivalent to %Y-%m-%d (the ISO 8601 date format). (C99)

%G The ISO 8601 week-based year (see NOTES) with century as a decimal number. The 4-digit year corresponding to the ISO week number (see %V). This has the same format and value as %Y, except that if the ISO week number belongs to the previous or next year, that year is used instead. (TZ) (Calculated from tm_year, tm_yday, and tm_wday.)

%g Like %G, but without century, that is, with a 2-digit year (00?99). (TZ) (Calculated from tm_year, tm_yday, and tm_wday.)

%h Equivalent to %b. (SU)

%H The hour as a decimal number using a 24-hour clock (range 00 to 23). (Calculated from tm_hour.)

%I The hour as a decimal number using a 12-hour clock (range 01 to 12). (Calculated from tm_hour.)

%j The day of the year as a decimal number (range 001 to 366). (Calculated from tm_yday.)

%k The hour (24-hour clock) as a decimal number (range 0 to 23); single digits are preceded by a blank. (See also %H.) (Calculated from tm_hour.) (TZ)

%l The hour (12-hour clock) as a decimal number (range 1 to 12); single digits are preceded by a blank. (See also %I.) (Calculated from tm_hour.) (TZ)

%m The month as a decimal number (range 01 to 12). (Calculated from tm_mon.)

%M The minute as a decimal number (range 00 to 59). (Calculated from tm_min.)

%n A newline character. (SU)

%O Modifier: use alternative numeric symbols, see below. (SU)

%p Either "AM" or "PM" according to the given time value, or the

corresponding strings for the current locale. Noon is treated as "PM" and midnight as "AM". (Calculated from tm_hour.) (The specific string representations used for "AM" and "PM" in the current locale can be obtained by calling nl_langinfo(3) with AM_STR and PM_STR, respectively.)

%P Like %p but in lowercase: "am" or "pm" or a corresponding string for the current locale. (Calculated from tm_hour.) (GNU)

%r The time in a.m. or p.m. notation. (SU) (The specific format used in the current locale can be obtained by calling nl_lang? info(3) with T_FMT_AMPM as an argument.) (In the POSIX locale this is equivalent to %I:%M:%S %p.)

%R The time in 24-hour notation (%H:%M). (SU) For a version in? cluding the seconds, see %T below.

%s The number of seconds since the Epoch, 1970-01-01 00:00:00 +0000 (UTC). (TZ) (Calculated from mktime(tm).)

%S The second as a decimal number (range 00 to 60). (The range is up to 60 to allow for occasional leap seconds.) (Calculated from tm_sec.)

%t A tab character. (SU)

%T The time in 24-hour notation (%H:%M:%S). (SU)

%u The day of the week as a decimal, range 1 to 7, Monday being 1. See also %w. (Calculated from tm_wday.) (SU)

%U The week number of the current year as a decimal number, range 00 to 53, starting with the first Sunday as the first day of week 01. See also %V and %W. (Calculated from tm_yday and tm_wday.)

%V The ISO 8601 week number (see NOTES) of the current year as a decimal number, range 01 to 53, where week 1 is the first week that has at least 4 days in the new year. See also %U and %W. (Calculated from tm_year, tm_yday, and tm_wday.) (SU)

%w The day of the week as a decimal, range 0 to 6, Sunday being 0. See also %u. (Calculated from tm_wday.)

%W The week number of the current year as a decimal number, range

00 to 53, starting with the first Monday as the first day of week 01. (Calculated from tm_yday and tm_wday.)

%x The preferred date representation for the current locale without the time. (The specific format used in the current locale can be obtained by calling nl_langinfo(3) with D_FMT as an argument for the %x conversion specification, and with ERA_D_FMT for the %Ex conversion specification.) (In the POSIX locale this is equivalent to %m/%d/%y.)

%X The preferred time representation for the current locale without the date. (The specific format used in the current locale can be obtained by calling nl_langinfo(3) with T_FMT as an argument for the %X conversion specification, and with ERA_T_FMT for the %EX conversion specification.) (In the POSIX locale this is equivalent to %H:%M:%S.)

%y The year as a decimal number without a century (range 00 to 99). (The %Ey conversion specification corresponds to the year since the beginning of the era denoted by the %EC conversion specifi? cation.) (Calculated from tm_year)

%Y The year as a decimal number including the century. (The %EY conversion specification corresponds to the full alternative year representation.) (Calculated from tm_year)

%z The +hhmm or -hhmm numeric timezone (that is, the hour and minute offset from UTC). (SU)

%Z The timezone name or abbreviation.

%+ The date and time in date(1) format. (TZ) (Not supported in glibc2.)

%% A literal '%' character.

Some conversion specifications can be modified by preceding the conver? sion specifier character by the E or O modifier to indicate that an al? ternative format should be used. If the alternative format or specifi? cation does not exist for the current locale, the behavior will be as if the unmodified conversion specification were used. (SU) The Single UNIX Specification mentions %Ec, %EC, %Ex, %EX, %Ey, %EY, %Od, %Oe,

%OH, %OI, %Om, %OM, %OS, %Ou, %OU, %OV, %Ow, %OW, %Oy, where the effect of the O modifier is to use alternative numeric symbols (say, roman nu? merals), and that of the E modifier is to use a locale-dependent alter? native representation. The rules governing date representation with the E modifier can be obtained by supplying ERA as an argument to a nl_langinfo(3). One example of such alternative forms is the Japanese era calendar scheme in the ja_JP glibc locale.

RETURN VALUE

Provided that the result string, including the terminating null byte, does not exceed max bytes, strftime() returns the number of bytes (ex? cluding the terminating null byte) placed in the array s. If the length of the result string (including the terminating null byte) would exceed max bytes, then strftime() returns 0, and the contents of the array are undefined.

Note that the return value 0 does not necessarily indicate an error. For example, in many locales %p yields an empty string. An empty for? mat string will likewise yield an empty string.

ENVIRONMENT

The environment variables TZ and LC_TIME are used.

ATTRIBUTES

For an explanation of the terms used in this section, see at? tributes(7).

??

?Interface ? Attribute ? Value ?

??

?strftime() ? Thread safety ? MT-Safe env locale ?

??

CONFORMING TO

SVr4, C89, C99. There are strict inclusions between the set of conver? sions given in ANSI C (unmarked), those given in the Single UNIX Speci? fication (marked SU), those given in Olson's timezone package (marked TZ), and those given in glibc (marked GNU), except that %+ is not sup? ported in glibc2. On the other hand glibc2 has several more exten?

sions. POSIX.1 only refers to ANSI C; POSIX.2 describes under date(1) several extensions that could apply to strftime() as well. The %F con? version is in C99 and POSIX.1-2001.

In SUSv2, the %S specifier allowed a range of 00 to 61, to allow for the theoretical possibility of a minute that included a double leap second (there never has been such a minute).

NOTES

ISO 8601 week dates

%G, %g, and %V yield values calculated from the week-based year defined by the ISO 8601 standard. In this system, weeks start on a Monday, and are numbered from 01, for the first week, up to 52 or 53, for the last week. Week 1 is the first week where four or more days fall within the new year (or, synonymously, week 01 is: the first week of the year that contains a Thursday; or, the week that has 4 January in it). When three or fewer days of the first calendar week of the new year fall within that year, then the ISO 8601 week-based system counts those days as part of week 52 or 53 of the preceding year. For example, 1 January 2010 is a Friday, meaning that just three days of that calendar week fall in 2010. Thus, the ISO 8601 week-based system considers these days to be part of week 53 (%V) of the year 2009 (%G); week 01 of ISO 8601 year 2010 starts on Monday, 4 January 2010. Similarly, the first two days of January 2011 are considered to be part of week 52 of the year 2010.

Glibc notes

Glibc provides some extensions for conversion specifications. (These extensions are not specified in POSIX.1-2001, but a few other systems provide similar features.) Between the '%' character and the conver? sion specifier character, an optional flag and field width may be spec? ified. (These precede the E or O modifiers, if present.)

The following flag characters are permitted:

_ (underscore) Pad a numeric result string with spaces.

- (dash) Do not pad a numeric result string.

0 Pad a numeric result string with zeros even if the conversion

specifier character uses space-padding by default.

^ Convert alphabetic characters in result string to uppercase.

Swap the case of the result string. (This flag works only with certain conversion specifier characters, and of these, it is only really useful with %Z.)

An optional decimal width specifier may follow the (possibly absent) flag. If the natural size of the field is smaller than this width, then the result string is padded (on the left) to the specified width.

BUGS

If the output string would exceed max bytes, errno is not set. This makes it impossible to distinguish this error case from cases where the format string legitimately produces a zero-length output string. POSIX.1-2001 does not specify any errno settings for strftime().

Some buggy versions of gcc(1) complain about the use of %c: warning: `%c' yields only last 2 digits of year in some locales. Of course pro? grammers are encouraged to use %c, as it gives the preferred date and time representation. One meets all kinds of strange obfuscations to circumvent this gcc(1) problem. A relatively clean one is to add an intermediate function

```
size_t
my_strftime(char *s, size_t max, const char *fmt,
        const struct tm *tm)
{
    return strftime(s, max, fmt, tm);
}
```

Nowadays, gcc(1) provides the -Wno-format-y2k option to prevent the warning, so that the above workaround is no longer required.

EXAMPLES

RFC 2822-compliant date format (with an English locale for %a and %b)

```
"%a, %d %b %Y %T %z"
```

RFC 822-compliant date format (with an English locale for %a and %b)

```
"%a, %d %b %y %T %z"
```

The program below can be used to experiment with strftime().

Some examples of the result string produced by the glibc implementation of strftime() are as follows:

```
$ ./a.out '%m'
Result string is "11"
$ ./a.out '%5m'
Result string is "00011"
$ ./a.out '%_5m'
Result string is "    11"
```

Program source

```
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
int
main(int argc, char *argv[])
{
    char outstr[200];
    time_t t;
    struct tm *tmp;
    t = time(NULL);
    tmp = localtime(&t);
    if (tmp == NULL) {
        perror("localtime");
        exit(EXIT_FAILURE);
    }
    if (strftime(outstr, sizeof(outstr), argv[1], tmp) == 0) {
        fprintf(stderr, "strftime returned 0");
        exit(EXIT_FAILURE);
    }
    printf("Result string is \"%s\"\n", outstr);
    exit(EXIT_SUCCESS);
}
```

date(1), time(2), ctime(3), nl_langinfo(3), setlocale(3), sprintf(3), strptime(3)

COLOPHON

This page is part of release 5.10 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man-pages/.