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*.

Ordinary characters placed in the format string are copied to *s* without conversion. *Conversion specifications* are introduced by a '%' character, and terminated by a *conversion specifier character*, and are replaced in *s* as follows:

%a The abbreviated weekday name according to the current locale.

%A The full weekday name according to the current locale.

%b The abbreviated month name according to the current locale.

%B The full month name according to the current locale.

%c The preferred date and time representation for the current locale.

%C The century number (year/100) as a 2-digit integer. (SU)

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

%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 ambiguous 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)

%E Modifier: use alternative format, see below. (SU)

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

%G The ISO 8601 year 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)

%g Like %G, but without century, i.e., with a 2-digit year (00-99). (TZ)

%h Equivalent to %b. (SU)

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

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

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

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

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

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

%n A newline character. (SU)

%O Modifier: use alternative format, 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'.

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

%r The time in a.m. or p.m. notation. In the POSIX locale this is equivalent to '%I:%M:%S %p'. (SU)

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

%s The number of seconds since the Epoch, i.e., since 1970-01-01 00:00:00 UTC. (TZ)

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

%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. (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.

%V The ISO 8601:1988 week number 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 current year, and with Monday as the first day of the week. See also %U and %W. (SU)

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

%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.

%x The preferred date representation for the current locale without the time.

%X The preferred time representation for the current locale without the date.

%y The year as a decimal number without a century (range 00 to 99).

%Y The year as a decimal number including the century.

%z The time-zone as hour offset from GMT. Required to emit RFC 822-conformant dates (using "%a, %d %b %Y %H:%M:%S %z"). (GNU)

%Z The time zone or 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 conversion specifier character by the E or O *modifier* to indicate that an alternative format should be used. If the alternative format or specification does not exist for the current locale, the behaviour 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 numerals), and that of the E modifier is to use a locale-dependent alternative representation.

The broken-down time structure *tm* is defined in <*time.h*>. See also **ctime**(3).

RETURN VALUE

The **strftime**() function returns the number of characters placed in the array *s*, not including the terminating null byte, provided the string, including the terminating null byte, fits. Otherwise, it returns 0, and the contents of the array is undefined. (Thus at least since libc 4.4.4; very old versions of libc, such as libc 4.4.1, would return *max* if the array was too small.)

Note that the return value 0 does not necessarily indicate an error; for example, in many locales %p yields an empty string.

ENVIRONMENT

The environment variables TZ and LC_TIME are used.

CONFORMING TO

SVr4, C89, C99. There are strict inclusions between the set of conversions given in ANSI C (unmarked), those given in the Single Unix Specification (marked SU), those given in Olson's timezone package (marked TZ), and those given in glibc (marked GNU), except that %+ is not supported in glibc2. On the other hand glibc2 has several more extensions. POSIX.1 only refers to ANSI C; POSIX.2 describes under **date**(1) several extensions that could apply to **strftime**() as well. The %F conversion is in C99 and POSIX.1-2001.

In SUSv2, the %S specified 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).

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 conversion specifier character, an optional *flag* and field *width* may be specified. (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 upper case.

Swap the case of the result string. (This flag only works 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

Some buggy versions of gcc complain about the use of %c: *warning: '%c' yields only last 2 digits of year in some locales*. Of course programmers are encouraged to use %c, it gives the preferred date and time representation. One meets all kinds of strange obfuscations to circumvent this gcc 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);
}
```

EXAMPLE

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

```
#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);
} /* main */
```

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"
```

SEE ALSO

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