NAME

strftime, cftime, ascftime – convert date and time to string

SYNOPSIS

#include <time.h>

size_t strftime(const char **s*, **size_t** *maxsize*, **const char** **format*,
const struct tm **timeptr*);

int cftime(char **s*, **char** **format*, **const time_t** **clock*);

int ascftime(char **s*, **const char** **format*, **const struct tm** **timeptr*);

MT-LEVEL

MT-Safe

DESCRIPTION

strftime( ), **ascftime( )**, and **cftime( )** place bytes into the array pointed to by *s* as controlled by the string pointed to by *format*. The *format* string consists of zero or more conversion specifications and ordinary characters. A conversion specification consists of a '%' (percent) character and one or two terminating conversion characters that determine the conversion specification's behavior. All ordinary characters (including the terminating null byte) are copied unchanged into the array pointed to by *s*. If copying takes place between objects that overlap, the behavior is undefined. For **strftime( )**, no more than *maxsize* bytes are placed into the array.

If *format* is **(char *)0**, then the locale's default format is used. For **strftime( )** the default format is the same as **%c**; for **cftime( )** and **ascftime( )** the default format is the same as **%C**. **cftime( )** and **ascftime( )** first try to use the value of the environment variable **CFTIME**, and if that is undefined or empty, the default format is used.

Each conversion specification is replaced by appropriate characters as described in the following list. The appropriate characters are determined by the **LC_TIME** category of the program's locale and by the values contained in the structure pointed to by *timeptr* for **strftime( )** and **ascftime( )**, and by the time represented by *clock* for **cftime( )**.

	%%	same as %
	%a	locale's abbreviated weekday name
	%A	locale's full weekday name
	%b	locale's abbreviated month name
	%B	locale's full month name
	%c	locale's appropriate date and time representation
Solaris	**%C**	locale's date and time representation as produced by **date**(1)
XPG4	**%C**	century number (the year divided by 100 and truncated to an integer as a decimal number [1,99]); single digits are preceded by **0**
	%d	day of month [1,31]; single digits are preceded by **0**
	%D	date as **%m/%d/%y**
	%e	day of month [1,31]; single digits are preceded by a space
	%h	locale's abbreviated month name
	%H	hour (24-hour clock) [0,23]; single digits are preceded by **0**
	%I	hour (12-hour clock) [1,12]; single digits are preceded by **0**
	%j	day number of year [1,366]; single digits are preceded by **0**
	%k	hour (24-hour clock) [0,23]; single digits are preceded by a blank

%l	hour (12-hour clock) [1,12]; single digits are preceded by a blank
%m	month number [1,12]; single digits are preceded by **0**
%M	minute [00,59]; leading zero is permitted but not required
%n	insert a newline
%p	locale's equivalent of either a.m. or p.m.
%r	appropriate time representation in 12-hour clock format with **%p**
%R	time as **%H**:**%M**
%S	seconds [00,61]
%t	insert a tab
%T	time as **%H**:**%M**:**%S**
%u	weekday as a decimal number [1,7], with 1 representing Sunday
%U	week number of year as a decimal number [00,53], with Sunday as the first day of week 1
%V	week number of the year as a decimal number [01,53], with Monday as the first day of the week. If the week containing 1 January has four or more days in the new year, then it is considered week 1; otherwise, it is week 53 of the previous year, and the next week is week 1.
%w	weekday as a decimal number [0,6], with 0 representing Sunday
%W	week number of year as a decimal number [00,53], with Monday as the first day of week 1
%x	locale's appropriate date representation
%X	locale's appropriate time representation
%y	year within century [00,99]
%Y	year, including the century (for example 1993)
%Z	time zone name or abbreviation, or no bytes if no time zone information exists

If a conversion specification does not correspond to any of the above or to any of the modified conversion specifications listed below, the behavior is undefined and **0** is returned.

The difference between **%U** and **%W** (and also between modified conversion specifications **%OU** and **%OW**) lies in which day is counted as the first of the week. Week number 1 is the first week in January starting with a Sunday for **%U** or a Monday for **%W**. Week number 0 contains those days before the first Sunday or Monday in January for **%U** and **%W**, respectively.

Modified Conversion Specifications

Some conversion specifications can be modified by the **E** and **O** modifiers to indicate that an alternate format or specification should be used rather than the one normally used by the unmodified conversion specification. If the alternate format or specification does not exist in the current locale, the behavior will be as if the unmodified specification were used.

%Ec	locale's alternate appropriate date and time representation
%EC	name of the base year (period) in the locale's alternate representation
%Ex	locale's alternate date representation
%EX	locale's alternate time representation
%Ey	offset from **%EC** (year only) in the locale's alternate representation

- **%EY** full alternate year representation
- **%Od** day of the month using the locale's alternate numeric symbols
- **%Oe** same as **%Od**
- **%OH** hour (24-hour clock) using the locale's alternate numeric symbols
- **%OI** hour (12-hour clock) using the locale's alternate numeric symbols
- **%Om** month using the locale's alternate numeric symbols
- **%OM** minutes using the locale's alternate numeric symbols
- **%OS** seconds using the locale's alternate numeric symbols
- **%OU** week number of the year (Sunday as the first day of the week) using the locale's alternate numeric symbols
- **%Ow** number of the weekday (Sunday=0) using the locale's alternate numeric symbols
- **%OW** week number of the year (Monday as the first day of the week) using the locale's alternate numeric symbols
- **%Oy** year (offset from **%C**) in the locale's alternate representation and using the locale's alternate numeric symbols

Selecting the Output Language

By default, the output of **strftime( )**, **cftime( )**, and **ascftime( )** appear in U.S. English. The user can request that the output of **strftime( )**, **cftime( )**, or **ascftime( )** be in a specific language by setting the **LC_TIME** category using **setlocale( )**.

Time Zone

Local time zone information is used as though **tzset**(3C) were called.

RETURN VALUES

strftime( ), **cftime( )**, and **ascftime( )** return the number of characters placed into the array pointed to by *s*, not including the terminating null character. If the total number of resulting characters including the terminating null character is more than *maxsize*, **strftime()** returns **0** and the contents of the array are indeterminate.

EXAMPLES

The following example illustrates the use of **strftime( )**. It shows what the string in *str* would look like if the structure pointed to by *tmptr* contains the values corresponding to Thursday, August 28, 1986 at 12:44:36.

strftime (str, strsize, "%A %b %d %j", tmptr)

This results in *str* containing "Thursday Aug 28 240".

FILES

/usr/lib/locale/*locale***/LC_TIME/time** locale specific date and time information

SEE ALSO

date(1), **ctime**(3C), **mktime**(3C), **setlocale**(3C), **strptime**(3C), **tzset**(3C), **TIMEZONE**(4), **strftime**(4), **environ**(5), **xpg4**(5)

NOTES

The range of values for **%S** is [00,61] rather than [00,59] to allow for the occasional leap second and even more occasional double leap second.