|  |  |
| --- | --- |
| strptime {base} | R Documentation |

**Date-time Conversion Functions to and from Character**

**Description**

Functions to convert between character representations and objects of classes "POSIXlt" and "POSIXct" representing calendar dates and times.

**Usage**

## S3 method for class 'POSIXct'

format(x, format = "", tz = "", usetz = FALSE, ...)

## S3 method for class 'POSIXlt'

format(x, format = "", usetz = FALSE, ...)

## S3 method for class 'POSIXt'

as.character(x, ...)

strftime(x, format = "", tz = "", usetz = FALSE, ...)

strptime(x, format, tz = "")

**Arguments**

|  |  |
| --- | --- |
| x | An object to be converted: a character vector for strptime, an object which can be converted to "[POSIXlt](http://127.0.0.1:14695/library/base/help/POSIXlt)" for strftime. |
| tz | A character string specifying the time zone to be used for the conversion. System-specific (see [as.POSIXlt](http://127.0.0.1:14695/library/base/help/as.POSIXlt)), but "" is the current time zone, and "GMT" is UTC. Invalid values are most commonly treated as UTC, on some platforms with a warning. |
| format | A character string. The default for the format methods is "%Y-%m-%d %H:%M:%S" if any element has a time component which is not midnight, and "%Y-%m-%d" otherwise. If [options](http://127.0.0.1:14695/library/base/help/options)("digits.secs") is set, up to the specified number of digits will be printed for seconds. |
| ... | Further arguments to be passed from or to other methods. |
| usetz | logical. Should the time zone abbreviation be appended to the output? This is used in printing times, and more reliable than using "%Z". |

**Details**

The format and as.character methods and strftime convert objects from the classes "[POSIXlt](http://127.0.0.1:14695/library/base/help/POSIXlt)" and "[POSIXct](http://127.0.0.1:14695/library/base/help/POSIXct)" to character vectors.

strptime converts character vectors to class "POSIXlt": its input x is first converted by [as.character](http://127.0.0.1:14695/library/base/help/as.character). Each input string is processed as far as necessary for the format specified: any trailing characters are ignored.

strftime is a wrapper for format.POSIXlt, and it and format.POSIXct first convert to class "POSIXlt" by calling [as.POSIXlt](http://127.0.0.1:14695/library/base/help/as.POSIXlt) (so they also work for class "[Date](http://127.0.0.1:14695/library/base/help/Date)"). Note that only that conversion depends on the time zone.

The usual vector re-cycling rules are applied to x and format so the answer will be of length of the longer of these vectors.

Locale-specific conversions to and from character strings are used where appropriate and available. This affects the names of the days and months, the AM/PM indicator (if used) and the separators in formats such as %x and %X, via the setting of the [LC\_TIME](http://127.0.0.1:14695/library/base/help/LC_TIME) locale category. The ‘current locale’ of the descriptions might mean the locale in use at the start of the **R** session or when these functions are first used.

The details of the formats are platform-specific, but the following are likely to be widely available: most are defined by the POSIX standard. A *conversion specification* is introduced by %, usually followed by a single letter or O or E and then a single letter. Any character in the format string not part of a conversion specification is interpreted literally (and %% gives %). Widely implemented conversion specifications include

%a

Abbreviated weekday name in the current locale on this platform. (Also matches full name on input: in some locales there are no abbreviations of names.)

%A

Full weekday name in the current locale. (Also matches abbreviated name on input.)

%b

Abbreviated month name in the current locale on this platform. (Also matches full name on input: in some locales there are no abbreviations of names.)

%B

Full month name in the current locale. (Also matches abbreviated name on input.)

%c

Date and time. Locale-specific on output, "%a %b %e %H:%M:%S %Y" on input.

%C

Century (00–99): the integer part of the year divided by 100.

%d

Day of the month as decimal number (01–31).

%D

Date format such as %m/%d/%y: the C99 standard says it should be that exact format (but not all OSes comply).

%e

Day of the month as decimal number (1–31), with a leading space for a single-digit number.

%F

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

%g

The last two digits of the week-based year (see %V). (Accepted but ignored on input.)

%G

The week-based year (see %V) as a decimal number. (Accepted but ignored on input.)

%h

Equivalent to %b.

%H

Hours as decimal number (00–23). As a special exception strings such as 24:00:00 are accepted for input, since ISO 8601 allows these.

%I

Hours as decimal number (01–12).

%j

Day of year as decimal number (001–366).

%m

Month as decimal number (01–12).

%M

Minute as decimal number (00–59).

%n

Newline on output, arbitrary whitespace on input.

%p

AM/PM indicator in the locale. Used in conjunction with %I and **not** with %H. An empty string in some locales (and the behaviour is undefined if used for input in such a locale).

Some platforms accept %P for output, which uses a lower-case version: others will output P.

%r

The 12-hour clock time (using the locale's AM or PM). Only defined in some locales.

%R

Equivalent to %H:%M.

%S

Second as integer (00–61), allowing for up to two leap-seconds (but POSIX-compliant implementations will ignore leap seconds).

%t

Tab on output, arbitrary whitespace on input.

%T

Equivalent to %H:%M:%S.

%u

Weekday as a decimal number (1–7, Monday is 1).

%U

Week of the year as decimal number (00–53) using Sunday as the first day 1 of the week (and typically with the first Sunday of the year as day 1 of week 1). The US convention.

%V

Week of the year as decimal number (01–53) as defined in ISO 8601. If the week (starting on Monday) containing 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. (Accepted but ignored on input.)

%w

Weekday as decimal number (0–6, Sunday is 0).

%W

Week of the year as decimal number (00–53) using Monday as the first day of week (and typically with the first Monday of the year as day 1 of week 1). The UK convention.

%x

Date. Locale-specific on output, "%y/%m/%d" on input.

%X

Time. Locale-specific on output, "%H:%M:%S" on input.

%y

Year without century (00–99). On input, values 00 to 68 are prefixed by 20 and 69 to 99 by 19 – that is the behaviour specified by the 2004 and 2008 POSIX standards, but they do also say ‘it is expected that in a future version the default century inferred from a 2-digit year will change’.

%Y

Year with century. Note that whereas there was no zero in the original Gregorian calendar, ISO 8601:2004 defines it to be valid (interpreted as 1BC): see <https://en.wikipedia.org/wiki/0_(year)>. Note that the standards also say that years before 1582 in its calendar should only be used with agreement of the parties involved.

For input, only years 0:9999 are accepted.

%z

Signed offset in hours and minutes from UTC, so -0800 is 8 hours behind UTC. Values up to +1400 are accepted as from **R** 3.1.1: previous versions only accepted up to +1200. (Standard only for output.)

%Z

(Output only.) Time zone abbreviation as a character string (empty if not available). This may not be reliable when a time zone has changed abbreviations over the years.

Where leading zeros are shown they will be used on output but are optional on input. Names are matched case-insensitively on input: whether they are capitalized on output depends on the platform and the locale. Note that abbreviated names are platform-specific (although the standards specify that in the C locale they must be the first three letters of the capitalized English name: this convention is widely used in English-language locales but for example the French month abbreviations are not the same on any two of Linux, macOS, Solaris and Windows). Knowing what the abbreviations are is essential if you wish to use %a, %b or %h as part of an input format: see the examples for how to check.

When %z or %Z is used for output with an object with an assigned time zone an attempt is made to use the values for that time zone — but it is not guaranteed to succeed.

Not in the standards and less widely implemented are

%k

The 24-hour clock time with single digits preceded by a blank.

%l

The 12-hour clock time with single digits preceded by a blank.

%s

(Output only.) The number of seconds since the epoch.

%+

(Output only.) Similar to %c, often "%a %b %e %H:%M:%S %Z %Y". May depend on the locale.

For output there are also %O[dHImMUVwWy] which may emit numbers in an alternative locale-dependent format (e.g., roman numerals), and %E[cCyYxX] which can use an alternative ‘era’ (e.g., a different religious calendar). Which of these are supported is OS-dependent. These are accepted for input, but with the standard interpretation.

Specific to **R** is %OSn, which for output gives the seconds truncated to 0 <= n <= 6 decimal places (and if %OS is not followed by a digit, it uses the setting of [getOption](http://127.0.0.1:14695/library/base/help/getOption)("digits.secs"), or if that is unset, n = 0). Further, for strptime %OS will input seconds including fractional seconds. Note that %S does not read fractional parts on output.

The behaviour of other conversion specifications (and even if other character sequences commencing with % *are* conversion specifications) is system-specific. Some systems document that the use of multi-byte characters in format is unsupported: UTF-8 locales are unlikely to cause a problem.

**Value**

The format methods and strftime return character vectors representing the time. NA times are returned as NA\_character\_. The elements are restricted to 256 bytes, plus a time zone abbreviation if usetz is true. (On known platforms longer strings are truncated at 255 or 256 bytes, but this is not guaranteed by the C99 standard.)

strptime turns character representations into an object of class "[POSIXlt](http://127.0.0.1:14695/library/base/help/POSIXlt)". The time zone is used to set the isdst component and to set the "tzone" attribute if tz != "". If the specified time is invalid (for example "2010-02-30 08:00") all the components of the result are NA. (NB: this does means exactly what it says – if it is an invalid time, not just a time that does not exist in some time zone.)

**Printing years**

Everyone agrees that years from 1000 to 9999 should be printed with 4 digits, but the standards do not define what is to be done outside that range. For years 0 to 999 most OSes pad with zeros or spaces to 4 characters, and Linux outputs just the number.

OS facilities will probably not print years before 1 CE (aka 1 AD) ‘correctly’ (they tend to assume the existence of a year 0: see <https://en.wikipedia.org/wiki/0_(year)>, and some OSes get them completely wrong). Common formats are -45 and -045.

Years after 9999 and before -999 are normally printed with five or more characters.

Some platforms support modifiers from POSIX 2008 (and others). On Linux the format "%04Y" assures a minimum of four characters and zero-padding. The internal code (as used on Windows and by default on macOS) uses zero-padding by default, and formats %\_4Y and %\_Y can be used for space padding and no padding.

**Time zone offsets**

Offsets from GMT (also known as UTC) are part of the conversion between timezones and to/from class "POSIXct", but cause difficulties as they often computed incorrectly.

They conventionally have the opposite sign from time-zone specifications (see [Sys.timezone](http://127.0.0.1:14695/library/base/help/Sys.timezone)): positive values are East of the meridian. Although there have been time zones with offsets like 00:09:21 (Paris in 1900), and 00:44:30 (Liberia until 1972), offsets are usually treated as whole numbers of minutes, and are most often seen in RFC 822 email headers in forms like -0800 (e.g., used on the Pacific coast of the US in winter).

Format %z can be used for input or output: it is a character string, conventionally plus or minus followed by two digits for hours and two for minutes: the standards say that an empty string should be output if the offset is unknown, but some systems use the offsets for the time zone in use for the current year.

**Note**

The default formats follow the rules of the ISO 8601 international standard which expresses a day as "2001-02-28" and a time as "14:01:02" using leading zeroes as here. (The ISO form uses no space to separate dates and times: **R** does by default.)

For strptime the input string need not specify the date completely: it is assumed that unspecified seconds, minutes or hours are zero, and an unspecified year, month or day is the current one. (However, if a month is specified, the day of that month has to be specified by %d or %e since the current day of the month need not be valid for the specified month.) Some components may be returned as NA (but an unknown tzone component is represented by an empty string).

If the time zone specified is invalid on your system, what happens is system-specific but it will probably be ignored.

Remember that in most time zones some times do not occur and some occur twice because of transitions to/from ‘daylight saving’ (also known as ‘summer’) time. strptime does not validate such times (it does not assume a specific time zone), but conversion by [as.POSIXct](http://127.0.0.1:14695/library/base/help/as.POSIXct) will do so. Conversion by strftime and formatting/printing uses OS facilities and may return nonsensical results for non-existent times at DST transitions.

Much less comprehensive support for output specifications was provided on Windows before **R** 3.1.0.

In a C locale %c is required to be "%a %b %e %H:%M:%S %Y". As Windows does not comply (and uses a date format not understood outside N. America), that format is used by **R** on Windows in all locales.

That %A %a %B %b on input match both full and abbreviated names caused problems in some locales prior to **R** 3.0.3: e.g. in French on macOS juillet was matched by jui, the abbreviation for June.

**References**

International Organization for Standardization (2004, 2000, ...) *ISO 8601. Data elements and interchange formats – Information interchange – Representation of dates and times.* For links to versions available on-line see (at the time of writing) <http://dotat.at/tmp/ISO_8601-2004_E.pdf> and <http://www.qsl.net/g1smd/isopdf.htm>; for information on the current official version, see <http://www.iso.org/iso/iso8601>.

The POSIX 1003.1 standard, which is in some respects stricter than ISO 8601.

**See Also**

[DateTimeClasses](http://127.0.0.1:14695/library/base/help/DateTimeClasses) for details of the date-time classes; [locales](http://127.0.0.1:14695/library/base/help/locales) to query or set a locale.

Your system's help page on strftime to see how to specify their formats. (On some systems, including Windows, strftime is replaced by more comprehensive internal code.)

**Examples**

## locale-specific version of date()

format(Sys.time(), "%a %b %d %X %Y %Z")

## time to sub-second accuracy (if supported by the OS)

format(Sys.time(), "%H:%M:%OS3")

## read in date info in format 'ddmmmyyyy'

## This will give NA(s) in some locales; setting the C locale

## as in the commented lines will overcome this on most systems.

## lct <- Sys.getlocale("LC\_TIME"); Sys.setlocale("LC\_TIME", "C")

x <- c("1jan1960", "2jan1960", "31mar1960", "30jul1960")

z <- strptime(x, "%d%b%Y")

## Sys.setlocale("LC\_TIME", lct)

z

## read in date/time info in format 'm/d/y h:m:s'

dates <- c("02/27/92", "02/27/92", "01/14/92", "02/28/92", "02/01/92")

times <- c("23:03:20", "22:29:56", "01:03:30", "18:21:03", "16:56:26")

x <- paste(dates, times)

strptime(x, "%m/%d/%y %H:%M:%S")

## time with fractional seconds

z <- strptime("20/2/06 11:16:16.683", "%d/%m/%y %H:%M:%OS")

z # prints without fractional seconds

op <- options(digits.secs = 3)

z

options(op)

## time zones name are not portable, but 'EST5EDT' comes pretty close.

(x <- strptime(c("2006-01-08 10:07:52", "2006-08-07 19:33:02"),

"%Y-%m-%d %H:%M:%S", tz = "EST5EDT"))

attr(x, "tzone")

## An RFC 822 header (Eastern Canada, during DST)

strptime("Tue, 23 Mar 2010 14:36:38 -0400", "%a, %d %b %Y %H:%M:%S %z")

## Make sure you know what the abbreviated names are for you if you wish

## to use them for input (they are matched case-insensitively):

format(seq.Date(as.Date('1978-01-01'), by = 'day', len = 7), "%a")

format(seq.Date(as.Date('2000-01-01'), by = 'month', len = 12), "%b")