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 'getdate.3p' command

$ man getdate.3p

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

getdate ? convert user format date and time

SYNOPSIS

#include <time.h>

struct tm *getdate(const char *string);

DESCRIPTION

The getdate() function shall convert a string representation of a date or time into a broken-down time.

The external variable or macro getdate_err, which has type int, is used by getdate() to return error values. It is unspecified whether get? date_err is a macro or an identifier declared with external linkage, and whether or not it is a modifiable lvalue. If a macro definition is suppressed in order to access an actual object, or a program defines an identifier with the name getdate_err, the behavior is undefined.

Templates are used to parse and interpret the input string. The tem? plates are contained in a text file identified by the environment vari? able DATEMSK. The DATEMSK variable should be set to indicate the full

pathname of the file that contains the templates. The first line in the template that matches the input specification is used for interpreta? tion and conversion into the internal time format.

The following conversion specifications shall be supported:

%% Equivalent to %.

%a Abbreviated weekday name.

%A Full weekday name.

%b Abbreviated month name.

%B Full month name.

%c Locale's appropriate date and time representation.

%C Century number [00,99]; leading zeros are permitted but not re? quired.

%d Day of month [01,31]; the leading 0 is optional.

%D Date as %m/%d/%y.

%e Equivalent to %d.

%h Abbreviated month name.

%H Hour [00,23].

%I Hour [01,12].

%m Month number [01,12].

%M Minute [00,59].

%n Equivalent to <newline>.

%p Locale's equivalent of either AM or PM.

%r The locale's appropriate representation of time in AM and PM notation. In the POSIX locale, this shall be equivalent to %I:%M:%S %p.

%R Time as %H:%M.

%S Seconds [00,60]. The range goes to 60 (rather than stopping at 59) to allow positive leap seconds to be expressed. Since leap seconds cannot be predicted by any algorithm, leap second data must come from some external source.

%t Equivalent to <tab>.

%T Time as %H:%M:%S.

%w Weekday number (Sunday = [0,6]).

%x Locale's appropriate date representation.

%X Locale's appropriate time representation.

%y Year within century. When a century is not otherwise specified, values in the range [69,99] shall refer to years 1969 to 1999 inclusive, and values in the range [00,68] shall refer to years 2000 to 2068 inclusive.

Note: It is expected that in a future version of this stan? dard the default century inferred from a 2-digit year will change. (This would apply to all commands ac? cepting a 2-digit year as input.)

%Y Year as "ccyy" (for example, 2001).

%Z Timezone name or no characters if no timezone exists. If the timezone supplied by %Z is not the timezone that getdate() ex? pects, an invalid input specification error shall result. The getdate() function calculates an expected timezone based on in? formation supplied to the function (such as the hour, day, and month).

The match between the template and input specification performed by getdate() shall be case-insensitive.

The month and weekday names can consist of any combination of upper and lowercase letters. The process can request that the input date or time specification be in a specific language by setting the LC_TIME category (see setlocale()).

Leading zeros are not necessary for the descriptors that allow leading zeros. However, at most two digits are allowed for those descriptors, including leading zeros. Extra white space in either the template file or in string shall be ignored.

The results are undefined if the conversion specifications %c, %x, and %X include unsupported conversion specifications.

The following rules apply for converting the input specification into the internal format:

* If %Z is being scanned, then getdate() shall initialize the broken-down time to be the current time in the scanned timezone. Other?

wise, it shall initialize the broken-down time based on the current local time as if localtime() had been called.

* If only the weekday is given, the day chosen shall be the day, starting with today and moving into the future, which first matches the named day.
* If only the month (and no year) is given, the month chosen shall be the month, starting with the current month and moving into the fu? ture, which first matches the named month. The first day of the month shall be assumed if no day is given.
* If no hour, minute, and second are given, the current hour, minute, and second shall be assumed.
* If no date is given, the hour chosen shall be the hour, starting with the current hour and moving into the future, which first matches the named hour.

If a conversion specification in the DATEMSK file does not correspond to one of the conversion specifications above, the behavior is unspeci? fied.

The getdate() function need not be thread-safe.

RETURN VALUE

Upon successful completion, getdate() shall return a pointer to a struct tm. Otherwise, it shall return a null pointer and set get? date_err to indicate the error.

ERRORS

The getdate() function shall fail in the following cases, setting get? date_err to the value shown in the list below. Any changes to errno are unspecified.

1. The DATEMSK environment variable is null or undefined.
2. The template file cannot be opened for reading.
3. Failed to get file status information.
4. The template file is not a regular file.
5. An I/O error is encountered while reading the template file.
6. Memory allocation failed (not enough memory available).
7. There is no line in the template that matches the input.

8. Invalid input specification. For example, February 31; or a time is specified that cannot be represented in a time_t (representing the time in seconds since the Epoch).

The following sections are informative.

EXAMPLES

1. The following example shows the possible contents of a template:

```
%m
%A %B %d, %Y, %H:%M:%S
%A
%B
%m/%d/%y %I %p
%d,%m,%Y %H:%M
at %A the %dst of %B in %Y
run job at %I %p,%B %dnd
%A den %d. %B %Y %H.%M Uhr
```

2. The following are examples of valid input specifications for the template in Example 1:

```
getdate("10/1/87 4 PM");
getdate("Friday");
getdate("Friday September 18, 1987, 10:30:30");
getdate("24,9,1986 10:30");
getdate("at monday the 1st of december in 1986");
getdate("run job at 3 PM, december 2nd");
```

If the LC_TIME category is set to a German locale that includes freitag as a weekday name and oktober as a month name, the follow? ing would be valid:

```
getdate("freitag den 10. oktober 1986 10.30 Uhr");
```

3. The following example shows how local date and time specification can be defined in the template:

```
????????????????????????????????????????????
?       Invocation        ? Line in Template ?
????????????????????????????????????????????
?getdate("11/27/86")      ? %m/%d/%y         ?
```

?getdate("27.11.86") ? %d.%m.%y ?

?getdate("86-11-27") ? %y-%m-%d ?

?getdate("Friday 12:00:00") ? %A %H:%M:%S ?

??

4. The following examples help to illustrate the above rules assuming that the current date is Mon Sep 22 12:19:47 EDT 1986 and the LC_TIME category is set to the default C or POSIX locale:

??

? Input ? Line in Template ? Date ?

??

?Mon ? %a ? Mon Sep 22 12:19:47 EDT 1986 ?

?Sun ? %a ? Sun Sep 28 12:19:47 EDT 1986 ?

?Fri ? %a ? Fri Sep 26 12:19:47 EDT 1986 ?

?September ? %B ? Mon Sep 1 12:19:47 EDT 1986 ?

?January ? %B ? Thu Jan 1 12:19:47 EST 1987 ?

?December ? %B ? Mon Dec 1 12:19:47 EST 1986 ?

?Sep Mon ? %b %a ? Mon Sep 1 12:19:47 EDT 1986 ?

?Jan Fri ? %b %a ? Fri Jan 2 12:19:47 EST 1987 ?

?Dec Mon ? %b %a ? Mon Dec 1 12:19:47 EST 1986 ?

?Jan Wed 1989 ? %b %a %Y ? Wed Jan 4 12:19:47 EST 1989 ?

?Fri 9 ? %a %H ? Fri Sep 26 09:00:00 EDT 1986 ?

?Feb 10:30 ? %b %H:%S ? Sun Feb 1 10:00:30 EST 1987 ?

?10:30 ? %H:%M ? Tue Sep 23 10:30:00 EDT 1986 ?

?13:30 ? %H:%M ? Mon Sep 22 13:30:00 EDT 1986 ?

??

APPLICATION USAGE

Although historical versions of getdate() did not require that <time.h> declare the external variable getdate_err, this volume of POSIX.1?2017 does require it. The standard developers encourage applications to re? move declarations of getdate_err and instead incorporate the declara? tion by including <time.h>.

Applications should use %Y (4-digit years) in preference to %y (2-digit years).

RATIONALE

In standard locales, the conversion specifications %c, %x, and %X do not include unsupported conversion specifiers and so the text regarding results being undefined is not a problem in that case.

FUTURE DIRECTIONS

None.

SEE ALSO

ctime(), localtime(), setlocale(), strftime(), times()

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

COPYRIGHT