NAME

getdate, getdate_r − convert a date-plus-time string to broken-down time

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

#include <time.h>

struct tm *getdate(const char ****string*);**

extern int getdate_err;

int getdate_r(const char *restrict *string***, struct tm *restrict** *res***);**

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

getdate():
  _XOPEN_SOURCE >= 500

getdate_r():
  _GNU_SOURCE

DESCRIPTION

The function **getdate**() converts a string representation of a date and time, contained in the buffer pointed to by *string*, into a broken-down time. The broken-down time is stored in a *tm* structure, and a pointer to this structure is returned as the function result. This *tm* structure is allocated in static storage, and consequently it will be overwritten by further calls to **getdate**().

In contrast to **strptime**(3), (which has a *format* argument), **getdate**() uses the formats found in the file whose full pathname is given in the environment variable **DATEMSK**. The first line in the file that matches the given input string is used for the conversion.

The matching is done case insensitively. Superfluous whitespace, either in the pattern or in the string to be converted, is ignored.

The conversion specifications that a pattern can contain are those given for **strptime**(3). One more conversion specification is specified in POSIX.1-2001:

%Z Timezone name. This is not implemented in glibc.

When **%Z** is given, the structure containing the broken-down time is initialized with values corresponding to the current time in the given timezone. Otherwise, the structure is initialized to the broken-down time corresponding to the current local time (as by a call to **localtime**(3)).

When only the day of the week is given, the day is taken to be the first such day on or after today.

When only the month is given (and no year), the month is taken to be the first such month equal to or after the current month. If no day is given, it is the first day of the month.

When no hour, minute, and second are given, the current hour, minute, and second are taken.

If no date is given, but we know the hour, then that hour is taken to be the first such hour equal to or after the current hour.

getdate_r() is a GNU extension that provides a reentrant version of **getdate**(). Rather than using a global variable to report errors and a static buffer to return the broken down time, it returns errors via the function result value, and returns the resulting broken-down time in the caller-allocated buffer pointed to by the argument *res*.

RETURN VALUE

When successful, **getdate**() returns a pointer to a *struct tm*. Otherwise, it returns NULL and sets the global variable *getdate_err* to one of the error numbers shown below. Changes to*errno* are unspecified.

On success **getdate_r**() returns 0; on error it returns one of the error numbers shown below.

ERRORS

The following errors are returned via *getdate_err* (for **getdate**()) or as the function result (for **getdate_r**()):

1 The **DATEMSK** environment variable is not defined, or its value is an empty string.

2 The template file specified by **DATEMSK** cannot be opened for reading.

3 Failed to get file status information.

4 The template file is not a regular file.

5 An error was encountered while reading the template file.

6 Memory allocation failed (not enough memory available).

7 There is no line in the file that matches the input.

8 Invalid input specification.

ENVIRONMENT

DATEMSK
File containing format patterns.

TZ, **LC_TIME**
Variables used by **strptime**(3).

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes**(7).

Interface	Attribute	Value
getdate()	Thread safety	MT-Unsafe race:getdate env locale
getdate_r()	Thread safety	MT-Safe env locale

STANDARDS

POSIX.1-2001, POSIX.1-2008.

NOTES

The POSIX.1 specification for **strptime**(3) contains conversion specifications using the **%E** or **%O** modifier, while such specifications are not given for **getdate**(). In glibc,**getdate**() is implemented using **str p-time**(3), so that precisely the same conversions are supported by both.

EXAMPLES

The program below calls **getdate**() for each of its command-line arguments, and for each call displays the values in the fields of the returned *tm* structure. The following shell session demonstrates the operation of the program:

```
$ TFILE=$PWD/tfile
$ echo '%A' > $TFILE       # Full name of the day of the week
$ echo '%T' >> $TFILE      # Time (HH:MM:SS)
$ echo '%F' >> $TFILE      # ISO date (YYYY−MM−DD)
$ date
$ export DATEMSK=$TFILE
$ ./a.out Tuesday '2009−12−28' '12:22:33'
Sun Sep  7 06:03:36 CEST 2008
Call 1 ("Tuesday") succeeded:
    tm_sec   = 36
    tm_min   = 3
    tm_hour  = 6
    tm_mday  = 9
    tm_mon   = 8
    tm_year  = 108
    tm_wday  = 2
    tm_yday  = 252
```

```
            tm_isdst = 1
        Call 2 ("2009−12−28") succeeded:
            tm_sec   = 36
            tm_min   = 3
            tm_hour  = 6
            tm_mday  = 28
            tm_mon   = 11
            tm_year  = 109
            tm_wday  = 1
            tm_yday  = 361
            tm_isdst = 0
        Call 3 ("12:22:33") succeeded:
            tm_sec   = 33
            tm_min   = 22
            tm_hour  = 12
            tm_mday  = 7
            tm_mon   = 8
            tm_year  = 108
            tm_wday  = 0
            tm_yday  = 250
            tm_isdst = 1
```

Program source

```
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int
main(int argc, char *argv[])
{
    struct tm *tmp;

    for (size_t j = 1; j < argc; j++) {
        tmp = getdate(argv[j]);

        if (tmp == NULL) {
            printf("Call %zu failed; getdate_err = %d\n",
                   j, getdate_err);
            continue;
        }

        printf("Call %zu (\"%s\") succeeded:\n", j, argv[j]);
        printf("    tm_sec   = %d\n", tmp−>tm_sec);
        printf("    tm_min   = %d\n", tmp−>tm_min);
        printf("    tm_hour  = %d\n", tmp−>tm_hour);
        printf("    tm_mday  = %d\n", tmp−>tm_mday);
        printf("    tm_mon   = %d\n", tmp−>tm_mon);
        printf("    tm_year  = %d\n", tmp−>tm_year);
        printf("    tm_wday  = %d\n", tmp−>tm_wday);
        printf("    tm_yday  = %d\n", tmp−>tm_yday);
        printf("    tm_isdst = %d\n", tmp−>tm_isdst);
    }
```

```
        exit(EXIT_SUCCESS);
    }
```

SEE ALSO

time(2), **localtime**(3), **setlocale**(3), **strftime**(3), **strptime**(3)