Dates tool box

Generated by Doxygen 1.8.11

# **Contents**

1	Intro	duction		1
2	Usa	ge		2
3	File	Index		2
	3.1	File Lis	at	2
4	File	Docume	entation	2
	4.1	dates.c	File Reference	2
		4.1.1	Function Documentation	5
	4.2	dates.h	File Reference	28
		4.2.1	Enumeration Type Documentation	31
		4.2.2	Function Documentation	32
Inc	dex			57

#### 1 Introduction

This library is a tool box that facilitates the management of dates and times.

It is a superset of lower level POSIX functions. The functions of this toolbox manipulate instants (points) in time expressed as a date and time of day, in Gregorian calendar.

Instants are internally stored in the structure struct tm defined by POSIX, with a resolution of one second. This allows compatible access to low level POSIX functions, such as strftime() or strptime() (see man page of mktime()).

However, objects struct tm should be considered as abstract data types, and should not be initialized by hand. Instants should be initialized with tm\_makelocal(), tm\_makeutc(), tm\_makenow() and tm\_maketoday() instead. The use of these functions is compulsory, as well as easier than handling with struct tm.

Once initialized, tm\_set(), tm\_setdatefromstring(), tm\_settimefromstring() can be used to modify the instant.

Instants in time can be at will represented either in UTC or local time. Functions tm\_toutcrepresentation() and tm← \_tolocalrepresentation() allow to switch from one representation to the other. Functions tm\_isutcrepresentation(), tm\_islocalrepresentation() and tm\_getrepresentation() permit to know the current representation of an instant in time. These functions do not affect the instant in time but only the way it is yield. One could think of it as the unit with which the instant is expressed.

Daylight saving time is taken into account in local time representation but is not applicable to UTC:

- When local representation is used, calculations take daylight saving time rules into account. Days with DST change contain 23 or 25 hours when added or compared. Local time is appropriate for acquisition or display in user interfaces of desktop applications.
- On the contrary, in UTC, daylight saving time does not apply, and all days last 24 hours.

Functions tm\_isdaylightsavingtime(), tm\_isdaylightsavingextrawintertime(), tm\_isdaylightsavingextrasummertime() indicate whether or not Daylight saving time is in effect. Function tm\_hasdaylightsavingtimerules() indicates whether or not daylight saving time rules apply in local timezone.

Functions for calculation are tm\_add... and tm\_diff.... Functions for comparison are tm\_compare() and tm\_equals(). Functions for persistance are tm\_tobinary() and tm\_frombinary().

# 2 Usage

#### Usage requires including

```
* #define _BSD_SOURCE
* #include <time.h>
* #include "dates.h"
*
```

# 3 File Index

#### 3.1 File List

dates.h

Here is a list of all documented files with brief descriptions:

dates.c 2

# 4 File Documentation

## 4.1 dates.c File Reference

#### **Functions**

• static const char \* tm\_utctimezone (void)

Returns the name of UTC timezone.

static time\_t tm\_normalizetolocal (struct tm \*tm)

Initializes instant in time from local date and time data.

• static time\_t tm\_normalizetoutc (struct tm \*tm)

Initializes instant in time from UTC date and time data.

• static time\_t tm\_normalize (struct tm \*date)

Normalizes instant in time.

tm\_status tm\_makenow (struct tm \*tm)

Initializes (or reinitializes) an instant in time with current date and time.

tm\_status tm\_maketoday (struct tm \*tm)

Initializes (or reinitializes) instant in time with current date, beginning of day, local time.

- $\bullet \;\; tm\_status \; tm\_makelocal \; (struct \; tm \; *tm, \; int \; year, \; tm\_month \; month, \; int \; day, \; int \; hour, \; int \; min, \; int \; sec)\\$
- Initializes (or reinitializes) instant intime with local date and time attributes.

   static tm\_status tm\_makelocalfromcalendartime (time\_t timep, struct tm \*tm)

Initialize instant in time with absolute calendar time.

• static tm status tm makeutcfromcalendartime (time t timep, struct tm \*tm)

Initializes instant in time with absolute calendar time.

• tm\_status tm\_makeutc (struct tm \*tm, int year, tm\_month month, int day, int hour, int min, int sec)

Initializes (or reinitializes) instant in time with UTC date and time attributes.

• tm status tm setdatefromstring (struct tm \*tm, const char \*buf)

Sets date from string.

tm\_status tm\_settimefromstring (struct tm \*tm, const char \*buf)

28

Sets time from string.

• tm\_status tm\_gettimeintostring (struct tm dt, char \*str, size\_t max)

Formats time into string.

tm\_status tm\_getdateintostring (struct tm dt, char \*str, size\_t max)

Formats date into string.

tm\_status tm\_toutcrepresentation (struct tm \*date)

Switches representation of instant in time to UTC.

tm status tm tolocalrepresentation (struct tm \*date)

Switches representation of instant in time to local time.

• int tm\_isleapyear (int year)

Indicates leap years.

int tm\_getweeksinisoyear (int isoyear)

Returns the number of weeks in ISO year.

int tm\_getdaysinmonth (int year, tm\_month month)

Returns the number of days in the specified month and year.

int tm\_getsecondsinlocalday (int year, tm\_month month, int day)

Returns the number of seconds in the specified day, month and year.

int tm\_getfirstweekdayinmonth (int year, tm\_month month, tm\_dayofweek dow)

Returns the day of the first weekday in the specified month.

int tm getlastweekdayinmonth (int year, tm month month, tm dayofweek dow)

Returns the day of the last weekday in the specified month.

int tm\_getfirstweekdayinisoyear (int isoyear, tm\_dayofweek dow)

Returns the day of the first weekday in the specified ISO-year.

tm\_dayofweek tm\_getdayofweek (struct tm date)

Gets day of week.

• tm\_month tm\_getmonth (struct tm date)

Gets the month in year, in the Gregorian calendar.

int tm\_getyear (struct tm date)

Gets the year, in the Gregorian calendar.

int tm getday (struct tm date)

Gets the day of the month, in the Gregorian calendar.

int tm\_gethour (struct tm date)

Gets hours.

• int tm\_getminute (struct tm date)

Gets minutes.

• int tm\_getsecond (struct tm date)

Gets seconds.

• int tm\_getdayofyear (struct tm date)

Gets day of year.

int tm\_getisoweek (struct tm date)

Gets ISO week.

• int tm\_getisoyear (struct tm date)

Gets ISO year.

int tm\_isutcrepresentation (struct tm date)

Indicates that the representation of instant in time is UTC.

• int tm\_islocalrepresentation (struct tm date)

Indicates that the representation of instant in time is local time.

tm\_representation tm\_getrepresentation (struct tm date)

Gets the current representation of instant in time.

int tm\_hasdaylightsavingtimerules (void)

Indicates if the system local timezone does have any daylight saving time rules.

• int tm\_isdaylightsavingtime (struct tm date)

Indicates that daylight saving time is in effect.

• int tm\_isdaylightsavingextrasummertime (struct tm date)

Indicates that date and time will be repeated after DST looses effect.

int tm\_isdaylightsavingextrawintertime (struct tm date)

Indicates that date and time has already occured before DST lost effect.

• tm status tm todaylightsavingextrawintertime (struct tm \*date)

Changes the zone offset to the later of the two valid offsets at a local time-line overlap.

tm\_status tm\_todaylightsavingextrasummertime (struct tm \*date)

Changes the zone offset to the earlier of the two valid offsets at a local time-line overlap.

int tm getutcoffset (struct tm date)

Gets offset between UTC and local time.

• const char \* tm\_gettimezone (struct tm date)

Gets the name of the time zone (either UTC or local time depending on current representation).

• int tm\_getsecondsofday (struct tm date)

Gets seconds of day.

• tm\_status tm\_set (struct tm \*tm, int year, tm\_month month, int day, int hour, int min, int sec)

Sets instant in time with date and time attributes with regards to time representation.

• tm status tm addseconds (struct tm \*date, long int nbSecs)

Adds seconds to the instant of time.

tm status tm adddays (struct tm \*date, int nbDays)

Adds full days to the instant of time, changing year, month and day of month without altering hours, minutes and seconds (if possible).

tm\_status tm\_addmonths (struct tm \*date, int nbMonths)

Adds full months to the instant of time, changing year, month without altering day of month, hours, minutes and seconds (if possible).

tm\_status tm\_addyears (struct tm \*date, int nbYears)

Adds full years to the instant of time, changing year without altering month, day of month, hours, minutes and seconds (if possible).

• tm\_status tm\_trimtime (struct tm \*tm)

Sets the time value to 0am (beginning of the day) and keeps the date component unchanged.

• int tm\_equals (struct tm a, struct tm b)

Returns a value indicating whether two broken-down time have the same value (including representation).

• long int tm\_diffseconds (struct tm debut, struct tm fin)

Gets number of seconds between two dates.

int tm\_compare (const void \*pdebut, const void \*pfin)

Compares two dates.

• int tm\_diffcalendardays (struct tm debut, struct tm fin)

Gets number of partial days between two dates.

int tm diffdays (struct tm debut, struct tm fin, int \*seconds)

Gets number of complete days between two dates.

int tm\_diffweeks (struct tm debut, struct tm fin, int \*days, int \*seconds)

Gets number of complete weeks between two dates.

• int tm diffcalendarmonths (struct tm debut, struct tm fin)

Gets number of partial months between two dates.

• int tm\_diffmonths (struct tm debut, struct tm fin, int \*days, int \*seconds)

Gets number of complete months between two dates.

int tm\_diffcalendaryears (struct tm debut, struct tm fin)

Gets number of partial years between two dates.

int tm\_diffyears (struct tm debut, struct tm fin, int \*months, int \*days, int \*seconds)

Gets number of complete years between two dates.

• int tm\_diffisoyears (struct tm debut, struct tm fin)

Gets number of partial ISO years between two dates.

 void tm\_getintimezone (struct tm date, const char \*tz, int \*year, tm\_month \*month, int \*day, int \*hour, int \*minute, int \*second, int \*isdst)

Gets time in another target timezone.

time\_t tm\_tobinary (struct tm date)

Serializes the instant of time to a binary value that subsequently can be used to recreate the instant of time.

tm status tm frombinary (struct tm \*date, time t binary)

Deserializes a binary value and recreates an original serialized date and time.

#### 4.1.1 Function Documentation

4.1.1.1 tm\_status tm\_adddays ( struct tm \* date, int nbDays )

Adds full days to the instant of time, changing year, month and day of month without altering hours, minutes and seconds (if possible).

It takes into account leap years and the number of days in a month.

Behavior depends on the current representation of the instant of time: in local time representation, adding one day might correspond to adding 23, 24 or 25 hours, depending whether or not there is a daylight saving time change. In case representation for both instants of time is local, days including a switch between standard time and daylight saving time count for 23 or 25 hours rather than 24. I.e., adding 14 days to march the 14th, 2016, 9 am, local Paris time, yields march the 28th, 9 am (rather than 10 am if a multiple of 24 hours were added), that is only 335 hours.

In order to add an exact multiple of 24 hours, use tm\_addseconds() instead.

In local time, if adding days results in an hour that is not valid in the resulting day (in case of daylight saving time change from winter to summer rule), an extra hour is added. For example, the transition from standard time to daylight saving time occurs in the U.S. Pacific Time zone on March 14, 2010, at 2:00 A.M., when the time advances by one hour, to 3:00 A.M. This hour interval is an invalid time, that is, a time interval that does not exist in this time zone. Thus, adding one day to March 13, 2010, 2:30 A.M. will result in March 14, 2010, 3:30 A.M, rather than 2:30 A.M.

# Parameters

in,out	date	Pointer to broken-down time structure
in	nbDays	Number of days to add to date

#### Returns

TM\_OK or TM\_ERROR (in case of overflow)

## Remarks

Behavior depends on time representation. Time representation is kept unchanged.

# 4.1.1.2 tm\_status tm\_addmonths ( struct tm \* date, int nbMonths )

Adds full months to the instant of time, changing year, month without altering day of month, hours, minutes and seconds (if possible).

It takes into account leap years and the number of days in a month, then adjusts the day part of the resulting instant in time. If adding months results in a day that is not a valid day in the resulting month, the last day of the resulting month is used. I.e., adding three months to January, the 31st, yields April, the 30th.

Behavior depends on the current representation of the instant of time: in local time representation, adding one day might correspond to adding 23, 24 or 25 hours, depending whether or not there is a daylight saving time change.

In local time, if adding days results in an hour that is not valid in the resulting day (in case of daylight saving time change from winter to summer rule), an extra hour is added. For example, the transition from standard time to daylight saving time occurs in the U.S. Pacific Time zone on March 14, 2010, at 2:00 A.M., when the time advances by one hour, to 3:00 A.M. This hour interval is an invalid time, that is, a time interval that does not exist in this time zone. Thus, adding one month to February 14, 2010, 2:30 A.M. will result in March 14, 2010, 3:30 A.M, rather than 2:30 A.M.

#### **Parameters**

in,out	date	Pointer to broken-down time structure
in	nbMonths	Number of months to add to date

#### Returns

TM\_OK or TM\_ERROR (in case of overflow)

#### Remarks

Behavior depends on time representation. Time representation is kept unchanged.

4.1.1.3 tm\_status tm\_addseconds ( struct tm \* date, long int nbSecs )

Adds seconds to the instant of time.

# Parameters

in,out	date	Pointer to broken-down time structure
in	nbSecs	Number of seconds to add to date

## Returns

TM\_OK or TM\_ERROR (in case of overflow)

### Remarks

Representation is kept unchanged (either local time or UTC).

4.1.1.4 tm\_status tm\_addyears ( struct tm \* date, int nbYears )

Adds full years to the instant of time, changing year without altering month, day of month, hours, minutes and seconds (if possible).

Behaves as if tm addmonths() were called with argument nbMonths equal to 12 x nbYears.

# **Parameters**

in,out	date	Pointer to broken-down time structure
in	nbYears	Number of months to add to date

## Returns

TM\_OK or TM\_ERROR (in case of overflow)

## Remarks

Behavior depends on time representation. Time representation is kept unchanged.

4.1.1.5 int tm\_compare ( const void \* debut, const void \* fin )

Compares two dates.

### **Parameters**

in	debut	Pointer to broken-down time structure
in	fin	Pointer to broken-down time structure

# Returns

-1 if debut is before fin, 1 if debut is after fin, 0 if debut and fin are at same instant, independently of representation.

# Remarks

The representation of instants of time are not considered. Compatible for use with qsort().

4.1.1.6 int tm\_diffcalendardays ( struct tm debut, struct tm fin )

Gets number of partial days between two dates.

## **Parameters**

in	debut	Broken-down time structure
in	fin	Broken-down time structure

## Returns

Number of partial or complete days between debut and fin.

### Remarks

Partial days are counted as 1.

Behavior depends on time representation.

Both debut and fin should have identical representation, otherwise result is unspecified.

## 4.1.1.7 int tm\_diffcalendarmonths ( struct tm debut, struct tm fin )

Gets number of partial months between two dates.

#### **Parameters**

in	debut	Broken-down time structure
in	fin	Broken-down time structure

## Returns

Number of partial or complete months between debut and fin.

#### Remarks

Partial months are counted as 1.

Behavior depends on time representation.

Both debut and fin should have identical representation, otherwise result is unspecified.

# 4.1.1.8 int tm\_diffcalendaryears ( struct tm debut, struct tm fin )

Gets number of partial years between two dates.

#### **Parameters**

in	debut	Broken-down time structure
in	fin	Broken-down time structure

#### Returns

Number of partial or complete years between debut and fin.

# Remarks

Partial years are counted as 1.

Behavior depends on time representation.

Both debut and fin should have identical representation, otherwise result is unspecified.

# 4.1.1.9 int tm\_diffdays ( struct tm debut, struct tm fin, int \* seconds )

Gets number of complete days between two dates.

In case representation for both indtants of time is local, days including between standard time and daylight saving time count for 23 or 25 hours rather than 24. I.e., difference between march the 14th, 9 am and march the 28th, 9 am, 2016, local Paris time, is 14 days, even though it includes only 335 hours.

in	debut	Broken-down time structure
in	fin	Broken-down time structure
out	seconds	Remainder in seconds (optional)

4.1 dates.c File Reference 9

#### Returns

Number of complete days between debut and fin.

#### Remarks

Partial days are counted as 0.

Behavior depends on time representation.

Both debut and fin should have identical representation, otherwise result is unspecified.

4.1.1.10 int tm\_diffisoyears ( struct tm debut, struct tm fin )

Gets number of partial ISO years between two dates.

## **Parameters**

in	debut	Broken-down time structure
in	fin	Broken-down time structure

## Returns

Number of partial or complete ISO years between debut and fin.

## Remarks

Partial ISO years are counted as 1.

Behavior depends on time representation.

Both debut and fin should have identical representation, otherwise result is unspecified.

4.1.1.11 int tm\_diffmonths ( struct tm debut, struct tm fin, int \* days, int \* seconds )

Gets number of complete months between two dates.

# **Parameters**

in	debut	Broken-down time structure		
in	fin	Broken-down time structure		
out	days	Remainder in days (optional)		
out	seconds	Remainder in seconds (optional)		

# Returns

Number of complete months between debut and fin.

## Remarks

Partial months are counted as 0.

Behavior depends on time representation.

Both debut and fin should have identical representation, otherwise result is unspecified.

## 4.1.1.12 long int tm\_diffseconds ( struct tm debut, struct tm fin )

Gets number of seconds between two dates.

#### **Parameters**

in	debut	Broken-down time structure
in	fin	Broken-down time structure

# Returns

Number of seconds between debut and fin. Negative of debut is after fin, positive if debut is before fin.

## Remarks

The representation of instants of time are not considered.

A difference of 0 seconds means debut and fin correspond to the same instant, independently of representation.

4.1.1.13 int tm\_diffweeks ( struct tm debut, struct tm fin, int \* days, int \* seconds )

Gets number of complete weeks between two dates.

#### **Parameters**

in	debut	Broken-down time structure
in	fin	Broken-down time structure
out	days Remainder in days (optional)	
out	seconds	Remainder in seconds (optional)

# Returns

Number of complete weeks between debut and fin.

## Remarks

Behavior depends on time representation.

Both debut and fin should have identical representation, otherwise result is unspecified.

4.1.1.14 int tm\_diffyears ( struct tm debut, struct tm fin, int \* months, int \* days, int \* seconds )

Gets number of complete years between two dates.

in	debut	Broken-down time structure
in	fin	Broken-down time structure
out	months	Remainder in months (optional)
out	days	Remainder in days (optional)
out	seconds	Remainder in seconds (optional)

#### Returns

Number of complete years between debut and fin.

#### Remarks

Partial years are counted as 0.

Behavior depends on time representation.

Both debut and fin should have identical representation, otherwise result is unspecified.

4.1.1.15 int tm\_equals ( struct tm a, struct tm b )

Returns a value indicating whether two broken-down time have the same value (including representation).

#### **Parameters**

in	а	Broken-down time structure
in	b	Broken-down time structure

#### Remarks

Behavior depends on time representation. Use tm\_diffseconds() or tm\_compare() for an absolute date and time comparison.

## Returns

1 if the two broken-down time have the same value, 0 otherwise.

4.1.1.16 tm\_status tm\_frombinary ( struct tm \* date, time\_t binary )

Deserializes a binary value and recreates an original serialized date and time.

### **Parameters**

out	date	Pointer to broken-down time structure, in local timezone representation
in	binary	representation of instant (point in time).

## Returns

TM\_OK on sucess, TM\_ERROR otherwise (overflow).

# Remarks

The instant (point in time) is presented in local time representation by default.

4.1.1.17 tm\_status tm\_getdateintostring ( struct tm dt, char \* str, size\_t max )

Formats date into string.

Formats the date and time according to the preferred date (without the time) display format for the current locale and places the result in the character array str of size max. str should have been previously allocated elsewhere.

#### **Parameters**

in	dt	Broken-down time structure
in	max	Size of the previously allocated string str
out	str	null terminated string

# Returns

 $\texttt{TM\_OK}$  if the result string, including the terminating null byte, does not exceed max bytes,  $\texttt{TM\_ERROR}$  otherwise (and the contents of the string str are then undefined.)

## Remarks

Behavior depends on time representation. Makes call to strftime().

4.1.1.18 int tm\_getday ( struct tm date )

Gets the day of the month, in the Gregorian calendar.

#### **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

## Returns

Day of month

### Remarks

Behavior depends on time representation.

4.1.1.19 tm\_dayofweek tm\_getdayofweek ( struct tm date )

Gets day of week.

# **Parameters**

in	date	Broken-down time structure

## Returns

Day of week (1 = Monday, 7 = Sunday)

# Remarks

Behavior depends on time representation.

4.1.1.20 int tm\_getdayofyear ( struct tm date )

Gets day of year.

4.1 dates.c File Reference

## **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

## Returns

Day of year (1 = January, the 1st)

## Remarks

Behavior depends on time representation.

4.1.1.21 int tm\_getdaysinmonth ( int year, tm\_month month )

Returns the number of days in the specified month and year.

It interprets month and year as the month and year of the Gregorian calendar, taking leap years into account.

#### **Parameters**

in	year	The year specified as a 4-digit number (for example, 1996), interpreted as a year in the Gregorian calendar.
in	month	Month

## Returns

Number of days in month month of year year

4.1.1.22 int tm\_getfirstweekdayinisoyear ( int isoyear, tm\_dayofweek dow )

Returns the day of the first weekday in the specified ISO-year.

# **Parameters**

in	isoyear	year
in	dow	Day of week

## Returns

The day of the first weekday in the specified month.

4.1.1.23 int tm\_getfirstweekdayinmonth ( int year, tm\_month month, tm\_dayofweek dow )

Returns the day of the first weekday in the specified month.

#### **Parameters**

in	year	Year
in	month	Month
in	dow	Day of week

Generated by Doxygen

## Returns

The day of the first weekday in the specified month.

4.1.1.24 int tm\_gethour ( struct tm date )

Gets hours.

#### **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

## Returns

Hours (between 0 and 23)

# Remarks

Behavior depends on time representation.

4.1.1.25 void tm\_getintimezone ( struct tm date, const char \* tz, int \* year, tm\_month \* month, int \* day, int \* hour, int \* minute, int \* second, int \* is\_dst\_on )

Gets time in another target timezone.

Daylight saving times are considered.

# **Parameters**

in	date	Broken-down time structure, either in local timezone or UTC representation
in	tz	Target timezone (see "man tzset" for details on possible values for $\ensuremath{\text{tz}}$ )
out	year	Year at the time described in target timezone
out	month	Month at the time described in target timezone
out	day	Day at the time described in target timezone
out	hour	Hour at the time described in target timezone
out	minute	Minute at the time described in target timezone
out	second	Second at the time described in target timezone
out	is_dst_on	Indicates whether (1) or not (0) daylight saving time is in effect at the time described in
		target timezone

4.1.1.26 int tm\_getisoweek ( struct tm date )

Gets ISO week.

	in	date	Broken-down time structure
--	----	------	----------------------------

#### Returns

ISO 8601 week

## Remarks

Behavior depends on time representation.

ISO 8601 week date: The first week of a year (starting on Monday) is :

- the first week that contains at least 4 days of calendar year.
- the week that contains the first Thursday of a year.
- · the week with January 4 in it

4.1.1.27 int tm\_getisoyear ( struct tm date )

Gets ISO year.

#### **Parameters**

in (	date	Broken-down time structure
------	------	----------------------------

#### Returns

ISO 8601 year

## Remarks

Behavior depends on time representation.

4.1.1.28 int tm\_getlastweekdayinmonth ( int year, tm\_month month, tm\_dayofweek dow )

Returns the day of the last weekday in the specified month.

# Parameters

in	year	Year
in	month	Month
in	dow	Day of week

# Returns

The day of the last weekday in the specified month.

4.1.1.29 int tm\_getminute ( struct tm date )

Gets minutes.

# **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

## Returns

Minutes (between 0 and 59)

## Remarks

Behavior depends on time representation.

4.1.1.30 tm\_month tm\_getmonth ( struct tm date )

Gets the month in year, in the Gregorian calendar.

#### **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

## Returns

Month (1 = January, ..., 12=December)

# Remarks

Behavior depends on time representation.

4.1.1.31 tm\_representation tm\_getrepresentation ( struct tm date )

Gets the current representation of instant in time.

## **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

# Returns

Date and time representation (TM\_LOCAL or TM\_UTC)

4.1.1.32 int tm\_getsecond ( struct tm date )

Gets seconds.

#### Returns

Seconds (between 0 and 59)

#### Remarks

Behavior depends on time representation.

4.1.1.33 int tm\_getsecondsinlocalday ( int year, tm\_month month, int day )

Returns the number of seconds in the specified day, month and year.

It interprets day, month and year as the day, month and year of the Gregorian calendar, taking leap years and daylight saving time tules into account.

#### **Parameters**

in	year	Year
in	month	Month
in	day	Day of month

#### Returns

Number of seconds in day day of month month of year year, in local time

4.1.1.34 int tm\_getsecondsofday ( struct tm date )

Gets seconds of day.

### **Parameters**

in	date	Broken-down time structure

## Returns

Elapsed seconds since beginning of day.

## Remarks

Behavior depends on time representation.

4.1.1.35  $tm_status tm_gettimeintostring ( struct tm dt, char * str, size_t max )$ 

Formats time into string.

Formats the date and time according to the preferred time (without the date) display format for the current locale and places the result in the character array str of size max. str should have been previously allocated elsewhere.

#### **Parameters**

in	dt	Broken-down time structure
in	max	Size of the previously allocated string str
out	str	null terminated string.

#### Returns

 $\texttt{TM\_OK}$  if the result string, including the terminating null byte, does not exceed max bytes,  $\texttt{TM\_ERROR}$  otherwise (and the contents of the string str are then undefined.)

## Remarks

Behavior depends on time representation. Makes call to strftime().

4.1.1.36 const char\* tm\_gettimezone ( struct tm date )

Gets the name of the time zone (either UTC or local time depending on current representation).

## **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

### Returns

Timezone abbreviation

# Remarks

Behavior depends on time representation.

4.1.1.37 int tm\_getutcoffset ( struct tm date )

Gets offset between UTC and local time.

# **Parameters**

in	date	Broken-down time structure

#### Returns

Offset, in seconds, between UTC and time representation (local or UTC)

## Remarks

Behavior depends on time representation.

4.1.1.38 int tm\_getweeksinisoyear ( int isoyear )

Returns the number of weeks in ISO year.

4.1 dates.c File Reference 19

## **Parameters**

in	isoyear	year
----	---------	------

# Returns

The number of weeks in ISO year

4.1.1.39 int tm\_getyear ( struct tm date )

Gets the year, in the Gregorian calendar.

## **Parameters**

in	date	Broken-down time structure
	aato	Broken down time of dotard

#### Returns

Year

## Remarks

Behavior depends on time representation.

4.1.1.40 int tm\_hasdaylightsavingtimerules (void)

Indicates if the system local timezone does have any daylight saving time rules.

# Returns

0 if this timezone does not have any daylight saving time rules, or nonzero if there is a time, past, present or future when daylight saving time applies.

# Remarks

Behavior depends on time representation.

4.1.1.41 int tm\_isdaylightsavingextrasummertime ( struct tm date )

Indicates that date and time will be repeated after DST looses effect.

in	date	Broken-down time structure

## Returns

1 if time is duplicated (before DST change), 0 otherwise.

## Remarks

Behavior depends on time representation.

4.1.1.42 int tm\_isdaylightsavingextrawintertime ( struct tm date )

Indicates that date and time has already occured before DST lost effect.

#### **Parameters**

	in	date	Broken-down time structure
--	----	------	----------------------------

#### Returns

1 if time is duplicated (after DST change), 0 otherwise.

# Remarks

Behavior depends on time representation.

4.1.1.43 int tm\_isdaylightsavingtime ( struct tm date )

Indicates that daylight saving time is in effect.

## **Parameters**

in	date	Broken-down time structure

# Returns

1 if DST is set, 0 otherwise.

# Remarks

Behavior depends on time representation. In UTC represntation, 0 is returned.

4.1.1.44 int tm\_isleapyear ( int year )

Indicates leap years.

in	year	year

## Returns

1 if year is a leap year, 0 otherwise

4.1.1.45 int tm\_islocalrepresentation ( struct tm date )

Indicates that the representation of instant in time is local time.

## **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

# Returns

1 if date is in local time representation, 0 otherwise.

4.1.1.46 int tm\_isutcrepresentation ( struct tm date )

Indicates that the representation of instant in time is UTC.

## **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

# Returns

1 if date is in UTC representation, 0 otherwise.

4.1.1.47 tm\_status tm\_makelocal ( struct tm \* dt, int year, tm\_month month, int day, int hour, int min, int sec )

Initializes (or reinitializes) instant intime with local date and time attributes.

## **Parameters**

in	year	The year, specified as a 4-digit number (for example, 1996), interpreted as a year in the Gregorian calendar (local time)
in	month	The month (local time)
in	day	The day (1 through the number of days in month) of month (local time)
in	hour	The hours (0 through 23) (local time)
in	min	The minutes (0 through 59) (local time)
in	sec	The seconds (0 through 59) (local time)
out	dt	Pointer to broken-down time structure

## Returns

TM\_OK or TM\_ERROR (in case of overflow or invalid arguments)

# Remarks

The instant (point in time) is initialized in local time representation by default.

4.1.1.48 static tm\_status tm\_makelocalfromcalendartime ( time\_t timep, struct tm \* tm ) [static]

Initialize instant in time with absolute calendar time.

#### **Parameters**

in	timep	Absolute calendar time
out	tm	Pointer to broken-down time structure

### Returns

TM\_OK or TM\_ERROR (in case of overflow)

## Remarks

Default representation is set to local.

4.1.1.49 tm\_status tm\_makenow ( struct tm \* dt )

Initializes (or reinitializes) an instant in time with current date and time.

# **Parameters**

out	dt	Pointer to broken-down time structure
-----	----	---------------------------------------

# Returns

TM\_OK or TM\_ERROR (in case of overflow)

#### Remarks

The instant (point in time) is initialized in local time representation by default.

4.1.1.50 tm\_status tm\_maketoday ( struct tm \* dt )

Initializes (or reinitializes) instant in time with current date, beginning of day, local time.

Initializes (or reinitializes) an instant in time set to today's date, with the time component set to 00:00:00, local time.

out	dt	Pointer to broken-down time structure
-----	----	---------------------------------------

#### Returns

TM\_OK or TM\_ERROR (in case of overflow)

#### Remarks

The instant (point in time) is initialized in local time representation by default.

4.1.1.51 tm\_status tm\_makeutc ( struct tm \* dt, int year, tm\_month month, int day, int hour, int min, int sec )

Initializes (or reinitializes) instant in time with UTC date and time attributes.

#### **Parameters**

in	year	Year, UTC
in	month	Month, UTC
in	day	Day of month, UTC
in	hour	Hour of day, UTC
in	min	Minutes, UTC
in	sec	Seconds, UTC
out	dt	Pointer to broken-down time structure

# Returns

TM\_OK or TM\_ERROR (in case of overflow or invalid arguments)

# Remarks

The instant (point in time) is initialized in UTC representation by default.

4.1.1.52 static tm\_status tm\_makeutcfromcalendartime ( time\_t timep, struct tm \* tm ) [static]

Initializes instant in time with absolute calendar time.

## Parameters

in	timep	Absolute calendar time
out	tm	Pointer to broken-down time structure

# Returns

TM\_OK or TM\_ERROR (in case of overflow)

### Remarks

Default representation is set to UTC.

**4.1.1.53 static time\_t tm\_normalize ( struct tm \*** *date* **)** [static]

Normalizes instant in time.

## **Parameters**

in,out	date	Pointer to broken-down time structure	
in,out	date	Pointer to broken-down time structu	re

# Returns

Absolute calendar time

4.1.1.54 static time\_t tm\_normalizetolocal ( struct tm \* tm ) [static]

Initializes instant in time from local date and time data.

#### **Parameters**

in,out	tm	Pointer to broken-down time structure
--------	----	---------------------------------------

#### Returns

Absolute calendar time

## Remarks

Calls mktime. The tm\_normalizetolocal() function is equivalent to the POSIX standard function mktime()

4.1.1.55 static time\_t tm\_normalizetoutc ( struct tm \* tm ) [static]

Initializes instant in time from UTC date and time data.

## **Parameters**

	in,out	tm	Pointer to broken-down time structure
--	--------	----	---------------------------------------

## Returns

Absolute calendar time

## Remarks

Portable version of timegm(): set the TZ environment variable to UTC, call mktime and restore the value of TZ.

#### See also

man mktime and timegm

4.1.1.56 tm\_status tm\_set ( struct tm \* dt, int year, tm\_month month, int day, int hour, int min, int sec )

Sets instant in time with date and time attributes with regards to time representation.

4.1 dates.c File Reference 25

# **Parameters**

in	year	Year
in	month	Month
in	day	Day of month
in	hour	Hour of day
in	min	Minutes
in	sec	Seconds
out	dt	Pointer to broken-down time structure

#### Returns

TM\_OK or TM\_ERROR (in case of overflow)

# Remarks

Behavior depends on time representation. Time representation is kept unchanged.

4.1.1.57 tm\_status tm\_setdatefromstring ( struct tm \* dt, const char \* str )

Sets date from string.

Recognized formats are: the locale's date format, the locale's alternative date representation, the ISO 8601 date format (YYYY-mm-dd). A year specified on 2 digits is converted to the closest year on 4 digits.

#### **Parameters**

in	str	string representation of date (without time)
out	dt	Pointer to broken-down time structure

# Returns

TM\_OK or TM\_ERROR (in case of overflow)

# Remarks

Behavior depends on time representation. Time representation is kept unchanged. Makes use of strptime().

4.1.1.58  $tm\_status\ tm\_settimefromstring\ (\ struct\ tm*\ \textit{dt},\ const\ char*\ \textit{str}\ )$ 

Sets time from string.

Recognized formats are: the locale's time format, the locale's alternative time representation, HH:MM:SS, HH:MM, (where HH is between 0 and 23)

in	str	string representation of time (without date)
out	dt	Pointer to broken-down time structure

#### Returns

```
TM_OK or TM_ERROR (in case of overflow)
```

#### Remarks

Behavior depends on time representation. Time representation is kept unchanged.

```
4.1.1.59 time_t tm_tobinary ( struct tm date )
```

Serializes the instant of time to a binary value that subsequently can be used to recreate the instant of time.

This binary value is suitable for database recording. It identifies an instant of time unambiguously, whatever the representation (local time or UTC).

#### **Parameters**

## Returns

Binary representation of instant (point in time).

4.1.1.60 tm\_status tm\_todaylightsavingextrasummertime ( struct tm \* date )

Changes the zone offset to the earlier of the two valid offsets at a local time-line overlap.

This function only has any effect when the local time-line overlaps, such as at an autumn daylight savings cutover. In this scenario, there are two valid offsets for the local date-time. Calling this method will return a local zoned date-time with the earlier of the two selected.

# Parameters

in, out   date   Broken-down time structure
---

# Returns

TM\_OK if instant was shifted, TM\_ERROR otherwise.

#### Remarks

Behavior depends on time representation.

4.1.1.61 tm\_status tm\_todaylightsavingextrawintertime ( struct tm \* date )

Changes the zone offset to the later of the two valid offsets at a local time-line overlap.

This function only has any effect when the local time-line overlaps, such as at an autumn daylight savings cutover. In this scenario, there are two valid offsets for the local date-time. Calling this method will return a local zoned date-time with the later of the two selected.

4.1 dates.c File Reference 27

## **Parameters**

in, out date	Broken-down time structure
--------------	----------------------------

# Returns

TM\_OK if instant was shifted, TM\_ERROR otherwise.

## Remarks

Behavior depends on time representation.

4.1.1.62 tm\_status tm\_tolocalrepresentation ( struct tm \* date )

Switches representation of instant in time to local time.

#### **Parameters**

in,out	date	Pointer to broken-down time structure
--------	------	---------------------------------------

## Remarks

Has no effect if time representation is local time already.

4.1.1.63 tm\_status tm\_toutcrepresentation ( struct tm \* date )

Switches representation of instant in time to UTC.

# **Parameters**

in, out	date	Pointer to broken-down time structure

#### Remarks

Has no effect if time representation is UTC already.

# Returns

TM\_OK on success, TM\_ERROR otherwise.

4.1.1.64 tm\_status tm\_trimtime ( struct tm \* date )

Sets the time value to 0am (beginning of the day) and keeps the date component unchanged.

in,out	date	Pointer to broken-down time structure

#### Remarks

Behavior depends on time representation. Time representation is kept unchanged.

```
4.1.1.65 static const char* tm_utctimezone ( void ) [static]
```

Returns the name of UTC timezone.

Returns

The name of UTC timezone

## 4.2 dates.h File Reference

#### **Enumerations**

#### Data types

- enum tm\_status { TM\_OK = EXIT\_SUCCESS, TM\_ERROR = EXIT\_FAILURE }
   Values of status.
- enum tm representation { TM REP LOCAL, TM REP UTC }

Kinds of representation for instant in time.

enum tm\_dayofweek {

TM\_WEEKDAY\_MONDAY = 1, TM\_WEEKDAY\_TUESDAY, TM\_WEEKDAY\_WEDNESDAY, TM\_WEE ← KDAY THURSDAY,

TM\_WEEKDAY\_FRIDAY, TM\_WEEKDAY\_SATURDAY, TM\_WEEKDAY\_SUNDAY }

Days of week.

enum tm\_month {

TM\_MONTH\_JANUARY = 1, TM\_MONTH\_FEBRUARY, TM\_MONTH\_MARCH, TM\_MONTH\_APRIL, TM\_MONTH\_MAY, TM\_MONTH\_JUNE, TM\_MONTH\_JULY, TM\_MONTH\_AUGUST, TM\_MONTH\_SEPTEMBER, TM\_MONTH\_OCTOBER, TM\_MONTH\_NOVEMBER, TM\_MONTH\_DEC← EMBER }

Months.

### **Functions**

## Constructors

Parameter dt should have been previously allocated, otherwise behavior is unpredictable.

- tm\_status tm\_makenow (struct tm \*dt)
  - Initializes (or reinitializes) an instant in time with current date and time.
- tm\_status tm\_maketoday (struct tm \*dt)

Initializes (or reinitializes) instant in time with current date, beginning of day, local time.

- tm\_status tm\_makelocal (struct tm \*dt, int year, tm\_month month, int day, int hour, int min, int sec)

  Initializes (or reinitializes) instant intime with local date and time attributes.
- tm\_status tm\_makeutc (struct tm \*dt, int year, tm\_month month, int day, int hour, int min, int sec)

  Initializes (or reinitializes) instant in time with UTC date and time attributes.

## Setters

Parameter dt should have been previously allocated, otherwise behavior is unpredictable.

- tm\_status tm\_set (struct tm \*dt, int year, tm\_month month, int day, int hour, int min, int sec)

  Sets instant in time with date and time attributes with regards to time representation.
- tm\_status tm\_settimefromstring (struct tm \*dt, const char \*str)

Sets time from string.

• tm\_status tm\_setdatefromstring (struct tm \*dt, const char \*str)

Sets date from string.

#### **Formatters**

• tm\_status tm\_getdateintostring (struct tm dt, char \*str, size\_t max)

Formats date into string.

• tm\_status tm\_gettimeintostring (struct tm dt, char \*str, size\_t max)

Formats time into string.

#### Operators

Those operators take into account local day length (24, 23 or 25 hours) when representation is local time.

• tm status tm addseconds (struct tm \*date, long int nbSecs)

Adds seconds to the instant of time.

tm\_status tm\_adddays (struct tm \*date, int nbDays)

Adds full days to the instant of time, changing year, month and day of month without altering hours, minutes and seconds (if possible).

• tm status tm addmonths (struct tm \*date, int nbMonths)

Adds full months to the instant of time, changing year, month without altering day of month, hours, minutes and seconds (if possible).

tm\_status tm\_addyears (struct tm \*date, int nbYears)

Adds full years to the instant of time, changing year without altering month, day of month, hours, minutes and seconds (if possible).

tm status tm trimtime (struct tm \*date)

Sets the time value to 0am (beginning of the day) and keeps the date component unchanged.

• tm\_status tm\_todaylightsavingextrasummertime (struct tm \*date)

Changes the zone offset to the earlier of the two valid offsets at a local time-line overlap.

tm\_status tm\_todaylightsavingextrawintertime (struct tm \*date)

Changes the zone offset to the later of the two valid offsets at a local time-line overlap.

# Comparators

Those comparators take into account local day length (24, 23 or 25 hours) when representation is local time.

int tm\_equals (struct tm a, struct tm b)

Returns a value indicating whether two broken-down time have the same value (including representation).

• long int tm\_diffseconds (struct tm debut, struct tm fin)

Gets number of seconds between two dates.

int tm\_compare (const void \*debut, const void \*fin)

Compares two dates.

int tm diffcalendardays (struct tm debut, struct tm fin)

Gets number of partial days between two dates.

int tm\_diffdays (struct tm debut, struct tm fin, int \*seconds)

Gets number of complete days between two dates.

int tm\_diffweeks (struct tm debut, struct tm fin, int \*days, int \*seconds)

Gets number of complete weeks between two dates.

int tm\_diffcalendarmonths (struct tm debut, struct tm fin)

Gets number of partial months between two dates.

int tm\_diffmonths (struct tm debut, struct tm fin, int \*days, int \*seconds)

Gets number of complete months between two dates.

• int tm\_diffcalendaryears (struct tm debut, struct tm fin)

Gets number of partial years between two dates.

• int tm diffyears (struct tm debut, struct tm fin, int \*months, int \*days, int \*seconds)

Gets number of complete years between two dates.

int tm\_diffisoyears (struct tm debut, struct tm fin)

Gets number of partial ISO years between two dates.

#### Representation of instant in time

• tm\_status tm\_toutcrepresentation (struct tm \*date)

Switches representation of instant in time to UTC.

tm\_status tm\_tolocalrepresentation (struct tm \*date)

Switches representation of instant in time to local time.

• int tm\_isutcrepresentation (struct tm date)

Indicates that the representation of instant in time is UTC.

int tm islocalrepresentation (struct tm date)

Indicates that the representation of instant in time is local time.

tm\_representation tm\_getrepresentation (struct tm date)

Gets the current representation of instant in time.

## **Properties**

int tm hasdaylightsavingtimerules (void)

Indicates if the system local timezone does have any daylight saving time rules.

int tm\_isdaylightsavingtime (struct tm date)

Indicates that daylight saving time is in effect.

• int tm\_isdaylightsavingextrasummertime (struct tm date)

Indicates that date and time will be repeated after DST looses effect.

• int tm\_isdaylightsavingextrawintertime (struct tm date)

Indicates that date and time has already occured before DST lost effect.

## Getters

• int tm\_getyear (struct tm date)

Gets the year, in the Gregorian calendar.

tm month tm getmonth (struct tm date)

Gets the month in year, in the Gregorian calendar.

int tm\_getday (struct tm date)

Gets the day of the month, in the Gregorian calendar.

int tm\_gethour (struct tm date)

Gets hours.

• int tm\_getminute (struct tm date)

Gets minutes.

int tm\_getsecond (struct tm date)

Gets seconds.

• int tm\_getdayofyear (struct tm date)

Gets day of year.

tm\_dayofweek tm\_getdayofweek (struct tm date)

Gets day of week.

• int tm getisoweek (struct tm date)

Gets ISO week.

• int tm\_getisoyear (struct tm date)

Gets ISO year.

• int tm\_getutcoffset (struct tm date)

Gets offset between UTC and local time.

const char \* tm\_gettimezone (struct tm date)

Gets the name of the time zone (either UTC or local time depending on current representation).

• int tm getsecondsofday (struct tm date)

Gets seconds of day.

### **Helpers**

 void tm\_getintimezone (struct tm date, const char \*tz, int \*year, tm\_month \*month, int \*day, int \*hour, int \*minute, int \*second, int \*is\_dst\_on) Gets time in another target timezone.

## Calendar properties

• int tm\_isleapyear (int year)

Indicates leap years.

• int tm\_getweeksinisoyear (int isoyear)

Returns the number of weeks in ISO year.

• int tm\_getdaysinmonth (int year, tm\_month month)

Returns the number of days in the specified month and year.

• int tm\_getsecondsinlocalday (int year, tm\_month month, int day)

Returns the number of seconds in the specified day, month and year.

• int tm\_getfirstweekdayinmonth (int year, tm\_month month, tm\_dayofweek dow)

Returns the day of the first weekday in the specified month.

• int tm\_getlastweekdayinmonth (int year, tm\_month month, tm\_dayofweek dow)

Returns the day of the last weekday in the specified month.

int tm\_getfirstweekdayinisoyear (int isoyear, tm\_dayofweek dow)

Returns the day of the first weekday in the specified ISO-year.

#### **Serializers**

• time t tm tobinary (struct tm date)

Serializes the instant of time to a binary value that subsequently can be used to recreate the instant of time.

• tm\_status tm\_frombinary (struct tm \*date, time\_t binary)

Deserializes a binary value and recreates an original serialized date and time.

- 4.2.1 Enumeration Type Documentation
- 4.2.1.1 enum tm\_dayofweek

Days of week.

#### **Enumerator**

TM\_WEEKDAY\_MONDAY Monday (1)

TM\_WEEKDAY\_TUESDAY Tuesday (2)

TM\_WEEKDAY\_WEDNESDAY Wednesday (3)

TM\_WEEKDAY\_THURSDAY Thursday (4)

TM\_WEEKDAY\_FRIDAY Friday (5)

TM\_WEEKDAY\_SATURDAY Saturday (6)

TM\_WEEKDAY\_SUNDAY Sunday (7)

#### 4.2.1.2 enum tm\_month

Months.

**Enumerator** 

TM\_MONTH\_JANUARY January (1)

TM\_MONTH\_FEBRUARY February (2)

TM\_MONTH\_MARCH March (3)

TM\_MONTH\_APRIL April (4)

TM\_MONTH\_MAY May (5)

TM\_MONTH\_JUNE June (6)

TM\_MONTH\_JULY July (7)

TM\_MONTH\_AUGUST August (8)

TM\_MONTH\_SEPTEMBER September (9)

TM\_MONTH\_OCTOBER October (10)

TM\_MONTH\_NOVEMBER November (11)

TM\_MONTH\_DECEMBER December (12)

#### 4.2.1.3 enum tm\_representation

Kinds of representation for instant in time.

**Enumerator** 

TM\_REP\_LOCAL Local time.
TM REP\_UTC UTC.

4.2.1.4 enum tm\_status

Values of status.

**Enumerator** 

TM\_OK Sucess.
TM\_ERROR Error.

4.2.2 Function Documentation

4.2.2.1 tm\_status tm\_adddays ( struct tm \* date, int nbDays )

Adds full days to the instant of time, changing year, month and day of month without altering hours, minutes and seconds (if possible).

It takes into account leap years and the number of days in a month.

Behavior depends on the current representation of the instant of time: in local time representation, adding one day might correspond to adding 23, 24 or 25 hours, depending whether or not there is a daylight saving time change. In case representation for both instants of time is local, days including a switch between standard time and daylight saving time count for 23 or 25 hours rather than 24. I.e., adding 14 days to march the 14th, 2016, 9 am, local Paris time, yields march the 28th, 9 am (rather than 10 am if a multiple of 24 hours were added), that is only 335 hours.

In order to add an exact multiple of 24 hours, use tm\_addseconds() instead.

In local time, if adding days results in an hour that is not valid in the resulting day (in case of daylight saving time change from winter to summer rule), an extra hour is added. For example, the transition from standard time to daylight saving time occurs in the U.S. Pacific Time zone on March 14, 2010, at 2:00 A.M., when the time advances by one hour, to 3:00 A.M. This hour interval is an invalid time, that is, a time interval that does not exist in this time zone. Thus, adding one day to March 13, 2010, 2:30 A.M. will result in March 14, 2010, 3:30 A.M, rather than 2:30 A.M.

#### **Parameters**

in,out	date	Pointer to broken-down time structure
in	nbDays	Number of days to add to date

#### Returns

TM\_OK or TM\_ERROR (in case of overflow)

#### Remarks

Behavior depends on time representation. Time representation is kept unchanged.

4.2.2.2 tm\_status tm\_addmonths ( struct tm \* date, int nbMonths )

Adds full months to the instant of time, changing year, month without altering day of month, hours, minutes and seconds (if possible).

It takes into account leap years and the number of days in a month, then adjusts the day part of the resulting instant in time. If adding months results in a day that is not a valid day in the resulting month, the last day of the resulting month is used. I.e., adding three months to January, the 31st, yields April, the 30th.

Behavior depends on the current representation of the instant of time: in local time representation, adding one day might correspond to adding 23, 24 or 25 hours, depending whether or not there is a daylight saving time change.

In local time, if adding days results in an hour that is not valid in the resulting day (in case of daylight saving time change from winter to summer rule), an extra hour is added. For example, the transition from standard time to daylight saving time occurs in the U.S. Pacific Time zone on March 14, 2010, at 2:00 A.M., when the time advances by one hour, to 3:00 A.M. This hour interval is an invalid time, that is, a time interval that does not exist in this time zone. Thus, adding one month to February 14, 2010, 2:30 A.M. will result in March 14, 2010, 3:30 A.M, rather than 2:30 A.M.

## **Parameters**

in,out	date	Pointer to broken-down time structure
in	nbMonths	Number of months to add to date

## Returns

TM\_OK or TM\_ERROR (in case of overflow)

## Remarks

Behavior depends on time representation. Time representation is kept unchanged.

4.2.2.3 tm\_status tm\_addseconds ( struct tm \* date, long int nbSecs )

Adds seconds to the instant of time.

#### **Parameters**

in,out	date	Pointer to broken-down time structure
in	nbSecs	Number of seconds to add to date

## Returns

TM\_OK or TM\_ERROR (in case of overflow)

## Remarks

Representation is kept unchanged (either local time or UTC).

4.2.2.4  $tm\_status\ tm\_addyears\ (\ struct\ tm\ *\ \textit{date,}\ int\ \textit{nbYears}\ )$ 

Adds full years to the instant of time, changing year without altering month, day of month, hours, minutes and seconds (if possible).

Behaves as if tm\_addmonths() were called with argument nbMonths equal to 12 x nbYears.

#### **Parameters**

in,out	date	Pointer to broken-down time structure
in	nbYears	Number of months to add to date

## Returns

TM\_OK or TM\_ERROR (in case of overflow)

## Remarks

Behavior depends on time representation. Time representation is kept unchanged.

4.2.2.5 int tm\_compare ( const void \* debut, const void \* fin )

Compares two dates.

# **Parameters**

i	n	debut	Pointer to broken-down time structure
i	n	fin	Pointer to broken-down time structure

## Returns

-1 if debut is before fin, 1 if debut is after fin, 0 if debut and fin are at same instant, independently of representation.

4.2 dates.h File Reference 35

#### Remarks

The representation of instants of time are not considered. Compatible for use with qsort().

4.2.2.6 int tm\_diffcalendardays ( struct tm debut, struct tm fin )

Gets number of partial days between two dates.

#### **Parameters**

in	debut	Broken-down time structure
in	fin	Broken-down time structure

# Returns

Number of partial or complete days between debut and fin.

#### Remarks

Partial days are counted as 1.

Behavior depends on time representation.

Both debut and fin should have identical representation, otherwise result is unspecified.

4.2.2.7 int tm\_diffcalendarmonths ( struct tm debut, struct tm fin )

Gets number of partial months between two dates.

### **Parameters**

in	debut	Broken-down time structure
in	fin	Broken-down time structure

#### Returns

Number of partial or complete months between debut and fin.

# Remarks

Partial months are counted as 1.

Behavior depends on time representation.

Both debut and fin should have identical representation, otherwise result is unspecified.

4.2.2.8 int tm\_diffcalendaryears ( struct tm debut, struct tm fin )

Gets number of partial years between two dates.

in	debut	Broken-down time structure
in	fin	Broken-down time structure

#### Returns

Number of partial or complete years between debut and fin.

#### Remarks

Partial years are counted as 1.

Behavior depends on time representation.

Both debut and fin should have identical representation, otherwise result is unspecified.

4.2.2.9 int tm\_diffdays ( struct tm debut, struct tm fin, int \* seconds )

Gets number of complete days between two dates.

In case representation for both indtants of time is local, days including between standard time and daylight saving time count for 23 or 25 hours rather than 24. I.e., difference between march the 14th, 9 am and march the 28th, 9 am, 2016, local Paris time, is 14 days, even though it includes only 335 hours.

#### **Parameters**

in	debut	Broken-down time structure
in	fin	Broken-down time structure
out	seconds	Remainder in seconds (optional)

#### Returns

Number of complete days between debut and fin.

#### Remarks

Partial days are counted as 0.

Behavior depends on time representation.

Both debut and fin should have identical representation, otherwise result is unspecified.

4.2.2.10 int tm\_diffisoyears ( struct tm debut, struct tm fin )

Gets number of partial ISO years between two dates.

#### **Parameters**

in	debut	Broken-down time structure
in	fin	Broken-down time structure

#### Returns

Number of partial or complete ISO years between debut and fin.

#### Remarks

Partial ISO years are counted as 1.

Behavior depends on time representation.

Both debut and fin should have identical representation, otherwise result is unspecified.

4.2 dates.h File Reference

37

4.2.2.11 int tm\_diffmonths ( struct tm debut, struct tm fin, int \* days, int \* seconds )

Gets number of complete months between two dates.

#### **Parameters**

in	debut	Broken-down time structure
in	fin	Broken-down time structure
out	days	Remainder in days (optional)
out	seconds	Remainder in seconds (optional)

## Returns

Number of complete months between debut and fin.

#### Remarks

Partial months are counted as 0.

Behavior depends on time representation.

Both debut and fin should have identical representation, otherwise result is unspecified.

4.2.2.12 long int tm\_diffseconds ( struct tm debut, struct tm fin )

Gets number of seconds between two dates.

#### **Parameters**

ir	debut	Broken-down time structure
ir	fin	Broken-down time structure

## Returns

Number of seconds between debut and fin. Negative of debut is after fin, positive if debut is before fin.

## Remarks

The representation of instants of time are not considered.

A difference of 0 seconds means debut and fin correspond to the same instant, independently of representation.

4.2.2.13 int tm\_diffweeks ( struct tm debut, struct tm fin, int \* days, int \* seconds )

Gets number of complete weeks between two dates.

## **Parameters**

	in	debut	Broken-down time structure
ſ	in	fin	Broken-down time structure
	out	days	Remainder in days (optional)
	out	seconds	Remainder in seconds (optional)

#### Returns

Number of complete weeks between debut and fin.

#### Remarks

Behavior depends on time representation.

Both debut and fin should have identical representation, otherwise result is unspecified.

4.2.2.14 int tm\_diffyears ( struct tm debut, struct tm fin, int \* months, int \* days, int \* seconds )

Gets number of complete years between two dates.

#### **Parameters**

in	debut	Broken-down time structure
in	fin	Broken-down time structure
out	months Remainder in months (optional)	
out	days Remainder in days (optional)	
out	seconds	Remainder in seconds (optional)

## Returns

Number of complete years between debut and fin.

## Remarks

Partial years are counted as 0.

Behavior depends on time representation.

Both debut and fin should have identical representation, otherwise result is unspecified.

4.2.2.15 int tm\_equals ( struct tm a, struct tm b )

Returns a value indicating whether two broken-down time have the same value (including representation).

## **Parameters**

in	а	Broken-down time structure
in	b	Broken-down time structure

## Remarks

Behavior depends on time representation. Use tm\_diffseconds() or tm\_compare() for an absolute date and time comparison.

## Returns

1 if the two broken-down time have the same value, 0 otherwise.

39

4.2.2.16  $tm\_status tm\_frombinary ( struct tm * date, time\_t binary )$ 

Deserializes a binary value and recreates an original serialized date and time.

## **Parameters**

out	date	Pointer to broken-down time structure, in local timezone representation
in	binary	representation of instant (point in time).

## Returns

TM\_OK on sucess, TM\_ERROR otherwise (overflow).

## Remarks

The instant (point in time) is presented in local time representation by default.

4.2.2.17 tm\_status tm\_getdateintostring ( struct tm dt, char \* str, size\_t max )

Formats date into string.

Formats the date and time according to the preferred date (without the time) display format for the current locale and places the result in the character array str of size max. str should have been previously allocated elsewhere.

## **Parameters**

in	dt	Broken-down time structure
in	max	Size of the previously allocated string str
out	str	null terminated string

## Returns

 ${\tt TM\_OK}$  if the result string, including the terminating null byte, does not exceed  ${\tt max}$  bytes,  ${\tt TM\_ERROR}$  otherwise (and the contents of the string  ${\tt str}$  are then undefined.)

## Remarks

Behavior depends on time representation. Makes call to strftime().

4.2.2.18 int tm\_getday ( struct tm date )

Gets the day of the month, in the Gregorian calendar.

## **Parameters**

in	date	Broken-down time structure

## Returns

Day of month

## Remarks

Behavior depends on time representation.

4.2.2.19 tm\_dayofweek tm\_getdayofweek ( struct tm date )

Gets day of week.

## **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

## Returns

Day of week (1 = Monday, 7 = Sunday)

#### Remarks

Behavior depends on time representation.

4.2.2.20 int tm\_getdayofyear ( struct tm date )

Gets day of year.

## **Parameters**

_			
	in	date	Broken-down time structure

## Returns

Day of year (1 = January, the 1st)

## Remarks

Behavior depends on time representation.

4.2.2.21 int tm\_getdaysinmonth ( int year, tm\_month month )

Returns the number of days in the specified month and year.

It interprets month and year as the month and year of the Gregorian calendar, taking leap years into account.

## **Parameters**

in	year	The year specified as a 4-digit number (for example, 1996), interpreted as a year in the Gregorian calendar.
in	month	Month

## Returns

Number of days in month month of year year

4.2.2.22 int tm\_getfirstweekdayinisoyear ( int isoyear, tm\_dayofweek dow )

Returns the day of the first weekday in the specified ISO-year.

## **Parameters**

in	isoyear	year
in	dow	Day of week

#### Returns

The day of the first weekday in the specified month.

4.2.2.23 int tm\_getfirstweekdayinmonth ( int year, tm\_month, tm\_dayofweek dow )

Returns the day of the first weekday in the specified month.

#### **Parameters**

in	year	Year
in	month	Month
in	dow	Day of week

## Returns

The day of the first weekday in the specified month.

4.2.2.24 int tm\_gethour ( struct tm date )

Gets hours.

#### **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

## Returns

Hours (between 0 and 23)

## Remarks

Behavior depends on time representation.

4.2.2.25 void tm\_getintimezone ( struct tm *date*, const char \* *tz*, int \* *year*, tm\_month \* *month*, int \* *day*, int \* *hour*, int \* *minute*, int \* *second*, int \* *is\_dst\_on* )

Gets time in another target timezone.

Daylight saving times are considered.

## **Parameters**

in	date	Broken-down time structure, either in local timezone or UTC representation
in	tz	Target timezone (see "man tzset" for details on possible values for tz)
out	year	Year at the time described in target timezone
out	month	Month at the time described in target timezone
out	day	Day at the time described in target timezone
out	hour	Hour at the time described in target timezone
out	minute	Minute at the time described in target timezone
out	second	Second at the time described in target timezone
out	is_dst_on	Indicates whether (1) or not (0) daylight saving time is in effect at the time described in
		target timezone

4.2.2.26 int tm\_getisoweek ( struct tm date )

Gets ISO week.

## **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

## Returns

ISO 8601 week

## Remarks

Behavior depends on time representation.

ISO 8601 week date: The first week of a year (starting on Monday) is :

- the first week that contains at least 4 days of calendar year.
- the week that contains the first Thursday of a year.
- · the week with January 4 in it

4.2.2.27 int tm\_getisoyear ( struct tm date )

Gets ISO year.

## **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

## Returns

ISO 8601 year

## Remarks

Behavior depends on time representation.

4.2.2.28 int tm\_getlastweekdayinmonth ( int year, tm\_month month, tm\_dayofweek dow )

Returns the day of the last weekday in the specified month.

## **Parameters**

in	year	Year
in	month	Month
in	dow	Day of week

## Returns

The day of the last weekday in the specified month.

4.2.2.29 int tm\_getminute ( struct tm date )

Gets minutes.

## **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

## Returns

Minutes (between 0 and 59)

## Remarks

Behavior depends on time representation.

4.2.2.30 tm\_month tm\_getmonth ( struct tm date )

Gets the month in year, in the Gregorian calendar.

## **Parameters**

in date Broken-down time structur
-----------------------------------

## Returns

```
Month (1 = January, ..., 12=December)
```

## Remarks

Behavior depends on time representation.

4.2.2.31 tm\_representation tm\_getrepresentation ( struct tm date )

Gets the current representation of instant in time.

#### **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

## Returns

Date and time representation (TM\_LOCAL or TM\_UTC)

4.2.2.32 int tm\_getsecond ( struct tm date )

Gets seconds.

## **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

## Returns

Seconds (between 0 and 59)

#### Remarks

Behavior depends on time representation.

4.2.2.33 int tm\_getsecondsinlocalday ( int year, tm\_month month, int day )

Returns the number of seconds in the specified day, month and year.

It interprets day, month and year as the day, month and year of the Gregorian calendar, taking leap years and daylight saving time tules into account.

#### **Parameters**

in	year	Year
in	month	Month
in	day	Day of month

## Returns

Number of seconds in day day of month month of year year, in local time

4.2.2.34 int tm\_getsecondsofday ( struct tm date )

Gets seconds of day.

#### **Parameters**

in	date	Broken-down time structure

## Returns

Elapsed seconds since beginning of day.

## Remarks

Behavior depends on time representation.

4.2.2.35 tm\_status tm\_gettimeintostring ( struct tm dt, char \* str, size\_t max )

Formats time into string.

Formats the date and time according to the preferred time (without the date) display format for the current locale and places the result in the character array str of size max. str should have been previously allocated elsewhere.

## **Parameters**

in	dt	Broken-down time structure	
in	max	Size of the previously allocated string str	
out	str	null terminated string.	

## Returns

 $\texttt{TM\_OK}$  if the result string, including the terminating null byte, does not exceed max bytes,  $\texttt{TM\_ERROR}$  otherwise (and the contents of the string str are then undefined.)

## Remarks

Behavior depends on time representation. Makes call to strftime().

4.2 dates.h File Reference 47

4.2.2.36 const char\* tm\_gettimezone ( struct tm date )

Gets the name of the time zone (either UTC or local time depending on current representation).

## **Parameters**

1n   <i>date</i>   Broken-down time structure	in	date	Broken-down time structure
---	----	------	----------------------------

## Returns

Timezone abbreviation

#### Remarks

Behavior depends on time representation.

4.2.2.37 int tm\_getutcoffset ( struct tm date )

Gets offset between UTC and local time.

## **Parameters**

in	date	Broken-down time structure
----	------	----------------------------

## Returns

Offset, in seconds, between UTC and time representation (local or UTC)

## Remarks

Behavior depends on time representation.

4.2.2.38 int tm\_getweeksinisoyear ( int isoyear )

Returns the number of weeks in ISO year.

## **Parameters**

in	isoyear	year

## Returns

The number of weeks in ISO year

4.2.2.39 int tm\_getyear ( struct tm date )

Gets the year, in the Gregorian calendar.

## **Parameters**

#### Returns

Year

#### Remarks

Behavior depends on time representation.

## 4.2.2.40 int tm\_hasdaylightsavingtimerules (void)

Indicates if the system local timezone does have any daylight saving time rules.

## Returns

0 if this timezone does not have any daylight saving time rules, or nonzero if there is a time, past, present or future when daylight saving time applies.

## Remarks

Behavior depends on time representation.

## 4.2.2.41 int tm\_isdaylightsavingextrasummertime ( struct tm date )

Indicates that date and time will be repeated after DST looses effect.

## **Parameters**

in	data	Broken-down time structure
T11	uale	Dioken-down time structure

## Returns

1 if time is duplicated (before DST change), 0 otherwise.

## Remarks

Behavior depends on time representation.

## 4.2.2.42 int tm\_isdaylightsavingextrawintertime ( struct tm date )

Indicates that date and time has already occured before DST lost effect.

## **Parameters**

in	date	Broken-down time structure

## Returns

1 if time is duplicated (after DST change), 0 otherwise.

## Remarks

Behavior depends on time representation.

4.2.2.43 int tm\_isdaylightsavingtime ( struct tm date )

Indicates that daylight saving time is in effect.

## **Parameters**

in date Broken-dov	vn time structure
--------------------	-------------------

## Returns

1 if DST is set, 0 otherwise.

## Remarks

Behavior depends on time representation. In UTC representation, 0 is returned.

4.2.2.44 int tm\_isleapyear ( int year )

Indicates leap years.

## Parameters

ın	year	year

## Returns

1 if year is a leap year, 0 otherwise

4.2.2.45 int tm\_islocalrepresentation ( struct tm date )

Indicates that the representation of instant in time is local time.

## **Parameters**

in	date	Broken-down time structure

## Returns

1 if date is in local time representation, 0 otherwise.

4.2.2.46 int tm\_isutcrepresentation ( struct tm date )

Indicates that the representation of instant in time is UTC.

## **Parameters**

in date Broken-down time	structure
--------------------------	-----------

## Returns

1 if date is in UTC representation, 0 otherwise.

4.2.2.47 tm\_status tm\_makelocal ( struct tm \* dt, int year, tm\_month, int day, int hour, int min, int sec )

Initializes (or reinitializes) instant intime with local date and time attributes.

## **Parameters**

in	year	The year, specified as a 4-digit number (for example, 1996), interpreted as a year in the Gregorian calendar (local time)
in	month	The month (local time)
in	day	The day (1 through the number of days in month) of month (local time)
in	hour	The hours (0 through 23) (local time)
in	min	The minutes (0 through 59) (local time)
in	sec	The seconds (0 through 59) (local time)
out	dt	Pointer to broken-down time structure

## Returns

TM\_OK or TM\_ERROR (in case of overflow or invalid arguments)

## Remarks

The instant (point in time) is initialized in local time representation by default.

4.2.2.48  $tm_status tm_makenow ( struct tm * dt )$ 

Initializes (or reinitializes) an instant in time with current date and time.

## **Parameters**

out	dt	Pointer to broken-down time structure
-----	----	---------------------------------------

## Returns

TM\_OK or TM\_ERROR (in case of overflow)

## Remarks

The instant (point in time) is initialized in local time representation by default.

## 4.2.2.49 tm\_status tm\_maketoday ( struct tm \* dt )

Initializes (or reinitializes) instant in time with current date, beginning of day, local time.

Initializes (or reinitializes) an instant in time set to today's date, with the time component set to 00:00:00, local time.

#### **Parameters**

	out	dt	Pointer to broken-down time structure	
--	-----	----	---------------------------------------	--

## Returns

TM\_OK or TM\_ERROR (in case of overflow)

## Remarks

The instant (point in time) is initialized in local time representation by default.

4.2.2.50 tm\_status tm\_makeutc ( struct tm \* dt, int year, tm\_month month, int day, int hour, int min, int sec )

Initializes (or reinitializes) instant in time with UTC date and time attributes.

#### **Parameters**

in	year	Year, UTC	
in	month	Month, UTC	
in	day	Day of month, UTC	
in	hour	Hour of day, UTC	
in	min	Minutes, UTC	
in	sec	Seconds, UTC	
out	dt	Pointer to broken-down time structure	

## Returns

TM\_OK or TM\_ERROR (in case of overflow or invalid arguments)

## Remarks

The instant (point in time) is initialized in UTC representation by default.

4.2.2.51 tm\_status tm\_set ( struct tm \* dt, int year, tm\_month month, int day, int hour, int min, int sec )

Sets instant in time with date and time attributes with regards to time representation.

#### **Parameters**

in	year	Year	
in	month	Month	
in	day	Day of month	
in	hour	Hour of day	
in	min	Minutes	
in	sec	Seconds	
out	dt	Pointer to broken-down time structure	

#### Returns

TM\_OK or TM\_ERROR (in case of overflow)

#### Remarks

Behavior depends on time representation. Time representation is kept unchanged.

4.2.2.52 tm\_status tm\_setdatefromstring ( struct tm \* dt, const char \* str )

Sets date from string.

Recognized formats are: the locale's date format, the locale's alternative date representation, the ISO 8601 date format (YYYY-mm-dd). A year specified on 2 digits is converted to the closest year on 4 digits.

#### **Parameters**

in	str	string representation of date (without time)
out	dt	Pointer to broken-down time structure

#### Returns

TM\_OK or TM\_ERROR (in case of overflow)

## Remarks

Behavior depends on time representation. Time representation is kept unchanged. Makes use of strptime().

4.2.2.53 tm\_status tm\_settimefromstring ( struct tm \* dt, const char \* str )

Sets time from string.

Recognized formats are: the locale's time format, the locale's alternative time representation, HH:MM:SS, HH:MM, (where HH is between 0 and 23)

#### **Parameters**

in	str	string representation of time (without date)
out	dt	Pointer to broken-down time structure

## Returns

TM\_OK or TM\_ERROR (in case of overflow)

## Remarks

Behavior depends on time representation. Time representation is kept unchanged.

4.2.2.54 time\_t tm\_tobinary ( struct tm date )

Serializes the instant of time to a binary value that subsequently can be used to recreate the instant of time.

This binary value is suitable for database recording. It identifies an instant of time unambiguously, whatever the representation (local time or UTC).

#### **Parameters**

in	date	Broken-down time structure, either in local timezone or UTC representation
----	------	--

#### Returns

Binary representation of instant (point in time).

4.2.2.55 tm\_status tm\_todaylightsavingextrasummertime ( struct tm \* date )

Changes the zone offset to the earlier of the two valid offsets at a local time-line overlap.

This function only has any effect when the local time-line overlaps, such as at an autumn daylight savings cutover. In this scenario, there are two valid offsets for the local date-time. Calling this method will return a local zoned date-time with the earlier of the two selected.

#### **Parameters**

in,out	date	Broken-down time structure
--------	------	----------------------------

## Returns

TM\_OK if instant was shifted, TM\_ERROR otherwise.

## Remarks

Behavior depends on time representation.

4.2.2.56 tm\_status tm\_todaylightsavingextrawintertime ( struct tm \* date )

Changes the zone offset to the later of the two valid offsets at a local time-line overlap.

This function only has any effect when the local time-line overlaps, such as at an autumn daylight savings cutover. In this scenario, there are two valid offsets for the local date-time. Calling this method will return a local zoned date-time with the later of the two selected.

## **Parameters**

in,out	date	Broken-down time structure
--------	------	----------------------------

## Returns

TM\_OK if instant was shifted, TM\_ERROR otherwise.

## Remarks

Behavior depends on time representation.

4.2.2.57 tm\_status tm\_tolocalrepresentation ( struct tm \* date )

Switches representation of instant in time to local time.

## **Parameters**

in,out dat	Pointer to broken-down time structure
------------	---------------------------------------

## Remarks

Has no effect if time representation is local time already.

4.2.2.58 tm\_status tm\_toutcrepresentation ( struct tm \* date )

Switches representation of instant in time to UTC.

#### **Parameters**

Pointer to broken-down time structu	date	in,out
-------------------------------------	------	--------

## Remarks

Has no effect if time representation is UTC already.

## Returns

TM\_OK on success, TM\_ERROR otherwise.

4.2.2.59 tm\_status tm\_trimtime ( struct tm \* date )

Sets the time value to 0am (beginning of the day) and keeps the date component unchanged.

## **Parameters**

in,out	date	Pointer to broken-down time structure

## Remarks

Behavior depends on time representation. Time representation is kept unchanged.

# Index

dates.c, 2	tm_normalizetoutc, 24
tm_adddays, 5	tm_set, 24
tm_addmonths, 5	tm_setdatefromstring, 25
tm_addseconds, 6	tm_settimefromstring, 25
tm_addyears, 6	tm_tobinary, 26
tm_compare, 7	tm_todaylightsavingextrasummertime, 26
tm_diffcalendardays, 7	tm_todaylightsavingextrawintertime, 26
tm_diffcalendarmonths, 7	tm_tolocalrepresentation, 27
tm_diffcalendaryears, 8	tm_toutcrepresentation, 27
tm_diffdays, 8	tm_trimtime, 27
tm_diffisoyears, 9	tm_utctimezone, 28
tm_diffmonths, 9	dates.h, 28
tm_diffseconds, 9	TM_ERROR, 32
tm_diffweeks, 10	TM_MONTH_APRIL, 32
tm_diffyears, 10	TM_MONTH_AUGUST, 32
tm_equals, 11	TM_MONTH_DECEMBER, 32
tm_frombinary, 11	TM_MONTH_FEBRUARY, 32
tm_getdateintostring, 11	TM_MONTH_JANUARY, 32
tm_getday, 12	TM_MONTH_JULY, 32
tm_getdayofweek, 12	TM_MONTH_JUNE, 32
tm_getdayofyear, 12	TM_MONTH_MARCH, 32
tm_getdaysinmonth, 13	TM_MONTH_MAY, 32
tm_getfirstweekdayinisoyear, 13	TM MONTH NOVEMBER, 32
tm_getfirstweekdayinmonth, 13	TM_MONTH_OCTOBER, 32
tm_gethour, 14	TM_MONTH_SEPTEMBER, 32
tm_getintimezone, 14	TM OK, 32
tm getisoweek, 14	TM REP LOCAL, 32
tm_getisoyear, 15	TM REP UTC, 32
tm_getlastweekdayinmonth, 15	TM WEEKDAY FRIDAY, 31
tm_getminute, 15	TM WEEKDAY MONDAY, 31
tm_getmonth, 16	TM_WEEKDAY_SATURDAY, 31
tm_getrepresentation, 16	TM WEEKDAY SUNDAY, 31
tm_getsecond, 16	TM_WEEKDAY_THURSDAY, 31
tm_getsecondsinlocalday, 17	TM_WEEKDAY_TUESDAY, 31
tm_getsecondsofday, 17	TM_WEEKDAY_WEDNESDAY, 31
tm_gettimeintostring, 17	tm_adddays, 32
tm gettimezone, 18	tm_addmonths, 33
tm_getutcoffset, 18	tm_addseconds, 33
tm_getweeksinisoyear, 18	tm_addyears, 34
tm_getyear, 19	tm_compare, 34
tm_hasdaylightsavingtimerules, 19	tm_dayofweek, 31
tm_isdaylightsavingextrasummertime, 19	tm_diffcalendardays, 35
tm_isdaylightsavingextrawintertime, 20	tm_diffcalendarmonths, 35
tm_isdaylightsavingtime, 20	tm_diffcalendaryears, 35
tm_isleapyear, 20	tm diffdays, 36
tm_islocalrepresentation, 21	tm_diffisoyears, 36
tm isutcrepresentation, 21	tm_diffmonths, 36
tm_makelocal, 21	tm_diffseconds, 37
tm_makelocalfromcalendartime, 22	tm_diffweeks, 37
tm_makenow, 22	tm_diffyears, 38
tm_maketoday, 22	tm equals, 38
tm_makeutc, 23	tm_frombinary, 38
tm_makeutcfromcalendartime, 23	tm_getdateintostring, 40
tm normalize, 23	tm getday, 40
tm_normalizetolocal, 24	tm_getdayofweek, 41
_ ,	

58 INDEX

tm_getdayofyear, 41	dates.h, 32
tm_getdaysinmonth, 41	TM_MONTH_MARCH
tm_getfirstweekdayinisoyear, 42	dates.h, 32
tm_getfirstweekdayinmonth, 42	TM_MONTH_MAY
tm_gethour, 42	dates.h, 32
tm_getintimezone, 42	TM_MONTH_NOVEMBER
tm_getisoweek, 43	dates.h, 32
tm_getisoyear, 43	TM_MONTH_OCTOBER
tm_getlastweekdayinmonth, 44	dates.h, 32
tm_getminute, 44	TM_MONTH_SEPTEMBER
tm_getmonth, 44	dates.h, 32
tm_getrepresentation, 45	TM_OK
tm_getsecond, 45	dates.h, 32
tm_getsecondsinlocalday, 45	TM_REP_LOCAL
tm_getsecondsofday, 46	dates.h, 32
tm_gettimeintostring, 46	TM_REP_UTC
tm_gettimezone, 46	dates.h, 32
tm_getutcoffset, 48	TM_WEEKDAY_FRIDAY
tm_getweeksinisoyear, 48	dates.h, 31
tm_getyear, 48	TM_WEEKDAY_MONDAY
tm_hasdaylightsavingtimerules, 49	dates.h, 31
tm_isdaylightsavingextrasummertime, 49	TM_WEEKDAY_SATURDAY
tm_isdaylightsavingextrawintertime, 49	dates.h, 31
tm_isdaylightsavingtime, 50	TM_WEEKDAY_SUNDAY
tm_isleapyear, 50	dates.h, 31
tm_islocalrepresentation, 50	TM_WEEKDAY_THURSDAY
tm_isutcrepresentation, 50	dates.h, 31
tm_makelocal, 51	TM_WEEKDAY_TUESDAY
tm_makenow, 51	dates.h, 31
tm_maketoday, 51	TM_WEEKDAY_WEDNESDAY
tm_makeutc, 52	dates.h, 31
tm_month, 31	tm_adddays
tm_representation, 32	dates.c, 5
tm_set, 52	dates.h, 32
tm_setdatefromstring, 53	tm addmonths
tm_settimefromstring, 53	dates.c, 5
tm_status, 32	dates.h, 33
tm_tobinary, 53	tm addseconds
tm_todaylightsavingextrasummertime, 54	dates.c, 6
tm_todaylightsavingextrawintertime, 54	dates.h, 33
tm_tolocalrepresentation, 55	tm addyears
tm_toutcrepresentation, 55	dates.c, 6
tm_trimtime, 55	dates.h, 34
TM ERROR	tm_compare
dates.h, 32	dates.c, 7
TM MONTH APRIL	dates.h, 34
dates.h, 32	
•	tm_davofweek
	tm_dayofweek
TM_MONTH_AUGUST	dates.h, 31
dates.h, 32	dates.h, 31 tm_diffcalendardays
dates.h, 32 TM_MONTH_DECEMBER	dates.h, 31 tm_diffcalendardays dates.c, 7
dates.h, 32 TM_MONTH_DECEMBER dates.h, 32	dates.h, 31 tm_diffcalendardays dates.c, 7 dates.h, 35
dates.h, 32 TM_MONTH_DECEMBER dates.h, 32 TM_MONTH_FEBRUARY	dates.h, 31 tm_diffcalendardays dates.c, 7 dates.h, 35 tm_diffcalendarmonths
dates.h, 32  TM_MONTH_DECEMBER dates.h, 32  TM_MONTH_FEBRUARY dates.h, 32	dates.h, 31 tm_diffcalendardays dates.c, 7 dates.h, 35 tm_diffcalendarmonths dates.c, 7
dates.h, 32  TM_MONTH_DECEMBER dates.h, 32  TM_MONTH_FEBRUARY dates.h, 32  TM_MONTH_JANUARY	dates.h, 31 tm_diffcalendardays dates.c, 7 dates.h, 35 tm_diffcalendarmonths dates.c, 7 dates.h, 35
dates.h, 32  TM_MONTH_DECEMBER dates.h, 32  TM_MONTH_FEBRUARY dates.h, 32  TM_MONTH_JANUARY dates.h, 32	dates.h, 31 tm_diffcalendardays dates.c, 7 dates.h, 35 tm_diffcalendarmonths dates.c, 7 dates.h, 35 tm_diffcalendaryears
dates.h, 32  TM_MONTH_DECEMBER dates.h, 32  TM_MONTH_FEBRUARY dates.h, 32  TM_MONTH_JANUARY dates.h, 32  TM_MONTH_JULY	dates.h, 31 tm_diffcalendardays dates.c, 7 dates.h, 35 tm_diffcalendarmonths dates.c, 7 dates.h, 35 tm_diffcalendaryears dates.c, 8
dates.h, 32  TM_MONTH_DECEMBER dates.h, 32  TM_MONTH_FEBRUARY dates.h, 32  TM_MONTH_JANUARY dates.h, 32	dates.h, 31 tm_diffcalendardays dates.c, 7 dates.h, 35 tm_diffcalendarmonths dates.c, 7 dates.h, 35 tm_diffcalendaryears

INDEX 59

dates.c, 8	dates.h, 44
dates.h, 36	tm_getminute
tm_diffisoyears	dates.c, 15
dates.c, 9	dates.h, 44
dates.h, 36	tm_getmonth
tm_diffmonths	dates.c, 16
dates.c, 9	dates.h, 44
dates.h, 36	tm_getrepresentation
tm_diffseconds	dates.c, 16
dates.c, 9	dates.h, 45
dates.h, 37	tm_getsecond
tm_diffweeks	dates.c, 16
dates.c, 10	dates.h, 45
dates.h, 37	tm_getsecondsinlocalday
tm_diffyears	dates.c, 17
dates.c, 10	dates.h, 45
dates.h, 38	tm_getsecondsofday
tm_equals	dates.c, 17
dates.c, 11	dates.h, 46
dates.h, 38	tm_gettimeintostring
tm_frombinary	dates.c, 17
dates.c, 11	dates.h, 46
dates.h, 38	tm_gettimezone
tm_getdateintostring	dates.c, 18
dates.c, 11	dates.h, 46
dates.h, 40	tm_getutcoffset
tm_getday	dates.c, 18
dates.c, 12	dates.h, 48
dates.h, 40	tm_getweeksinisoyear
tm_getdayofweek	dates.c, 18
dates.c, 12	dates.h, 48
dates.h, 41	tm_getyear
tm_getdayofyear	dates.c, 19
dates.c, 12	dates.h, 48
dates.h, 41	tm_hasdaylightsavingtimerules
tm_getdaysinmonth	dates.c, 19
dates.c, 13	dates.h, 49
dates.h, 41	tm_isdaylightsavingextrasummertime
tm_getfirstweekdayinisoyear	dates.c, 19
dates.c, 13	dates.h, 49
dates.h, 42	tm_isdaylightsavingextrawintertime
tm_getfirstweekdayinmonth	dates.c, 20
dates.c, 13	dates.h, 49
dates.h, 42	tm_isdaylightsavingtime
tm_gethour	dates.c, 20
dates.c, 14	dates.h, 50
dates.h, 42	tm_isleapyear
tm_getintimezone	dates.c, 20
dates.c, 14	dates.h, 50
dates.h, 42	tm_islocalrepresentation
tm_getisoweek	dates.c, 21
dates.c, 14	dates.h, 50
dates.h, 43	tm_isutcrepresentation
tm_getisoyear	dates.c, 21
dates.c, 15	dates.h, 50
dates.h, 43	tm_makelocal
tm_getlastweekdayinmonth	dates.c, 21
dates.c, 15	dates.h, 51

60 INDEX

```
tm_makelocalfromcalendartime
    dates.c, 22
tm_makenow
    dates.c, 22
    dates.h, 51
tm maketoday
    dates.c, 22
    dates.h, 51
tm makeutc
    dates.c, 23
    dates.h, 52
tm_makeutcfromcalendartime
    dates.c, 23
tm_month
    dates.h, 31
tm_normalize
    dates.c, 23
tm normalizetolocal
    dates.c, 24
tm_normalizetoutc
    dates.c, 24
tm_representation
    dates.h, 32
tm_set
    dates.c, 24
    dates.h, 52
tm_setdatefromstring
    dates.c, 25
    dates.h, 53
tm_settimefromstring
    dates.c, 25
    dates.h, 53
tm status
    dates.h, 32
tm_tobinary
    dates.c, 26
    dates.h, 53
tm_todaylightsavingextrasummertime
    dates.c, 26
     dates.h, 54
tm_todaylightsavingextrawintertime
    dates.c, 26
    dates.h, 54
tm_tolocalrepresentation
    dates.c, 27
    dates.h, 55
tm_toutcrepresentation
    dates.c, 27
    dates.h, 55
tm_trimtime
    dates.c, 27
    dates.h, 55
tm utctimezone
    dates.c, 28
```