Dates tool box

Generated by Doxygen 1.8.11

Contents

1 Introduction 1

2 Usage 2

3 File Index 2

3.1 File List . . . 2

4 File Documentation 2

4.1 dates.c File Reference . . . 2

4.1.1 Function Documentation . . . 5

4.2 dates.h File Reference . . . 27

4.2.1 Enumeration Type Documentation . . . 30

4.2.2 Function Documentation . . . 31

Index 55

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 initializeed 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

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

dates.c 2

dates.h 27

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.
- 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)

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.
- 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.

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.

4.1.1.10 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.

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.

4.1.1.15 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`	*a*	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

TM_OK if the result string, including the terminating null byte, does not exceed max bytes, 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.

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

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 `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.

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

4.1.1.27 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.

Parameters

in	*date*	Broken-down time structure

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

TM_OK if the result string, including the terminating null byte, does not exceed max bytes, 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.

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

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.

Parameters

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.

Parameters

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.

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.

4.1.1.51 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

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.

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 * *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.

4.1.1.59 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

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

Returns

Binary representation of instant (point in time).

4.1.1.60 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.

4.1.1.61 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.

4.1.1.62 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.

4.1.1.63 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.

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 * *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.

4.2.2.5 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().

4.2.2.6 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.

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.

4.2.2.9 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.

4.2.2.11 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

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.

4.2.2.13 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
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`	*a*	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.

4.2.2.16 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

`TM_OK` if the result string, including the terminating null byte, does not exceed `max` bytes, `TM_ERROR` otherwise (and the contents of the string `str` are then undefined.)

Remarks

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

4.2.2.18 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 *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 structure

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

`TM_OK` if the result string, including the terminating null byte, does not exceed `max` bytes, `TM_ERROR` otherwise (and the contents of the string `str` are then undefined.)

Remarks

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

4.2.2.36 const char* tm_gettimezone (struct tm *date*)

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

Parameters

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

in	*date*	Broken-down time structure

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	*date*	Broken-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-down time structure

Returns

1 if DST is set, 0 otherwise.

Remarks

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

4.2.2.44 int tm_isleapyear (int *year*)

Indicates leap years.

Parameters

in	*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 *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_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.

4.2.2.56 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.

4.2.2.57 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_adddays, 5
- tm_addmonths, 5
- tm_addseconds, 6
- tm_addyears, 6
- tm_compare, 7
- tm_diffcalendardays, 7
- tm_diffcalendarmonths, 7
- tm_diffcalendaryears, 8
- tm_diffdays, 8
- tm_diffisoyears, 9
- tm_diffmonths, 9
- tm_diffseconds, 9
- tm_diffweeks, 10
- tm_diffyears, 10
- tm_equals, 11
- tm_frombinary, 11
- tm_getdateintostring, 11
- tm_getday, 12
- tm_getdayofweek, 12
- tm_getdayofyear, 12
- tm_getdaysinmonth, 13
- tm_getfirstweekdayinisoyear, 13
- tm_getfirstweekdayinmonth, 13
- tm_gethour, 14
- tm_getintimezone, 14
- tm_getisoweek, 14
- tm_getisoyear, 15
- tm_getlastweekdayinmonth, 15
- tm_getminute, 15
- tm_getmonth, 16
- tm_getrepresentation, 16
- tm_getsecond, 16
- tm_getsecondsinlocalday, 17
- tm_getsecondsofday, 17
- tm_gettimeintostring, 17
- tm_gettimezone, 18
- tm_getutcoffset, 18
- tm_getweeksinisoyear, 18
- tm_getyear, 19
- tm_hasdaylightsavingtimerules, 19
- tm_isdaylightsavingextrasummertime, 19
- tm_isdaylightsavingextrawintertime, 20
- tm_isdaylightsavingtime, 20
- tm_isleapyear, 20
- tm_islocalrepresentation, 21
- tm_isutcrepresentation, 21
- tm_makelocal, 21
- tm_makelocalfromcalendartime, 22
- tm_makenow, 22
- tm_maketoday, 22
- tm_makeutc, 23
- tm_makeutcfromcalendartime, 23
- tm_normalize, 23
- tm_normalizetolocal, 24
- tm_normalizetoutc, 24
- tm_set, 24
- tm_setdatefromstring, 25
- tm_settimefromstring, 25
- tm_tobinary, 26
- tm_tolocalrepresentation, 26
- tm_toutcrepresentation, 26
- tm_trimtime, 27
- tm_utctimezone, 27

dates.h, 27
- TM_ERROR, 31
- TM_MONTH_APRIL, 31
- TM_MONTH_AUGUST, 31
- TM_MONTH_DECEMBER, 31
- TM_MONTH_FEBRUARY, 31
- TM_MONTH_JANUARY, 31
- TM_MONTH_JULY, 31
- TM_MONTH_JUNE, 31
- TM_MONTH_MARCH, 31
- TM_MONTH_MAY, 31
- TM_MONTH_NOVEMBER, 31
- TM_MONTH_OCTOBER, 31
- TM_MONTH_SEPTEMBER, 31
- TM_OK, 31
- TM_REP_LOCAL, 31
- TM_REP_UTC, 31
- TM_WEEKDAY_FRIDAY, 30
- TM_WEEKDAY_MONDAY, 30
- TM_WEEKDAY_SATURDAY, 30
- TM_WEEKDAY_SUNDAY, 30
- TM_WEEKDAY_THURSDAY, 30
- TM_WEEKDAY_TUESDAY, 30
- TM_WEEKDAY_WEDNESDAY, 30
- tm_adddays, 31
- tm_addmonths, 32
- tm_addseconds, 32
- tm_addyears, 33
- tm_compare, 33
- tm_dayofweek, 30
- tm_diffcalendardays, 34
- tm_diffcalendarmonths, 34
- tm_diffcalendaryears, 34
- tm_diffdays, 35
- tm_diffisoyears, 35
- tm_diffmonths, 35
- tm_diffseconds, 36
- tm_diffweeks, 36
- tm_diffyears, 37
- tm_equals, 37
- tm_frombinary, 37
- tm_getdateintostring, 39
- tm_getday, 39
- tm_getdayofweek, 40
- tm_getdayofyear, 40
- tm_getdaysinmonth, 40

tm_getfirstweekdayinisoyear, 41
tm_getfirstweekdayinmonth, 41
tm_gethour, 41
tm_getintimezone, 41
tm_getisoweek, 42
tm_getisoyear, 42
tm_getlastweekdayinmonth, 43
tm_getminute, 43
tm_getmonth, 43
tm_getrepresentation, 44
tm_getsecond, 44
tm_getsecondsinlocalday, 44
tm_getsecondsofday, 45
tm_gettimeintostring, 45
tm_gettimezone, 45
tm_getutcoffset, 47
tm_getweeksinisoyear, 47
tm_getyear, 47
tm_hasdaylightsavingtimerules, 48
tm_isdaylightsavingextrasummertime, 48
tm_isdaylightsavingextrawintertime, 48
tm_isdaylightsavingtime, 49
tm_isleapyear, 49
tm_islocalrepresentation, 49
tm_isutcrepresentation, 49
tm_makelocal, 50
tm_makenow, 50
tm_maketoday, 50
tm_makeutc, 51
tm_month, 30
tm_representation, 31
tm_set, 51
tm_setdatefromstring, 52
tm_settimefromstring, 52
tm_status, 31
tm_tobinary, 52
tm_tolocalrepresentation, 53
tm_toutcrepresentation, 53
tm_trimtime, 53

TM_ERROR
dates.h, 31
TM_MONTH_APRIL
dates.h, 31
TM_MONTH_AUGUST
dates.h, 31
TM_MONTH_DECEMBER
dates.h, 31
TM_MONTH_FEBRUARY
dates.h, 31
TM_MONTH_JANUARY
dates.h, 31
TM_MONTH_JULY
dates.h, 31
TM_MONTH_JUNE
dates.h, 31
TM_MONTH_MARCH
dates.h, 31
TM_MONTH_MAY
dates.h, 31
TM_MONTH_NOVEMBER
dates.h, 31
TM_MONTH_OCTOBER
dates.h, 31
TM_MONTH_SEPTEMBER
dates.h, 31
TM_OK
dates.h, 31
TM_REP_LOCAL
dates.h, 31
TM_REP_UTC
dates.h, 31
TM_WEEKDAY_FRIDAY
dates.h, 30
TM_WEEKDAY_MONDAY
dates.h, 30
TM_WEEKDAY_SATURDAY
dates.h, 30
TM_WEEKDAY_SUNDAY
dates.h, 30
TM_WEEKDAY_THURSDAY
dates.h, 30
TM_WEEKDAY_TUESDAY
dates.h, 30
TM_WEEKDAY_WEDNESDAY
dates.h, 30
tm_adddays
dates.c, 5
dates.h, 31
tm_addmonths
dates.c, 5
dates.h, 32
tm_addseconds
dates.c, 6
dates.h, 32
tm_addyears
dates.c, 6
dates.h, 33
tm_compare
dates.c, 7
dates.h, 33
tm_dayofweek
dates.h, 30
tm_diffcalendardays
dates.c, 7
dates.h, 34
tm_diffcalendarmonths
dates.c, 7
dates.h, 34
tm_diffcalendaryears
dates.c, 8
dates.h, 34
tm_diffdays
dates.c, 8
dates.h, 35
tm_diffisoyears
dates.c, 9

dates.h, 35
tm_diffmonths
dates.c, 9
dates.h, 35
tm_diffseconds
dates.c, 9
dates.h, 36
tm_diffweeks
dates.c, 10
dates.h, 36
tm_diffyears
dates.c, 10
dates.h, 37
tm_equals
dates.c, 11
dates.h, 37
tm_frombinary
dates.c, 11
dates.h, 37
tm_getdateintostring
dates.c, 11
dates.h, 39
tm_getday
dates.c, 12
dates.h, 39
tm_getdayofweek
dates.c, 12
dates.h, 40
tm_getdayofyear
dates.c, 12
dates.h, 40
tm_getdaysinmonth
dates.c, 13
dates.h, 40
tm_getfirstweekdayinisoyear
dates.c, 13
dates.h, 41
tm_getfirstweekdayinmonth
dates.c, 13
dates.h, 41
tm_gethour
dates.c, 14
dates.h, 41
tm_getintimezone
dates.c, 14
dates.h, 41
tm_getisoweek
dates.c, 14
dates.h, 42
tm_getisoyear
dates.c, 15
dates.h, 42
tm_getlastweekdayinmonth
dates.c, 15
dates.h, 43
tm_getminute
dates.c, 15
dates.h, 43
tm_getmonth
dates.c, 16
dates.h, 43
tm_getrepresentation
dates.c, 16
dates.h, 44
tm_getsecond
dates.c, 16
dates.h, 44
tm_getsecondsinlocalday
dates.c, 17
dates.h, 44
tm_getsecondsofday
dates.c, 17
dates.h, 45
tm_gettimeintostring
dates.c, 17
dates.h, 45
tm_gettimezone
dates.c, 18
dates.h, 45
tm_getutcoffset
dates.c, 18
dates.h, 47
tm_getweeksinisoyear
dates.c, 18
dates.h, 47
tm_getyear
dates.c, 19
dates.h, 47
tm_hasdaylightsavingtimerules
dates.c, 19
dates.h, 48
tm_isdaylightsavingextrasummertime
dates.c, 19
dates.h, 48
tm_isdaylightsavingextrawintertime
dates.c, 20
dates.h, 48
tm_isdaylightsavingtime
dates.c, 20
dates.h, 49
tm_isleapyear
dates.c, 20
dates.h, 49
tm_islocalrepresentation
dates.c, 21
dates.h, 49
tm_isutcrepresentation
dates.c, 21
dates.h, 49
tm_makelocal
dates.c, 21
dates.h, 50
tm_makelocalfromcalendartime
dates.c, 22
tm_makenow
dates.c, 22

dates.h, 50
tm_maketoday
dates.c, 22
dates.h, 50
tm_makeutc
dates.c, 23
dates.h, 51
tm_makeutcfromcalendartime
dates.c, 23
tm_month
dates.h, 30
tm_normalize
dates.c, 23
tm_normalizetolocal
dates.c, 24
tm_normalizetoutc
dates.c, 24
tm_representation
dates.h, 31
tm_set
dates.c, 24
dates.h, 51
tm_setdatefromstring
dates.c, 25
dates.h, 52
tm_settimefromstring
dates.c, 25
dates.h, 52
tm_status
dates.h, 31
tm_tobinary
dates.c, 26
dates.h, 52
tm_tolocalrepresentation
dates.c, 26
dates.h, 53
tm_toutcrepresentation
dates.c, 26
dates.h, 53
tm_trimtime
dates.c, 27
dates.h, 53
tm_utctimezone
dates.c, 27