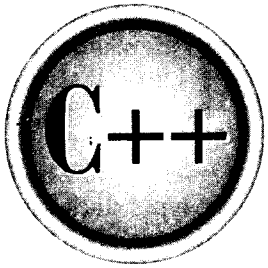
The Complete Reference



Chapter 28

Time, Date, and Localization Functions

The standard function library defines several functions that deal with the date and time. It also defines functions that handle the geopolitical information associated with a program. These functions are described here.

The time and date functions require the header **<ctime>**. (A C program must use the header file **time.h**.) This header defines three time-related types: **clock_t**, **time_t**, and **tm**. The types **clock_t** and **time_t** are capable of representing the system time and date as some sort of integer. This is called the *calendar time*. The structure type **tm** holds the date and time broken down into its elements. The **tm** structure is defined as shown here:

```
struct tm {
  int tm_sec;  /* seconds, 0-61 */
  int tm_min;  /* minutes, 0-59 */
  int tm_hour; /* hours, 0-23 */
  int tm_mday; /* day of the month, 1-31 */
  int tm_mon;  /* months since Jan, 0-11 */
  int tm_year; /* years from 1900 */
  int tm_wday; /* days since Sunday, 0-6 */
  int tm_yday; /* days since Jan 1, 0-365 */
  int tm_isdst /* Daylight Saving Time
                  indicator */
}
```

The value of **tm_isdst** will be positive if daylight saving time is in effect, zero if it is not in effect, and negative if there is no information available. This form of the time and date is called the *broken-down time*.

In addition, **<ctime>** defines the macro **CLOCKS_PER_SEC**, which is the number of system clock ticks per second.

The geopolitical environmental functions require the header **<clocale>**. (A C program must use the header file **locale.h**.)

asctime

```
#include <ctime>
char *asctime(const struct tm *ptr);
```

The **asctime( )** function returns a pointer to a string that contains the information stored in the structure pointed to by *ptr* converted into the following form:

day month date hours:minutes:seconds year\n\0

For example:

```
Fri Apr 15 12:05:34 2005
```

The structure pointer passed to **asctime( )** is usually obtained from either **localtime( )** or **gmtime( )**.

The buffer used by **asctime( )** to hold the formatted output string is a statically allocated character array and is overwritten each time the function is called. If you wish to save the contents of the string, you must copy it elsewhere.

Related functions are **localtime( )**, **gmtime( )**, **time( )**, and **ctime( )**.

clock

```
#include <ctime>
clock_t clock(void);
```

The **clock( )** function returns a value that approximates the amount of time the calling program has been running. To transform this value into seconds, divide it by **CLOCKS_PER_SEC**. A value of –1 is returned if the time is not available.

Related functions are **time( )**, **asctime( )**, and **ctime( )**.

ctime

```
#include <ctime>
char *ctime(const time_t *time);
```

The **ctime( )** function returns a pointer to a string of the form

day month year hours:minutes:seconds year\n\0

given a pointer to the calendar time. The calendar time is often obtained through a call to **time( )**.

The buffer used by **ctime( )** to hold the formatted output string is a statically allocated character array and is overwritten each time the function is called. If you wish to save the contents of the string, it is necessary to copy it elsewhere.

Related functions are **localtime( )**, **gmtime( )**, **time( )**, and **asctime( )**.

difftime

```
#include <ctime>
double difftime(time_t time2, time_t time1);
```

The **difftime( )** function returns the difference, in seconds, between *time1* and *time2*. That is, *time2 –time1*.

Related functions are **localtime( )**, **gmtime( )**, **time( )**, **asctime( )**.

gmtime

```
#include <ctime>
struct tm *gmtime(const time_t *time);
```

The **gmtime( )** function returns a pointer to the broken-down form of *time* in the form of a **tm** structure. The time is represented in Coordinated Universal Time (UTC), which is essentially Greenwich mean time. The *time* value is usually obtained through a call to **time( )**. If the system does not support UTC, **NULL** is returned.

The structure used by **gmtime( )** to hold the broken-down time is statically allocated and is overwritten each time the function is called. If you wish to save the contents of the structure, you must copy it elsewhere.

Related functions are **localtime( )**, **time( )**, and **asctime( )**.

localeconv

```
#include <clocale>
struct lconv *localeconv(void);
```

The **localeconv( )** function returns a pointer to a structure of type **lconv**, which contains various geopolitical environmental information relating to the way numbers are formatted. The **lconv** structure is organized as shown here:

```
struct lconv {
  char *decimal_point;      /* decimal point character
                               for nonmonetary values */
  char *thousands_sep;      /* thousands separator
                               for nonmonetary values */
  char *grouping;           /* specifies grouping for
```

```
                                nonmonetary values */
    char *int_curr_symbol;    /* international currency symbol */
    char *currency_symbol;    /* local currency symbol */
    char *mon_decimal_point; /* decimal point character for
                                monetary values */
    char *mon_thousands_sep; /* thousands separator for
                                monetary values */
    char *mon_grouping;       /* specifies grouping for
                                monetary values */
    char *positive_sign;      /* positive value indicator for
                                monetary values */
    char *negative_sign;      /* negative value indicator for
                                monetary values */
    char int_frac_digits;     /* number of digits displayed to the
                                right of the decimal point for
                                monetary values displayed using
                                international format */
    char frac_digits;         /* number of digits displayed to the
                                right of the decimal point for
                                monetary values displayed using
                                local format */
    char p_cs_precedes;       /* 1 if currency symbol precedes
                                positive value, 0 if currency
                                symbol follows value */
    char p_sep_by_space;      /* 1 if currency symbol is
                                separated from value by a space,
                                0 otherwise */
    char n_cs_precedes;       /* 1 if currency symbol precedes
                                a negative value, 0 if currency
                                symbol follows value */
    char n_sep_by_space;      /* 1 if currency symbol is
                                separated from a negative
                                value by a space, 0 if
                                currency symbol follows value */
    char p_sign_posn;         /* indicates position of
                                positive value symbol */
    char n_sign_posn;         /* indicates position of
                                negative value symbol */
}
```

The **localeconv( )** function returns a pointer to the **lconv** structure. You must not alter the contents of this structure. Refer to your compiler's documentation for implementation-specific information relating to the **lconv** structure.

A related function is **setlocale( )**.

localtime

```
#include <ctime>
struct tm *localtime(const time_t *time);
```

The **localtime( )** function returns a pointer to the broken-down form of *time* in the form of a **tm** structure. The time is represented in local time. The *time* value is usually obtained through a call to **time( )**.

The structure used by **localtime( )** to hold the broken-down time is statically allocated and is overwritten each time the function is called. If you wish to save the contents of the structure, you must copy it elsewhere.

Related functions are **gmtime( )**, **time( )**, and **asctime( )**.

mktime

```
#include <ctime>
time_t mktime(struct tm *time);
```

The **mktime( )** function returns the calendar-time equivalent of the broken-down time found in the structure pointed to by *time*. The elements **tm_wday** and **tm_yday** are set by the function, so they need not be defined at the time of the call.

If **mktime( )** cannot represent the information as a valid calendar time, –1 is returned.

Related functions are **time( )**, **gmtime( )**, **asctime( )**, and **ctime( )**.

setlocale

```
#include <clocale>
char *setlocale(int type, const char *locale);
```

The **setlocale( )** function allows certain parameters that are sensitive to the geopolitical environment of a program's execution to be queried or set. If *locale* is null, **setlocale( )** returns a pointer to the current localization string. Otherwise, **setlocale( )** attempts to use the string specified by *locale* to set the locale parameters as specified by *type*. Refer to your compiler's documentation for the localization strings that it supports.

At the time of the call, *type* must be one of the following macros:

LC_ALL
LC_COLLATE
LC_CTYPE
LC_MONETARY
LC_NUMERIC
LC_TIME

LC_ALL refers to all localization categories. **LC_COLLATE** affects the operation of the **strcoll( )** function. **LC_CTYPE** alters the way the character functions work. **LC_MONETARY** determines the monetary format. **LC_NUMERIC** changes the decimal-point character for formatted input/output functions. Finally, **LC_TIME** determines the behavior of the **strftime( )** function.

The **setlocale( )** function returns a pointer to a string associated with the *type* parameter.

Related functions are **localeconv( )**, **time( )**, **strcoll( )**, and **strftime( )**.

strftime

```
#include <ctime>
size_t strftime(char *str, size_t maxsize, const char *fmt,
                const struct tm *time);
```

The **strftime( )** function places time and date information, along with other information, into the string pointed to by *str* according to the format commands found in the string pointed to by *fmt* and using the broken-down time *time*. A maximum of *maxsize* characters will be placed into *str*.

The **strftime( )** function works a little like **sprintf( )** in that it recognizes a set of format commands that begin with the percent sign (%) and places its formatted output into a string. The format commands are used to specify the exact way various time and date information is represented in *str*. Any other characters found in the format string are placed into *str* unchanged. The time and date displayed are in local time. The format commands are shown in the table below. Notice that many of the commands are case sensitive.

The **strftime( )** function returns the number of characters placed in the string pointed to by *str* or zero if an error occurs.

Command	Replaced By
%a	Abbreviated weekday name
%A	Full weekday name

Command	Replaced By
%b	Abbreviated month name
%B	Full month name
%c	Standard date and time string
%d	Day of month as a decimal (1-31)
%H	Hour (0-23)
%I	Hour (1-12)
%j	Day of year as a decimal (1-366)
%m	Month as decimal (1-12)
%M	Minute as decimal (0-59)
%p	Locale's equivalent of AM or PM
%S	Second as decimal (0-60)
%U	Week of year, Sunday being first day (0-53)
%w	Weekday as a decimal (0-6, Sunday being 0)
%W	Week of year, Monday being first day (0-53)
%x	Standard date string
%X	Standard time string
%y	Year in decimal without century (0-99)
%Y	Year including century as decimal
%Z	Time zone name
%%	The percent sign

Related functions are **time( )**, **localtime( )**, and **gmtime( )**.

time

```
#include <ctime>
time_t time(time_t *time);
```

The **time( )** function returns the current calendar time of the system. If the system has no time, –1 is returned.

The **time( )** function can be called either with a null pointer or with a pointer to a variable of type **time_t**. If the latter is used, the variable will also be assigned the calendar time.

Related functions are **localtime( )**, **gmtime( )**, **strftime( )**, and **ctime( )**.