NAME

asctime, ctime, gmtime, localtime, mktime, asctime_r, ctime_r, gmtime_r, localtime_r − transform date and time to broken-down time or ASCII

SYNOPSIS

#include <time.h>

char *asctime(const struct tm ***tm*);**
char *asctime_r(const struct tm ***tm*, char ***buf***);**

char *ctime(const time_t ***timep*);**
char *ctime_r(const time_t ***timep*, char ***buf***);**

struct tm *gmtime(const time_t ***timep*);**
struct tm *gmtime_r(const time_t ***timep*, struct tm ***result***);**

struct tm *localtime(const time_t ***timep*);**
struct tm *localtime_r(const time_t ***timep*, struct tm ***result***);**

time_t mktime(struct tm ***tm*);**

DESCRIPTION

The **ctime**(), **gmtime**() and **localtime**() functions all take an argument of data type *time_t* which represents calendar time. When interpreted as an absolute time value, it represents the number of seconds elapsed since 00:00:00 on January 1, 1970, Coordinated Universal Time (UTC).

The **asctime**() and **mktime**() functions both take an argument representing broken-down time which is a representation separated into year, month, day, etc.

Broken-down time is stored in the structure *tm* which is defined in *<time.h>* as follows:

```
struct tm {
    int tm_sec;         /* seconds */
    int tm_min;         /* minutes */
    int tm_hour;        /* hours */
    int tm_mday;        /* day of the month */
    int tm_mon;         /* month */
    int tm_year;        /* year */
    int tm_wday;        /* day of the week */
    int tm_yday;        /* day in the year */
    int tm_isdst;       /* daylight saving time */
};
```

The members of the *tm* structure are:

tm_sec The number of seconds after the minute, normally in the range 0 to 59, but can be up to 60 to allow for leap seconds.

tm_min The number of minutes after the hour, in the range 0 to 59.

tm_hour
The number of hours past midnight, in the range 0 to 23.

tm_mday
The day of the month, in the range 1 to 31.

tm_mon
The number of months since January, in the range 0 to 11.

tm_year
: The number of years since 1900.

tm_wday
: The number of days since Sunday, in the range 0 to 6.

tm_yday
: The number of days since January 1, in the range 0 to 365.

tm_isdst
: A flag that indicates whether daylight saving time is in effect at the time described. The value is positive if daylight saving time is in effect, zero if it is not, and negative if the information is not available.

The call **ctime(*t*)** is equivalent to **asctime(localtime(*t*))**. It converts the calendar time *t* into a string of the form

```
"Wed Jun 30 21:49:08 1993\n"
```

The abbreviations for the days of the week are 'Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', and 'Sat'. The abbreviations for the months are 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', and 'Dec'. The return value points to a statically allocated string which might be overwritten by subsequent calls to any of the date and time functions. The function also sets the external variable *tzname* (see **tzset**(3)) with information about the current time zone. The re-entrant version **ctime_r**() does the same, but stores the string in a user-supplied buffer of length at least 26. It need not set *tzname*.

The **gmtime**() function converts the calendar time *timep* to broken-down time representation, expressed in Coordinated Universal Time (UTC). It may return NULL when the year does not fit into an integer. The return value points to a statically allocated struct which might be overwritten by subsequent calls to any of the date and time functions. The **gmtime_r**() function does the same, but stores the data in a user-supplied struct.

The **localtime**() function converts the calendar time *timep* to broken-time representation, expressed relative to the user's specified time zone. The function acts as if it called **tzset**(3) and sets the external variables *tzname* with information about the current time zone, *timezone* with the difference between Coordinated Universal Time (UTC) and local standard time in seconds, and *daylight* to a non-zero value if daylight savings time rules apply during some part of the year. The return value points to a statically allocated struct which might be overwritten by subsequent calls to any of the date and time functions. The **localtime_r**() function does the same, but stores the data in a user-supplied struct. It need not set *tzname*.

The **asctime**() function converts the broken-down time value *tm* into a string with the same format as **ctime**(). The return value points to a statically allocated string which might be overwritten by subsequent calls to any of the date and time functions. The **asctime_r**() function does the same, but stores the string in a user-supplied buffer of length at least 26.

The **mktime**() function converts a broken-down time structure, expressed as local time, to calendar time representation. The function ignores the specified contents of the structure members *tm_wday* and *tm_yday* and recomputes them from the other information in the broken-down time structure. If structure members are outside their legal interval, they will be normalized (so that, e.g., 40 October is changed into 9 November). Calling **mktime**() also sets the external variable *tzname* with information about the current time zone. If the specified broken-down time cannot be represented as calendar time (seconds since the epoch), **mktime**() returns a value of (time_t)(−1) and does not alter the *tm_wday* and *tm_yday* members of the broken-down time structure.

RETURN VALUE

Each of these functions returns the value described, or NULL (−1 in case of **mktime**()) in case an error was detected.

NOTES

The four functions **asctime**(), **ctime**(), **gmtime**() and **localtime**() return a pointer to static data and hence are not thread-safe. Thread-safe versions **asctime_r**(), **ctime_r**(), **gmtime_r**() and **localtime_r**() are specified by SUSv2, and available since libc 5.2.5.

In many implementations, including *glibc*, a 0 in *tm_mday* is interpreted as meaning the last day of the preceding month.

The glibc version of struct tm has additional fields

```
long tm_gmtoff;           /* Seconds east of UTC */
const char *tm_zone;      /* Timezone abbreviation */
```

defined when _BSD_SOURCE was set before including <*time.h*>. This is a BSD extension, present in 4.3BSD-Reno.

CONFORMING TO

SVr4, POSIX.1-2001, 4.3BSD, C89, C99.

SEE ALSO

date(1), **gettimeofday**(2), **time**(2), **utime**(2), **clock**(3), **difftime**(3), **strftime**(3), **strptime**(3), **tzset**(3), **time**(7)