NAME

mktime – converts a tm structure to a calendar time

SYNOPSIS

#include <time.h>

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

MT-LEVEL

Unsafe

DESCRIPTION

mktime( ) converts the time represented by the **tm** structure pointed to by *timeptr* into a calendar time (the number of seconds since 00:00:00 UTC, January 1, 1970).

The **tm** structure contains the following members:

```
int     tm_sec;      /* seconds after the minute [0, 61]  */
int     tm_min;      /* minutes after the hour [0, 59] */
int     tm_hour;     /* hour since midnight [0, 23] */
int     tm_mday;     /* day of the month [1, 31] */
int     tm_mon;      /* months since January [0, 11] */
int     tm_year;     /* years since 1900 */
int     tm_wday;     /* days since Sunday [0, 6] */
int     tm_yday;     /* days since January 1 [0, 365] */
int     tm_isdst;    /* flag for daylight savings time */
```

In addition to computing the calendar time, **mktime( )** normalizes the supplied **tm** structure. The original values of the **tm_wday** and **tm_yday** components of the structure are ignored, and the original values of the other components are not restricted to the ranges indicated in the definition of the structure. On successful completion, the values of the **tm_wday** and **tm_yday** components are set appropriately, and the other components are set to represent the specified calendar time, but with their values forced to be within the appropriate ranges. The final value of **tm_mday** is not set until **tm_mon** and **tm_year** are determined.

The original values of the components may be either greater than or less than the specified range. For example, a **tm_hour** of –1 means 1 hour before midnight, **tm_mday** of 0 means the day preceding the current month, and **tm_mon** of –2 means 2 months before January of **tm_year**.

If **tm_isdst** is positive, the original values are assumed to be in the alternate timezone. If it turns out that the alternate timezone is not valid for the computed calendar time, then the components are adjusted to the main timezone. Likewise, if **tm_isdst** is zero, the original values are assumed to be in the main timezone and are converted to the alternate timezone if the main timezone is not valid. If **tm_isdst** is negative, the correct timezone is determined and the components are not adjusted.

Local timezone information is used as if **mktime( )** had called **tzset( )** (see **ctime**(3C)).

mktime( ) returns the specified calendar time. If the calendar time cannot be represented, the function returns the value (**time_t**)–1.

EXAMPLES

What day of the week is July 4, 2001?

```
#include <stdio.h>
#include <time.h>

static char *const wday[ ] = {
        "Sunday", "Monday", "Tuesday", "Wednesday",
        "Thursday", "Friday", "Saturday", "-unknown-"
};
struct tm time_str;
/*...*/
time_str.tm_year        = 2001 - 1900;
time_str.tm_mon         = 7 - 1;
time_str.tm_mday        = 4;
time_str.tm_hour        = 0;
time_str.tm_min         = 0;
time_str.tm_sec         = 1;
time_str.tm_isdst       = −1;
if (mktime(&time_str)== −1)
  time_str.tm_wday=7;
printf("%s\n", wday[time_str.tm_wday]);
```

SEE ALSO

ctime(3C), **getenv**(3C), **TIMEZONE**(4)

NOTES

tm_year of the **tm** structure must be for year 1970 or later. Calendar times before 00:00:00 UTC, January 1, 1970 or after 03:14:07 UTC, January 19, 2038 cannot be represented.