NAME

asctime, ctime, gmtime, localtime, mktime − transform binary date and time to ASCII

SYNOPSIS

#include <time.h>

char *asctime(const struct tm **timeptr*);

char *ctime(const time_t **timep*);

struct tm *gmtime(const time_t **timep*);

struct tm *localtime(const time_t **timep*);

time_t mktime(struct tm **timeptr*);

extern char **tzname*[2];
long int *timezone*;
extern int *daylight*;

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 binary 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 61 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 **ctime()** function converts the calendar time *timep* 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* with information about the current time zone.

The **gmtime()** function converts the calendar time *timep* to broken-down time representation, expressed in Coordinated Universal Time (UTC).

The **localtime()** function converts the calendar time *timep* to broken-time representation, expressed relative to the user's specified time zone. The function 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 standard US daylight savings time rules apply.

The **asctime()** function converts the broken-down time value *timeptr* 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 **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.

CONFORMING TO
SVID 3, POSIX, BSD 4.3, ISO 9899

SEE ALSO
date(1), **gettimeofday**(2), **time**(2), **tzset**(3), **difftime**(3), **strftime**(3), **newctime**(3).