time

Gets the system time.

time_t time(time_t *_timer_);

Routine	Required Header	Compatibility
time	<time.h>	ANSI, Win 95, Win NT

For additional compatibility information, see Compatibility in the Introduction.

Libraries

LIBC.LIB	Single thread static library, retail version
LIBCMT.LIB	Multithread static library, retail version
MSVCRT.LIB	Import library for MSVCRT.DLL, retail version

Return Value

time returns the time in elapsed seconds. There is no error return.

Parameter

timer
Storage location for time

Remarks

The **time** function returns the number of seconds elapsed since midnight (00:00:00), January 1, 1970, coordinated universal time, according to the system clock. The return value is stored in the location given by *timer*. This parameter may be **NULL**, in which case the return value is not stored.

Example

```
/* TIMES.C illustrates various time and date functions including:
 *      time            _ftime          ctime       asctime
 *      localtime       gmtime          mktime      _tzset
 *      _strtime        _strdate        strftime
 *
 * Also the global variable:
 *      _tzname
 */

#include <time.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/timeb.h>
#include <string.h>
```

```
void main()
{
    char tmpbuf[128], ampm[] = "AM";
    time_t ltime;
    struct _timeb tstruct;
    struct tm *today, *gmt, xmas = { 0, 0, 12, 25, 11, 93 };

    /* Set time zone from TZ environment variable. If TZ is not set,
     * the operating system is queried to obtain the default value
     * for the variable.
     */
    _tzset();

    /* Display operating system-style date and time. */
    _strtime( tmpbuf );
    printf( "OS time:\t\t\t\t%s\n", tmpbuf );
    _strdate( tmpbuf );
    printf( "OS date:\t\t\t\t%s\n", tmpbuf );

    /* Get UNIX-style time and display as number and string. */
    time( &ltime );
    printf( "Time in seconds since UTC 1/1/70:\t%ld\n", ltime );
    printf( "UNIX time and date:\t\t\t%s", ctime( &ltime ) );

    /* Display UTC. */
    gmt = gmtime( &ltime );
    printf( "Coordinated universal time:\t\t%s", asctime( gmt ) );

    /* Convert to time structure and adjust for PM if necessary. */
    today = localtime( &ltime );
    if( today->tm_hour > 12 )
    {
   strcpy( ampm, "PM" );
   today->tm_hour -= 12;
    }
    if( today->tm_hour == 0 )  /* Adjust if midnight hour. */
   today->tm_hour = 12;

    /* Note how pointer addition is used to skip the first 11
     * characters and printf is used to trim off terminating
     * characters.
     */
    printf( "12-hour time:\t\t\t\t%.8s %s\n",
       asctime( today ) + 11, ampm );

    /* Print additional time information. */
    _ftime( &tstruct );
    printf( "Plus milliseconds:\t\t\t%u\n", tstruct.millitm );
    printf( "Zone difference in seconds from UTC:\t%u\n",
             tstruct.timezone );
    printf( "Time zone name:\t\t\t\t%s\n", _tzname[0] );
    printf( "Daylight savings:\t\t\t%s\n",
             tstruct.dstflag ? "YES" : "NO" );

    /* Make time for noon on Christmas, 1993. */
    if( mktime( &xmas ) != (time_t)-1 )
   printf( "Christmas\t\t\t\t%s\n", asctime( &xmas ) );

    /* Use time structure to build a customized time string. */
    today = localtime( &ltime );

    /* Use strftime to build a customized time string. */
    strftime( tmpbuf, 128,
```

```
         "Today is %A, day %d of %B in the year %Y.\n", today );
    printf( tmpbuf );
}
```

Output

```
OS time:                                21:51:03
OS date:                                05/03/94
Time in seconds since UTC 1/1/70:       768027063
UNIX time and date:                     Tue May 03 21:51:03 1994
Coordinated universal time:             Wed May 04 04:51:03 1994
12-hour time:                           09:51:03 PM
Plus milliseconds:                      279
Zone difference in seconds from UTC:    480
Time zone name:
Daylight savings:                       YES
Christmas                               Sat Dec 25 12:00:00 1993

Today is Tuesday, day 03 of May in the year 1994.
```