



Linux Ubuntu 26.04 Manual Pages on command 'gettimeofday.2'

\$ man gettimeofday.2

gettimeofday(2) System Calls Manual gettimeofday(2)

NAME

gettimeofday, settimeofday - get / set time

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <sys/time.h>
```

[[deprecated]]

```
int gettimeofday(struct timeval *restrict tv,  
                 struct timezone *_Nullable restrict tz);
```

[[deprecated]]

```
int settimeofday(const struct timeval *tv,  
                 const struct timezone *_Nullable tz);
```

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

settimeofday():

Since glibc 2.19:

 _DEFAULT_SOURCE

glibc 2.19 and earlier:

 _BSD_SOURCE

DESCRIPTION

The functions gettimeofday() and settimeofday() can get and set the time as well as a time zone.

The tv argument is a struct timeval (as specified in <sys/time.h>):

```

struct timeval {
    time_t    tv_sec;    /* seconds */
    suseconds_t tv_usec; /* microseconds */
};

```

and gives the number of seconds and microseconds since the Epoch (see time(2)).

The tz argument is a struct timezone:

```

struct timezone {
    int tz_minuteswest; /* minutes west of Greenwich */
    int tz_dsttime;     /* type of DST correction */
};

```

If either tv or tz is NULL, the corresponding structure is not set or returned. (However, compilation warnings will result if tv is NULL.)

The use of the timezone structure is obsolete; the tz argument should normally be specified as NULL. See VERSIONS.

Under Linux, there are some peculiar "warp clock" semantics associated with the settimeofday(2) system call if on the very first call (after booting) that has a non-NULL tz argument, the tv argument is NULL and the tz_minuteswest field is nonzero. (The tz_dsttime field should be zero for this case.) In such a case it is assumed that the CMOS clock is on local time, and that it has to be incremented by this amount to get UTC system time. No doubt it is a bad idea to use this feature.

RETURN VALUE

gettimeofday() and settimeofday() return 0 for success. On error, -1 is returned and errno is set to indicate the error.

ERRORS

EFAULT One of tv or tz pointed outside the accessible address space.

EINVAL (settimeofday()): timezone is invalid.

EINVAL (settimeofday()): tv.tv_sec is negative or tv.tv_usec is outside the range [0, 999,999].

EINVAL (since Linux 4.3)

(settimeofday()): An attempt was made to set the time to a value less than the current value of the CLOCK_MONOTONIC clock (see clock_gettime(2)).

EPERM The calling process has insufficient privilege to call settimeofday(); under Linux the CAP_SYS_TIME capability is required.

VERSIONS

C library/kernel differences

On some architectures, an implementation of `gettimeofday()` is provided in the `vdso(7)`.

The kernel accepts `NULL` for both `tv` and `tz`. The `timezone` argument is ignored by `glibc` and `musl`, and not passed to/from the kernel. Android's `bionic` passes the `timezone` argument to/from the kernel, but Android does not update the kernel `timezone` based on the device `timezone` in `Settings`, so the kernel's `timezone` is typically `UTC`.

STANDARDS

None.

HISTORY

4.3BSD, SVr4. SUSv1 and POSIX.1-2001 XSI described `gettimeofday()` but not `settimeofday()`. POSIX.1-2008 XSI marked `gettimeofday()` as obsolete, recommending the use of `clock_gettime(2)` instead. Likewise, `clock_settime(2)` should be used instead of `settimeofday()`. POSIX.1-2024 removed `gettimeofday()`.

Traditionally, the fields of `struct timeval` were of type `long`.

The `tz_dsttime` field

On a non-Linux kernel, with `glibc`, the `tz_dsttime` field of `struct timezone` will be set to a nonzero value by `gettimeofday()` if the current `timezone` has ever had or will have a daylight saving rule applied. In this sense it exactly mirrors the meaning of `daylight(3)` for the current zone. On Linux, with `glibc`, the setting of the `tz_dsttime` field of `struct timezone` has never been used by `settimeofday()` or `gettimeofday()`. Thus, the following is purely of historical interest.

On old systems, the field `tz_dsttime` contains a symbolic constant (values are given below) that indicates in which part of the year Daylight Saving Time is in force. (Note: this value is constant throughout the year: it does not indicate that DST is in force, it just selects an algorithm.) The daylight saving time algorithms defined are as follows:

```
DST_NONE    /* not on DST */
DST_USA     /* USA style DST */
DST_AUST    /* Australian style DST */
DST_WET     /* Western European DST */
DST_MET     /* Middle European DST */
DST_EET     /* Eastern European DST */
DST_CAN     /* Canada */
```

```
DST_GB    /* Great Britain and Eire */
```

```
DST_RUM    /* Romania */
```

```
DST_TUR    /* Turkey */
```

```
DST_AUSTALT /* Australian style with shift in 1986 */
```

Of course, it turned out that the period in which Daylight Saving Time is in force cannot be given by a simple algorithm, one per country; indeed, this period is determined by unpredictable political decisions. So this method of representing timezones has been abandoned.

NOTES

The time returned by `gettimeofday()` is affected by discontinuous jumps in the system time (e.g., if the system administrator manually changes the system time). If you need a monotonically increasing clock, see `clock_gettime(2)`.

Macros for operating on `timeval` structures are described in `timeradd(3)`.

SEE ALSO

`date(1)`, `adjtimex(2)`, `clock_gettime(2)`, `time(2)`, `ctime(3)`, `ftime(3)`, `timeradd(3)`, `capabilities(7)`, `time(7)`, `vdso(7)`, `hwclock(8)`