



Linux Ubuntu 26.04 Manual Pages on command 'ntp_adjtime.3'

\$ man ntp_adjtime.3

adjtimex(2) System Calls Manual adjtimex(2)

NAME

adjtimex, clock_adjtime, ntp_adjtime - tune kernel clock

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

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

int adjtimex(struct timex *buf);

int clock_adjtime(clockid_t clk_id, struct timex *buf);

int ntp_adjtime(struct timex *buf);
```

DESCRIPTION

Linux uses David L. Mills' clock adjustment algorithm (see RFC 5905). The system call adjtimex() reads and optionally sets adjustment parameters for this algorithm. It takes a pointer to a timex structure, updates kernel parameters from (selected) field values, and returns the same structure updated with the current kernel values. This structure is declared as follows:

```
struct timex {
    int modes; /* Mode selector */

    long offset; /* Time offset; nanoseconds, if STA_NANO
                 status flag is set, otherwise
                 microseconds */

    long freq; /* Frequency offset; see NOTES for units */

    long maxerror; /* Maximum error (microseconds) */
```

```

long esterror; /* Estimated error (microseconds) */

int status; /* Clock command/status */

long constant; /* PLL (phase-locked loop) time constant */

long precision; /* Clock precision
                (microseconds, read-only) */

long tolerance; /* Clock frequency tolerance (read-only);
                see NOTES for units */

struct timeval time;

/* Current time (read-only, except for
   ADJ_SETOFFSET); upon return, time.tv_usec
   contains nanoseconds, if STA_NANO status
   flag is set, otherwise microseconds */

long tick; /* Microseconds between clock ticks */

long ppsfreq; /* PPS (pulse per second) frequency
              (read-only); see NOTES for units */

long jitter; /* PPS jitter (read-only); nanoseconds, if
             STA_NANO status flag is set, otherwise
             microseconds */

int shift; /* PPS interval duration
           (seconds, read-only) */

long stabil; /* PPS stability (read-only);
            see NOTES for units */

long jitcnt; /* PPS count of jitter limit exceeded
            events (read-only) */

long calcnt; /* PPS count of calibration intervals
            (read-only) */

long errcnt; /* PPS count of calibration errors
            (read-only) */

long stbcnt; /* PPS count of stability limit exceeded
            events (read-only) */

int tai; /* TAI offset, as set by previous ADJ_TAI
         operation (seconds, read-only,
         since Linux 2.6.26) */

```

```
/* Further padding bytes to allow for future expansion */
```

```
};
```

The `modes` field determines which parameters, if any, to set. (As described later in this page, the constants used for `ntp_adjtime()` are equivalent but differently named.) It is a bit mask containing a bitwise OR combination of zero or more of the following bits:

ADJ_OFFSET

Set time offset from `buf.offset`. Since Linux 2.6.26, the supplied value is clamped to the range (-0.5s, +0.5s). In older kernels, an `EINVAL` error occurs if the supplied value is out of range.

ADJ_FREQUENCY

Set frequency offset from `buf.freq`. Since Linux 2.6.26, the supplied value is clamped to the range (-32768000, +32768000). In older kernels, an `EINVAL` error occurs if the supplied value is out of range.

ADJ_MAXERROR

Set maximum time error from `buf.maxerror`.

ADJ_ESTERROR

Set estimated time error from `buf.terror`.

ADJ_STATUS

Set clock status bits from `buf.status`. A description of these bits is provided below.

ADJ_TIMECONST

Set PLL time constant from `buf.constant`. If the `STA_NANO` status flag (see below) is clear, the kernel adds 4 to this value.

ADJ_SETOFFSET (since Linux 2.6.39)

Add `buf.time` to the current time. If `buf.status` includes the `ADJ_NANO` flag, then `buf.time.tv_usec` is interpreted as a nanosecond value; otherwise it is interpreted as microseconds.

The value of `buf.time` is the sum of its two fields, but the field `buf.time.tv_usec` must always be nonnegative. The following example shows how to normalize a timeval with nanosecond resolution.

```
while (buf.time.tv_usec < 0) {  
    buf.time.tv_sec -= 1;  
    buf.time.tv_usec += 1000000000;
```

}

ADJ_MICRO (since Linux 2.6.26)

Select microsecond resolution.

ADJ_NANO (since Linux 2.6.26)

Select nanosecond resolution. Only one of ADJ_MICRO and ADJ_NANO should be specified.

ADJ_TAI (since Linux 2.6.26)

Set TAI (Atomic International Time) offset from buf.constant.

ADJ_TAI should not be used in conjunction with ADJ_TIMECONST, since the latter mode also employs the buf.constant field.

For a complete explanation of TAI and the difference between TAI and UTC, see BIPM

ADJ_TICK

Set tick value from buf.tick.

Alternatively, modes can be specified as either of the following (multibit mask) values, in which case other bits should not be specified in modes:

ADJ_OFFSET_SINGLESHOT

Old-fashioned adjtime(3): (gradually) adjust time by value specified in buf.offset, which specifies an adjustment in microseconds.

ADJ_OFFSET_SS_READ (functional since Linux 2.6.28)

Return (in buf.offset) the remaining amount of time to be adjusted after an earlier

ADJ_OFFSET_SINGLESHOT operation. This feature was added in Linux 2.6.24, but did not work correctly until Linux 2.6.28.

Ordinary users are restricted to a value of either 0 or ADJ_OFFSET_SS_READ for modes. Only the superuser may set any parameters.

The buf.status field is a bit mask that is used to set and/or retrieve status bits associated

with the NTP implementation. Some bits in the mask are both readable and settable, while others are read-only.

STA_PLL (read-write)

Enable phase-locked loop (PLL) updates via ADJ_OFFSET.

STA_PPSFREQ (read-write)

Enable PPS (pulse-per-second) frequency discipline.

STA_PPSTIME (read-write)

Enable PPS time discipline.

STA_FLL (read-write)

Select frequency-locked loop (FLL) mode.

STA_INS (read-write)

Insert a leap second after the last second of the UTC day, thus extending the last minute of the day by one second. Leap-second insertion will occur each day, so long as this flag remains set.

STA_DEL (read-write)

Delete a leap second at the last second of the UTC day. Leap second deletion will occur each day, so long as this flag remains set.

STA_UNSYNC (read-write)

Clock unsynchronized.

STA_FREQHOLD (read-write)

Hold frequency. Normally adjustments made via ADJ_OFFSET result in dampened frequency adjustments also being made. So a single call corrects the current offset, but as offsets in the same direction are made repeatedly, the small frequency adjustments will accumulate to fix the long-term skew.

This flag prevents the small frequency adjustment from being made when correcting for an ADJ_OFFSET value.

STA_PPSSIGNAL (read-only)

A valid PPS (pulse-per-second) signal is present.

STA_PPSJITTER (read-only)

PPS signal jitter exceeded.

STA_PPSWANDER (read-only)

PPS signal wander exceeded.

STA_PPSERROR (read-only)

PPS signal calibration error.

STA_CLOCKERR (read-only)

Clock hardware fault.

STA_NANO (read-only; since Linux 2.6.26)

Resolution (0 = microsecond, 1 = nanoseconds). Set via ADJ_NANO, cleared via ADJ_MI? CRO.

STA_MODE (since Linux 2.6.26)

Mode (0 = Phase Locked Loop, 1 = Frequency Locked Loop).

STA_CLK (read-only; since Linux 2.6.26)

Clock source (0 = A, 1 = B); currently unused.

Attempts to set read-only status bits are silently ignored.

clock_adjtime ()

The clock_adjtime() system call (added in Linux 2.6.39) behaves like adjtimex() but takes an additional clk_id argument to specify the particular clock on which to act.

ntp_adjtime ()

The ntp_adjtime() library function (described in the NTP "Kernel Application Program API", KAPI) is a more portable interface for performing the same task as adjtimex(). Other than the following points, it is identical to adjtimex():

- ? The constants used in modes are prefixed with "MOD_" rather than "ADJ_", and have the same suffixes (thus, MOD_OFFSET, MOD_FREQUENCY, and so on), other than the exceptions noted in the following points.
- ? MOD_CLKA is the synonym for ADJ_OFFSET_SINGLESHOT.
- ? MOD_CLKB is the synonym for ADJ_TICK.
- ? There is no synonym for ADJ_OFFSET_SS_READ, which is not described in the KAPI.

RETURN VALUE

On success, adjtimex() and ntp_adjtime() return the clock state; that is, one of the following values:

TIME_OK Clock synchronized, no leap second adjustment pending.

TIME_INS Indicates that a leap second will be added at the end of the UTC day.

TIME_DEL Indicates that a leap second will be deleted at the end of the UTC day.

TIME_OOP Insertion of a leap second is in progress.

TIME_WAIT A leap-second insertion or deletion has been completed. This value will be returned until the next ADJ_STATUS operation clears the STA_INS and STA_DEL flags.

TIME_ERROR The system clock is not synchronized to a reliable server. This value is returned when any of the following holds true:

- ? Either STA_UNSYNC or STA_CLOCKERR is set.
- ? STA_PPSSIGNAL is clear and either STA_PPSFREQ or STA_PPSTIME is set.
- ? STA_PPSTIME and STA_PPSJITTER are both set.
- ? STA_PPSFREQ is set and either STA_PPSWANDER or STA_PPSJITTER is set.

The symbolic name TIME_BAD is a synonym for TIME_ERROR, provided for backward compatibility.

On failure, these calls return -1 and set errno to indicate the error.

Page 7/8

STANDARDS

adjtimex()

clock_adjtime()

Linux.

The preferred API for the NTP daemon is ntp_adjtime().

NOTES

In struct timex, freq, ppsfreq, and stabil are ppm (parts per million) with a 16-bit fractional part, which means that a value of 1 in one of those fields actually means 2^{-16} ppm, and $2^{16}=65536$ is 1 ppm. This is the case for both input values (in the case of freq) and output values.

The leap-second processing triggered by STA_INS and STA_DEL is done by the kernel in timer context. Thus, it will take one tick into the second for the leap second to be inserted or deleted.

SEE ALSO

clock_gettime(2), clock_settime(2), settimeofday(2), adjtime(3), ntp_gettime(3), capabilities(7), time(7), adjtimex(8), hwclock(8)

NTP "Kernel Application Program Interface"

Linux man-pages 6.17

2026-02-08

adjtimex(2)