Rocky Enterprise Linux 9.2 Manual Pages on command 'clock_adjtime.2'

$ man clock_adjtime.2

ADJTIMEX(2) Linux Programmer's Manual ADJTIMEX(2)

NAME

adjtimex, clock_adjtime, ntp_adjtime - tune kernel clock

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 parame?

ters for this algorithm. It takes a pointer to a timex structure, up?

dates 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 de? scribed later in this page, the constants used for ntp_adjtime() are equivalent but differently named.) It is a bit mask containing a bit? wise-or combination of zero or more of the following bits:

ADJ_OFFSET

Set time offset from buf.offset. Since Linux 2.6.26, the sup? plied 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.esterror.

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 follow? ing 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 ?http://www.bipm.org/en/bipm/tai/tai.html?

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 speci? fied in modes:

ADJ_OFFSET_SINGLESHOT

Old-fashioned adjtime(3): (gradually) adjust time by value spec? ified in buf.offset, which specifies an adjustment in microsec? onds.

ADJ_OFFSET_SS_READ (functional since Linux 2.6.28)

Return (in buf.offset) the remaining amount of time to be ad? justed 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_OFF?
SET_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 re?
mains 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 di?
rection 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_MICRO.

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.
* The 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 oper? ation 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_PP? STIME is set.
* STA_PPSTIME and STA_PPSJITTER are both set.
* STA_PPSFREQ is set and either STA_PPSWANDER or STA_PP? SJITTER is set.

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

Note that starting with Linux 3.4, the call operates asynchronously and the return value usually will not reflect a state change caused by the call itself.

On failure, these calls return -1 and set errno.

ERRORS

EFAULT buf does not point to writable memory.

EINVAL (kernels before Linux 2.6.26)

An attempt was made to set buf.freq to a value outside the range (-33554432, +33554432).

EINVAL (kernels before Linux 2.6.26)

An attempt was made to set buf.offset to a value outside the permitted range. In kernels before Linux 2.0, the permitted range was (-131072, +131072). From Linux 2.0 onwards, the per? mitted range was (-512000, +512000).

EINVAL An attempt was made to set buf.status to a value other than those listed above.

EINVAL The clk_id given to clock_adjtime() is invalid for one of two reasons. Either the System-V style hard-coded positive clock ID value is out of range, or the dynamic clk_id does not refer to a valid instance of a clock object. See clock_gettime(2) for a discussion of dynamic clocks.

EINVAL An attempt was made to set buf.tick to a value outside the range 900000/HZ to 1100000/HZ, where HZ is the system timer interrupt frequency.

ENODEV The hot-pluggable device (like USB for example) represented by a dynamic clk_id has disappeared after its character device was opened. See clock_gettime(2) for a discussion of dynamic clocks.

EOPNOTSUPP

The given clk_id does not support adjustment.

EPERM buf.modes is neither 0 nor ADJ_OFFSET_SS_READ, and the caller does not have sufficient privilege. Under Linux, the CAP_SYS_TIME capability is required.

ATTRIBUTES

For an explanation of the terms used in this section, see at? tributes(7).

??

?Interface ? Attribute ? Value ?

??

?ntp_adjtime() ? Thread safety ? MT-Safe ?

??

CONFORMING TO

None of these interfaces is described in POSIX.1

adjtimex() and clock_adjtime() are Linux-specific and should not be used in programs intended to be portable.

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 val? ues.

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 sec? ond 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" ?http://www.slac.stanford.edu/comp/unix/package/rtems/src/ssrlApps/ntpNanoclock/api.htm?

COLOPHON

This page is part of release 5.10 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man-pages/.