

NAME

clock_gettime, **clock_settime**, **clock_getres** - get/set/calibrate date and time

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

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

int

```
clock_gettime(clockid_t clock_id, struct timespec *tp);
```

int

```
clock_settime(clockid_t clock_id, const struct timespec *tp);
```

int

```
clock_getres(clockid_t clock_id, struct timespec *tp);
```

DESCRIPTION

The **clock_gettime()** and **clock_settime()** system calls allow the calling process to retrieve or set the value used by a clock which is specified by *clock_id*.

The *clock_id* argument can be a value obtained from `clock_getcpuclockid(3)` or `pthread_getcpuclockid(3)` as well as the following values:

CLOCK_REALTIME

CLOCK_REALTIME_PRECISE

CLOCK_REALTIME_FAST

CLOCK_REALTIME_COARSE

 Increments as a wall clock should.

CLOCK_MONOTONIC

CLOCK_MONOTONIC_PRECISE

CLOCK_MONOTONIC_FAST

CLOCK_MONOTONIC_COARSE

 Increments in SI seconds.

CLOCK_UPTIME

CLOCK_UPTIME_PRECISE

CLOCK_UPTIME_FAST

CLOCK_BOOTTIME

 Starts at zero when the kernel boots and increments monotonically in SI seconds while the

machine is running.

CLOCK_VIRTUAL

Increments only when the CPU is running in user mode on behalf of the calling process.

CLOCK_PROF

Increments when the CPU is running in user or kernel mode.

CLOCK_SECOND

Returns the current second without performing a full time counter query, using an in-kernel cached value of the current second.

CLOCK_PROCESS_CPUTIME_ID

Returns the execution time of the calling process.

CLOCK_THREAD_CPUTIME_ID

Returns the execution time of the calling thread.

The clock IDs *CLOCK_REALTIME*, *CLOCK_MONOTONIC*, and *CLOCK_UPTIME* perform a full time counter query. The clock IDs with the *_FAST* suffix, i.e., *CLOCK_REALTIME_FAST*, *CLOCK_MONOTONIC_FAST*, and *CLOCK_UPTIME_FAST*, do not perform a full time counter query, so their accuracy is one timer tick. Similarly, *CLOCK_REALTIME_PRECISE*, *CLOCK_MONOTONIC_PRECISE*, and *CLOCK_UPTIME_PRECISE* are used to get the most exact value as possible, at the expense of execution time. The clock IDs *CLOCK_REALTIME_COARSE* and *CLOCK_MONOTONIC_COARSE* are aliases of corresponding IDs with *_FAST* suffix for compatibility with other systems. Finally, *CLOCK_BOOTTIME* is an alias for *CLOCK_UPTIME* for compatibility with other systems.

The structure pointed to by *tp* is defined in *<sys/timespec.h>* as:

```
struct timespec {
    time_t    tv_sec;           /* seconds */
    long      tv_nsec; /* and nanoseconds */
};
```

Only the super-user may set the time of day, using only *CLOCK_REALTIME*. If the system *securelevel(7)* is greater than 1 (see *init(8)*), the time may only be advanced. This limitation is imposed to prevent a malicious super-user from setting arbitrary time stamps on files. The system time can still be adjusted backwards using the *adjtime(2)* system call even when the system is secure.

The resolution (granularity) of a clock is returned by the **clock_getres()** system call. This value is placed in a (non-NULL) **tp*.

RETURN VALUES

Upon successful completion, the value 0 is returned; otherwise the value -1 is returned and the global

variable *errno* is set to indicate the error.

ERRORS

The following error codes may be set in *errno*:

[EINVAL] The *clock_id* or *timespec* argument was not a valid value.

[EPERM] A user other than the super-user attempted to set the time.

SEE ALSO

date(1), adjtime(2), clock_getcpuclockid(3), ctime(3), pthread_getcpuclockid(3)

STANDARDS

The **clock_gettime()**, **clock_settime()**, and **clock_getres()** system calls conform to IEEE Std 1003.1b-1993 ("POSIX.1b"). The clock IDs *CLOCK_REALTIME_FAST*, *CLOCK_REALTIME_PRECISE*, *CLOCK_MONOTONIC_FAST*, *CLOCK_MONOTONIC_PRECISE*, *CLOCK_UPTIME*, *CLOCK_UPTIME_FAST*, *CLOCK_UPTIME_PRECISE*, *CLOCK_SECOND* are FreeBSD extensions to the POSIX interface.

HISTORY

The **clock_gettime()**, **clock_settime()**, and **clock_getres()** system calls first appeared in FreeBSD 3.0.