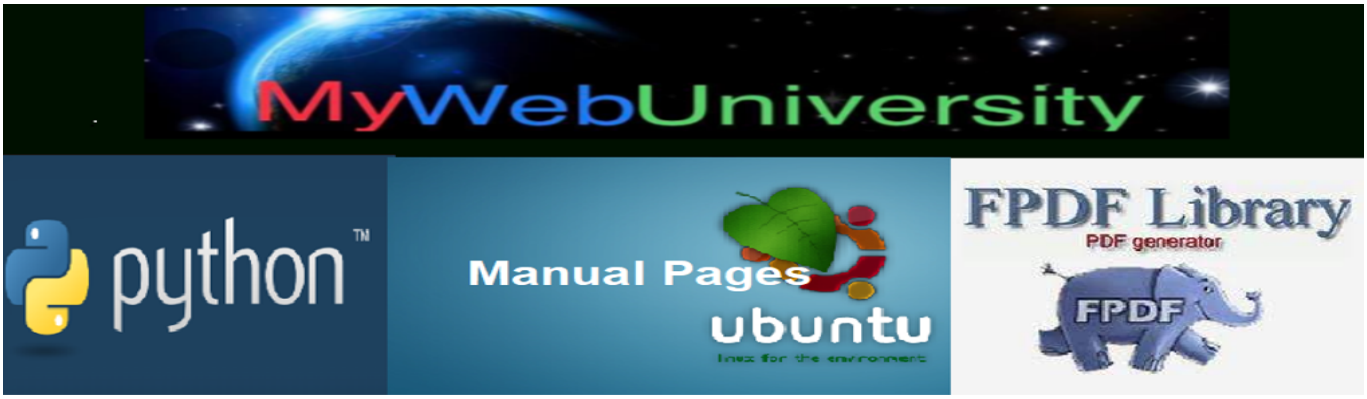

Rocky Enterprise Linux 9.2 Manual Pages on command 'nanosleep.2'

$ man nanosleep.2

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

NAME

nanosleep - high-resolution sleep

SYNOPSIS

```
#include <time.h>

int nanosleep(const struct timespec *req, struct timespec *rem);
```

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

nanosleep(): _POSIX_C_SOURCE >= 199309L

DESCRIPTION

nanosleep() suspends the execution of the calling thread until either at least the time specified in *req has elapsed, or the delivery of a signal that triggers the invocation of a handler in the calling thread or that terminates the process.

If the call is interrupted by a signal handler, nanosleep() returns -1, sets errno to EINTR, and writes the remaining time into the structure pointed to by rem unless rem is NULL. The value of *rem can then be used to call nanosleep() again and complete the spec? ified pause (but see NOTES).

The structure timespec is used to specify intervals of time with nanosecond precision. It is defined as follows:

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

The value of the nanoseconds field must be in the range 0 to 999999999.

Compared to sleep(3) and usleep(3), nanosleep() has the following advantages: it provides a higher resolution for specifying the sleep interval; POSIX.1 explicitly specifies that it does not interact with signals; and it makes the task of resuming a sleep that has been interrupted by a signal handler easier.

RETURN VALUE

On successfully sleeping for the requested interval, nanosleep() returns 0. If the call is interrupted by a signal handler or encounters an error, then it returns -1, with errno set to indicate the error.

ERRORS

EFAULT Problem with copying information from user space.

EINTR The pause has been interrupted by a signal that was delivered to the thread (see signal(7)). The remaining sleep time has been written into *rem so that the thread can easily call nanosleep() again and continue with the pause.

EINVAL The value in the tv_nsec field was not in the range 0 to 999999999 or tv_sec was negative.

CONFORMING TO

POSIX.1-2001, POSIX.1-2008.

NOTES

If the interval specified in req is not an exact multiple of the granularity underlying clock (see time(7)), then the interval will be rounded up to the next multiple. Further? more, after the sleep completes, there may still be a delay before the CPU becomes free to once again execute the calling thread.

The fact that nanosleep() sleeps for a relative interval can be problematic if the call is repeatedly restarted after being interrupted by signals, since the time between the inter? ruptions and restarts of the call will lead to drift in the time when the sleep finally completes. This problem can be avoided by using clock_nanosleep(2) with an absolute time value.

POSIX.1 specifies that nanosleep() should measure time against the CLOCK_REALTIME clock. However, Linux measures the time using the CLOCK_MONOTONIC clock. This probably does not matter, since the POSIX.1 specification for clock_settime(2) says that discontinuous changes in CLOCK_REALTIME should not affect nanosleep():

Setting the value of the CLOCK_REALTIME clock via clock_settime(2) shall have no effect on threads that are blocked waiting for a relative time service based upon

this clock, including the nanosleep() function; ... Consequently, these time ser? vices shall expire when the requested relative interval elapses, independently of the new or old value of the clock.

Old behavior

In order to support applications requiring much more precise pauses (e.g., in order to control some time-critical hardware), nanosleep() would handle pauses of up to 2 millisec? onds by busy waiting with microsecond precision when called from a thread scheduled under a real-time policy like SCHED_FIFO or SCHED_RR. This special extension was removed in kernel 2.5.39, and is thus not available in Linux 2.6.0 and later kernels.

BUGS

If a program that catches signals and uses nanosleep() receives signals at a very high rate, then scheduling delays and rounding errors in the kernel's calculation of the sleep interval and the returned remain value mean that the remain value may steadily increase on successive restarts of the nanosleep() call. To avoid such problems, use clock_nanosleep(2) with the TIMER_ABSTIME flag to sleep to an absolute deadline.

In Linux 2.4, if nanosleep() is stopped by a signal (e.g., SIGTSTP), then the call fails with the error EINTR after the thread is resumed by a SIGCONT signal. If the system call is subsequently restarted, then the time that the thread spent in the stopped state is not counted against the sleep interval. This problem is fixed in Linux 2.6.0 and later ker? nels.

SEE ALSO

clock_nanosleep(2), restart_syscall(2), sched_setscheduler(2), timer_create(2), sleep(3), usleep(3), time(7)

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/.