Rocky Enterprise Linux 9.2 Manual Pages on command 'time.7'

$ man time.7

TIME(7)   Linux Programmer's Manual   TIME(7)

NAME

    time - overview of time and timers

DESCRIPTION

  Real time and process time

    Real time is defined as time measured from some fixed point, either

    from a standard point in the past (see the description of the Epoch and

    calendar time below), or from some point (e.g., the start) in the life

    of a process (elapsed time).

    Process time is defined as the amount of CPU time used by a process.

    This is sometimes divided into user and system components. User CPU

    time is the time spent executing code in user mode. System CPU time is

    the time spent by the kernel executing in system mode on behalf of the

    process (e.g., executing system calls). The time(1) command can be

    used to determine the amount of CPU time consumed during the execution

    of a program. A program can determine the amount of CPU time it has

    consumed using times(2), getrusage(2), or clock(3).

  The hardware clock

Most computers have a (battery-powered) hardware clock which the kernel reads at boot time in order to initialize the software clock. For fur?ther details, see rtc(4) and hwclock(8).

The software clock, HZ, and jiffies

The accuracy of various system calls that set timeouts, (e.g., se?lect(2), sigtimedwait(2)) and measure CPU time (e.g., getrusage(2)) is limited by the resolution of the software clock, a clock maintained by the kernel which measures time in jiffies. The size of a jiffy is de?termined by the value of the kernel constant HZ.

The value of HZ varies across kernel versions and hardware platforms. On i386 the situation is as follows: on kernels up to and including 2.4.x, HZ was 100, giving a jiffy value of 0.01 seconds; starting with 2.6.0, HZ was raised to 1000, giving a jiffy of 0.001 seconds. Since kernel 2.6.13, the HZ value is a kernel configuration parameter and can be 100, 250 (the default) or 1000, yielding a jiffies value of, respec?tively, 0.01, 0.004, or 0.001 seconds. Since kernel 2.6.20, a further frequency is available: 300, a number that divides evenly for the com?mon video frame rates (PAL, 25 HZ; NTSC, 30 HZ).

The times(2) system call is a special case. It reports times with a granularity defined by the kernel constant USER_HZ. User-space appli?cations can determine the value of this constant using sysconf(_SC_CLK_TCK).

System and process clocks; time namespaces

The kernel supports a range of clocks that measure various kinds of elapsed and virtual (i.e., consumed CPU) time. These clocks are de?scribed in clock_gettime(2). A few of the clocks are settable using clock_settime(2). The values of certain clocks are virtualized by time namespaces; see time_namespaces(7).

High-resolution timers

Before Linux 2.6.21, the accuracy of timer and sleep system calls (see below) was also limited by the size of the jiffy.

Since Linux 2.6.21, Linux supports high-resolution timers (HRTs), op?tionally configurable via CONFIG_HIGH_RES_TIMERS. On a system that

supports HRTs, the accuracy of sleep and timer system calls is no
longer constrained by the jiffy, but instead can be as accurate as the
hardware allows (microsecond accuracy is typical of modern hardware).
You can determine whether high-resolution timers are supported by
checking the resolution returned by a call to clock_getres(2) or look?
ing at the "resolution" entries in /proc/timer_list.

HRTs are not supported on all hardware architectures. (Support is pro?
vided on x86, arm, and powerpc, among others.)

The Epoch

UNIX systems represent time in seconds since the Epoch, 1970-01-01
00:00:00 +0000 (UTC).

A program can determine the calendar time via the clock_gettime(2)
CLOCK_REALTIME clock, which returns time (in seconds and nanoseconds)
that have elapsed since the Epoch; time(2) provides similar informa?
tion, but only with accuracy to the nearest second. The system time
can be changed using clock_settime(2).

Broken-down time

Certain library functions use a structure of type tm to represent bro?
ken-down time, which stores time value separated out into distinct com?
ponents (year, month, day, hour, minute, second, etc.). This structure
is described in ctime(3), which also describes functions that convert
between calendar time and broken-down time. Functions for converting
between broken-down time and printable string representations of the
time are described in ctime(3), strftime(3), and strptime(3).

Sleeping and setting timers

Various system calls and functions allow a program to sleep (suspend
execution) for a   specified period of time; see nanosleep(2),
clock_nanosleep(2), and sleep(3).

Various system calls allow a process to set a timer that expires at
some point in the future, and optionally at repeated intervals; see
alarm(2), getitimer(2), timerfd_create(2), and timer_create(2).

Timer slack

Since Linux 2.6.28, it is possible to control the "timer slack" value

for a thread. The timer slack is the length of time by which the ker‐

nel may delay the wake-up of certain system calls that block with a

timeout.   Permitting this delay allows the kernel to coalesce wake-up

events, thus possibly reducing the number of system wake-ups and saving

power.   For more details, see the description of PR_SET_TIMERSLACK in

prctl(2).

SEE ALSO

date(1), time(1), timeout(1), adjtimex(2), alarm(2), clock_gettime(2),

clock_nanosleep(2), getitimer(2), getrlimit(2), getrusage(2),

gettimeofday(2), nanosleep(2), stat(2), time(2), timer_create(2),

timerfd_create(2), times(2), utime(2), adjtime(3), clock(3),

clock_getcpuclockid(3), ctime(3), ntp_adjtime(3), ntp_gettime(3),

pthread_getcpuclockid(3), sleep(3), strftime(3), strptime(3),

timeradd(3), usleep(3), rtc(4), time_namespaces(7), hwclock(8)

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/.

Linux   2020-04-11   TIME(7)