

Re: Premature wakeup of time.sleep()

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2005-09/msg01852.html>

- *From:* bokr@xxxxxx (Bengt Richter)
 - *Date:* Tue, 13 Sep 2005 19:58:38 GMT
-

On Mon, 12 Sep 2005 19:33:07 -0400, Peter Hansen <peter@xxxxxxxxxxxx> wrote:

>Steve Horsley wrote:

>> I think the sleep times are quantised to the granularity of the system
>> clock, which varies from os to os. From memory, windows 95 has a 55mS
>> timer, NT is less (19mS?), Linux and solaris 1mS. All this is from
>

>For the record, the correct value for NT/XP family is about 15.6 ms
>(possibly exactly 64 per second and thus 15.625ms, but offhand I don't
>recall, though I'm sure Google does).

>

What "correct value" are you referring to? The clock chip interrupts,
clock chip resolution, or OS scheduling/dispatching?

For NT4 at least, I don't know of anything near 15.6 ms ;-)

Speaking generally (based on what I've done myself implementing
time-triggered wakeups of suspended tasks in a custom kernel eons ago),
a sleep-a-while function is not as desirable as a sleep-until function,
because the first is relative and needs to get current time and
add the delta and then use the sleep-until functionality. So if there
is a timing glitch during the sleep-a-while call, there will be a glitch
in the wake-up time.

Once you have a wakeup time, you can put your sleeper in a time-ordered queue,
at which time you might or might not check if wake-up time is already past,
which could happen. So you can put the task in a ready queue for the OS to
run again when it gets around to it, or since the sleeper is the caller, you
have the option of a hot return as if the sleep call hadn't happened.

Assuming you queue the task, when is the queue going to be checked to see
if it's time to wake anyone up? Certainly when the next OS time slice is
up. That might be ~55ms or 10ms (NT4) or 1ms (modern OS & processor). At that
point (roughly on the strike of e.g. 10ms, depending on how many bad high-priority
interrupt service routines there are that glitch it by suspending interrupts
too long or queueing too much super-priority deferred processing)), we look at
the wake-up queue. And a number of tasks might get wakened. Some will have had
wake-up times that were really just microseconds after they were queued, but it
has taken 10ms (or whatever) for the OS to finish another task's time slice and

Re: Premature wakeup of time.sleep()

check, so the effect is to wake up on the OS basic slicing time. Waking up is only being queued for resumption however, so there might be more delay, depending on relative priorities etc. Some OS's might