Subject: Re: usecs
Posted by Tom1 on Fri, 07 Dec 2018 09:02:00 GMT
View Forum Message <> Reply to Message

Hi,

In my opinion, two timing methods are needed. One is the 'wall clock', i.e. something that OS supplies as UTC (or local time, but rather as UTC). This may be synchronized by e.g. NTP or adjusted by user and as such it is always just an approximation of time. There are no guarantees of its correctness or quality.

The other timing source needed is a counter for interval measurement and monotonicity (or 'steadyness' as they seem to call it in C++ lib) is a vital property of it. Monotonic counters are in the end the only solid reference for any serious timing.

(I use these monotonic counters for e.g. creating a 'UTC wall clock' that is synchronized way beyond the accuracy of any OS based clock. The timing requirements of my application are far beyond what e.g. NTP can supply over network.)

For the above reasons, I would suggest using only such APIs for msecs() and usecs() that guarantee in documentation the monotonicity of the clock. Using a source that looks fine on my system and yours does not guarantee similar behavior on the other client's system. After all these are highly hardware dependent in addition to the software platform used.

On POSIX I would stick with "clock_gettime(CLOCK_MONOTONIC,&now);". (This is what I would like to have on Windows too, but that's just a wish...)

For Windows, I would carefully read:

https://docs.microsoft.com/en-us/windows/desktop/SysInfo/acquiring-high-resolution-time-stamps

and then go with QueryPerformanceCounter (QPC).

-

How about adding a future proof solution to U++: "int64 nsecs_monotonic()"? I guess the name says it all. The resolution will be as good as the platform can supply and monotonicity will guarantee its usefulness for any application.

Best regards,

Tom