Add new optional time bases v3 Proposal for C2x

Jens Gustedt

ISO/IEC JTC 1/SC 22/WG14 **N2460**

November 18, 2019 **v 3**

Add new optional time bases v3

Proposal for C2x

Jens Gustedt
INRIA and ICube, Université de Strasbourg, France

We propose the inclusion of optional macros for time bases that are modelled after ISO 9945's `CLOCK_MONOTONIC`, `CLOCK_PROCESS_CPU_ID`, and `CLOCK_THREAD_CPU_ID`.
History: This is one part of a follow-up of N2402 and N2417, which had been denied adequate treatment in the Ithaca 2019 meeting of WG14.

1. INTRODUCTION

The interfaces in `time.h` to manipulate time values have grown mostly unattended over the years and present several problems that could be easily avoided with more modern, redesigned interfaces. This paper is concerned with the following problem:

— The standard allows implementations to add more time bases than **`TIME_UTC`** but gives no guidance in which direction to go with such new base values.
— POSIX already provides normalized semantics for some other time bases than **`TIME_UTC`**, and it would be good if we could avoid that practices with similar named time base emerge that diverge from these.

1.1. Strategy

C11 and C17 left the addition of new time bases completely to the implementation. Although it is a good principle to leave room for extensions, certain of them already have a connotation in other normative context. In particular, ISO 9945 already provides specifications for four different time bases, two for elapsed time measurement (`CLOCK_REALTIME` and `CLOCK_MONOTONIC`), and two for CPU time (`CLOCK_PROCESS_CPUTIME_ID` and `CLOCK_THREAD_CPUTIME_ID`).
C11's **`timespec_get`** and **`TIME_UTC`** are modeled after ISO 9945's `clock_gettime` and `CLOCK_REALTIME`, so we propose not handle the latter, and to suppose that the specification for **`TIME_UTC`** is sufficient.
For the other three, we propose to add optional macros to the standard, such that the names, if defined, bind implementations to a particular semantic. ISO 9945 and ISO 9899 differ slightly in their interfaces and have different terminology, so we propose to have macro names according to C's terminology with a prefix `TIME`:

— `TIME_MONOTONIC` for a time base that is not affected by changes to the time environment, if such a change is possible on the platform. The intent is to provide a measure of time as perceived by the execution platform in its current physical reference system. (This is in contrast to calendar time as measured by **`TIME_UTC`** which is subject to normative and cultural adjustments.)
— `TIME_CPU` which is the cpu time that is accounted for the whole execution. The intent is to provide a value that is consistent with the return of the **`clock`** function as specified by the C standard.
— `TIME_THREAD_CPU` which is the same, but accounted on a per thread base.

Since these macros will generally have different values from the ones provided by ISO 9945 (there the constants have the opaque type `clockid_t`) we can impose positive values without invalidating components of ISO 9945.
As a general strategy we propose to modify 7.27.1 p2 as follows:

> ...
> **TIME_UTC**
> TIME_MONOTONIC
> which expands ~~to an~~ integer constants, ~~that~~ designat~~es~~ing the ~~UTC~~real time and monotonic time base, respectively; and
> TIME_CPU
> TIME_THREAD_CPU
> which expand to integer values, designating overall and thread-specific cpu-time bases, respectively.

And then to add a new paragraph:

> 3 The definition of macros for time bases other than **TIME_UTC** are optional. If defined, the corresponding time bases are supported by **timespec_get** and their values are positive.

1.2. Elapsed time

ISO 9945 has two different "clocks" for measurement of elapsed time, CLOCK_REALTIME and CLOCK_MONOTONIC. They differ eventually in the starting point of the measurement (*epoch* vs. boot time) and, more importantly, concerning their behavior when the system time is set:

— CLOCK_REALTIME changes when the clock is set to a new value, *e.g* if a background time daemon adjusts to a drift indicated by a time servers, or if calendar time is adjusted with a leap second. This is the only clock in ISO 9945 that is mandatory, and as such plays a similar role as **TIME_UTC** for ISO 9899.
— CLOCK_MONOTONIC is guaranteed not to be affected by such changes of the system clock and to measure physical time as perceived by the platform.

We propose to model the latter by TIME_MONOTONIC and to add text to the end of 7.27.2.5 p3 (**timespec_get**):

> TIME_MONOTONIC is the same, but the reference point may or may not be the same epoch as for **TIME_UTC** or any other implementation-defined time point; this point shall not change during the program execution and the result shall not be affected by any implementation-specific functions or external events that set the system time; it is implementation-defined if this base accounts for time during which the execution platform is suspended.[FNT]

With the attached footnote:

> [FNT] An execution platform may for example not be able to track physical time that elapsed during suspension in a low power consumption mode.

QUESTION 1. *Shall we adopt* TIME_MONOTONIC *as proposed in N2460?*

1.3. CPU time

In C17, CPU time of a program execution can be measured by means of the **clock** function. Unfortunately this functions has several problems, the most sever being that it may overflow without notice. Another disadvantage of **clock** is that there is one legacy C implementation that gets this function fundamentally wrong when compared to the C standard: it accounts for elapsed (wallclock) time instead of CPU time. This repeatably leads to confusion when code is ported from or to conforming platforms. For these reasons we think that **clock** is best deprecated and replaced by an appropriate time base for **timespec_get**.

ISO 9945 has two such "clocks" which we propose to adapt to the needs of the C standard. Because implementations might need to dynamically distinguish different values for these bases for concurrent program executions (processes) or threads, the specifications of the values exempts them from being compile time constants (see above) and we add in 7.25.1 p3:

> The value of `TIME_CPU` shall be different from the constants **`TIME_UTC`** and `TIME_MONOTONIC` and shall not change during the same program execution. The macro `TIME_THREAD_CPU` shall not be defined if the implementation does not support threads; its value shall be different from the above, shall be the same for all invocations from the same thread, and the value provided for one thread shall not be used by a different thread as `base` argument of **`timespec_get`**.

For **`timespec_get`** itself the text proposal in 7.27.2.5 is then quite simple:

> For `base` set to `TIME_CPU` and `TIME_THREAD_CPU` the result is similar, but the call measures the amount of processor time associated with the program execution or thread, respectively.

Calls with `TIME_PROCESS_CPU` could replace calls of **`clock`**, provided we know the resolution of this time base.
If supported, calls with `TIME_THREAD_CPU` implement a new feature that allows to distinguish the cost of threads individually.

QUESTION 2. *Shall we adopt* `TIME_CPU` *as proposed in N2460?*

QUESTION 3. *Shall we adopt* `TIME_THREAD_CPU` *as proposed in N2460?*