How do I upgrade a 32-bit tick count to a 64-bit one?

devblogs.microsoft.com/oldnewthing/20220107-00

January 7, 2022

Raymond Chen

There are a number of Windows functions that produce 32-bit tick counts, like `GetMessageTime`. But you may have a general policy in your program of always working with 64-bit tick counts to avoid timer rollover problems. How can you upgrade these to 64-bit tick counts?

Well, clearly, the 32-bit timestamp is ambiguous, since the 32-bit timer rolls over every 49 days (approximately). But there is only one point within the previous 49 days that has the specified 32-bit timestamp. It is probably a reasonable assumption that the timestamp refers to something that occurred within the past 49 days, because most of these system-provided timestamps are reporting the time at which something occurred in the recent past, like the time the message or input was generated prior to delivery. In practice, the timestamp age will be on the order of tens of milliseconds, maybe a few seconds if your program is having a bad day. A delay of over 49 days is extremely unlikely.

All we have to do is reconstruct the upper 32 bits of the timestamp.

```
uint64_t UpgradeTickCount(uint32_t then, uint64_t now)
{
    auto diff = static_cast<uint32_t>(now) - then;
    return now - diff;
}
```

First, we calculate the 32-bit elapsed time between the provided timestamp and the current time. This works even in the face of timer rollover thanks to unsigned integer arithmetic, which is modulo 2^{32} for `uint32_t`.

Now we know how long ago the 32-bit timestamp was, assuming it represents a time at most 49 days in the past. We can then apply that same adjustment to the current 64-bit time to obtain a full 64-bit timestamp.

The value for `now` need not be the current time. It just needs to be any 64-bit time value that is at or later than the `then` value.

Now, there are some functions that report the most recent time an event occurred, and it's possible that those events are extremely rare. For example, the `GetLastInputInfo` function tells you the last time the system received input, but a server in a closet might go for an extended period without receiving any input.

What you can do is establish a baseline by querying the timestamp and immediately upgrading it to a 64-bit timestamp. (You have no way of knowing whether that timestamp is less than 49 days old, so you may as well assume that it is.) Each time you want to upgrade a new timestamp, you check whether it is the same as the previous 32-bit timestamp: If so, then assume it's the same old timestamp. If not, then assume that the new event occurred within the past 49 days. If you perform this check at least once every 49 days (pretty easy to arrange), then that will be a valid assumption.

```
uint64_t UpgradeTickCount2(uint32_t then, uint64_t previous, uint64_t now)
{
    if (static_cast<uint32_t>(then) == static_cast<uint32_t>(previous)) {
        return previous;
    } else {
        return UpgradeTickCount(then, now);
    }
}
```

Still, to get to 49 days, it means you're not applying monthly security updates like some sort of monster who welcomes your new botnet overlords. But at least you got timestamps.

Raymond Chen

Follow