The ISP Column
A monthly column on things Internet

March 2026
Geoff Huston

The Last Time

It's fading from our collective memory, but almost thirty years ago the global IT industry was gripped by Y2K fever. In the years leading up to the year 2000 it was observed that many of the older software systems at the time represented a date using just the last two digits of the year. In this mode of date representation, the transition to the new century would look like the local clock had been turned backward, and that would cause poorly written software to crash or spin into an infinite loop. It was a rather obscure problem, but it was taken up in mainstream media as a sure sign of impeding techno-doom, with dire predictions of jammed lifts, failing phone systems, and much more.

At the time much money and time was spent in auditing IT systems to ensure that they could handle this time transition to a new millennium without any fuss, and many companies, including the phone company I worked for at the time, called their technical staff into the office to babysit their IT systems when the local wall clock time got to midnight. Nothing happened.

The reason why this was a non-event was that few programming languages and fewer operating systems used a local representation of the time with a two-digit year core. We may have used two digit year codes in handwritten dates, but that was not how the date and time was handled within software stacks in most cases. About the only Y2K candidate at the time was COBOL, which had a variant of the date in a 6-digit numeric form, so that the millennium transition would cause the date to move from "991231" to "000101". Even so, the amount of COBOL in the world of 1999 was not exactly large, and by and large time rolled over to the new millennium without much in the way of serious mayhem and failure.

Far more common at that time was an internal representation of time as the number of units of time since some defined epoch. For example, in Excel the time is stored as a fractional count of the number of days since 1 January 1900. The Unix operating system's internal representation of the time is the number of seconds since the 1 January 1970, UTC. That internal representation of time meant that the millennium occurred when the internal clock counter transitioned from the value of 946,684,799 to 946,684,800. Hardly an auspicious event, nor was it one that would trigger software mayhem! Windows uses a 64-bit count of the number of milliseconds since 1 January 1601.

The Global Positioning System (GPS) counts the number of weeks since January 6, 1980 using a 10-bit counter (1,024 weeks). The last two rollovers of this counter have already happened at midnight on 21 August 1999, and 31 March 2019. It has always been within the responsibility of the GPS user to consult other sources and choose the correct current 1024-week-block for GPS applications.

A larger table of various time representations can be found at https://en.wikipedia.org/wiki/Syxstem_time.

The problem with time counters rolling over is that software systems deal with time differences all the time. If software simply assumes a monotonically increasing sequence of counter values, then at rollover time these time differences will become negative, which may cause the code to fail in surprising ways.

So, when will these various time counters roll over?

Unix Time

The Unix call *time*() is a useful system call on a generic Unix system and returns the current time in form of an integer timestamp (*time_t* integer second counter). In a 32-bit system, *time_t* is usually defined as a signed 32-bit integer. However, that's not all that common these days, as most modern computer platforms support a signed 64-bit integer, and if the local software libraries have been updated to have 64-bit support, then the generic data type used for the seconds counter is an unsigned 64-bit integer.

When applications are compiled, the compiler will impose a uniform view of the internal data objects, so a 32-bit Unix application or library still expects the kernel to return a 32-bit value even if the kernel is running on a 64-bit architecture.

This could be a problem, because even though 64-bit platforms are heading towards ubiquity in the coming years, legacy application software will continue to be used.

If you are using a 32-bit signed integer platform, or running 32-bit legacy applications and using a seconds counter with an epoch of 1 January 1970, then counter rollover will occur on 19 January 2038. The use of an unsigned 32-bit time counter will roll over on 7 February, 2106. If you are using a 64-bit representation of time, then the seconds time counter will roll on Dec 4, in the year 292,277,026,596!

Windows

32-Bit Windows applications, or Windows applications defining _USE_32BIT_TIME_T, can be hit by the year 2038 problem too if they use the *time_t* data type.

NTP

What about the Network Time Protocol? NTP, used to synchronise a device's clock with a number of reference clocks over a network, uses a representation of the time counter as a two-part 64-Bit timestamp for most parts of the protocol. The first 32 bits are the number of seconds since January 1, 1900, and the second 32 bits are a fractional part. This gives NTP 136 years between rollovers of the seconds counter, leading to February 7, 2036 as the first rollover event for the seconds counter.

The NTPv4 specification defines an NTP date format as a 64-bit second counter, which spans 584 billion years. It's not clear how NTP implementations will cope with the roll of the low-order 32 bits of the second counter, but we have about a decade to figure that out!

File Systems

HFS+ timestamps are stored as an unsigned 32-bit integer representing the number of seconds since midnight, January 1, 1904 (UTC). That means that the filesystem timestamps will rollover on 5 February 2040.

FAT32 timestamps use 16 bits for the calendar date and 16 bits for the time of day, so the system can hold timestamps in the range from 1980 to 2107.

NFSv4 uses the 64-but Unix time structure, so the rollover time is well into the future, as already noted.

ZFS similarly supports the Unix time structures.

Summary of Rollover Times

Unix - 32-bit	19 January 2038
Unix - 64-bit	4 December 292,277,026,596
Windows - 32-bit	19 January 2038
Windows - 64-bit	4 December 292,277,026,596
NTP	7 February 2036
HFS+	5 February 2040
FAT32	31 Dec 2107
NFSv4	4 December 292,277,026,596
ZFS	4 December 292,277,026,596

Conclusions

All this seems like a replay of the Y2K story. Older systems, which have not been re-factored to run on current (64-bit) hardware platforms, will encounter one or more of the forthcoming counter rollover events in 2036, 2038 and 2040.

Yes, this requires some due diligence on the part of operators of IT and network services to ensure that their platforms and applications are re-factored as required to use 64-bit timestamps. As long as that preparation is done well in advance then we should expect that February 2036, January 2038 and February 2040 will all happen without much in the way of unplanned drama!

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

Author

Geoff Huston AM, B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

www.potaroo.net