DOI:10.1145/1941487.1941505

Article development led by [acmqueue](https://queue.acm.org)
queue.acm.org

Finding a lasting solution to the leap seconds problem has become increasingly urgent.

BY POUL-HENNING KAMP

The One-Second War

THANKS TO A secretive conspiracy working mostly below the public radar, your time of death may be a minute later than presently expected. But don't expect to live any longer, unless you happen to be responsible for time synchronization in a large network of computers, in which case this coup will lower your stress level a bit every other year or so.

We're talking about the abolishment of leap seconds, a crude hack added 40 years ago to paper over the fact that planets make lousy clocks compared with quantum mechanical phenomena.

Timekeeping used to be astronomers' work, and the trouble it caused was very academic. To the rural

population, sunrise, midday, and sunset were plenty precise for all relevant purposes.

Timekeeping became a problem for non-astronomers only when ships started to navigate where they could not see land. Finding your latitude is easy: measure the height of the midday sun over the horizon, look at the table in your almanac, done. Finding your longitude is possible only if you know the time of day precisely, and the sun will not tell you that unless you know your longitude.

If you know your longitude, however, the sun will tell you the time very precisely. Using that time, you can make tables of other nonsolar astronomical events—for example, the transits of the

moons of Jupiter, which can then be used to estimate time from that longitude.

This is why Greenwich Observatory in the U.K. and the U.S. Naval Observatory were funded by their respective admiralities. The British empire staked some money on this question, and while the astronomers won on dirty play, the audience vastly preferred John Harrison's chronometers because you did not need to see the transits of the moons of Jupiter to know what time it was. Harrison's chronometer just told you, any time you wanted to know.⁴

Ever since, astronomers have lost ground as "time lords."

Time zones, made necessary by

transcontinental railroads, reduced the number of necessary observatories to nearly nothing. Previously, every respectable city, with or without a university, had somebody whose job it was to figure out proper time. With time zones and a telegraph, you could service all of the United States from the Naval Observatory.

The next loss was the length a second, which astronomers had defined as "1/31,556,925.9747 of the tropical year in 1900," neither a very practical nor very reproducible definition.

Louis Essen's atomic clock won that battle, and SI (International System of Units) seconds became 9,192,631,770 periods of hyperfine radiation from a

cesium-133 atom. A new time scale was created to count these seconds.

Civil time was still kept using a different and varying length of a second, depending on what astronomers had measured the earth's rotation to for each year.

Having variable-length seconds did not work for anybody, not even the astronomers, so in 1970 it was decided to use SI seconds and do full-second step adjustments—leap seconds—starting January 1, 1972.² In practice, this works by astronomers sending the rest of the world a telegram twice a year to tell us how long the last minute of June and December will be: 59, 60, or 61 seconds.

There is a certain irony in the fact

that the UTC (*Universal Time Coordinated*) time scale depends on the rotation of one particular rock in the less fashionable western part of the galaxy. I am pretty sure that, should humans ever colonize other rocks, leap seconds will not be in the luggage.

How Leap Seconds Became a Problem

Until the advent of big synchronized networks of computers, leap seconds bothered nobody. Many computers used the frequency of the electrical grid to count time, and most had their time initially set from somebody's wristwatch. The number of people who actually cared probably numbered fewer than two dozen worldwide.

Therefore, Unix didn't bother with leap seconds. In the `time_t` definition from Unix, all minutes have 60 seconds, all hours 3,600 seconds, and all days 86,400 seconds. This definition carried over to Posix and The Open Group where it is presumably gold-plated for all eternity.

Then something shifted deep under the surface of the earth. We can only guess what it might have been, but there was no need for leap seconds for seven straight years: from the end of 1998 to the end of 2005. This was, more or less, the time when the Internet happened and everybody bought PCs with Windows. Most of the people who hacked Perl to implement the dot-com revolution had never heard of leap seconds.

This is what Microsoft had to say on the subject of leap seconds: "[...]after the leap second occurs, the NTP (Network Time Protocol) client that is running Windows Time service is one second faster than the actual time."³

Unix systems running NTP will paper over the leap second, but there is no standard that says how this should be done. Your system might do one of the scenarios shown in the accompanying figure.

Or it might do something entirely different. Some systems have resorted to slowing down the clock by 1/3600th for the last hour before the leap second, hoping that nobody notices that seconds suddenly are 277 microseconds long.

That's in theory. In practice it depends on the systems getting notice of the leap second and handling it as intended. In this context *systems* are also the NTP servers from which the rest of the computers get their time: at the 2008 leap second, more than one in seven in the public NTP pool servers got it wrong.

The Effort to "Fix" Leap Seconds

By early 2005 when the first leap second in seven years finally began to look likely, some people started to worry about a "Y2K-lite" event. Some bright person inside the U.S. military-industrial complex thought, "Wait a minute, why do we need leap seconds in the first place?" and proposed to the ITU-R (International Telecommunication Union, Radiocommunication Sector) that they be abolished, preferably before December 2005.

Nice try, but one should never underestimate the paper tiger in a UN organization.

The December 2005 leap second came, Armageddon did not, but it was painfully obvious to everybody who paid attention that there were massive amounts of software that needed fixing, before leap seconds would not cause trouble. Even the HBG time signal from the Swiss time reference system did it wrong.

Another leap second occurred in December 2008, and the situation had not changed in any measurable way, but at least the Swiss got it right this time.

Since then the proposal, known to insiders as TF.460-7, has been the subject of "further study" in "Study Group 7A,"

and all sorts of secret scientific brotherhoods, from AAU to CCTF, have had their chance to weigh in. Many have, but few have clear-cut positions.

What is the Problem with Leap Seconds?

The problem is that more systems care about time at the second level.

Air Traffic Control systems perform anti-collision tests many times a second because a plane moves 300 meters in a second. A one-second hiccup in input data from the radar is not trivial in a tightly packed airspace around a major airport.

Medical products and semiconductors are produced in time-critical processes in complex continuous production facilities. On December 8, 2010, a 70-msec power glitch hit a Toshiba flash chip manufacturing facility, and 20% of the products scheduled to ship in January and February 2011 had to be scrapped: "Once the line is stopped, we can't just resume production," said Toshiba spokesman Hiroko Yamazaki.⁵

Technically, there is no problem with leap seconds that we IT professionals cannot tolerate. We just have to make sure that all computers know about leap seconds and that all programs, operating systems, and applications know how to deal with them.

The first part of that problem is we have only six months to tell all computers and software about leap seconds, because that is all the warning we get from the astronomers. In practice, we often have 10 months' notice; for example, we were told on February 2 that there will be no leap second in December of this year.¹

Unfortunately, this advantage is negated by some time signals—for example, the DCF77 signal from Germany, announcing the leap second only one hour ahead of time.

The other part of the problem—changing `time_t` to know about leap seconds—has nasty results: time is suddenly not a fixed radix quantity anymore. How much code finds the current day by `d = t/86400` or tests if two events are further apart than a minute by `if (t1 >= t2 + 60)?` Nobody knows. How much of such code needs to be fixed if we change the `time_t` definition? Nobody knows.

The Y2K experience indicates it

Sensitivities in leap seconds.

23:59:57	23:59:57	23:59:57
23:59:58	23:59:58	23:59:58
23:59:59	23:59:59	23:59:59
23:59:59	00:00:00	(halt for 1 sec)
00:00:00	00:00:00	00:00:00
00:00:01	00:00:01	00:00:00

would be expensive to find out, because relative to Y2K, the questions are a lot harder than “2 digits or 4 digits.”

How do we tell if code that does `s += 3600` intends this to mean “one hour from now” or “same time, next hour?” The original programmer did not expect there to be any difference, so the documentation will not tell us.

The Cost of Uncertainty

The next time Bulletin C tells us to insert a leap second, probably in 2012, a lot of people will have to kick into action. Any critical bits installed since December 2008 and any bits older than that that failed to “do the right thing” with the December 2008 leap second will need to be pondered, and a plan made for what to do: test, fix, hope, or shut down.

Unsurprisingly, many plants and systems simply give up trying to predict what their multivendor heterogeneous systems will do with a leap second, and they sidestep the issue by moving or scheduling planned maintenance downtime to cover the leap second. For them, that is the cheapest way to make sure that no robot arms get out of sync with the assembly line and that no space-shuttle computers hiccup while in space.

I’m told from usually reliable sources that the entire U.S. nuclear deterrent is in “a special mode” for one hour on either side of a leap second and that the cost runs into “two-digit million dollars.”

But What Do Leap Seconds Actually Do?

Leap seconds make sure the sun is due south at noon by adjusting noon to happen when the sun is due south at the reference location. This very important job is handled by the International Earth Rotation Service (IERS).

Leap seconds are not a viable long-term solution because the earth’s rotation is not constant: tides and internal friction cause the planet to lose momentum and slow down the rotation, leading to a quadratic difference between earth rotation and atomic time. In the next century we will need a leap second every year, often twice every year; and 2,500 years from now we will need a leap second every month.

On the other hand, if we stop plugging leap seconds into our time scale,

There is no problem with leap seconds that we IT professionals cannot tolerate. We just have to make sure all computers know about leap seconds and that all programs, operating systems, and applications know how to deal with them.

noon on the clock will be midnight in the sky some 3,000 years from now, unless we fix that by adjusting our time zones.

Actually, the sun is not due south at noon, and certainly not with a second’s precision, for more than an infinitesimal number of people who are probably totally unaware of it. Our system of one-hour-wide time zones means that only those who live exactly on a longitude divisible by 15 have the chance, provided that their governments have not put them in a different time zone. For example, all of China is one time zone, despite the 75–120E span of longitude.

Of the remaining few lucky people, many are out of luck during the part of the year when their government has decided to have daylight saving time—although that could possibly put a select few of those who lost on the first criterion back in luck during that part of the year. Finally, it is really only a couple of times a year that the sun is precisely due south, for interesting orbital and geophysical reasons.

The people who really do care about UTC time being synchronized to earth rotation are those who use UTC time as an estimator for earth rotation: those who point things on earth at things in the sky—in other words, astronomers and their telescopes, and satellite operators and their antennae. Actually, that should more accurately be *some* of those people: many of them have long since given up on using UTC as an earth rotation estimator, because the ± 1 -second tolerance is not sufficient for their needs. Instead, they pick up Bulletin A or B from the IERS FTP server, which gives daily values with microsecond precision.

The Cost-Benefit Equation

Most of those involved on the “Abolish Leap Seconds” side of the debate claim a cost-benefit equation that essentially says: “cost of fixing all computers to deal correctly with leap seconds = infinity” over “benefits of leap seconds = next to nothing.” QED: case closed.

The vocal leaders of the “Preserve the Leap Seconds” campaign (not to be confused with the “Campaign for Real Time”) have a different take on the equation: “cost of unknown consequences of decoupling civil time from earth rotation = [a lot...infinity]” over

“programmers should fix their past mistakes for free.” QED: case closed.

Not a lot of common ground there, and not a lot of data supporting either proposition, although Y2K experience, as well as the principles of a capitalist economy, dictate that getting programmers to handle leap seconds correctly will be expensive.

A Possible Compromise?

Warner Losh, a fellow time-and-computer nerd, and I both have extensive hands-on experience with leap-second handling in critical systems, and we have tried to suggest a compromise on leap seconds that would vastly reduce the costs and risks involved: schedule the darn things 20 years in advance instead of only six months in advance.

If we know when leap seconds are to occur 20 years in advance, we can code them into tables in our operating systems, and suddenly 99.9% of our computers will do the right thing when leap seconds happen, because they know when they will happen. The remaining 0.1% of the systems, involving ready, cold spares on shelves, autonomous computers on the South Pole, and similar systems, get 20 years to update stored tables rather than six months to do so.

The astronomical flip side of this proposal is that the difference between earth rotation and UTC time would likely exceed the current one-second tolerance limit, at least until geophysicists get a better understanding of the currently not understood fluctuations in earth rotation.

The IT flip side is that we would still have a variable radix time scale: most minutes would be 60 seconds, but a few would be 61 seconds, and code that really cares about time intervals would have to do the right thing instead of just adding 86,400 seconds per day.

So far, nobody has tried, or if they tried, they failed to inject this idea into the official standards process in ITU-R. It is not clear to me that it would even be possible to inject this idea unless a national government, seconded by another, officially raises it at the ITU plenary assembly.

What Happens Next?

Proposal TF-460-7 to abolish leap seconds will come up for plenary vote at the

ITU-R in January 2012, and if it, modulo amendments, collects a supermajority of 70% of the votes, leap seconds would cease beginning in approximately 2018.

If the proposal fails to gain 70% of the votes, then leap seconds will continue, and we had better start fixing computers to deal properly, or at least more predictably, with them.

As I understand the voting rules of ITU-R, only country representatives can vote, one vote per country. If my experience is anything to go by, finding out who votes on behalf of your country and how they intend to vote may not be immediately obvious to the casually inquiring citizen.

The Philosophical Issues

One of my Jewish friends explained to me that all the rules Jews must follow are not meant to make sense; they are meant to make life so difficult that you never take it for granted. In the same spirit, Van Halen used brown M&Ms to test for lack of attention, and I use leap seconds: if a system has not documented and tested what happens on leap seconds, I don't trust it to get anything else right, either.

But Linus Torvalds' observation that “95% of all programmers think they are in the top 5%, and the rest are certain they are above average” should not be taken lightly: very few programmers have any idea what the difference is between “wall-clock time” and “interval time,” and leap seconds are way past rocket science for them. (For example, Posix defines only a `pthread_cond_timedwait()`, which takes wall-clock time but not an interval-time version of the call.)

When a large fraction of the world economy is run by the creations of lousy programmers, and when embedded systems are increasingly capable of killing people, do we raise the bar and demand that programmers pay attention to pointless details such as leap seconds, or do we remove leap seconds?

As an old-timer in the IT business, I'm firmly for the first option: we should always strive to do things better, and do them right, and pointless details makes for good checkboxes. As a frequent user of technological marvels built by the lowest bidder, however, the second option is not unattractive—particularly when the pilots tell us they “have to turn

the entire plane off and on again before we can start all the motors.”

As a time-nut, a small and crazy fraternity that thinks running an atomic clock in your basement is a requirement for a good life (let me know if you need a copy of my 400GB recording of the European VLF spectrum during a leap second...), I would miss leap seconds. They are quaint and interesting, and their present rate of one every couple of years makes for a wonderful chance to inspire young nerds with tales of wonders in physics and geophysics.

But once every couple of years is not nearly often enough to ensure that IT systems handle them correctly.

I wish we could somehow get the 20-year horizon compromise on the table next January, but failing that, if the choice is only between keeping leap seconds or abolishing leap seconds, they will have to go—before they kill somebody through bad standards writing and bad programming.

Related articles on queue.acm.org

Principles of Robust Timing over the Internet

Julien Ridoux, Darryl Veitch
<http://queue.acm.org/detail.cfm?id=1773943>

You Don't Know Jack about Network Performance

Kevin Fall, Steve McCanne
<http://queue.acm.org/detail.cfm?id=1066069>

Fighting Physics: A Tough Battle

Jonathan M. Smith
<http://queue.acm.org/detail.cfm?id=1530063>

References

1. International Earth Rotation and Reference Systems Service. Information on UTC-TAI; <http://data.iers.org/products/16/14433/orig/bulletinc-041.txt>.
2. International Earth Rotation and Reference Systems Service. Relationship between TAI and UTC; http://hpiers.obspm.fr/eop-pc/earthor/utc/TAI-UTC_tab.html.
3. Microsoft. How the Windows Time service treats a leap second (2006). (November 1); <http://support.microsoft.com/kb/909614>.
4. Sobel, D. *Longitude*. Walker and Company, 2005.
5. Williams, M. Power glitch hits Toshiba's flash memory production line. *ComputerWorld* (Dec. 2010); http://www.computerworld.com/s/article/9200738/Power_glitch_hits_Toshiba_s_flash_memory_production_line.

Poul-Henning Kamp (phk@FreeBSD.org) has programmed computers for 26 years and is the inspiration behind bikeshed.org. His software has been widely adopted as “under the hood” building blocks in both open source and commercial products. His most recent project is the Varnish HTTP accelerator, which is used to speed up large Web sites such as Facebook.