One

LOSING TRACK OF TIME

On 14 September 2004 around eight hundred aircraft were making long-distance flights above Southern California. A mathematical mistake was about to threaten the lives of the tens of thousands of people onboard. Without warning, the Los Angeles Air Route Traffic Control Center lost radio voice contact with all the aircraft. A justifiable amount of panic ensued.

The radios were down for about three hours, during which time the controllers used their personal mobile phones to contact other traffic control centres to get the aircraft to retune their communications. There were no accidents but, in the chaos, ten aircraft flew closer to each other than regulations allowed (5 nautical miles horizontally or 2,000 feet vertically); two pairs passed within 2 miles of each other. Four hundred flights on the ground were delayed and a further six hundred cancelled. All because of a maths error.

Official details are scant on the precise nature of what went wrong but we do know it was due to a timekeeping error within the computers running the control centre. It seems the air traffic control system kept track of time by starting at 4,294,967,295 and counting down once a millisecond. Which meant that it would take 49 days, 17 hours, 2 minutes and 47.295 seconds to reach 0.

Usually, the machine would be restarted before that happened, and the countdown would begin again from 4,294,967,295. From what I can tell, some people were aware of the potential issue so it was policy to restart the system at least every thirty days. But this was just a way of working around the problem; it did nothing to correct the underlying mathematical error, which was that nobody had checked how many milliseconds there would be in the probable runtime of the system. So, in 2004, it accidentally ran for fifty days straight, hit zero, and shut down. Eight hundred aircraft travelling through one of the world's biggest cities were put at risk because, essentially, someone didn't choose a big enough number.

People were quick to blame the issue on a recent upgrade of the computer systems to run a variation of the Windows operating system. Some of the early versions of Windows (most notably Windows 95) suffered from exactly the same problem. Whenever you started the program, Windows would count up once every millisecond to give the 'system time' that would drive all the other programs. But once the Windows system time hit 4,294,967,295, it would loop back to zero. Some programs – drivers, which allow the operating system to interact with external devices – would have an issue with time suddenly racing backwards. These drivers need to keep track of time to make sure the devices are regularly responding and do not freeze for too long. When Windows told them that time had abruptly started to go backwards, they would crash and take the whole system down with them.

It is unclear if Windows itself was directly to blame or if it was a new piece of computer code within the control centre system itself. But, either way, we do know that the number 4,294,967,295 is to blame. It wasn't big enough for people's home desktop computers in the 1990s and it was not big enough for air traffic control in the early 2000s. Oh, and it was not big enough in 2015 for the Boeing 787 Dreamliner aircraft.

The problem with the Boeing 787 lay in the system that controlled the electrical power generators. It seems they kept track of time using a counter that would count up once every 10 milliseconds (so, a hundred times a second) and it capped out at 2,147,483,647 (suspiciously close to half of 4,294,967,295 . . .). This means that the Boeing 787 could lose electrical power if turned on continuously for 248 days, 13 hours, 13 minutes and 56.47 seconds. This was long enough that most planes would be restarted before there was a problem but short enough that power could, feasibly, be lost. The Federal Aviation Administration described the situation like this:

> The software counter internal to the generator control units (GCUs) will overflow after 248 days of continuous power, causing that GCU to go into failsafe mode. If the four main GCUs (associated with the engine-mounted generators) were powered up at the same time, after 248 days of continuous power, all four GCUs will go into failsafe mode at the same time, resulting in a loss of all AC electrical power regardless of flight phase.

I believe that 'regardless of flight phase' is official FAA speak for 'This could go down mid-flight.' Their official line on airworthiness was the requirement of 'repetitive maintenance tasks for electrical power deactivation'. That is to say, anyone with a Boeing 787 had to remember to turn it off and on again. It's the classic computer programmer fix. Boeing has since updated its program to fix the problem, so preparing the plane for take-off no longer involves a quick restart.

When 4.3 billion milliseconds is just not enough

So why would Microsoft, Los Angeles Air Route Traffic Control Center and Boeing all limit themselves to this seemingly arbitrary number of around 4.3 billion (or half of it) when keeping track of time? It certainly seems to be a widespread problem. There is a massive clue if you look at the number 4,294,967,295 in binary. Written in the 1s and 0s of computer code, it becomes 11111111111111111111111111111111; a string of thirty-two consecutive ones.

Most humans never need to go near the actual circuits or binary code on which computers are built. They only need to worry about the programs and apps which run on their devices and, occasionally, the operating system on which those programs run (such as Windows or iOS). All these use the normal digits of 0 to 9 in the base-10 numbers we all know and love.

But beneath it all lies binary code. When people use Windows on a computer or iOS on a phone, they are interacting only with the graphical user interface, or GUI (delightfully pronounced 'gooey'). Below the GUI is where it gets messy. There are layers of computer code taking the mouse clicks and swipe lefts of the human using the device and converting them into the harsh machine code of 1s and 0s that is the native language of computers.

If you had space for only five digits on a piece of paper, the largest number you could write down would be 99,999. You've filled every spot with the largest digit available. What the Microsoft, air traffic control and Boeing systems all had in common is that they were 32-bit binary-number systems, which means the default is that the largest number they can write down is thirty-two 1s in binary, or 4,294,967,295 in base-10.

It was slightly worse in systems that wanted to use one of the thirty-two spots for something else. If you wanted to use that piece of paper with room for five symbols to write down a negative number, you'd need to leave the first spot free for a positive or negative sign, which would mean that you could now write down all the whole numbers between −9,999 and +9,999. It's believed Boeing's system used such 'signed numbers', so, with the first spot taken,* they only had room for a maximum of thirty-one 1s, which translates into 2,147,483,647. Counting only centiseconds rather than milliseconds bought them some time – but not enough.

Thankfully, this is a can that can be kicked far enough down the road that it does not matter. Modern computer systems are generally 64-bit, which allows for much bigger numbers by default. The maximum possible value is of course still finite, so any computer system is assuming that it will eventually be turned off and on again. But if a 64-bit system counts milliseconds, it will not hit that limit until 584.9 million years have passed. So you don't need to worry: it will need a restart only twice every billion years.

Calendars

The analogue methods of timekeeping we used before the invention of computers would, at least, never run out of room.

* Of course, you cannot save a + or – symbol in a binary number, so a system is used to indicate positive or negative using the binary itself, but it still takes up a bit of space.

The hands of a clock can keep spinning around; new pages can be added to the calendar as the years go by. Forget milliseconds: with only good old-fashioned days and years to worry about, you will not have any maths mistakes ruining your day.

Or so thought the Russian shooting team as they arrived at the 1908 Olympic Games in London a few days before the international shooting was scheduled to start on 10 July. But if you look at the results of the 1908 Olympics, you'll see that all the other countries did well but there are no Russian results for any shooting event. And that is because what was 10 July for the Russians was 23 July in the UK (and indeed most of the rest of the world). The Russians were using a different calendar.

It seems odd that something as straightforward as a calendar can go so wrong that a team of international athletes show up at the Olympics two weeks late. But calendars are far more complex than you'd expect; it seems that dividing the year up into predictable days is not easy and there are different solutions to the same problems.

The universe has given us only two units of time: the year and the day. Everything else is the creation of humankind to try to make life easier. As the protoplanetary disc congealed and separated into the planets as we know them, the Earth was made with a certain amount of angular momentum, sending it flying around the sun, spinning as it goes. The orbit we ended up in gave us the length of the year, and the rate of the Earth's spin gave us the length of the day.

Except they don't match. There is no reason they should! It was just where the chunks of rock from that protoplanetary disc happened to fall, billions of years ago. The year-long orbit of the Earth around the sun now takes 365 days, 6 hours, 9 minutes and 10 seconds. For simplicity, we can call that 365 and a quarter days.

This means that, if you celebrate New Year's Eve after a year of 365 days, the Earth still has a quarter of a day of movement before you'll be back to exactly where you were last New Year's Eve. The Earth is tearing around the sun at a speed of around 30 kilometres every second, so this New Year's Eve you will be over 650,000 kilometres away from wherever you were last year. So, if your New Year's resolution was to not be late for things, you're already way behind.

This goes from being a minor inconvenience to becoming a major problem because the Earth's orbital year controls the seasons. The northern hemisphere summer occurs around the same point in the Earth's orbit every year because this is where the Earth's tilt aligns with the position of the sun. After every 365-day year, the calendar year moves a quarter of a day away from the seasons. After four years, summer would start a day later. In less than four hundred years, within the lifespan of a civilization, the seasons would drift by three months. After eight hundred years, summer and winter would swap places completely.

To fix this, we had to tweak the calendar to have the same number of days as the orbit. Somehow, we needed to break away from having the same number of days every year, but without having a fraction of a day; people get upset if you restart the day at a time other than midnight. We needed to link a year to the Earth's orbit without breaking the tie between a day and the Earth's rotation.

The solution that most civilizations came up with was to vary the number of days in any given year so there is a fractional number of days per year on average. But there is no single way to do that, which is why there are still a few competing calendars around today (which all start at different points in history). If you ever have access to a friend's phone, go into the settings and change their calendar to the

Buddhist one. Suddenly, they're living in the 2560s. Maybe try to convince them they have just woken up from a coma.

Our main modern calendar is a descendant of the Roman Republican calendar. They had only 355 days, which was substantially fewer than required, so an entire extra month was inserted between February and March, adding an extra twenty-two or twenty-three days to the year. In theory, this adjustment could be used to keep the calendar aligned with the solar year. In practice, it was up to the reigning politicians to decide when the extra month should be inserted. As this decision could either lengthen their year of ruling or shorten that of an opponent, the motivation was not always to keep the calendar aligned.

A political committee is rarely a good solution to a mathematical problem. The years leading up to 46BCE were known as the 'years of confusion', as extra months came and went, with little relation to when they were needed. A lack of notice could also mean that people travelling away from Rome would have to guess what the date back at home was.

In 46BCE Julius Caesar decided to fix this with a new, predictable calendar. Every year would have 365 days – the closest whole number to the true value – and the bonus quarter days would be saved up until every fourth year, which would have a single bonus day. The leap year with an extra leap day was born!

To get everything back into alignment in the first place, the year 46BCE had a possible-world-record 445 days. In addition to the bonus month between February and March, two more months were inserted between November and December. Then, from 45BCE, leap years were inserted every four years to keep the calendar in synch.

Well, almost. There was an initial clerical error, where the last year in a four-year period was double-counted as the first year of the next period, so leap years were actually put in every three years. But this was spotted, fixed and, by 3CE, everything was on track.

The audacity of Pope

But Julius Caesar was betrayed – albeit long after his death – by the 11 minutes and 15 seconds difference between the 365.25 days per year his calendar gave and the actual time between seasons of 365.242188792 days. An eleven-minute drift per year is not that noticeable to start with; the seasons move only one day every 128 years. But after a millennium or so of drift, it would accumulate. And the young upstart religion of Christianity had pinned their celebration of Easter to the timing of the seasons, and by the early 1500s there was a ten-day gap between the date and the actual start of spring.

And now for a niche fact. There is an oft-repeated statement that the Julian calendar years of 365.25 days were too long compared to the Earth's orbit. But that is incorrect! The Earth's orbit is 365 days, 6 hours, 9 minutes and 10 seconds: slightly more than 365.25 days. The Julian calendar is too *short* compared to the orbit. But it is too *long* compared to the seasons. Bizarrely, the seasons don't even exactly match the orbital year.

We're now at the level of calendar resolution when other orbital mechanics come into play. As the Earth orbits, the direction it is leaning also changes, going from pointing directly at the sun to pointing away every 13,000 years. A calendar perfectly matching the Earth's orbit will still swap the seasons every 13,000 years. If we factor the Earth's axial precession (the change in how it leans) into its orbit, the time between seasons is 365 days, 5 hours, 48 minutes and 45.11 seconds.

The movement of the Earth's tilt buys us an extra 20 minutes and 24.43 seconds per orbit. So the true *sidereal* (literally, 'of the stars') year based on the orbit is longer than the Julian calendar, but the *tropical* year based on the seasons (which we actually care about) is shorter. It's because the seasons depend on the tilt of the Earth relative to the sun, not on the actual position of the Earth. You have my permission to photocopy this part of the book and hand it to anyone who gets the type of year wrong. Maybe suggest their New Year resolution should be to understand what a new year actually is.

Sidereal year

31,558,150 seconds = 365.2563657 days
365 days, 6 hours, 9 minutes, 10 seconds

Tropical year

31,556,925 seconds = 365.2421875 days
365 days, 5 hours, 48 minutes, 45 seconds

This slight mismatch between the Julian and tropical years was unnoticeable enough that, by 1500CE, pretty much all of Europe and parts of Africa were using the Julian calendar. But the Catholic Church was sick of Jesus's death (celebrated according to the seasons) drifting away from his birth (celebrated on a set date). Pope Gregory XIII decided something had to be done. Everyone would need to update to a new calendar. Thankfully, if there's one thing a pope can do, it's convince a lot of people to change their behaviour for seemingly arbitrary reasons.

What we now know as the Gregorian calendar was not actually designed by Pope Greg – he was too busy doing pope things and convincing people to change their behaviour – but by the Italian doctor and astronomer Aloysius 'Luigi' Lilius. Luigi unfortunately died in 1576, two years before the calendar reform commission released his (slightly tweaked) calendar. With the slight nudge of a papal bull in 1582 to bully them into it, a decent chunk of the world swapped over to the new calendar system that year.

Luigi's breakthrough was to keep the standard every-fourth-year leap year of the Julian calendar but to take out three leap days every four hundred years. Leap years were all the years divisible by four, and all Luigi suggested was to remove the leap days from years which were also a multiple of 100 (apart from those that were also a multiple of 400). This now averages out to 365.2425 days per year; impressively close to the desired tropical year of around 365.2422 days.

Despite it being a mathematically better calendar, because this new system was born out of Catholic holidays and promulgated by the pope, anti-Catholic countries were duly anti-Gregorian calendar. England (and, by extension at the time, North America) clung to the old Julian calendar for another century and a half, during which time their calendar not only drifted another day away from the seasons but was also different to the one used in most of Europe.

This problem was exacerbated because the Gregorian calendar was backdated, recalibrating the year as if it, rather than the Julian option, had always been used. Through the use of pope power, it was decreed that ten dates would be taken from October 1582 and so, in Catholic countries, 4 October 1582 was directly followed by 15 October. All this does of course make historical dates a bit confusing. When the English forces landed on Île de Ré on 12 July 1627 as part of the Anglo-French War, the French forces were ready to fight back on 22 July. That is, on exactly the same day. At least, for both armies, it was a Thursday.

However, as the Gregorian calendar became more about seasonal convenience and less about doing what the pope said, other countries gradually switched over. A British Act of Parliament from 1750 points out that not only do England's dates differ from those in the rest of Europe, they also differ from those in Scotland. So England swapped over, but without any direct mention of the pope; they merely referred indirectly to 'a method of correcting the calendar'.

England (which still – barely – included parts of North America) swapped over in 1752, realigning its dates by removing eleven days from September. Thus, 2 September 1752 was followed by 14 September 1752. Despite what you may read online, no one complained about losing eleven days of their life and no one carried a placard demanding, 'Give us our eleven days.' I know this for sure: I went to the British Library in London, which houses a copy of every newspaper ever published in England and looked up contemporary reports. No mention of complaint, only ads selling new calendars. Calendar creators were having the time of their life.

The myth that people protested against the calendar change seems to have come from political debates before an election in 1754. The opposition party was attacking everything the other party had done during its term in office, including the changes to the calendar and stealing eleven days. This was captured in *An Election Entertainment*, an oil painting by William Hogarth. The only contemporary concerns were expressed by people who did not want to pay a full 365 days' worth of tax on a year with fewer days. Legitimately, one might say.

Russia did not swap calendars until 1918, when it started February on the 14th rather than on the 1st to bring themselves back into alignment with everyone else on the Gregorian calendar. Which must have caught a lot of people off guard. Imagine waking up thinking you had two weeks only to find it's already Valentine's Day. This new calendar means the Russians would have been on time for the 1920 Olympics, had they been invited, but in the interim Russia had become Soviet Russia and was not invited for political reasons. The next Olympic Games attended by Russian athletes was in Helsinki in 1952, where they finally won a gold medal in shooting.

Despite all these improvements, our current Gregorian calendar is still not quite perfect. An average of 365.2425 days per year is good, but it's not exactly 365.2421875. We're still out by twenty-seven seconds a year. This means that our current Gregorian calendar will drift a whole day once every 3,213 years. The seasons will still reverse once every half a million years. And you will be alarmed to know that there are currently no plans to fix this!

In fact, on such long timescales, we have other problems to worry about. As well as the Earth's axis of rotation moving about, the orbital path of the Earth moves around as well. The path is an ellipse, and the closest and most distant locations do a lap around the solar system about once every 112,000 years. But even then the gravitational tug of other planets can mess it up. The solar system is a sloshy mess.

But astronomy does give Julius Caesar the last laugh. The unit of a light-year, that is, the distance travelled by light in a year (in a vacuum) is specified using the Julian year of 365.25 days. So we measure our current cosmos using a unit in part defined by an ancient Roman.

The day time will stand still

At 3.14 a.m. on Tuesday, 19 January 2038 many of our modern microprocessors and computers are going to stop working. And all because of how they store the current date and time. Individual computers already have enough problems keeping track of how many seconds have passed while they are turned on; things get worse when they also need to keep completely up to date with the date. Computer timekeeping has all the ancient problems of keeping a calendar in synch with the planet *plus* the modern limitations of binary encoding.

When the first precursors to the modern internet started to come online in the early 1970s a consistent timekeeping standard was required. The Institute of Electrical and Electronics Engineers threw a committee of people at the problem and, in 1971, they suggested that all computer systems could count sixtieths of a second from the start of 1971. The electrical power driving the computers was already coming in at a rate of 60 Hertz, so it simplified things to use this frequency within the system. Very clever. Except that a 60-Hertz system would exceed the space in a 32-digit binary number in a little over two years and three months. Not so clever.

So the system was recalibrated to count the number of whole seconds since the start of 1970. This number was stored as a signed 32-digit binary number which allowed for a maximum of 2,147,483,647 seconds: a total of over sixty-eight years from 1970. And this was put in place by members of the generation who in the sixty-eight years leading up to 1970 had seen humankind go from the Wright Brothers inventing the first powered aeroplane to humans dancing on the Moon. They were sure that, by the year 2038, computers would have changed beyond all recognition and no longer use Unix time.

Yet here we are. More than halfway there and we're still on the same system. The clock is literally ticking.

Computers have indeed changed beyond recognition, but the Unix time beneath them is still there. If you're running any flavour of Linux device or a Mac, it is there in the lower half of the operating system, right below the GUI. If you have a Mac within reach, open up the app Terminal, which is the gateway to how your computer actually works. Type in **date +%s** and hit Enter. Staring you in the face will be the number of seconds that have passed since 1 January 1970.

If you're reading this before Wednesday, 18 May 2033 it is still coming up on 2 billion seconds. What a party that will be. Sadly, in my time zone, it will be around 4.30 a.m. I remember a boozy night out on 13 February 2009 with some mates to celebrate 1,234,567,890 seconds having passed, at just after 11.31 p.m. My programmer friend Jon had written a program to give us the exact countdown; everyone else in the bar was very confused why we were celebrating Valentine's Day half an hour early.

Celebrations aside, we are now well over halfway through the count-up to destruction. After 2,147,483,647 seconds, everything stops. Microsoft Windows has its own timekeeping system, but MacOS is built directly on Unix. More importantly, many significant computer processors in everything from internet servers to your washing machine will be running some descendant of Unix. They are all vulnerable to the Y2K38 bug.

I don't blame the people who originally set up Unix time. They were working with what they had available back then.

The engineers of the 1970s figured that someone else, further into the future, would fix the problems they were causing (classic baby-boomers). And to be fair, sixty-eight years is a very long time. The first edition of this book was published in 2019 and, occasionally, I think about ways to future-proof it. Maybe I'll include 'at the time of writing' or carefully structure the language to allow for things to change and progress in the future so that it doesn't go completely out of date. You might be reading this after the 2 billion second mark in 2033; I've allowed for that. But at no point do I think about people reading it in 2087. That's sixty-eight years away!

Some steps have already been taken towards a solution. All the processors which use 32-digit binary numbers by default are known as 32-bit systems. When buying a new laptop, you may not have paused to check what the default binary encoding was, but Macs have been 64-bit for nearly a decade now and most commonly used computer servers will have gone up to 64 bits as well. Annoyingly, some 64-bit systems will still track time as a signed 32-bit number so they can still play nicely with their older computer friends but, for the most part, if you buy a 64-bit system it will be able to keep track of time for quite a while to come.

The largest value you can store in a signed 64-bit number is 9,223,372,036,854,775,807, and that number of seconds is equivalent to 292.3 billion years. It's times like this when the age of the universe becomes a useful unit of measurement: 64-bit Unix time will last until twenty-one times the current age of the universe from now. Until – and assuming we don't manage another upgrade in the meantime – on 4 December in the year 292277026596CE all the computers will go down. On a Sunday.

Once we live in an entirely 64-bit world, we are safe. The question is: will we upgrade all the multitude of microprocessors in our lives before 2038? We need either new processors or a patch that will force the old ones to use an unusually big number to store the time.

Here is a list of all the things I've had to update the software on recently: my light bulbs, a TV, my home thermostat and the media player that plugs into my TV. I am pretty certain they are all 32-bit systems. Will they be updated in time? Knowing my obsession with up-to-date firmware, probably. But there are going to be a lot of systems that will not get upgraded. There are also processors in my washing machine, dishwasher and car, and I have no idea how to update those.

It's easy to write this off as a second coming of the Y2K 'millennium bug' that wasn't. That was a case of higher level software storing the year as a two-digit number, which would run out after ninety-nine. Through a massive effort, almost everything was updated. But a disaster averted does not mean it was never a threat in the first place. It's risky to be complacent because Y2K was handled so well. Y2K38 will require updating far more fundamental computer code and, in some cases, the computers themselves.

See for yourself

If you want to see the Y2K38 bug in action for yourself, find an iPhone. This may work for other phones, or the iPhone may one day be updated to fix this. But, for now, the built-in stopwatch on the iPhone piggybacks on the internal clock and stores its value as a signed 32-bit number. The reliance on the clock means that, if you start the stopwatch and then change the time forwards, the time elapsed on the stopwatch will suddenly jump forward. By repeatedly moving the time and date on your phone forwards and backwards, you can ratchet up the stopwatch at an alarming rate. Until it hits the 32-bit limit and crashes.

When you really F-22 it up

How hard can it be to know what date it is? Or will be? I could safely state that 64-bit Unix time will run out on 4 December 292277026596CE because the Gregorian calendar is very predictable. In the short term, it is super easy and loops every few years. Allowing for the two types of year (leap and normal), and the seven possible days a year can start on, there are only fourteen calendars to choose from. When I was shopping for a 2019 calendar (non-leap year, starting on a Tuesday), I knew it would be the same as the one for 2013 so I could pick up a second-hand one at a discount price. Actually, for some retro charm, I hunted down one from 1985.

If you care about the sequence of years, the Gregorian calendar loops perfectly every four hundred years after a complete cycle of meta-leap years (the cycle of leaping leap years). So, the day you are enjoying now is exactly the same as the day it was four hundred years ago. You would think this would make it easy to program it into a computer. And it is, if the computer stays still. But as soon as the computer can move, it starts to get complicated.

Mistake from the internet

> GOOD LUCK EVERYONE!!! This year, December has 5 Mondays, 5 Saturdays and 5 Sundays. This happens once every 823 years. This is called money bags. So share it and money will arrive within 4 days. Based on Chinese Feng Shui. The one who does not share will be without money. Share within 11 minutes of reading. Can't hurt so I did it. JUST FOR FUN.

This is one of many popular internet memes which claim that something happens only every 823 years. I have no idea where the number

823 came from. But, for some reason, the internet is rife with claims that the current year is special and that this specialness will not be repeated for 823 years.

Now you can safely reply and say that nothing in the Gregorian calendar can happen less frequently than once every four hundred years. JUST FOR FUN.

And, given that there are only four possible month lengths and seven different starting days, there are actually only twenty-eight possible arrangements for the days of a month. So stuff like this actually happens every few years. (Not based on Chinese Feng Shui.)

In December 2005 the first F-22 Raptor fighter aircraft came into service. To quote the United States Air Force (USAF), 'The F-22 is a first-of-a-kind multi-mission fighter aircraft that combines stealth, supercruise, advanced maneuverability and integrated avionics to make it the world's most capable combat aircraft.' But, to be fair, this was taken from the budget statement in which the air force was trying to justify the expense. The USAF ran the numbers and estimated that, by 2009, the cost of getting each F-22 in the air was $150,389,000.

The F-22 certainly did have some really integrated avionics. In older aircraft, the pilot would be physically flying the plane with controls that used cables to raise and lower flaps, and so on. Not the F-22. Everything is done by computer. How else can you get advanced manoeuvrability and capable combat? Computers are the way forward. But, like planes, computers are all well and good – until they crash.

In February 2007 six F-22s were flying from Hawaii to Japan when all their systems crashed at once. All navigation systems went offline, the fuel systems went and even some of the communication systems were out. This was not triggered by an enemy attack or clever sabotage. The aircraft had merely flown over the International Date Line.

Everyone wants midday to be roughly when the sun is directly overhead: the moment when that part of the Earth is pointing straight at the sun. The Earth spins towards the east, so, when it is midday for you, everywhere to the east has already had midday (and has now overshot the sun), while everywhere to the west is waiting for their turn in the noon sun. This is why, as you move east, each time zone increases by an hour (or so).

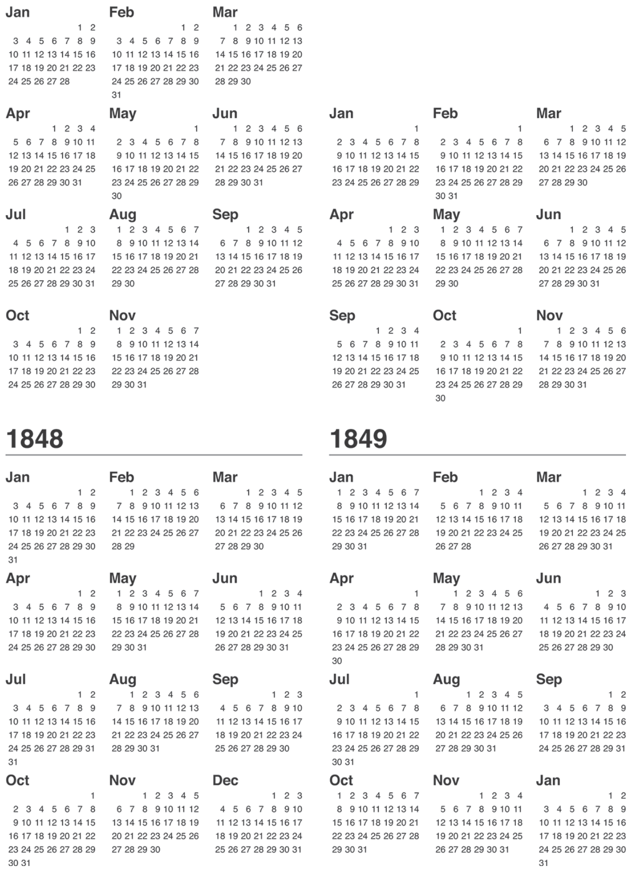
But this has to stop eventually; you can't go forward in time constantly while travelling east. If you were to magically lap the planet at a super-fast rate, you wouldn't get back to where you started and find it was a complete day in the future. At some point, the end of one day has to meet, well, the day before it. By stepping over the International Date Line, you go back (or forward) a complete day in the calendar.

If you're finding it hard to get your head around this, you're not alone. The International Date Line causes all sorts of confusion and whoever was programming the F-22 must have struggled to work it out. The US Air Force has not confirmed what went wrong (only that it was fixed within forty-eight hours), but it seems that time suddenly jumped by a day and the plane freaked out and decided that shutting everything down was the best course of action. Mid-flight attempts to restart the system proved unsuccessful so, while the planes could still fly, the pilots couldn't navigate. The planes had to limp home by following their nearby refuelling aircraft.

Modern fighter jet or ancient Roman rulers: sooner or later, time catches up with everyone.

Calen-duh

Programmer Nick Day emailed me when he noticed that the calendar on iOS devices seems to break in 1847. Suddenly, February has thirty-one days. And January has twenty-eight days. July is strangely unreliable; December has vanished completely. For the years before 1848, the year headers have disappeared. If you open the default calendar on an iPhone in 'year view', it takes only a few seconds of frantic swiping down to see this for yourself.



```
Jan                   Feb                   Mar
                1  2                  1  2      1  2  3  4  5  6
  3  4  5  6  7  8  9   3  4  5  6  7  8  9   7  8  9 10 11 12 13
 10 11 12 13 14 15 16  10 11 12 13 14 15 16  14 15 16 17 18 19 20
 17 18 19 20 21 22 23  17 18 19 20 21 22 23  21 22 23 24 25 26 27
 24 25 26 27 28        24 25 26 27 28 29 30  28 29 30
                       31

Apr                   May                   Jun
          1  2  3  4                     1      1  2  3  4  5  6
  5  6  7  8  9 10 11   2  3  4  5  6  7  8   7  8  9 10 11 12 13
 12 13 14 15 16 17 18   9 10 11 12 13 14 15  14 15 16 17 18 19 20
 19 20 21 22 23 24 25  16 17 18 19 20 21 22  21 22 23 24 25 26 27
 26 27 28 29 30 31     23 24 25 26 27 28 29  28 29 30 31
                       30

Jul                   Aug                   Sep
             1  2  3    1  2  3  4  5  6  7         1  2  3  4  5
  4  5  6  7  8  9 10   8  9 10 11 12 13 14   6  7  8  9 10 11 12
 11 12 13 14 15 16 17  15 16 17 18 19 20 21  13 14 15 16 17 18 19
 18 19 20 21 22 23 24  22 23 24 25 26 27 28  20 21 22 23 24 25 26
 25 26 27 28 29 30 31  29 30                 27 28 29 30 31

Oct                   Nov
                1  2    1  2  3  4  5  6  7
  3  4  5  6  7  8  9   8  9 10 11 12 13 14
 10 11 12 13 14 15 16  15 16 17 18 19 20 21
 17 18 19 20 21 22 23  22 23 24 25 26 27 28
 24 25 26 27 28 29 30  29 30 31
```

```
Jan                   Feb                   Mar
                   1                     1      1  2  3  4  5
  2  3  4  5  6  7  8   2  3  4  5  6  7  8   6  7  8  9 10 11 12
  9 10 11 12 13 14 15   9 10 11 12 13 14 15  13 14 15 16 17 18 19
 16 17 18 19 20 21 22  16 17 18 19 20 21 22  20 21 22 23 24 25 26
 23 24 25 26 27 28 29  23 24 25 26 27 28 29  27 28 29 30
                       30 31

Apr                   May                   Jun
             1  2  3    1  2  3  4  5  6  7         1  2  3  4  5
  4  5  6  7  8  9 10   8  9 10 11 12 13 14   6  7  8  9 10 11 12
 11 12 13 14 15 16 17  15 16 17 18 19 20 21  13 14 15 16 17 18 19
 18 19 20 21 22 23 24  22 23 24 25 26 27 28  20 21 22 23 24 25 26
 25 26 27 28 29 30 31  29 30                 27 28 29 30 31

Sep                   Oct                   Nov
          1  2  3  4                     1      1  2  3  4  5  6
  5  6  7  8  9 10 11   2  3  4  5  6  7  8   7  8  9 10 11 12 13
 12 13 14 15 16 17 18   9 10 11 12 13 14 15  14 15 16 17 18 19 20
 19 20 21 22 23 24 25  16 17 18 19 20 21 22  21 22 23 24 25 26 27
 26 27 28 29 30 31     23 24 25 26 27 28 29  28 29 30 31
                       30
```

```
1848

Jan                   Feb                   Mar
                1  2      1  2  3  4  5  6         1  2  3  4  5
  3  4  5  6  7  8  9   7  8  9 10 11 12 13   6  7  8  9 10 11 12
 10 11 12 13 14 15 16  14 15 16 17 18 19 20  13 14 15 16 17 18 19
 17 18 19 20 21 22 23  21 22 23 24 25 26 27  20 21 22 23 24 25 26
 24 25 26 27 28 29 30  28 29                 27 28 29 30
 31

Apr                   May                   Jun
                1  2    1  2  3  4  5  6  7            1  2  3  4
  3  4  5  6  7  8  9   8  9 10 11 12 13 14   5  6  7  8  9 10 11
 10 11 12 13 14 15 16  15 16 17 18 19 20 21  12 13 14 15 16 17 18
 17 18 19 20 21 22 23  22 23 24 25 26 27 28  19 20 21 22 23 24 25
 24 25 26 27 28 29 30  29 30 31              26 27 28 29 30 31

Jul                   Aug                   Sep
                1  2      1  2  3  4  5  6               1  2  3
  3  4  5  6  7  8  9   7  8  9 10 11 12 13   4  5  6  7  8  9 10
 10 11 12 13 14 15 16  14 15 16 17 18 19 20  11 12 13 14 15 16 17
 17 18 19 20 21 22 23  21 22 23 24 25 26 27  18 19 20 21 22 23 24
 24 25 26 27 28 29 31  28 29 30 31           25 26 27 28 29 30
 31

Oct                   Nov                   Dec
                   1         1  2  3  4  5               1  2  3
  2  3  4  5  6  7  8   6  7  8  9 10 11 12   4  5  6  7  8  9 10
  9 10 11 12 13 14 15  13 14 15 16 17 18 19  11 12 13 14 15 16 17
 16 17 18 19 20 21 22  20 21 22 23 24 25 26  18 19 20 21 22 23 24
 23 24 25 26 27 28 29  27 28 29 30           25 26 27 28 29 30 31
 30 31
```

```
1849

Jan                   Feb                   Mar
  1  2  3  4  5  6  7            1  2  3  4            1  2  3  4
  8  9 10 11 12 13 14   5  6  7  8  9 10 11   5  6  7  8  9 10 11
 15 16 17 18 19 20 21  12 13 14 15 16 17 18  12 13 14 15 16 17 18
 22 23 24 25 26 27 28  19 20 21 22 23 24 25  19 20 21 22 23 24 25
 29 30 31              26 27 28              26 27 28 29 30 31

Apr                   May                   Jun
                   1      1  2  3  4  5  6               1  2  3
  2  3  4  5  6  7  8   7  8  9 10 11 12 13   4  5  6  7  8  9 10
  9 10 11 12 13 14 15  14 15 16 17 18 19 20  11 12 13 14 15 16 17
 16 17 18 19 20 21 22  21 22 23 24 25 26 27  18 19 20 21 22 23 24
 23 24 25 26 27 28 29  28 29 30 31           25 26 27 28 29 30
 30

Jul                   Aug                   Sep
                   1         1  2  3  4  5                  1  2
  2  3  4  5  6  7  8   6  7  8  9 10 11 12   3  4  5  6  7  8  9
  9 10 11 12 13 14 15  13 14 15 16 17 18 19  10 11 12 13 14 15 16
 16 17 18 19 20 21 22  20 21 22 23 24 25 26  17 18 19 20 21 22 23
 23 24 25 26 27 28 29  27 28 29 30 31        24 25 26 27 28 29 31
 30 31

Oct                   Nov                   Jan
  1  2  3  4  5  6  7            1  2  3  4                  1  2
  8  9 10 11 12 13 14   5  6  7  8  9 10 11   3  4  5  6  7  8  9
 15 16 17 18 19 20 21  12 13 14 15 16 17 18  10 11 12 13 14 15 16
 22 23 24 25 26 27 28  19 20 21 22 23 24 25  17 18 19 20 21 22 23
 29 30 31              26 27 28 29 30 31     24 25 26 27 28 29 30
                                             31
```

But why 1847? As far as I can tell, Nick was the first person to spot this, and I could not find an obvious link to Unix time and 32- or 64-bit numbers. But we have a working theory . . .

Apple has more than one time available at its disposal and sometimes uses CFAbsoluteTime, that is, the number of seconds after 1 January 2001. And if CFAbsoluteTime is stored as a signed 64-bit number with some of the digits dedicated to decimal places (a double-precision floating-point value), there would be only 52 bits of space for the integer number of seconds.

The largest possible number held in a 52-digit binary number is 4,503,599,627,370,495, and if you count back that many microseconds (instead of seconds) from 1 January 2001, you land on Friday, 16 April 1858 . . . which could be why it breaks around this date . . . maybe. Well, it's the best we've got!

If any Apple engineers can provide a definite answer, please get in touch.