WAITING FOR 01-01-00

Brian Hayes

When I was a boy growing up in Toledo, Ohio, my best friend was Jimmy Liccata, whose father owned Tony's Gulf Station a few blocks away. One day Jimmy and I were snooping in the Liccata garage, doubtless up to no good, and discovered a carton of blank receipt pads from the gas station—the kind with alternating white and yellow pages, and a purple sheet of carbon paper to slip between them. The tablets were an irresistible attraction to a couple of first graders just beginning their initiation into the mysteries of the written word. I was enchanted by the magic of seeing all my scribbles reproduced in duplicate. But what I remember most vividly about those long-lost gas-station forms (imprinted with an orange-and-blue emblem that has itself disappeared from the American landscape) is the space for filling in the date. It read: "_____, 195__." Seeing that inscription gave me my first hint of the alarming velocity of time. All my life it had been the 1950s, and up to that moment the decade had seemed absolutely endless. Now I realized that the Fifties would eventually pass away, and all that stationery would be made obsolete. The close of the decade still seemed unimaginably far off—these events took place in the summer of 1956—but even if I could not quite see myself living in the new world of 196__, I knew it was coming, inexorably. Perhaps I even foresaw that stationery imprinted "_____, 19___" would someday expire. What a thought! A new millennium.

The calendrical milestone that seemed so astronomically remote to a boy in 1956 is now bearing down on us at a sure and steady speed of 3,600 seconds per hour. We have only five years left before a certain Friday night brings to an end, all at once, a month, a year, a decade, a century and a millennium. On the following Saturday morning it is not just preprinted stationery that will become obsolete. Among all the other transformations to be expected as the new age dawns, it seems likely that many computer programs will begin acting a bit strangely that day. Here are some hypothetical examples:

Shortly after midnight on January 1, 2000, a program meant to make automatic back-up copies of computer files starts replacing documents dated 01-01-00 with versions from 12-31-99 or earlier. Similarly, a programmer's tool that automatically links together the latest components of software under development begins including outdated modules.

Because many computers interpret 01-01-00 as the first day of 1900 rather than 2000, they also take it to be a Monday rather than a Saturday. As a result, traffic signals and school bells operate on their weekday schedule, and the display of arrivals and departures at the railroad station shows the Monday-morning commuter trains. Somewhere in the nation a bank is robbed because a time lock allows a vault to open on Saturday.

At the airport, all flights are canceled. A computer in the maintenance department has grounded the aircraft because they are 99 years overdue for airframe and engine overhauls. Furthermore, some pilots appear to have been on duty for 875,000 hours, in violation of union and FAA work rules.

At the local dairy, the oldest milk on hand is supposed to be shipped first, but in the early weeks of the new millennium milk from the year 00 is given precedence. Indeed, any milk remaining from December 1999 will not be scheduled for shipment until the end of 2099. Meanwhile at the bakery across town a computer calculates that bread dated 01-01-00 must be a century old, and sends it to the landfill.

The next time you open up your computerized laboratory notebook, the software informs you haughtily that all entries must have monotonically increasing time stamps. Then it accuses you of tampering with the system clock.

When you get your first bank statement of the new year, you find transactions listed in a curious nonchronological sequence, with all those from 2000 preceding those from 1999. Moreover, your $1,000 deposit made in late December has earned almost 100 years of interest, and so you have a balance of $400,000. Don't be too quick to spend it, however. The next day

Brian Hayes is a former editor of American Scientist. *Address: 211 Dacian Avenue, Durham, NC 27701. Internet: bhayes@mercury.interpath.net.*

your Visa bill arrives, and you owe $136 million. And when the phone bill comes in, that Happy New Year call you placed just before midnight has been charged as 53 million minutes. Then the library sends you a notice of some *seriously* overdue books.

The Millennium Cruise

Events like these are awaited with anxiety—and perhaps also a hint of mischievous glee, as long as no one gets hurt—by the small band of computer professionals who track the hazards of information technology. The main gathering place for this band is the Risks Forum *(1)*, an Internet mailing list and newsgroup moderated by Peter G. Neumann of SRI International. The foibles of computer clocks and calendars are a favorite topic in the forum. Most of the imaginary calamities mentioned above are based on speculations published in Risks over the past few years *(2)*. Indeed, the perils of 01-01-00 have been so often anticipated in the forum that one contributor has proposed a turn-of-the-century cruise *(3)* for those who want to be out of harm's way when the calendar "rolls over." While the rest of us are stuck ashore, arguing over whether the 21st century begins in 2000 or 2001, the canny Risks readers will slip away aboard a craft with nonelectronic controls and no computer-aided navigational equipment.

Most of the problems cited above arise from representing calendar dates with just six decimal digits, in a format that allows only two digits for the year. A program that adopts this representation is necessarily limited to a 100-year span. Most such programs interpret the years 00 through 99 as 1900 through 1999. Thus when the calendar rolls over from 99 to 00, the date does not advance into the next century but returns to 1900 again. Computers that rely on a calendar of this kind are destined to repeat the 20th century over and over; they live in a cyclic universe, where the future wraps around to become the past again.

The concepts of "before" and "after" are circular in such a world. A chronological sorting of bank transactions or batches of milk is sure to yield some strange results near the turn of the century. Similarly, the Office of Vital Statistics should not be surprised a few years from now to register an entire generation of newborn children who are older than their parents and grandparents. All these oddities are simple consequences of the arithmetic of the cyclic calendar, in which 99 + 1 = 00, but 00 < 99.

When the arithmetic is more complex than numeric comparison, the exact effect of calendar rollover depends on the details of how the arithmetic is done. Consider the case of a bank calculating interest on a sum of money deposited on 12-20-99 and withdrawn on 01-10-00. Subtracting 12-20-99 from 01-10-00 ought to yield not

Figure 1. The Last Judgment, as envisioned circa 1000, when the end of time seemed near. At 2000, apocalypse will be computerized. (From Georg Leidinger, *Miniaturen der Staatsbibliotek München.*)

quite –100 years, or more precisely –36,503 days. On getting such a result, one thing a computer might do is—to use the technical term—*barf*. A negative interval makes no sense in this context, and a prudently written program might well include an explicit check for this possibility; on finding a negative value, the program would stop and signal an error. Even without explicit error-checking, the negative number of days might bring the program to a halt; for example, the program might at some point attempt to take the logarithm of this number, which is an impossible operation. Or, the unexpected negative value might have just the opposite effect, causing the program *not* to halt but instead to enter an infinite loop. In all of these cases the program ultimately refuses to produce an answer, which may be the most benign failure mode available in the circumstances.

There are lots of other possible outcomes. Because a negative period of deposit is not to be expected, a programmer might write the interest-calculating procedure in such a way that it ignores the sign of the result when subtracting dates. This is the kind of arithmetic assumed in several of the hypothetical events cited above. A related but subtly different approach is to do the arithmetic with unsigned integers, a system of numbers in which negative values simply do

not exist. In one common implementation of unsigned computer arithmetic, 0 – 99 is equal to 65,437; if your bank uses this number of years in the interest calculation, it had better leave room on your statement for a 1,650-digit dollar amount.

Another possibility is that the minus sign would propagate all the way through the computation, so that you would be credited with, say, –$397,000 in interest. This result has a certain logic to it, since the bank thinks you withdrew your money in 1900 but did not deposit it until 1999. Happily, interest on credit-card balances and loans might also cross over to the other side of the ledger, so that you would receive a fabulous refund.

In still another variation, a negative number could turn up as the exponent in the formula for compound interest. In this case both your assets and your liabilities would dwindle away to trivial sums; at –6 percent per year, a $1,000 deposit would be reduced to $2.95. Here's one more amusing prospect: If the bogus negative value creeps into the computation in *two* places, your interest could be either a debit or a credit depending on whether the money was on deposit for an odd or an even number of days.

And there are yet more ways for this simple-seeming calculation to go awry. Much computer arithmetic is done with 16-bit numbers, which can represent a total of 65,536 (i.e., 2^{16}) distinct values. A widely adopted convention for 16-bit arithmetic allows the integers from 0 through 32,767 to represent themselves, whereas the remaining values are interpreted as the negative integers from –32,767 through –1. The 36,503 days between 01-10-00 and 12-20-99 exceed the maximum positive value in this system and would therefore be interpreted as a negative integer, namely –29,033.

It would be easy to dream up still more error mechanisms. Indeed, it begins to appear that with enough ingenuity virtually any result could be gotten from this calculation—even the right result. Most programs on most computers will probably continue to operate normally in the new millennium. But there will also be a great blooming of bugs on that Saturday morning five years from now.

The Centenarian's Complaint

There is a quick fix for some of the difficulties noted above. Although two decimal digits can represent no more than 100 years, any 100-year span could be chosen. For example, a program could be designed to interpret the years from 30 through 99 as 1930 through 1999, yet see 00 through 29 as 2000 through 2029. In this way the program might be able to survive the millennial crisis, or at least postpone it for a few decades. But the fix has a cost: The system would become more vulnerable to other faults. In particular, anyone whose birthdate is before 1930 might find the software distinctly unfriendly.

No matter how the dates are shifted or rearranged, a calendrical system that covers only a finite period is bound to bump up against a limit at some point. We do not even have to wait for a magic midnight such as 01-01-00 before the trouble starts. Computations done with six-digit dates already cause occasional bewilderment. For example, in 1992 Mary Bandar of Winona, Minn., was invited to join kindergarten classes when her name turned up among others identified in a database search for people born in "88"; at the time Bandar was 104 years old *(4)*. Similarly, C. G. Blodgett's auto-insurance premium tripled after his 101st birthday, apparently because he was classified as a high-risk youthful driver *(5)*. (There is something particularly disturbing about this story, even apart from the idea of insuring a one-year-old driver: Why did the company wait until his hundred-and-*first* birthday to raise the premium?) Another Risks anecdote tells of the hospital computer that interpreted the blood count of a 99-year-old man by standards appropriate to that of a newborn *(6)*.

There are also computer systems whose built-in time bombs have a fuse shorter than 100 years. On September 19, 1989, dozens of hospitals found that computers used for bookkeeping and administration had ceased to function *(7)*; it was not a coincidence that September 19, 1989, was the 32,768th day after January 1, 1900. A few weeks later computers running the Michigan Terminal System began to fail *(8)*; the date was November 16, 1989, which was 32,767 days after March 1, 1900. The satellites of the Global Positioning System keep track of the date by counting the weeks since January 6, 1980. The count is maintained as a 10-bit value, and thus it has a maximum range of 1,024 weeks. It follows that on December 21, 1999, the counter will roll over, and GPS receivers will think it is 1980 all over again *(9)*. And the grand prize for planned obsolescence goes to a personal computer sold in the mid-1980s by AT&T *(10)*: It had a clock that ran out of ticks at the end of 1990.

Even systems that will outlast the 1900s will not get very far into the new millennium. The clock runs out on the Unix operating system in 2038, on the Macintosh in 2040 and on MS-DOS in 2048. Many other computer systems span the interval from 1901 through 2099; the likely reason for choosing these particular boundary dates is that they simplify leap-year calculations (2000 is a normal leap year, but 1900 and 2100 are not).

The Dusty Deck

Abbreviated date formats were adopted for reasons that doubtless seemed compelling at the

time. Why waste storage on digits that are always 1 and 9? Likewise, why force people to type four digits when the first two are always the same? Furthermore, some of the tools that programmers rely on encourage or even enforce a limited temporal horizon. The COBOL programming language, used for many business and financial applications, defines a year as a two-character data type. Ada, the language mandated for most Department of Defense software, is one of the 1901-to-2099 systems. Everyone knew, when these decisions were made, that time would undo them, but the day of reckoning was remote. To the programmer cobbling together a bank accounting system in 1962 it would have seemed comically pretentious to imagine that the program might still be in use in 2000. Given the brevity of the human life span, and the rapid pace of technological evolution, building a machine with a design life of 100 years should not earn you criticism for shortsightedness.

Yet here comes 01-01-00. Risks Forum readers know how the day will go. The first trouble reports will come from New Zealand, which Peter Neumann calls the "king's taster" for computer clock problems. Then the wave of failures will wash over Asia, Africa and Europe; those of us in the Americas will see it coming hours in advance, and yet we will probably be unable to do much of anything to stop it.

As computer bugs go, the problems connected with truncated and cyclic dates are not very subtle or obscure, and many of them can doubtless be fixed by simple changes. Just leaving room for four-digit years should hold us for another eight millennia. The challenge of averting computer catastrophe on 01-01-00 is not in the bugs themselves; the challenge is the "dusty-deck problem." The term comes from the era of punched cards—and so does some of the software still running today. That accounting system written in 1962 may still be at the heart of a bank's daily operations, though by now it is encrusted with thick layers of additions and patches, and no one has a clear idea of how it actually works. "Legacy systems," they call such software. Altering a legacy system in an area as fundamental as the format of dates is rather like changing a tire without stopping the car. It's a delicate operation, and now is not too soon to get started.

The prevalence of calendar-related malfunctions among the reports appearing in the Risks Forum suggests just how tricky it is to get date computations right. I can offer further evidence. In the first draft of this column I was off by 10 in my calculation of the number of days between 12-20-99 and 01-10-00. David Schoonmaker, the managing editor of *American Scientist*, caught the error and showed me, with arithmetic of convincing simplicity, how to derive the correct answer. I had done the original calculation with a spreadsheet program, which I considered more trustworthy than my own arithmetical skills. Part of the error turned out to be the result of a careless typing mistake, but after correcting that slip, a one-day discrepancy remained. The puzzle was solved when I discovered that the spreadsheet program treats 1900 as a leap year.

The last time the calendrical odometer turned over a row of three zeroes, it was a turbulent moment in social history, with much of Christendom nervously awaiting the end of the world *(11)*. The millennium was thought to be the time of apocalypse, the appointed Day of Judgment. I cannot suppress the idea that many were perplexed, perhaps even disappointed, when 1000 came and went, and the sun continued to rise and set so reliably. But there were no computers then.

Notes

1. The Risks Forum is distributed through the Usenet newsgroup comp.risks and is also available by electronic mail, either through the BITNET LISTSERV mechanism (send the message SUBSCRIBE RISKS) or by requesting a subscription from risks-request@csl.sri.com. An archive of the forum is available through the anonymous ftp protocol from crvax.sri.com. The archive can be searched by means of the WAIS protocol, using the risks-digest.src database on the WAIS server cmns-moon.think.com. Excerpts from the forum appear in *Software Engineering Notes*, the quarterly journal of the ACM Special Interest Group for Software Engineering, and in the *Communications of the ACM*. The *Communications* excerpts of January 1991 (Vol. 34, No. 1, p. 170) concern clocks and calendars. Much material from the forum, with additional analysis and commentary, is collected in a recent book: Peter G. Neumann, 1995, *Computer-Related Risks*, New York: The ACM Press and Reading, Mass.: Addison-Wesley Publishing Company. See pp. 85–92 for a discussion of clock and calendar problems.
2. See especially: Paul Robinson. The danger of six-digit dates. Risks Forum 16:32, August 15, 1994.
3. Steve Peterson. Re: Turn of the century date problems. Risks Forum 14:45, March 29, 1993.
4. Ed Ravin. Call for the class of '88. Risks Forum 14:44, March 29, 1993.
5. Lee F. Breisacher. Risk of aging. Risks Forum 4:9, November 10, 1986.
6. David B. Benson. Another 100-year computer saga. Risks Forum 9:73, March 6, 1990.
7. Joe Morris. Hospital problems due to software bug. Risks Forum 9:26, September 20, 1989. Will Martin. Re: Hospital problems due to software bug. Risks Forum 9:27, September 21, 1989. Steve VanDevender. Re: Hospital problems due to software bug. Risks Forum 9:28, September 24, 1989.
8. Brian Randell. Another foretaste of the millennium. Risks Forum 9:45, November 17, 1989. Also corrigendum November 21, 1989.
9. Marc Auslander. Not enough bytes bites again. Risks Forum 16:48, October 21, 1994. Dave Moore. Re: Not enough bytes bites again. Risks Forum 16:49, October 24, 1994.
10. Daniel J Yurman. Re: AT&T machines and dates. Risks Forum 13:05, January 21, 1992.
11. Henri Focillon. 1969. *The Year 1000*. New York: Frederick Ungar Publishing Co.