

Y2K: 10 years later

By Paul Krill - January 4, 2010

It's hard to believe that 10 years have passed since the dreaded Millennium Bug put fear into the hearts of technology specialists, software developers, business executives, and legal departments everywhere.

Fears of massive system failures abounded, including worries about errant missile launches thanks to computers confused about what century we were in. But the calendar flipped from Dec. 31, 1999, to Jan. 1, 2000, with the world relatively unscathed from the Y2K switchover.

Ironically, the clock ticking to Jan. 19, 2038, poses a similar threat to some systems today. Languages such as C, C++, and early Unix languages stored dates in an odd way beginning in 1970, says Drake Coker, chief technologist for application development at Cobol provider Micro Focus. "That overflows 32 bits" on Jan. 19, 2038, Coker says. Older software will experience the problem, he says, but "it won't be as big" of a problem as Y2K.

The Y2K countdown: Tech's big drama

If you worked in technology, you might remember that New Year's Day was different from others. Instead of watching college football games or recovering from the previous night's celebrations, you may have had to work or at least be on call to keep watch over potential Y2K-generated mishaps.

These days, however, the biggest reminder of Y2K comes perhaps from TV reruns of the cult classic film, "Office Space," which had depressed computer specialist Peter Gibbons, played by Ron Livingston, explaining the Y2K switch to waitress Joanna, played by Jennifer Aniston. That scene certainly dates the movie.

For anyone who needs an explanation after all these years, the Millennium Bug referred to computer systems that used two-digit dates because programmers at the dawn of computing did not think far enough ahead to put in four digits. So when the 1990s made way for the year 2000, the date, "99" would then become "00," with systems believing the world had just reverted to 1900 instead of advancing to the year 2000. *"The fundamental issue was the date ranges," says Josh Aaron, president of Business Technology Partners, a technology consulting firm.*

A now-retired technologist who worked for British Telecom 10 years ago recalls the extensive efforts to tend to Y2K. "It all got done, and it was a hell of a lot of work

involved, and of course once you fix it, you've got to test it," says David Quinn, who ran the systems software group at BT.

Y2K was fixed because people prepared for it, Quinn says. "I wrote some of those systems" for billing and order management, he says. "I know the dates were wrong."

A decade after Y2K, technologists reflected back to InfoWorld on that time and the lessons learned, with some disagreement over whether Y2K turned out to be basically a nonevent because millions of dollars were spent in a heroic effort in advance to fix the problem or because the problem was overblown in the first place.

Was the Y2K Millennium Bug fear overblown?

"I think people felt duped because the world was predicting a disaster," Quinn says. There were even predictions that cars would stop running because of engine clock problems, he says.

"My recollection is that probably 70 percent of that concern turned out to be unfounded - but you had to do the research anyway. You couldn't take a chance" in mission-critical environments like financial services and health care, Aaron says. He says he could not recall an actual Y2K problem that could not be fixed in five minutes. "My opinion is it was a quiet day because people put the proper focus on it, did the right amount of due diligence, and [did] the work that needed to be done."

Concurring that due diligence took care of the problem, Chip Ahlswede, who worked for the U.S. House of Representatives at the time checking on Y2K compliance, says it was better to be safe than unprepared. "I think it was the preparation thing," says Ahlswede, who worked as a subcommittee clerk. He now is principal at Regal Strategies, a political consulting firm. He remembers Y2K's mild impacts. "As far as the government, there were a few systems that had glitches after the fact, but obviously no missiles were launched," Ahlswede says.

"The high publicity it garnered ensured that everyone took care of it," recalls Micro Focus' Coker. Corporate managers were afraid of getting sued as a result of Y2K problems, he says.

Another IT official, who worked at Sun Microsystems at the time, disagrees sharply. "I don't think that IT should look back and say, 'Hey, Y2K was successful because nothing happened,'" says Bill Roth, who is now chief marketing officer at LogLogic, which provides security and event management. He had been a Java marketing manager at Sun Microsystems during Y2K. "It is unlikely that anything would have happened other than people having their birth dates stated incorrectly or checks being predated by 100 years."

Although he stresses the Y2K issue was overblown, he acknowledges there were legitimate reasons to be concerned: "The problem was about poorly written, date-based mainframe applications." Still, "the power grid stayed up, our trading system stayed up,

and while a small amount of it was due to the diligence of people cleaning up Cobol code, it's unlikely that anything would have happened in any event," Roth says.

But Aaron points out potential mishaps. "You had to look at every single system assuming it was going to break," he says. Potential existed for mishaps such as two parties sharing a financial trade and one party seeing the date wrong, voiding the trade, and creating a massive number of trade breaks among financial institutions. "You would have wound up with customers that think they made a trade because they called in an order but the order never executed," Aaron says.

As awareness grew of the Millennium Bug, it was unclear how pervasive it was. And that meant that programmers had to review lots and lots of code to see where the problem might exist. The effort to prevent a Y2K disaster swung into earnest about two years before Dec. 31, 1999, Aaron notes. The result most of the time was that software did not need any changes to accommodate Y2K, Aaron says. Systems either already were set for the 2000 switch or just needed a simple work-around.

Many systems, such as embedded systems and chips, did not fail from Y2K because they did not even run on Julian calendars, says J. Greg Hanson, executive vice president at technology services company Criterion Systems: "The clock on the computer chip is not based on calendar time." Y2K was mostly a problem associated with business software, says Hanson, who led U.S. Air Force's \$345 million Y2K program but retired from the USAF prior to the year 2000.

Y2K's legacy: Better disaster planning and documentation

Despite the debate over how serious the Y2K problem would have been had companies not invested so much time examining code for the Millennium Bug, it's clear that the IT industry did learn some lasting lessons, Aaron says. These include doing continuity and disaster planning and documenting systems, he says.

Another lesson, says LogLogic's Roth, was that systems last longer than we think and need to be ready for the future. "You always have to be future-proofing the software you write, the hardware you build," he says.

But IT organizations still have not learned as much from Y2K as they should have, says Gartner analyst Dale Vecchio, who covered Y2K at the time of the switch. "I'd like to tell you that there were a lot of lessons learned, but I'm not sure that I've seen a lot," he says.

For example, the Y2K experience might have caused organizations to keep current with knowledge of their IT portfolio, he says. Instead, "once they passed the risk of Y2K, they went back to the same lack of knowledge and now, when faced with an aging portfolio and aging workforce, they don't know any more now than they knew then," Vecchio says.

Many older applications are still running, but the people with the skills necessary to run them are inching toward retirement age, says Vecchio. "Baby Boomer retirements [are] bringing back a recognition that [IT shops] don't understand these application portfolios to move forward," he says. One reason that IT doesn't understand its application portfolio is due to one response to the Y2K issue: At the time, many businesses replaced homegrown software with vendors' Y2K-certified package software. "Y2K drove a lot of packaged software sales," Vecchio says. But much of that software is a black box for IT and is harder to maintain knowledge of.