Hal Berghel

How Xday Figures in the Y2K Countdown

Here we are, eight months out and counting. We've already seen glimpses of things to come. Stories like that of Lynn Electric, Bluefield, WV. When the company tried to close its 1998 payroll "all documents reverted to 1944." (*Washington Post*, 1/1/99.)

In Sweden, airport police were unable to issue one-year temporary passports after midnight, Dec. 31, 1998. The computers refused to accept a termination date beyond Dec. 31, 1999. (Source: Bruce Taylor.)

Palace Produce sued Tec America Inc. because their system was incapable of handling credit cards with an expiration date past 01/99. They won their case and were awarded $250,000 in damages. (Source: Peter de Jager.)

Newspapers, magazines, and the Web are replete with similar stories. And there is every expectation these sorts of incidents will continue for the immediate future. Examples might include the June 30, 1999 fiscal roll-overs for many organizations. This may provide a useful estimate on the number of active legacy applications that have so far escaped repair.

Also, consider accounting and spreadsheet software that seem to take on a mind of their own when it comes to manual and automated attempts to reset to correct dates.

QUENTIN WEBB

Some system-sensitive processes will be reset to 1900 at century's end, resulting in system failure and data loss (exactly how many of these will occur is subject to wide speculation).

Some post hoc BIOS' repairs will conflict with software fixes, resulting in new and as yet unimagined categories of software and data conflicts. Perhaps OS vendors should put a new prompt in their toolbox akin to Microsoft's "General Protection Fault," something like "Generic Y2K Fault: Reboot without prejudice."

Small, local telephone companies may charge for 99-year calls those which span midnight next New Year's Eve.

An occasional blackout could result if a noncompliant computer in the grid get's confused about the date.

Valid credit cards may be rejected as out of date.

Small bank ATMs may only work with a fraction of the supported debit and credit cards.

Air Traffic Control problems may arise in the smaller, local airports spawning missed connections for the passengers and load-scheduling problems for airport hubs.

In some smaller corporations, security IDs may fail to work for awhile, necessitating doors left ajar, and a temporary, but large, lapse in security. It's not inconceivable that time-locks on small bank safes may fail to operate correctly.

Of course, the gloom-and-doom forecasters see things in a very different light.

The Essence of Y2K

As we examined in an earlier column (*Communications*, Mar., 1998), the generic problem underlying the Y2K bug is actu-

The Y2K problem is actually a hydra-headed monster that rears its ugly head in multifarious ways.

ally an inductive, logical fallacy. This new riddle of inductive reasoning (aka, Goodman's Paradox) derives from the fact that we cannot always distinguish law-like regularities from contingent or accidental ones. We blithely assumed that an operating system date stamp satisfied the predicate "___ is the current date on the Gregorian calendar" when, in fact, the predicate was "___ is the current date on the Gregorian calendar only before time, *t*; and not thereafter." The accuracy up to this point in computing history has been contingent and not law-like. In the normal course of events, this would be confirmed at *t* = midnight, Dec. 31, 1999. Literally billions of dollars are being spent to avoid this result. In the end, it is the same sort of short-sightedness behind the five-digit automobile odometer. It was inconceivable in 1920 that any car would last 100,000 miles, just as it was inconceivable in 1959 that COBOL legacy applications would still be in use in 1999.

The Y2K problem is actually a hydra-headed monster that rears its ugly head in multifarious ways. For instance, even if a patch handles the Y2K transition, it may not handle the leap year. Our astronomical year is 365.242 days on the Gregorian calendar. Falling short of 365.25 by .008 days/year causes big problems over time, so three-fourths of the century years are common (that is, non-leap), while those evenly divisible by 400 as, in the year 2000, are leap. So, any enduring Y2K fix has to accommodate the leap-year status in 2000 denied its century predecessor, 1900. In contrast, every year evenly divisible by 4000 is a common year despite the fact it's also evenly divisible by 400. Just think, our descendants may teleport themselves around the universe in search of quick fixes for the Y4K bug.

Moreover, even if the Y2K problem disappears, we will still have to deal with the sister problem of calculating the "elapsed time in Unix." A manual might say that Unix function *x* produces today's date in Register 1. In fact, it won't report the date at all, rather it will *calculate* the date on the basis of the number of seconds which have expired since January 1, 1970, until count, *t* = 2**31 (assuming the sign-bit is retained, this is Jan. 18, 2038). Just as with Wintel machines, we naively designed our computers for projectable predicates, but actually implemented nonprojectable ones in hardware.

There may be several layers of Y2K problems. In an earlier column, I used DOS as an example. Not only does DOS not roll-over automatically to the 21st century, it only recognizes dates reported by the real-time clock that fall within the interval 1980–2099. We observe that this only appears ad hoc and arbitrary until one traces back the origin of the PC. This gets even more convoluted when one looks at FILE_DATE values. In this case, the reported date is a compression of the DOS reported date according to the formula ((year-1980) x 512)+(month x 32)+day. Hence, Dec. 31, 1999 becomes the unsigned word integer 279Fh (10,143). It is interesting to note the roll-over for the file date would actually be 1980+2**7 years=2108, but since DOS will only recognize the 19th and 20th centuries, the file date will be undefined under DOS beyond 2099. As I observed before, the inductive fallacy which underlies Y2K is to be found in our hardware and the inner-most recesses of our BIOS.

Still another example involves the lack of orthodoxy when it comes to four-byte REPORT_DATE standards (ISO, Microsoft, European). There are several in use, all incompatible with one another, which will continue to present headaches long after the aftertaste of Y2K has past.

Y2K Solution Strategies

Of course, we've seen as many potential solutions as there are potential pitfalls. The following categorization of Y2K fixes is adapted from Capers Jones article "Finding Time for the Year 2000 Repairs" (www.spr.com).

1. Replacement of noncompliant applications with compliant ones. Most appropriate with popular, commercial software such as productivity applications and utilities. *Advantage:* Simplicity and

economy (assuming that everything goes according to plan). *Disadvantage:* Applies mostly to popular, commercial software.
2. Repair of noncompliant applications. Most appropriate with proprietary and legacy applications. *Advantage:* Infinitely customizable. *Disadvantage:* Probably not enough time left to handle large systems.
3. Termination of noncompliant programs on an as-needed basis. *Advantage:* A real managerial no-brainer, and it may be the default strategy after midnight next New Years eve. *Disadvantage:* It will require that automated processes be done manually or partially during the repair cycle, which will be very expensive.
4. Masking the data exchange between applications rather than try to fix the applications themselves. Masking consists of several forms, including *pivoting*—a pivot year is the internally defined boundary that software uses to distinguish between the 19th and 20th century. For example, if the pivot year were chosen to be 29, a program could interpret any two-digit date within the range 29–99 as 19xx; else, 20xx. Other fixes have set pivot years at 30, 40, 48, 51 "current + 20,"current + 70," 1980, and so forth. Of course, these application-specific fixes have to cooperate with each other. The test case will be the import and export of data between applications with different pivot dates which should produce some excitement.

Masking may also involve conversion of the two-byte date field into some other form. For example, 16-bit binary enumeration beginning at year 0 on the Gregorian calendar would carry us through for another 60,000+ years. The problem, of course, is that everyone has to agree to the convention, and it's not likely this agreement will surface in the next eight months.

Gregorian date	Julian date	
January 1, 0001	1721475	
1800	Operational range 1999-3501	2378497
1900		2415021
May 16, 1998		2450950
January 1, 2000 (xday)		2451545
2100		2488070
2400		2597642
August, 3501	3000000	implied

Figure 1. Gregorian vs. Julian dating

Another masking form is the encapsulation of date exchange information between application and data. To illustrate, the shift of a date downward by a multiple of 28 years mirrors the day-date correspondences of the current calendar. So, an application might retrieve the date of 03 from a file, reduce it to 75 for internal processing, and then convert it back by adding 28 back before storing, thereby avoiding any Y2K collision (hopefully).

Creating redundancy across a databases so that applications will only run on them if their date fields are the same size is another masking technique. An obvious downside is that this technique consumes twice the space until everything converges on the four-digit date field, but this pales in comparison with the enormous difficulty of keeping the databases synchronized.

Also, object-code interception which is predicated on the assumption the internal representations of dates is irrelevant if we can intercept, change, and maintain them as they are shared across applications and systems. One of the most innovative ideas for object-code interception is to be found in Bob Bemer's clever Xday (eXchange day) proposal.

Bob Bemer's Xday

Bob Bemer is a genuine computer pioneer. Among his many advertised accomplishments are the invention of the ESCape sequence, the creation of ASCII, coining the term "Cobol," and inventing its "Picture Clause." Since 1949 he's held prominent positions with IBM, Honeywell, and General Electric, to name but a few. Having been bitten by Y2K fever, Bemer came out of retirement to solve the problem. While his product, Vertex 2000, uses an object-code interception strategy, it's independent of Bemer's Xday strategy.

Bemer was early to recognize that a Y2K fix can take one of two strategies: internal (for example, converting two-digit date fields to four-digit fields) or external (for

YYYYMMDD	Xday	YYMMDD	MMDDYY	DDMMYY
18581116	400000	581115	111558	151158
19000101	415021	000101	010100	010100
19431003	431001	431003	100343	031043
19990101	451180	990101	010199	010199
19991231	451544	991231	123199	311299
20000101	451545	000101	010100	010100
20230224	460000	230224	022423	240223
21320831	500000	320931	083132	310832
35010814	999999	010814	081418	140801

Figure 2. Conflict-free Xdating

example, object-code interception). As it turns out, the former solution is more difficult and permanent, while the latter may be easily implemented but ephemeral. However, a quick fix with just a century's breathing room would be of enormous value at this late date.

That's where Xday comes in. Bemer observed that if we substitute for our Gregorian solar calendar a variant of its Julian parent, we have a unique way of representing each day into which every conceivable date format may be mapped. 1 A.D. is Julian 1721475; 1900 A.D. is 2415021; 2000 A.D. is 2451545; 2400 A.D. is Julian 2597642. Coincidentally, the most significant digit in the Julian date, Bemer observes, stays unchanged for 27 centuries, with 15 centuries remaining (Figure 1). Assuming that Y2K will no longer be a problem in 3500, we imply the first digit without reservation.

Now the magic comes in. Bemer points out that the next two digits of Xday (Julian date sans leading digit) fall within the range of 45 to 99 until August 14, 3501. This has enormous consequences for his object-code interception strategy because these two digits cannot be confused with existing day, month, or year values, regardless of the cultural format or order in which the fields appear (Figure 2). A collision could have occurred when 43-10-03 would have corresponded with Xday of 431001, but that's not much of a chink in the armor as it predates the computer era.

As if that isn't clever enough, it also turns out that Xday occupies the same six-digit space used for existing date formats, so it can be used interchangeably. If one has the time and energy, the Xday value could actually be used to replace the problematic date fields (ala the replacement strategy discussed earlier). On the other hand, it will also serve well (and is really intended) for object-code interception since it can't conflict with other date fields in data streams. Of course, the interchange requires some arithmetic (related algorithms may be found at www.software.ibm.com/year2000/tips15.html. We note that Xday = LillianDay + 299160), and there has yet to be a reliable estimate of overhead, and there are sundry implementation issues to be resolved, but none loom large enough to detract from Bemer's brilliant idea. Unfortunately for all of us, it's an idea whose time has not yet come—and time is running out.

I would be remiss if I failed to also mention that Xday methodology is a public-domain resource which involves no proprietary or patented ideas or concepts. As such, it is to be distinguished from the "Y2K Silver Bullet Patent" (#5,852,824) which was issued to Roger Brown by the U.S. Patent and Trademark Office as this column goes to press which involves proprietary encapsulation routines. An even earlier patent (#5,600,836) was awarded to TOCS (Turn of the Century Solution) in Nov. 1995 for another encapsulation routine.

Where Things Stand Today

As the sidebar illustrates, there is no shortage of information on Y2K on the Web, ranging from alarmist documents ravings to corporate repair strategies. For most of us who fall into the category of millenia moderates, Y2K is expected to be a major inconvenience which some authors liken to a granddaddy of all brownouts, a flood of the century, and an ice storm to end all ice storms. Such being the case, we'll have a considerable mess on our hands. But life as we know it will go on, and Y2K, too, will pass. Should that not happen, I herewith reserve the right to withdraw this statement and to deny under oath that I ever made it. C

Hal Berghel (www.acm.org/~hlb) is a professor of computer science at the University of Arkansas and a frequent contributor to the literature on cyberspace.

URL PEARLS

Our Web Ferret metalevel search engine found 4,000 URLs matching the query "year 2000" or "Y2K" before my connection timed out. Here are some sites I found interesting.

For general information

- *The Year 2000*; www.year2000.com. Peter de Jager's extensive Web site on the Y2K problem and an extensive listing of Y2K vendors.
- *ComputerWorld's* Y2K info site; www.computerworld. com/res/year_2000.html. Postings of related CW articles.
- *PC Magazine's* Y2K site; www.pcmag.com/Y2K. Postings of the magazine's Y2K articles.
- *Software Productivity Research's* Web site; www.spr.com. Software engineer and company founder Capers Jones' extensive writings on the Y2K problem.
- *The Year 2000 Problem and the New Riddle of Induction*; www.acm.org/~hlb/col-edit/digital_village/mar-98/dv_3-98.html. Analysis of the origin of the Y2K problem which appeared in the Mar. 1998 *Communication*.
- *Y2K News Magazine*; www.Y2Knews.com. Full-featured Web site on Y2K.
- *Information Technology Association of America*; www.itaa.org/year2000/. Extensive information for MIS managers.

Vendors of Y2K software

- *Commtec Systems 2025*; www.doitnow.com/~commtec/. Commercial vendor of Roger Brown's patented Y2K Silver Bullet software, QIPP.
- *BigiSoft*; www.bigisoft.com/. Vendor of Bob Bemer's Vertex 2000 software for IBM mainframes.
- *TOCS*; www.tocs.com. Vendor uses program encapsulation that takes advantage of the fact the calendar repeats every 28 years. By setting the clock back multiples of 28 years, the days, weeks and leap-year status remain unchanged.
- *Logica*; www.logica.com/year2000/. Vendor offers both data and program encapsulation.

Computing vendors

- *IBM's Y2K site*; www.software.ibm.com/year2000/index.html. IBM's spin on Y2K problems and solutions.
- *Microsoft Year 2000 Resource Center*; www.microsoft.com/technet/year2k/default.htm. Ditto for Microsoft.
- *Corel*; www.corel.com/2000/index.htm
- *Dell*; www.dell.com/year2000/.
- *Gateway*; www.gateway.com/.
- *Compaq*; www.compaq.com/year2000/.
- *Apple*; www.apple.com/about/ year2000/. Apple claims that "... since their introduction in 1984, Macintosh computers *have had the ability to make the transition* to the year 2000. In fact, the Mac OS and most Mac applications can handle internally generated dates correctly all the way to the year 29,940." (Italics added. However, it isn't clear whether "having the ability to make the "Y2K transition" and "being Y2K compliant" are equivalent expressions.)

Sundry Y2K compliance testing and repair utilities

- *RighTime Clock*; www.rightime.com. Diagnostic and Terminate and Stay Resident fix (TSR).
- *McAfee 2000 Toolbox*; www.nai.com/Y2K/. Diagnostic with both TSR and continuous BIOS-checking.
- *WRQ's Express 2000*; www.wrq.com. Data inventory and monitoring of applications.
- *AMS Group's DateSpy*; www.datespy.com. Date-sensitive searching in selected applications.

Miscellaneous

- *Gary North's Y2K Links and Forums*; www.garynorth.com. State-of-the-art alarmism.
- *Duh2000*; www.duh-2000.com. Stupidest things said about Y2K. Humor mixed with attitude.
- *Art Bell's Web site*; www.artbell.com. Considerable Y2K information archived by the popular late-night radio talk-show host.