Y2K Excuses

By Bob Bemer December 23, 1999

Six basic excuses are customarily given for not being ready to handle year values starting with "20" in computers:

- No room on punch cards
- Computer memory was too expensive
- Too inefficient and costly for extra input keystrokes
- Never thought the programs would last that long
- Had to be consistent with everyone else
- Computer manufacturers forced upward compatibility

All of these are specious, self-serving, untrue, and cover-ups by the thousands of people who didn't think correctly and acted badly. They now make excuses, not valid efforts, as in:

"An excuse is usually a very thin skin of falsehood stretched tightly over a barefaced lie."

No Room on Punch Cards

- Punch cards are actually an expandable medium, and can carry all the data one needs. Think of them as placed end-to-end in a long sequence just like perforated paper tape. Then think of them as pages in tax returns, each carrying the same key to tie them together (your Social Security number), and a sequence number for its place in the series. Need more room for data? Add another form, another card type, another sequence number. For all data, collate them together. All punch card users did that.

- U.S. Censuses used punch cards from 1890 on. By the early 1900s they carried 45 columns (or characters) [1]. By 1924 Hollerith became IBM. By 1930 new rectangular holes permitted 80 columns, even as today. Each census included people born before and after 1900, handling the equivalent of 4-digit years by recording a 2-digit age, which, when subtracted from the census year, gave the birth year. No confusion was possible; the cards for each census were never intermingled.

- But assume a real need to use just two columns for a year, independent of the year associated with the card. Is that possible? Yes, indeed.

With what I learned in 1949 at the RAND Corporation, I could wire an old plugboard today, using what were then called "emitters" and "selectors," to have a plain 5 in the decade column read in as "195," a minus 5 (an overpunched 11 meant "-") to read in as "185," and a plus 5 (an overpunched 12 meant "+") to read in as "205."

I could also show you how to print those 4-digit years from those cards, and how to sort them on those two columns so they were in order on 4-digit years!

- When stored programs did come into use, there were other ways that 4-digit years could have been expressed in just 2 positions. Don't forget that two digits can express 100 months, and we have only 12. 100 days, and we have only 31. Mathematicians call these "sparse sets." Suppose we programmed our computer to test the month value (we have to do it anyway, to make sure that only 01 to 12 were punched). We could have allowed 51 to 62 to also be legal values, but subtract 50 and make the 2-digit year begin with "20" rather than "19." Would that have caused more work, historically? Not a single thing would have changed until Y2K became a problem.

Computer Memory Was too Expensive

- Prof. Leon Kappelman debunked this in "Time To Debunk Y2K Myths" [See Footnote 2]. His model calculated cumulative savings using 2- and not 4-digit years through 1995. Taking his 1963 figures as 100% per either gigabyte of total storage or per-application costs, by 1972 these figures went to 5% and 7%. By 1983 they were 0.08% and 0.19%.

  With the huge increase in computer applications, odds are that later practice of this technique was penny-wise and pound-foolish.

- Dr. Fred Brooks, IBM's project manager for the IBM 360, says that "the cost of using four-digit years went down gradually, and the wisdom of using them went up gradually." He says the two lines crossed in "around 1970" [3].

- By 1980 storage was so cheap that I advocated storing the history and characteristics of every data item together with the item itself [4]!

- A modified Julian Day could have been used to mark time, rather than day/month/year or year/day-of-year. With this, 6 digits could handle 27 centuries, not just 1! No net extra programming or run time would have been needed.

Too Inefficient and Costly for Extra Keystrokes

"Windowing" is a technique used pervasively to try to fix the Y2K Problem. One chooses a pivot point for the year value, say 60. If the year value is 60 or greater, it is prefixed by a "19." Else a "20." But

this must be done the hard way, by changing one's existing programs (which is where all the difficulties come in), because the years are already stored as only 2 digits.

Let us now go back to 1972 when, Prof. Kappelman argues, it was cheap enough to store all 4 digits of the year. To 1972 when the ISO and ANSI standards for dates had been published. When the argument was the expense of the extra keystrokes to put in a "19" every time, when everyone knew that the century stayed the same.

Then everyone could have written their COBOL programs to say:

BIRTH_YEAR PIC 9999

instead of

BIRTH_YEAR PIC 99

Great! All of their programs will work as always in the year 2021! No Y2K Problem. But management still doesn't want to waste time and effort on typing in "19" for every year value. How can that be fixed? Very easily.

Remember that (as above) all data entry must be checked by validation programs, which users write, so that only legal and permissible values will pass into the computer.

Suppose people had added, to the (probably single) routine that validated year values, a program to prefix a "19." Thus accepting a year, entered via 2 digits, as a 4-digit value into the database (no longer so expensive).

Now, when the reality of the 21st century hits them, they simply change that one routine a little. Instead of the automatic "19" they detour into a little windowing code, which adds EITHER 19 or 20 according to the rule. Simple!

The Y2K problem would have never have existed, because these programs handled 4-digit years all along!

Inputters today would be cheerily typing "98," which would go into the database as "1998," and "02," which would go into the database as "2002." We're still going to be doing that anyway, because we'd still like to minimize the keystrokes and the area on the screen where the year is entered.

Why wait until now to use that ever so popular windowing method? When it could have been applied permanently way back in 1970? Excuse demolished!

Never Thought the Programs Would Last That Long

Ed Yourdon once said [5] that "We had the impression that not only would the hardware be thrown out in seven or eight years, the software would be thrown out, too."

Not an excuse. Even if the software is thrown out what happens to

all the data that was input and created during that time? Data persists. It's historical, and useful for continuity of our lives. How about those Egyptian stone tablets found to record sales of grain over two thousand years ago?

No, during those years of early computer use a mountain of century-disadvantaged (as our bureaucrats would specify it) date data was entered and created. How can century values be truly recovered for such data? Invalid excuse.

Had to Be Consistent with Everybody Else

The only reason to do the same as everyone else is if you're going to exchange data containing year values with someone else. That's a good reason to be standard. But guess what the latest Y2K discovery is? There ISN'T any standard for representing dates in computers that is universally agreed and used! Merrill Lynch is setting up "firewalls" and says it will refuse to do computerized business with any entity not representing dates their way. Alan Greenspan admits this is a big danger. Joel Willemssen, U.S. Government Accounting Office [6], uncovered over 500,000 different electronic data interchanges in just the Government's mission-critical systems. And not all of these use anywhere near the same date form.

Computer Manufacturers Forced Upward Compatibility

Some point to the IBM 360/370/390 progression as evidence, saying manufacturers had to be certain that all of a customer's old programs would still work. In a sense this is true, for in early days (with computers much harder to program) emulation programs were made for the new machine to simulate the way the old machine worked, to not lose production time to reprogramming.

But this fact was a major impetus to create FORTRAN and COBOL. FORTRAN was out in 1957, and COBOL in 1960. COBOL had the Picture Clause for data description, allowing the compiler to adjust automatically for change. Write "PIC 9999" as a characteristic of a year, not "PIC 99," and you were home free.

Such languages were designed to be machine-independent. You could run IBM programs on a UNIVAC machine, which is much more of a change than successive IBM computers were. End of argument.

REFERENCES:

1. http://www.maxmon.com/punch1.htm
2. http://www.techweb.com/se/directlink.cgi?IWK19980928S0075 "Time To Debunk Y2K Myths", 1998 September 28, L. Kappelman
3. Washington Post, Style Section, Page 1, pp. F1, F4, F5, 1999 July 18. "The Millennium Bug", by Gene Weingarten.
4. R.W.Bemer, "Incorrect data and social harm", DATA Magazine, Copenhagen, 10, No. 9, 43-46, 1980 Sep, 43-46.
5. D. Hayes, F. Buller, "Chances to Fix Problem Frittered for Decades", Kansas City Star, 1998 Sep 27, p. A-18 lead.
6. http://www.house.gov/reform/gmit/hearings/testimony/980903jw.htm