Subject: Y2k of 2038
Posted by Jaspah on Sun, 16 Apr 2006 02:07:21 GMT
View Forum Message <> Reply to Message

It's an interesting read. Someone with Yahoo Messenger should try this.

The bowls of ze IntarwebThis news has been published in one of the daily news paper...read on

Year 2038..Again the replica of Y2K
Note: This is just for FYI only, Please Don't try this. This is true and if you do this then your network based applications will not work.

The Year 2038 Problem
Triaging steps...

1. login to yahoo messenger
2. send instant message to anyone - fine its working...
3. now, change your system date to 19-Jan-2038, 03:14:07 AM or above
4. Confirm weather your date is changed
5. again send instant message to anyone...

Your YM crashes...

* * * YES ALL NETWORK BASED APPLICATION WILL NOT WORK NOW * * *

*Why...
What is it?*

Starting at GMT 03:14:07, Tuesday, January 19, 2038, It is expected to see lots of systems around the world breaking magnificently: satellites falling out of orbit, massive power outages (like the 2003 North American black out), hospital life support system failures, phone system interruptions, banking errors, etc. One second after this critical second, many of these systems will have wildly inaccurate date settings, producing all kinds of unpredictable

consequences. In short, many of the dire predictions for the year 2000 are much more likely to actually occur in the year 2038! Consider the year 2000 just a dry run. In case you think we can sit on this issue for another 30 years before addressing it, consider that reports of temporal echoes of the 2038 problem are already starting to appear in future date calculations for mortgages and vital statistics!

In the first month of the year 2038 C.E. many computers will encounter a date-related bug in their operating systems and/or in the applications they run. This can result in incorrect and wildly inaccurate dates being reported by the operating system and/or applications. The effect of this bug is hard to predict, because many applications are not prepared for the resulting "skip" in reported time anywhere from 1901 to a "broken record" repeat of the reported time at the second the bug occurs. Also, may make some small adjustment to the actual time the bug expresses itself. This bug to cause serious problems on many platforms, especially Unix and Unix-like platforms, because these systems will "run out of time".

What causes it?

Time_t is a data type used by C and C++ programs to represent dates and times internally. (Windows programmers out there might also recognize it as the basis for the CTime and CTimeSpan classes in MFC.) time_t is actually just an integer, a whole number, that counts the number of seconds since January 1, 1970 at 12:00 AM Greenwich Mean Time. A time_t value of 0 would be 12:00:00 AM (exactly midnight) 1-Jan-1970, a time_t value of 1 would be 12:00:01 AM

(one second after midnig ht) 1-Jan-1970, etc.. some example times and their exact time_t representations:

Date & time time_t representation

1-Jan-1970, 12:00:00 AM GMT 0
1-Jan-1970, 12:01:00 AM GMT 60
1-Jan-1970, 01:00:00 AM GMT 3 600
2-Jan-1970, 12:00:00 AM GMT 86 400
1-Jan-1971, 12:00:00 AM GMT 31 536 000

1-Jan-1972, 12:00:00 AM GMT 63 072 000
1-Jan-2038, 12:00:00 AM GMT 2 145 916 800
19-Jan-2038, 03:14:07 AM GMT 2 147 483 647

By the year 2038, the time_t representation for the current time will be over 2 140 000 000. And that's the problem. A modern 32-bit computer stores a "signed integer" data type, such as time_t, in 32 bits. The first of these bits is used for the positive/negative sign of the integer, while the remaining 31 bits are used to store the number itself.

The highest number these 31 data bits can store works out to exactly 2 147 483 647. A time_t value of this exact number, 2 147 483 647, represents Janu ary 19, 2038, at 7 seconds past 3:14 AM Greenwich Mean Time. So, at 3:14:07 AM GMT on that fateful day, every time_t used in a 32-bit C or C++ program will reach its upper limit.

One second later, on 19-January-2038 at 3:14:08 AM GMT, disaster strikes. When a signed integer reaches its maximum value and then gets incremented, it wraps around to its lowest possible negative value. This means a 32-bit signed integer, such as a time_t, set to its maximum value of 2 147 483 647 and then incremented by 1, will become -2 147 483 648. Note that "-" sign at the beginning of this large number. A time_t value of -2 147 483 648 would represent December 13, 1901 at 8:45:52 PM GMT.

So, if all goes normally, 19-January-2038 will suddenly become 13-December-1901 in every time_t across the globe, and every date calculation based on this figure will go haywire. And it gets worse. Most of the support functions that use the time_t data type cannot handle negative time_t values at all. They simply fail and return an error code.

I love all these OMGWTF ARMAGEDDON!?! theories people come up with.