Year 2038 Problem: Y2K38

S.N.Rawoof Ahamed, V.Saran Raj.

Department of Information Technology,

Dhanalakshmi College of Engineering,

Tambaram,Manimangalam, Chennai

Abstract **Our world has been facing many problems but few seemed to be more dangerous. The most famous bug was Y2K. Then Y2K10 somehow, these two were resolved. Now we are going to face Y2K38 bug. This bug will affect most embedded systems and other systems also which use signed 32 bit format for representing the date and time. From 1,january,1970 the number of seconds represented as signed 32 bit format.Y2K38 problem occurs on 19,january,2038 at 03:14:07 UTC (Universal Coordinated Time).After this time all bits will be starts from first i.e. the date will change again to 1,january,1970. There are no proper solutions for this problem.**

Index Terms-signed 32 bit integer, time_t, Y2K, Y2K38.

I. INTRODUCTION

In the year2038,there may be a problem which causes all software and systems that both store system time as a signed 32-bit integer, and interpret this number as the number of seconds since 00:00:00 UTC on Thursday, 1 January 1970. The time is represented in this way 03:14:07 UTC.On Tuesday, 19 January 2038 (2147483647 seconds after January 1st, 1970) the times beyond this moment will "wrap around" and be stored internally as a negative number, which these systems will interpret as a date in December 13, 1901 rather than January 19, 2038. This is caused by overflow. Programs that work with future dates will begin to run into problems much sooner. For example, a program that works with dates 23 years in the future will have to be fixed no later than 2015.

Because most 32-bit Unix-like systems store and manipulate time in this format, it is usually called Unix time, and so the year 2038 problem is often referred to as the Unix Millennium Bug. . But this does not mean that other operating systems will not be affected from this bug, they will be as much vulnerable to this problem if they are using such a time format.

As we know Y2K bug was the result of practice of abbreviating a four digit year to two digits. The practice of representing years in two digits creates a problem when one rolls over from x99 to x00.

Back in 2000, Y2K makes the computer failures due to the remedial work done or because of the reticence of the organizations to report problems. Another bug which is threatening to cause great damage is the Y2K38 bug.

The Y2K38 problems arise because of the similar reason which was responsible for the Y2K problems. As the Y2K problems resulted from not allocating enough bits for the representation of year, the Y2K38 problems are a result of not allocating enough bits for the representation of internal time. Most programs which are written in the C programming language will suffer from this problem as C programs use a library of routines called the standard time library (time.h) and this library uses a standard 4-byte format for the storage of time values and hence, the maximum value that can be represented using the 4-byte format is 2,147,483,647 which is equal to 19 January, 2038 at 03:14:07 UTC. And C is the mostly used programming language in the embedded software,

thus, making the Y2K38 bug even more dangerous as embedded systems. In C++ also it leads to the same problem. On 19, January, 2038 03:14:07, all these systems will stop working properly and the result will be massive power outages, hospital life support system failures, bank problems, satellites falling out of orbit.

II.AN EARLY PROBLEMS

The Year 2000 problem (Y2K problem) was a problem which resulted from the practice of abbreviating a four-digit year to two digits. It identifies two problems, Firstly, the practice of representing the year with two digits becomes problematic with logical errors arising upon rollover from x99 to x00. This has caused some date-related problems after 1 January 2000, since the year was represented in two digits like "... 97, 98, 99, 00 ..." ascending numbering assumption suddenly became invalid. Secondly, some programmers had misunderstood the rule that determines whether years that are exactly divisible by 100 are not leap years, and assumed the year 2000 would not be a leap year.

Companies and organizations worldwide checked, fixed, and upgraded their computer systems. The number of computer failures that occurred when the clocks rolled over into 2000 in spite of remedial work is not known.

Fig.1 The (French) sign reads “3 January 1900” instead of “3 January 2000”

Seconds, according to this, the latest time that it can store is 03:14:07 UTC, Tuesday, January 19, 2038. After this time, the sign bit of the 32-bit signed integer will be set and it will represent a negative number. As I said, the time is stored as number of seconds elapsed since 1st January 1970, this negative number will be added to compute the time as per the POSIX standards. But this being a negative number it will calculate the time by subtracting this many seconds from 1st January 1970 which will eventually generate a historical date-time which will cause the applications to fail. This time will be Friday, December 1901 and is called the wrap-around date. Applications written in C in many operating system will also be affected as the POSIX presentation of time is widely used there. The animation below visualizes actual scenario in an easier manner. This bug is often denoted as "Y2038", "Y2K38", or "Y2.038K" bug.

The effects of this bug is hard to predict. Some experts are of the opinion that this bug can cause serious problems in many platforms, especially Unix and Unix-like platforms, because these systems will “run out of 03:14:07, Tuesday, January 19, 2038, one can expect many systems around the world to break down, satellites to fall out of orbit, massive power outages (like the 2003 North American blackout), failures of life support systems, interruptions of phone systems, banking errors, etc. The maximum value of time before it rolls over to a negative (and invalid) value is 2,147,483,647, which translates into January 19, 2038 as shown in Fig 2.

```
Binary  : 10000000 00000000 00000000 00000000
Decimal : -2147483648
Date    : 1901-12-13 20:45:52 (UTC)
Date    : 2038-01-19 03:14:08 (UTC)
```

Fig.2 Example showing how the date would reset, represented as a signed 32bit integer (at 03:14:08 UTC on 19 January 2038).

Time_t is actually an integer which is counting the number of seconds that have passed since January 1, 1970 at 00:00:00 UTC. That is, a time_t value of 0 would give 00:00:00(midnight) 1, January, 1970, a time_t value which is equal to 1 would give 00:00:01(just a second after midnight) 1 January, 1970 and so on. Also one year has around 31,536,000 seconds, thus, a time_t value of 31,536,000 would return 1, January, 1971 and a value of 63,072,000 would represent 1, January, 1972.
The precise date of this occurrence is Tue Jan 19 03:14:07 2038. At this time, a machine prone to this bug will show the time Fri Dec 13 20:45:52 1901, hence it is possible that the media will call this The Friday 13th Bug.

```
#include <stdio.h> #include <time.h> int main ()

{

struct tm *tm_ptr; time_t i; i=2147483641;

for(;i<2147483651;i++){ gmtime_r(&i,tm_ptr);

printf("Date: %04d-%02d-%02d ",tm_ptr->tm_year+1900, tm_ptr->tm_mon+1, tm_ptr->tm_mday);

printf("Time: %02d:%02d:%02d\n", tm_ptr->tm_hour, tm_ptr->tm_min, tm_ptr->tm_sec);

}

exit (0);

}
```

Output:

Tue	Jan	19	03:14:01	2038
Tue	Jan	19	03:14:02	2038
Tue	Jan	19	03:14:03	2038
Tue	Jan	19	03:14:04	2038
Tue	Jan	19	03:14:05	2038
Tue	Jan	19	03:14:06	2038
Tue	Jan	19	03:14:07	2038
Fri	Dec	13	20:45:52	1901
Fri	Dec	13	20:45:52	1901
Fri	Dec	13	20:45:52	1901

Output of the program shows the date and time changes after 03:14:07 UTC, Tuesday, January 19, 2038.

The above program shows what will happen in 2038. All the time related functions, programs will give

erroneous result. All the variable quantities are studied and measured with respect to time as it is considered to be the independent variable in most computer applications so it becomes important for this time_t variable to return correct and exact value.

Some times and their time_t representation are,

Table 1. Exact time_t representations of Date and Time in signed 32-bits data type.

Date & Time	Representation
1-Jan-1970,12:00:00 AM GMT	0
1-Jan-1970,12:00:01 AM GMT	1
1-Jan-1970,12:01:00 AM GMT	60
1-Jan-1970,01:00:00 AM GMT	3600
2-Jan-1970,12:00:00 AM GMT	86400
3-Jan-1970,12:00:00 AM GMT	172800
1-Feb -1970,12:00:00 AM GMT	2678400
1-Jan-1971,12:00:00 AM GMT	31536000
1-Jan-2038,12:00:00 AM GMT	2145916800
19-Jan-2038 03:14:07 AM GMT	2147483647

IV. Y2K38 Solutions

1. Change the definition of time_t- This would result in code compatibility problems as there are certain applications which are dependent on the signed 32 bit representation of time. For example if time_t is changed to an unsigned 32 bit integer then, the applications that retrieve, store or manipulate the dates prior to 1970 will not be able to do so correctly. Though doing so will delay this problem to 2106.

2. Shift from 32 bit systems to 64 bit systems- Shifting from 32 bit systems to 64 bit systems gives us a new wrap around date which is over 20 times greater than the estimated age of universe i.e. about 292 billion years from now.

3. NetBSD's-Startingsolutionwithits6thmajor release, NetBSD is using a 64 bit time_t for both 32 bit and 64 bit architectures which supports 32 bit time_t in applications

4. Possible solution for Linux that can be implemented.

5. Redefine the time_t structure as 64-bit, This is not a solution as the binary compatibility of the software would break here. Programmes depending on the binary representation of time would be in trouble. So we cannot even think of this one.

6. Change time_t from 32-bit signed to 32-bit unsigned, This seems to be good at first look, but this would just delay(post-pone) the judgement day to the year 2106 as it will give some more scope by adding another usable bit. You will be in the same trouble by then. So this is a feasible solution but not a practical one.

7. Shift from 32-bit systems to 64-bit systems, Most 64-bit architectures use 64 bit storage to represent time_t. The new wrap-around date with this new (signed) 64 bit representation will not come before 290 billion years. It is positively predicted that by the year 2038 all 32-bit systems will be phased out and all systems will be 64-bit

CONCLUSIONS

The Y2K problem kept us engrossed and worried in the year 1999. However, the Y2K38 problem is not as obvious. This problem arises out of programs not allocating enough bits to internal time representation, which results from computing dates moving into the year 2038 and beyond in 32-bit operating systems. Even today, any date calculations forecasted beyond Tuesday, Jan 19, 03:14:07, 2038, will be erroneous.

Most of the Banks, Financial Institutions and Insurance Companies, which have servers running on Unix or Unix like OS, will need to react now, as many product and services offered by them are for longer duration such as Loans, Bonds and Polices, etc. Modifications in the source code of the current applications, as given here and/or switching to 64-bit computing is required to solve the problem

Y2K38 being a serious problem needs to be solved properly and within time. But, a solution that can be applied Universally is not yet there. Most of the solutions are either for particular software, system or for delaying this problem.

ACKNOWLEDGMENT

We would like to express our sincere thanks to Mr. Arul (HOD, Information Technology, D.C.E., Chennai) and Mrs.Geetha Rani (Faculty, Information Technology, D.C.E, Chennai) who were abundantly helpful and supported us throughout this research work.

REFERENCES

[1] http://en.wikipedia.org/wiki/Year_2000_problem

[2] http://www.2038bug.com/pivotal_gmtime_r.c.html

[3] http://www.rdg.opengroup.org/public/tech/base/y2k2000.html

[4] "The Open Group Base Specifications Issue 6 IEEE Std 1003.1, 2004 Edition (definition of epoch)". *IEEE and the Open Group*. The Open Group. 2004. Retrieved 7 March 2008.

[5] Sun Microsystems. "Java API documentation: System.currentTimeMillis". Retrieved 7 May 2007.

[6] http://en.wikipedia.org/wiki/Year_2038_problem

[7] Millennium bug hits retailers from BBC News, 29 December 1999.

[8] http://www.crn.com.au/News/163864,bank-of-queensland-hit-by- y201k-glitch.aspx

[9] http://www.informationweek.com/desktop/25-years-from-today-a- time-for-bugs/d/d-id/1108280?

[10] http://stablecross.com/files/End_Of_Time.html

[11] http://webarchive.org/web/20060408161959/http://ununium.org/articles/uuutime

[12] https://code.google.com/p/android/issues/detail?id=16899

[13] http://books.google.es/books?id=vEN-kKdtCwC&pg=PA49&dq=292,277,026,596&cd=1#v=onepage&q=292%2C277%2C026%2C596&f=false

[14] http://substitute.livejournal.com/1430908.html

[15] "The Case for Windowing: Technique by Raymond B. Howard, Year/2000 Journal, Mar/Apr 1998.