西暦2000年問題の技術的な教訓

技術本部　IT コンサルティング室長　田中　薫

１．はじめに

去年、あれほど騒がれていた2000年問題は、今となっては、どこかへいってしまった感じである。マスコミもニュースバリューがなくなると、取り上げなくなるようだ。実際は大方のところでは2000年を無事にクリアできた、ということだろう。年末・年始に出勤したり泊まりがけで対応された方々には、ほんとうにご苦労様と申し上げたい。

「2000年になるとすべてのコンピュータが狂いだす」とマスコミで喧伝されてからようやく本格化した感がある2000年問題への対応は、計算機に依存するシステム作りにおける設計・アーキテクチャ上の問題点をいろいろと含んでいるように感じる。

2000年も半ばを過ぎた現在、そういった観点を中心に、2000年問題とは何だったのかについて考えてみる。

２．Ｙ２Ｋの原因と現象

西暦2000年問題とは、そもそも次のような原因や現象が発生することで、システムのインフラにあたるハードウェアやOS、あるいはアプリケーションプログラムの動作が正しくなされないことから、計算機の処理内容と結果が信用できなくなる事象である。

⑴　日付の年を西暦の下２桁だけ保持することによる各種の問題点

年月日の年は、西暦の下２桁だけで通用するという慣例に従って、システム内部に蓄積する日付データの年として２桁しか持たなかった。そのため、2000年になったときに1900年代か2000年代かの区別がつかなくなったり、ソート順が狂ったりしてデータ処理に支障をきたすことになる。ここに最大の問題点があり、内部的には４桁で持っているが、実際の処理には２桁しか使っていないケースも含め、2000年問題の対応を混乱させている。なぜ２桁しか持たなかったのかが責められているようだが、原因はそう単純ではないと考える。

⑵　2000年がうるう年であることを忘れていた問題点

うるう年の定義からすると、西暦年が４で割り切れて100で割り切れない年はうるう年、100で割り切れて400で割り切れない年は平年、400で割り切れる年はうるう年ということで、少し複雑ではあるが2000年はうるう年である。1900年はうるう年ではなかったので、2000年も平年として扱うプログラムが一部にあったことは事実であり、２月29日は2000年対応テストで必ずテストすべき日付に入っていた。

⑶　ユーザーインタフェース上の問題点

計算機の内部では年を４桁で持っていても、ユーザーに入力してもらうときは年を省略したり、できるだけ短い桁数で入力できるように入力・出力において操作方法のルールを決めていることが多い。

そのルールが2000年以降になったときに破綻する可能性がある。例えば、年の入力を２桁で済ますように入力された年に1900を加えるというルールになっていたとすれば、00/01/02は1900年と解釈されることになってしまう。また、日本では和暦入力も許したいということで、75以下の数字を入れると昭和、それ以上なら西暦というルールを設定したシステムもある。

さらに、アメリカ、ヨーロッパでは日付の順番が違い、MM/DD/YY や DD/MM/YY が習慣的に使われていることも、混乱に拍車をかける状況となっている。

⑷　PC のカレンダー機能の問題

初期の PC では、内部のクロックに連動するカレンダーは、西暦年の下２桁しか持っていなかった。その後、2000

年対応のために１桁追加されたが、一部の PC で年の桁上がりが正しく処理できなかった。2000年対応のために BIOS の更新をさせられた人も多い。

⑸ 日本固有の問題点

2000年問題の一部ではないが、日本では2000年（平成12年）から、「成人の日」と「体育の日」が、それぞれ１月と10月の第２月曜日になることに祝日に関する法律が改正になった。これに関しては2000年問題にまぎれて、あまり意識されていないが、システムカレンダーで営業日を設定している担当者は要注意である。しかし、いまどき祝日をプログラム中にハードコーディングしているシステムがあるとは思えないので、大きな問題ではないだろう。

日本では、西暦と和暦を混在して使い分けているので、2000年の影響は比較的少ないという意見が一時ささやかれたが、和暦で日付を持つような見通しの悪いシステムの設計はすべきでない。入力・出力には和暦を使うとしても、内部で持つ日付は西暦または元号の変更に影響されない通算年で持つべきだ。

３．歴の歴史と特異年

2000年問題の騒動の発端である「年の下２桁」という点では、多くのシステムが引っかかったようだが、その「下２桁だけで済む」というのは誰がいつ判断したのだろうか。

初期のコンピュータはメモリ、２次記憶ともにコストが高く、できるだけ記憶量を減らすために年は２桁しか持たなかったという説明がされているが、筆者はこれを信じない。1970年代の汎用機である IBM 系のメインフレームの数値はパック型10進数で持つのが通常であったことを考えれば、２桁も３桁も、記憶領域は２バイトであり、節約はできないからだ。

例えば、西暦４桁（３バイト）を持つのが大きな負担だとしても、1900年をベースとする３桁で持っていれば、2000年問題は起きず、問題発生は2899年まで先送りできたはずだ。では、そこまで見通しのきく SE がいなかったのであろうか。どちらかといえば、そのほうが真実に近いのだろう。

暦の歴史を辿ってみると、いろいろと面白いことが見つかる。一部の例をあげると、1582年10月４日まではユリウス暦といい、４年ごとにうるう年が必ずある暦であった。これを1582年10月15日からは今のグレゴリオ暦に変わり、４年に一度のうるう年を、100で割り切れる年は通常の年にし、さらに400で割り切れる年はうるう年とすることになった（ちなみに季節とカレンダーのずれを修正するために10月４日の次の日は15日とされた）。また、ユリウス暦からグレゴリオ暦への切り替えは世界全体で行われたわけではなく、一部にユリウス暦を適用している地域が残っていたのだが、その話は今のテーマとは関係ないので追求しない。

この経緯から考えると1700年、1800年、1900年は例外的に平年となり、2000年は正常に戻ってうるう年になる。

実は、西暦で表示するときに社会一般が1900年代を省略して、下２桁で年を表わす習慣があり、システムの設計者もそれに習って、年を下２桁で格納した、というのが真実かつ最大の原因だろう。そこに、2000年は上記のようにうるう年の特異年にあたり、状況をさらに複雑にしたといえる。

４．2000年問題の類似問題が発生する可能性

それでは、Ｙ２Ｋ問題は再発しないのだろうか。もちろん、2000年は今年１回きりであるが、それと類似した問題は次のように発生する可能性があると考える。

⑴ UNIX、PC のタイムスタンプ

初期の16/32ビットベースのコンピュータ OS である UNIX では、日付・時刻の表わし方として、1970年１月１日00:00:00からの経過秒数をタイムスタンプとして32ビットの整数で持つ、という仕様があった。この値があふれるのは2039年頃になるはずで、一部の人は2039問題として気にしている。もっとも、最近の UNIX は、タイムスタンプを64ビットで持つというように変更してきており、アプリケーション側の対応を誤らなければ、問題ないだろう。

また、MS-DOS も1980年からの年数を７ビットで持つという仕様があり、こちらは、2107年までしか表わせない。

1999年８月に、一部のカーナビゲーション・システムが動かなくなった。これも、日付を限られたビット数でもっていたため、それがあふれて処理できなくなったことが原因だった。

⑵ データベースの smalldatetime

SQL Server などでは、datetime の他に smalldatetime というデータ型があり、４バイトと小容量なのでよく使われるが、これも上記のタイムスタンプと同じで2039年ころにオーバーフローしてしまう。

datetime の方は西暦9999年まで表現できる代わりに８バイトを占めてしまう。今後、設計するシステムに、どちらのデータ型を使うべきかは明確であろう。

⑶ 数値の桁数

数値、特に金額を格納するときに、その桁数を何桁にするか頭を悩ました人は多いだろう。日本の国家予算は70兆円から80兆円になろうとしているが、これは数値の桁数でいうと14桁になる。いまどきのコンピュータではないと思うが、20年位前の汎用機では COBOL の数値桁数は15桁くらいだった記憶がある。今後、何十年かすれば日本の国家予算は15桁をオーバーすることになる。これと類似した問題の発生は、筆者が知っている限りでも、実際に次のように存在する。

① 一回の取引で100億円以上になることはない、という前提で金額を10桁で持つシステムがあり、その後に100億円以上の取引が発生したため、取引を２つの伝票に分けて入力した。

② 株価が1000円台だった頃の証券取引のシステムでは、株の単価を４桁で持つものがあった。その後（皆さんもご存知のように）１万円を超える株価が発生し、現在では１億円を超えるものもある。

③ 顧客コードや社員コードなど文字列の固定長で持つものが、収容能力を超えるコードが発生したために桁数を増加させたり、数字だけでなくアルファベットもコードに使うように拡張する例はよく見かける。

以上、いずれも処理できる桁数に上限があるか、または、システムの設計時に決められており、それを超えるときにソフトウェアが対応できなくなることが原因で発生すると考えられる。もともとのインフラ（ハード、OS）に制限があるものについては使用を止めて、もっと上限値の大きいインフラに置き換えるしかない。さらに、システムの設計時に決めた最大値についてはソフトウェアを改善するしかない。

今回のＹ２Ｋはこの両原因が一緒に発生したために混乱が大きくなった側面もある。

５．2000年問題の再発防止

それでは、年の桁数はどれだけ持てばよいのか。今回の処置で西暦年数を４桁で持つようにしたところが多いと思われるが、西暦10000年になったときに同じ騒ぎを繰り返すのだろうか。今から8000年も後なので、そのときの人々に対応を任すという考え方はある。もし、今のコンピュータをタイムカプセルに入れて、西暦10000年に開けるとして、そのときにも正常に動くようにしておきたいなら、今からでも西暦年数については５桁や６桁を持つ必要がある。

時間はいつまで続くのか誰にもわからない。自分の作ったプログラムを半永久的に正常に動作するようにしたいと思えば、コンピュータが使う数値とか文字列などの上限をなくせばいい、と考えるだろう。しかし、それはコンピュータの処理効率と相容れない要素であり、これまではCPU能力の有効活用の観点から整数演算の最大値は決められてきた。もちろん、任意桁数を処理するライブラリは存在するが、コンピュータのもともとの処理効率からすると非常に遅く使い道が限定されている。

CPU能力向上を目的別に分類すると、第一期はCPU能力向上そのものが目的であり、第二期はユーザーインタフェース、特にGUI（Graphical User Interface）を向上させるために使われ、第三期は処理データの柔軟性をコンピュータがカバーする方向だと思う。現時点は第二期が終了し、これからは第三期であると思う。それは、次のいくつかの方向で現在も改善が進んでいる方向と考えている。

⑴ データの型や桁数に合わせて処理を変える

データのオブジェクト化とも考えられるが、年や金額の格納データ型に、このやり方が生かされれば、2000年問題の再発を考える必要はなくなるかもしれない。

⑵ データを交換するときにデータの意味付けに関する情報もいっしょに渡す

交換する２社間だけに通用するフォーマットやルールを定める方法では、多対多のデータ交換の場面では使用できない。複数の参加者を横断するルールを決める必要があり、現実世界ではなかなか難しい。具体的には、XML（eXtensible Markup Language）がこの方向の典型である。

⑶ ネットワーク上での協調・分散処理

分散処理は、C/S（クライアント／サーバー・システム）の分散処理で、その方向性が見えたかに思えたが、マシン能力とソフトウェアシステムの構築技術の両面で未熟であることが明確になった。

協調・分散システムとして稼動させるためには、その単位となる部分の自立性が大切であり、今後も模索が続くであろう。

６．何が問題だったのか

SEの観点からいうと、システムの設計・構築時に前提とした事柄が変わってきたとき、システムは当初の機能を果たさなくなるのは当然で、変わった前提に合わせてシステムの改造が必要となる。業務的な前提条件はよく考慮されるが、年を２桁で持つか４桁で持つかは、それほど明確な前提条件ではない。

それでは、年を２桁で持つと決めた彼ら（SE）は、2000年がくるとは思わなかったのであろうか。

推測だが、少なくとも1960年から1970年代のSEは、自分の作ったシステムが2000年まで使われるとは、よもや思っていなかっただろう。確かに、それらのシステムは、現在では使われていないが、その初期のアプリケーションシステムが、アーキテクチャを決定し、その後に作られたシステムは、そのアーキテクチャを踏襲して受け入れた結果が2000年問題につながったと考える。

いちばん簡単な例が、最初に作られたシステムでデータをデータベースなどに格納する際に「年を２桁で持つ」という仕様で作成され、その後に追加されるシステムがそのデータを参照していれば、そのシステムでも年を２桁で扱わざるを得ない、というものだ。今までのシステムの前提を見直して、再設計・再構築する必要があると誰かが考えても、そうするだけの余裕がない場合もあったのだろう。

また、一度設定されたアーキテクチャを覆すことは難し

い、ということもある。例えば、初期のアーキテクチャを決定した人が、その後に部長や役員になっていると、一介のSEがアーキテクチャの検討をし直すことは、上司のあら捜しをしている、とみなされかねない。

さらに、アーキテクチャの根本から考え直すという困難さに比較すると、既に設定されている路線（アーキテクチャ）に沿ってシステムを開発するのは容易であり、安易な方向に流れることは一概に批判できない。

しかし、企業の情報システムは、５年、10年単位でアーキテクチャを見直すべきであり、2000年問題はその良い機会だった。もし、2000年問題を対症療法で乗り切ったところは要注意で、今からでもアーキテクチャを見直すことを勧める。

７．おわりに

2000年問題は、意外に奥が深い。当初は西暦の下２桁だけの問題と思っていた。しかし、うるう年の問題もあり、PCのクロックが2000年を迎える際に正しく更新されるかなど、対策をしても次々と問題点が出てくる様子を見て、これは、一種のアーキテクチャの変更なのではないか、と考え直したものである。

アーキテクチャとして包含されていて、普段はあまり意識しない前提条件が変更されたとき、旧来のシステムを修正しようとしても、なかなか一筋縄ではいかず、渦中の担当者は「最初から作り直した方が良かった」と思ったことだろう。

しかし、世界的に騒がれたおかげで対策も無事に済み、大きな混乱がなかったことは幸いであった。この事象を良薬として、今後の糧にしていきたい。