Y2K対応で発生した注意すべき事例

当社は、Y2K（西暦2000年問題）のシステム対応を行うなかで発生した障害などの事例を、全社レベルで一元管理し、共有化している。その中から他のシステムでも参考となりそうな障害事例について本稿で紹介する。これが他山の石として、残された数カ月の間での対応の一助となれば幸いである。

報告されたいくつかの要注意事例

全社レベルで報告された事例を紹介する。

事例①：UNIXサーバーコマンドのバグ

1999年１月５日以降、データ登録ができなくなる現象が発生した。そのサーバーはY2K対応パッチ（修正版）を適用済みであったにもかかわらず本現象が発生した。原因を調べたところ、適用していた対応パッチは、障害を起こしたコマンド部分の修正について未対応版であることが判明した。

事例②：アプリケーション日付判断のバグ

日付エリアが99999999のデータが引き落とされる現象が発生した。バグ自体は、Y2K日付に関わるものでなかった。当該システムは再構築されたシステムであり、当初からY2K対応については考慮済みというステータスで管理されていた。

事例③：シェルなどの仕様理解不十分

Y2K日付でのテストを行ったところ、日付が正しく算出されなかった。原因としては、①Cシェル（UNIX）の変数の仕様、②シェル内での前ゼロサプレス、③関数の使用方法の誤り――であった。

①のCシェルの仕様では、先頭が０の数値は８進数として扱われる。年号を２桁で扱っていた場合、2000年１月28日は000128となり、シェル内で８進数とみなされ、数字中の８を不正と判断しエラーとなった。

②では、シェルの中で３カ月後を求める計算をし、その結果の値でテーブルの検索処理をしていた。99年10月（9910）から３カ月後の2000年１月（0001）を求める計算をしたところ、結果が0001でなく、前ゼロがサプレスされて１となり、後ろの検索処理が正しく行われなかった。

③では、日付関数の年号から1900を引く機能を利用し、年号の下２桁を求める処理を行っていた。2000年についてその処理を行ったが、00ではなく、結果は100であった。

事例④：プログラムの調査漏れ

最終確認テスト中に、Y2K日付のデータの処理がエラーとなった。日付の項目名称を頼りにプログラムの調査を行っていた。しかし、古いプログラムで用いられていた項目名称が調査対象に含まれておらず、結果として対応の漏れが発生した。

事例⑤：外部データのレイアウト変更

これは、当社の事例ではないが、外部から受け取ったデータを処理したところ、システムがダウンした。外部からの入力データが何

の連絡もないままY2K対応されていたため、レイアウトの不整合が発生し、システムトラブルにつながった例と推察される。

事例⑥：機器そのものへの注意

WS（ワークステーション）においても、電源を抜いた状態で内蔵電池が切れると、PC（パソコン）同様に、各種設定情報がクリアされる。その場合、ブートディスクのデフォルトが内蔵ディスクでない機種については、特別な対応が必要となる。また、内蔵電池が特殊なものの場合には交換に時間がかかることもある。2000年を迎えるに当たって、停電が発生したり、コンセントを抜いておく対応により、WSの再立ち上げができず、復旧に時間がかかる可能性がある。

要注意事例からいえること

前述の事例から次のようなことがいえる。

①調査に終わりはない。新規システムも油断は禁物

プログラムの日付関連の調査には限界がある。Y2K問題が意識された後に構築したシステムでも、未来・過去の日付を求める処理などにおいてはきちんとした確認が必要である。その手段としては、テストが有効であるが、現実（期間・コスト・重要度）を意識しながら優先づけして対応することが必要である（100%対応するのは現実的には難しい）。

②Y2K対応パッチに終わりはない

Y2K対応パッチと称していても、早い段階では未対応のものがあったり、バグが内在する可能性がある。また、Y2K対応パッチが1999年中にすべて出揃う保証もない。ただし、最新パッチを当て続けるとなると、デグレード確認テストの負荷が増える（きりがない）。

情報収集は継続的に実施する必要があるが、適用はある時点で見切るという判断をしていくことも必要である。

③他社システムとのインタフェース変更は同期を取って

各社それぞれでY2K対応を行っているが、外部とのデータのやり取りをしているファイルレイアウトを変更する際には、相手先と連絡を密に取りながら、タイミングを合わせて切り替えていくことが必要である。

④最後はコンティンジェンシープランの策定と体制作り

内蔵電池の問題では、ソフトウェアに関係のないところで問題が発生する可能性が指摘された。また、OS（基本ソフト）などのパッチやアプリケーションにおいても100%完璧な対応ができていない場合もある。これらの不測の事態に備えるため、開発関係者だけでなく実際の業務現場の方々と協同でコンティンジェンシープランを策定していくことが必要となる。また、それらの事態に備えるという意味では、権限を持つ責任者と一緒になって年末年始の体制作りを進めていくことも非常に重要なことである。

（野村総合研究所　櫻井善紀）