B-013

時刻同期プロトコル内の時刻変換による 32 bit Linux の 2038 年問題対策

Solving the Year 2038 Problem in 32 bit Linux by Converting Time in Clock Synchronization Protocols

矢吹 潤† 岡部 亮† 堀井 圭祐† 外山 正勝†
Jun Yabuki Ryo Okabe Keisuke Horii Masakatsu Toyama

1. はじめに

32 bit 版の Linux には，2038 年を迎えると時刻管理変数がオーバフローする問題（2038 年問題）がある[1]．産業用計算機のような製品寿命が長い製品においては，2038 年を越えて運用される可能性があるため，この問題の対策が必要とされる．

本問題の対策として，Linux Kernel や標準 C ライブラリ内に変更を加える方法が考えられるが，改修が広範囲にわたるため，改修量や保守性の維持が問題となる．そこで本稿では，システム固有の時刻を設定する対策を提案する．これは，システム内部時計に対してシステム固有の時刻を設定することで，システム内部時計の時刻が 2038 年を迎えないようにする．また，時刻同期プロトコル NTP および PTP の一実装に対する本対策の適用について考察する．

2. 背景と課題

2.1 背景

Linux では，1970 年 1 月 1 日午前 0 時（UNIX Epoch）からの経過秒数をカウントすることでシステム内部時計の時刻を管理している．時刻を表現するデータ型（以下，`time_t`）は符号付き整数型で定義されている．そのため，32 bit の CPU アーキテクチャにおいては，`time_t` は 32 bit の符号付き整数型として扱われるため，2 の 31 乗で表現可能な 2,147,483,648 秒（2038 年 1 月 19 日 3 時 14 分 17 秒）を超えるとオーバフローしてしまう．この時，`time_t` を利用している Linux 内のソフトウェアで不具合が発生し，システムが正常な動作を継続不可能になると考えらえる．このような現象が 2038 年問題と呼ばれている．

産業用計算機のような製品寿命が長い製品においては，20 年以上継続して使用されるものも多く存在し，2038 年を越えて運用される可能性があるため，この問題の対策が必要となる．

Linux では，図 1 に示すように `time_t` が利用されている．Linux Kernel，標準 C ライブラリ，アプリケーションにおいて想定される 2038 年問題による影響を下記に示す．

- **Linux Kernel**
  `time_t` を利用する時刻関係のシステムコール（`clock_gettime` や `time` など）が影響を受ける．デバイスドライバでも `time_t` 型を利用している場合は影響を受ける．
- **標準 C ライブラリ**
  システム内部時計の時刻の設定や取得を行うために定義されている `time_t` および `time_t` を利用するライブラリ関数（`mktime` や `localtime` など）が影響を受ける．これらのライブラリ関数は内部で Linux Kernel のシステムコールを呼び出しているため，Linux Kernel の改修の影響も受ける．

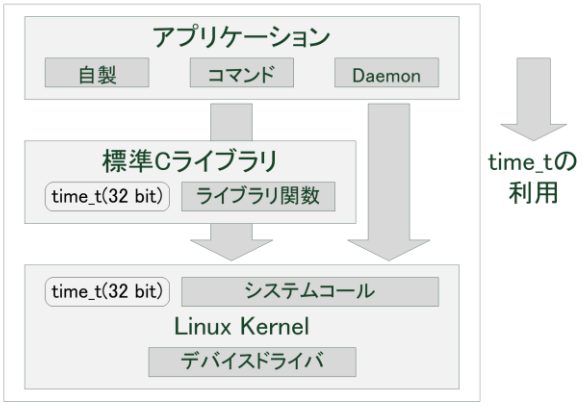

図 1　Linux における `time_t` の利用

- **アプリケーション**
  上記で示した時刻関連のライブラリ関数，もしくは直接システムコールを呼び出しているアプリケーションが影響を受ける．これは，自製アプリケーションに限らず Linux に含まれるコマンドや Daemon プログラムなども対象となる．

2.2 課題

Linux Kernel や標準 C ライブラリ内に変更を加えて 2038 年問題に対応する場合，プログラムの改修が必要な箇所が広範囲に及ぶため，改修や検証が困難であるという問題がある．また，コミュニティやディストリビュータから提供されるパッチの適用が困難になるため，システムの保守性の維持が問題となる．本稿では，上記の問題を回避するため，Linux Kernel や標準 C ライブラリ内に変更を加えずに 2038 年問題を解決することを課題とする．

3. 対策方針の候補

2038 年問題の解決策として我々が示す 3 つの対策方針を説明する．

(1) `time_t` を 64 bit の符号付き整数型に変更
システム内の全ての `time_t` の定義を 32 bit の符号付き整数型から 64 bit の符号付き整数型に変更することで，`time_t` で 2038 年以降を表現可能にする．

(2) システム内部時計に固有の時刻を設定
システム内部時計に対して敢えて現在の時刻ではなくシステム固有の時刻を設定することで，システム内部時計の時刻が 2038 年を迎えないようにする．

(3) OSS コミュニティによる対策を反映
OSS コミュニティの動向をウォッチしつつ，2038 年問題の対策が完了した時点でソフトウェアをアップデートする．

† 三菱電機株式会社　情報技術総合研究所 Information Technology R&D Center, Mitsubishi Electric Corporation

4. 対策方針の選定

本稿では，システム上で動作するアプリケーションがシステム出荷時に固定される，組込みシステムを対象とする．したがって，時刻に関する対策を講じた際の影響範囲は，時刻同期や時刻表示を行うアプリケーションなどに限定することが可能である．

3 章で示した 3 つの対策方針のうち，対策方針(1)は Linux Kernel や標準 C ライブラリ内に変更を加える必要があるため，2.2 節で述べた課題を達成できない．対策方針(3)は OSS コミュニティによる対策の完了時期が未定であり，対象システムは出荷後にソフトフェアのアップデートが不可であるため，選外となる．

対策方針(2)は，Linux Kernel や標準 C ライブラリに変更を加えずに対策が可能である．対象システムにおいては，現在の時刻を必要とするアプリケーションの改修のみで対策が可能である．そのため，本稿では対策方針(2)を提案する．

以降，時刻サーバから配信される現在の時刻を外部時刻，本対策によりシステム内部時計に設定する過去の時刻を固有時刻と表記する．

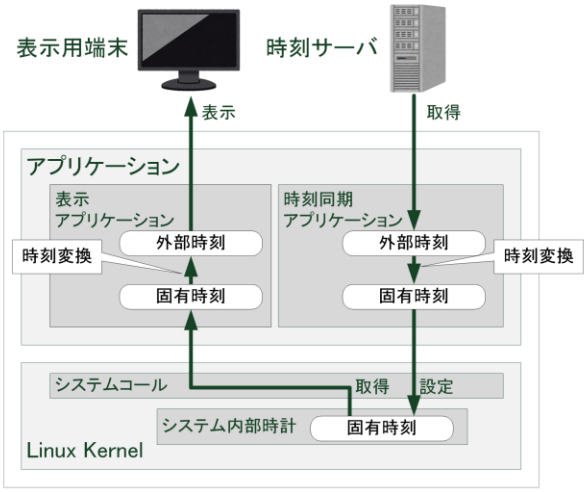

図 2　対策方針(2)適用時の構成

システム内部時計に固有時刻を設定して運用するためには，図 2 に示すように，システム外部から取得した時刻をシステム内部時計に設定する際や，システム内部時計の時刻をシステム外部へ出力する際，これらを行う各アプリケーションで時刻変換を行う必要がある．例えば，時刻同期の際には，時刻サーバから取得した外部時刻を固有時刻に変換する．時刻の表示の際には，システム内部時計から取得した固有時刻を外部時刻に変換する．これによって，システム外部では現在の時刻を使用しながら，システム内部時計には時刻同期されたシステム固有の時刻を使用することが可能となる．

なお，時刻の扱い方はアプリケーションによって異なるため，アプリケーションごとに時刻変換を行う方法を個別に設計する必要がある．

5. 時刻同期アプリケーションへの適用

時刻同期プロトコル NTP（Network Time Protocol）の一実装である NTP クライアント（ntp 4.2.8p11）および PTP（Precision Time Protocol）の一実装である PTP スレーブ（linuxptp 1.8）において時刻変換を行う方法を検討する．

ntp, linuxptp では，時刻サーバからプロトコル独自のデータ型で時刻を取得し，アプリケーション内でその時刻を `time_t` に換算し，システム内部時計に設定することで時刻同期を行う．2038 年以降は，外部時刻を `time_t` に換算することによってオーバフローするため，外部時刻を `time_t` に換算する前の，プロトコル独自のデータ型で時刻を扱う段階で時刻変換を行う必要がある．

ntp における時刻同期では，NTP サーバから NTP 独自のデータ型で時刻を取得し，ntp 内でその時刻を `time_t` に換算し，システム内部時計に設定する．そこで，ntp への対策として，図 3 に示すように，時刻サーバから時刻を取得した直後に時刻変換を行うようにした． 図 3 では，NTP 独自のデータ型の時刻のうち，外部時刻を外部 NTP 時刻，固有時刻を固有 NTP 時刻と表記している．

固有時刻は外部時刻の 40 年過去の時刻と仮定し，時刻サーバから取得した時刻を 40 年過去の時刻となるように時刻変換を行うようにした．

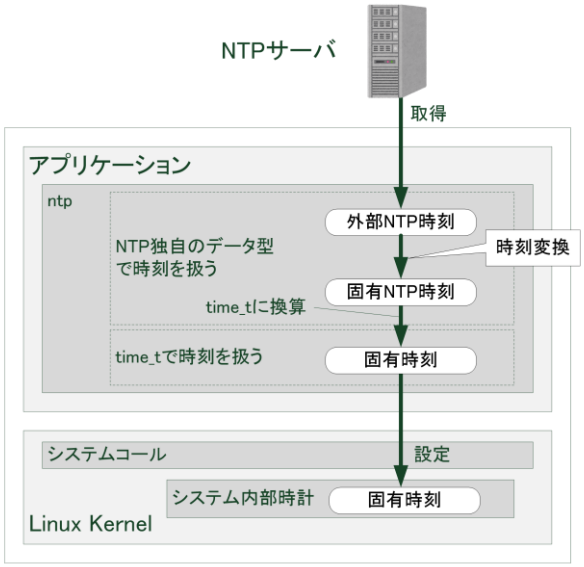

図 3　ntp への対策方針(2)適用時の構成

ntp と linuxptp においては，アプリケーション内で上記の時刻変換を行うことで対策が可能であると考えられる．これにより，Linux Kernel や標準 C ライブラリ内に変更を加えずに 2038 年問題を解決する見込みを得た．

6. おわりに

本稿では，システム内部時計の時刻が 2038 年を迎えないようにする対策を提案し，アプリケーション内で時刻変換を行うことで，Linux Kernel や標準 C ライブラリ内に変更を加えずに 2038 年問題を解決する見込みを得た．

今後は，実際にアプリケーション内に時刻変換を行う処理を追加し，本対策による 2038 年問題の解決の実現性を確認していく．

参考文献

[1] 坂井丈泰, “GPS における週番号の決定と時刻表現に関する諸問題”,測位航法学会論文誌, Vol.2, No.2 pp. 13-20 (2011).