オペレーティングシステムの動的バージョンアップ

7H－3

三浦 雅弘　平山 秀昭

(株)東芝　情報・通信システム技術研究所

1　はじめに

ソフトウェアのバグは，コンピュータシステムの信頼性を低下させる一因である．バグによる障害を防止するためにNバージョンプログラミングやリカバリブロックといった手法が考えられているが，いまだ現実的な実装をみておらず，またバグ自体を防止するためのソフトウェアエンジニアリングの手法も検討されているが，やはり十分なものとなっていない．このため，ソフトウェアのバグ修正のためにバージョンアップを行なう必要がたびたび生じる．さらに，ソフトウェアの機能拡張のためにバージョンアップを行なう場合もある．

とくに，バージョンアップを行ないたいソフトウェアがOSである場合，ユーザのサイトによってはOSを停止することによって多大な影響を及ぼす可能性があるため，OSを動的に(動作させたままで)バージョンアップすることができれば非常に有用である．

我々は，動的バージョンアップが可能な特別なOSを製作するのではなく，市場で受け入れられているUNIX[1]等の標準的なOSの動的バージョンアップを可能にすることを目指している．今回はUNIX上に実装したバージョンアップ機構について紹介する．

2　概要

本バージョンアップ法では，OSカーネルを関数単位で置き換える．具体的には，図1に示すように，カーネルメモリを確保して，置き換えたい関数の新版をロードし，旧版の先頭に新版への分岐命令を書き込む．バージョンアップのより詳細な手順は次節で説明する．

本バージョンアップ機構は，バージョンアップ用に追加したシステムコールと，そのシステムコールを使ってバージョンアップを行なうプログラムとからなる．

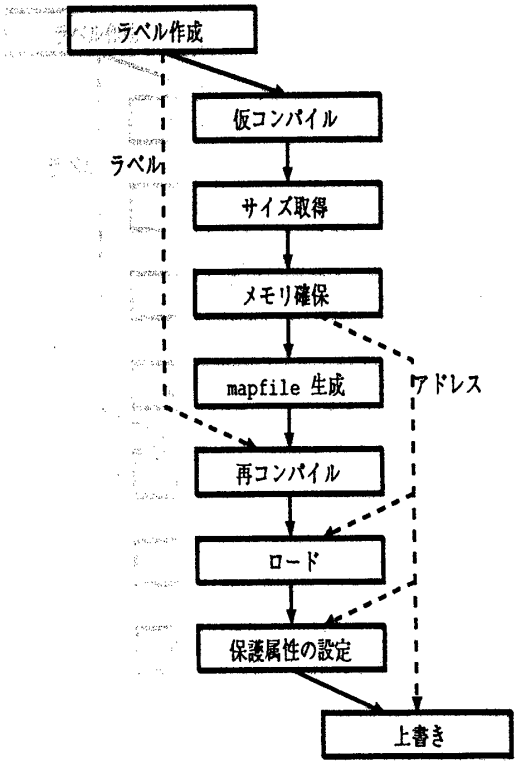

図1: バージョンアップ手順

3　バージョンアップ手順

本節では，バージョンアップの具体的な手順を説明する．

3.1　カーネルコードからのラベル作成

置き換えたい関数の新版がカーネル内の他の関数や変数を参照している場合，関数のリンク時に参照を解決する必要がある．このため，まずカーネルコードを走査してシンボルとその値を抽出し，シンボルのラベル定義のみを行なうアセンブラプログラムを生成する．

3.2　新版関数の仮コンパイルによるサイズ取得

新版の関数をロードするためのカーネルメモリを確保するためには，そのまえに新版の関数のサイズを知る必要がある．このため，新版の関数をコンパイルし，生成されたオブジェクトを調べて，関数のサイズを取得する．

この段階では，新版の関数が最終的に配置されるアドレスが不明であるので，分岐命令などは正しくコード生成できない．

3.3　カーネルメモリの確保

新版の関数をロードするのに必要なサイズ分のカーネルメモリを確保する．

カーネルメモリ確保用のシステムコールをカーネルに追加し，バージョンアップ・プログラムはこのシステムコールを呼び出す．

3.4　mapfile の生成

UNIX の ELF リンカ ld は，通常はオブジェクトファイルから実行ファイルへのマッピングを自動的に行なう．こ

[0]Dynamic Version Up for Operating Systems

[0]MIURA Masahiro, Hideaki HIRAYAMA:
[0]Information & Communication Systems Laboratory,
[0]TOSHIBA Corporation

[1]UNIX は，X/Open カンパニーリミテッドがライセンスしている米国ならびに他の国における登録商標です．

のマッピングは, ld の mapfile オプションを使うことによってユーザが変更できる. なお mapfile は通常のテクストファイルである.

今回は, 新版の関数を, 確保したカーネルメモリに配置したときに正しく動作するようリンクする必要がある. このため, そのようなリンクを行なうよう指定する mapfile を生成する.

3.5 生成した mapfile を使っての再コンパイル

生成した mapfile およびラベル定義のアセンブラプログラムを使って, 新版の関数を再コンパイルする. ここで生成したコードが最終的なコードとなる.

3.6 新版関数のロード

新版の関数を, 確保したカーネルメモリへロードする.

ユーザのメモリ空間からカーネルメモリへコピーするシステムコールを追加してある. バージョンアップ・プログラムはまずバッファを用意し, 新版の関数のコードをいったんディスクからバッファへ読み込んだあと, 上記のシステムコールを呼び出して, バッファからカーネルメモリへコピーする.

3.7 メモリブロックの保護属性の変更

新版関数をロードしたメモリブロックに, 読み込み可・書き込み不可・実行可といった保護属性を付ける.

メモリブロック保護属性の変更用のシステムコールをカーネルに追加し, バージョンアップ・プログラムはこのシステムコールを呼び出す.

3.8 分岐命令の旧版関数への上書き

旧版の関数の先頭部分に, 確保したカーネルメモリのアドレス (つまり新版関数の先頭アドレス) への分岐命令を上書きする.

メモリ空間中のテクスト領域へのデータ書き込みは, 通常 MMU (メモリ管理ユニット) の働きにより不可能である. ここでは, MMU をパスして物理アドレスに直接データを書き込むシステムコールを追加し, バージョンアップ・プログラムはこのシステムコールを呼び出すことによって分岐命令を上書きする.

4 汎用性の評価

このバージョンアップ機構を実装するために追加したシステムコールは, 以下の 4 つである.

`ov_kmem_alloc()`: カーネルメモリを確保する.

`ov_mmu_chgprot()`: メモリブロックの保護情報を変更する.

`ov_pawrite()`: 物理アドレスに直接データを書き込む.

`ov_copyin()`: ユーザのメモリ空間からカーネルメモリへデータをコピーする.

これらのシステムコールはいずれも, UNIX 以外の OS にも同様の機能があるものばかりであり, UNIX 固有の機能は用いていない.

したがって, このバージョンアップ機構を他の OS に実装するのは比較的容易であると思われる.

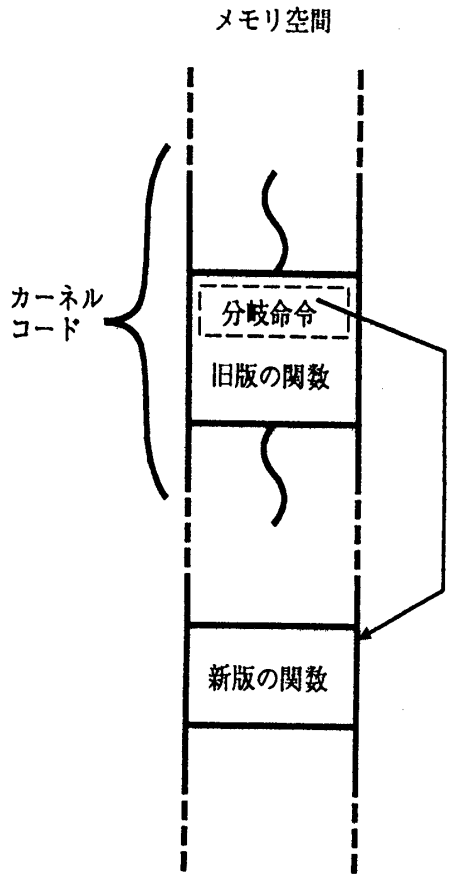

図 2: 分岐命令の上書き

5 おわりに

オペレーティング・システムの動的バージョンアップ法について説明した.

ここで紹介した方法は, 関数単位の置き換えを行う以外のことはできない, 制約の厳しいものである. しかし, 小規模なバグ修正などには十分に有用な方法であり, UNIX という既存の OS に容易に実装することができた.

今後は, より制約の緩い方法を検討していきたい.

参考文献

[1] Maurice J. Bach, "The Design of the UNIX Operating System", Prentice-Hall, Inc.

[2] S. J. Leffler, M. K. Mckusick, M. J. Karels, J. S. Quarterman, "The Design and Implementation of the 4.3BSD UNIX Operating System", Addison-Wesley Publishing Company, Inc.