特 別 研 究 報 告 書

題 目

TwinOSのバージョンアップの検討

研究室担当教員の署名

提 出 者

牛尾 裕

岡山大学工学部 情報工学科

平成21年2月9日 提出

要約

近年，計算機の性能が向上し，1台の計算機上に複数のOSを走行させ多様なサービスを実現する機能が注目されている．このような機能を持った技術として，1台のマシン上で複数のOSを同時に走行させるTwinOSという技術が研究されている．TwinOSでは，メモリ分割機能や共存OSメモリ展開機能など様々な機能を実装し，2つのLinuxの同時走行を実現している．

一方，カーネルは常に改良されるものであり，セキュリティや安定性の向上のため，カーネルをバージョンアップすることは非常に重要である．ここで，現在のTwinOSのソースコードには，カーネルのバージョンが低い問題がある．TwinOSのカーネルをバージョンアップするには，Linuxカーネルからの変更場所を明確にし，機能毎にモジュール化して考えることが重要である．しかし，現在のTwinOSは様々な機能が混在しており，機能別にモジュール化されていない．

本論文では，TwinOSのバージョンアップ手法を検討した．TwinOSをバージョンアップするには，まずLinuxとTiwnOSの差異を整理し，モジュール化する．次に，Linuxのバージョン間による差異を明確化し，先ほどのモジュールを新しいバージョンのカーネルに適用できるよう改変する．このような手順でTwinOSをバージョンアップできることを明らかにし，これを基に実装を行った．まず，LinuxとTwinOSの差異を整理したところ，69個のファイルが変更または追加されていることが分かった．この結果を基に，LinuxとTwinOSの差異を5つのモジュールに分割した．5つのモジュールのうち，メモリ分割機能を新しいバージョンのカーネルに適用して動作を確認した．

残された課題は，他の4つのモジュールのLinux 2.6への適用とLinux 2.4からLinux 2.6へのTwinOSの適用以外への適用可能性を探ることである．

目次

1 はじめに **1**

2 TwinOS **2**

2.1 特徴 2

2.2 起動方式 3

2.3 現在のTwinOS の問題点 4

3 バージョンアップ方式 **6**

3.1 目的 6

3.2 要求 6

3.3 バージョンアップ手順 7

4 TwinOS のバージョンアップ **9**

4.1 Linux 2.4 とTwinOS 2.4 の差異の整理 9

4.2 TwinOS の機能のモジュール化 10

4.2.1 TwinOS モジュール 10

4.2.2 TwinOS モジュールの確認 11

4.3 TwinOS 2.4 の実装における各機能の検討項目 11

4.4 Linux 2.4 とLinux 2.6 の差異の整理 13

5 TwinOS のモジュールの改変 **16**

5.1 メモリ分割機能 16

5.1.1 Linux のメモリ管理 16

5.1.2 TwinOS 2.4 のメモリ分割機能 17

5.1.3 TwinOS 2.6 のメモリ分割機能 18

5.1.4 動作確認 20

6 おわりに **23**

謝辞 **24**

参考文献 **25**

図 目 次

2.1 TwinOSの構成 3
2.2 TwinOS の起動方式 4

3.1 TwinOSのバージョンアップ手順 8

5.1 メモリマップの構造 17
5.2 BIOSが返却したメモリマップ 18
5.3 TwinOS 2.4 のメモリマップ変更フローチャート 19
5.4 TwinOS 2.4のメモリマップエントリの分割 20
5.5 TwinOS 2.6のメモリマップ変更フローチャート 21
5.6 Linux 2.6のメモリマップ 22
5.7 TwinOS 2.6のメモリマップエントリの分割 22

表 目 次

4.1 変更されたファイル一覧 14
4.2 追加されたファイル一覧 15

第 1 章

はじめに

近年，計算機の性能が向上し，1 台の計算機上に複数の OS を走行させ多様なサービスを実現する機能が注目されている．このような機能を持った技術として，1 台のマシン上で複数の OS を同時に走行させる TwinOS という技術が研究されている．TwinOS では，メモリ分割機能や共存 OS メモリ展開機能など様々な機能を実装し，二つの Linux の同時走行を実現している．

一方，カーネルは常に改良されるものであり，セキュリティや安定性の向上のため，カーネルをバージョンアップすることは非常に重要である．ここで，現在の TwinOS のソースコードには，カーネルのバージョンが低い問題がある．TwinOS のカーネルをバージョンアップするには，Linux カーネルからの変更場所を明確にし，機能毎にモジュール化して考えることが重要である．しかし，現在の TwinOS は様々な機能が混在しており，機能別にモジュール化されていない．

本論文では，TwinOS のバージョンアップ手法を検討し，実装した結果を述べる．これを通じて，カーネルの種類，およびカーネルのバージョンに依存しないバージョンアップの方式を検討する．

第 2 章

TwinOS

2.1 特徴

TwinOSは，1台の計算機上で二つのLinuxを独立に走行させる方式であり，以下の特徴を持つ．

(1) 相互に他OSの処理負荷の影響を受けない

(2) 両OSとも入出力性能を十分に利用できる

このため，1台の計算機ハードウェアにおいて，CPUやメモリ，入出力機器といった各資源について効果的な共有と占有が必要である．2つのOSの独立性を保つためには，共有するハードウェア資源を最小限とすることが有効であるため，CPUのみを共有する．CPU以外のハードウェアは分割し，分割したそれぞれを各OSに占有させる．分割するハードウェアのうち，入出力機器は，OSの起動時に指定したものだけを占有させる． 図2.1 にTwinOSの構成を示し，各ハードウェアの分割，および共有方法を以下に示す.

(1) CPU

CPUは時分割で制御する．具体的には，起動時に起動処理を順次逐次的に行い，起動後はタイマ割り込みを利用してOSを切替える．

(2) メモリ

上位と下位に2分割する．具体的には，最初に起動するOS(以降，先行OSと呼ぶ) にメモリの老番アドレスを，先行OSから起動するOS(以降，共存OSと呼ぶ) に若番アドレスを割り当てる．

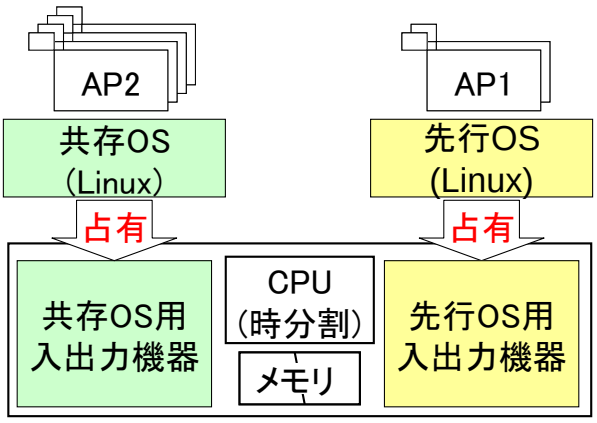

図 2.1 TwinOS の構成

(3) 入出力機器

各 OS 毎に指定された入出力機器のみを占有制御する．割り込み/例外処理は，当該割り込みを発生した入出力機器を占有制御している OS の割り込み処理が呼び出されるように制御する．このため，走行していない OS が占有する入出力機器からの割り込みの場合は，OS を切り替える．

また，各方式の実現においては，各 OS の改修を最小化するとともに，改修量が同等な場合は 1 つの OS に改修を集中させ局所化することとしている．これにより，開発工数の削減だけではなく，将来における Linux 以外の OS への本手法適用の際の工数削減を考慮している．

2.2 起動方式

TwinOS の起動方式を 図2.2 に示し，以下で詳細を述べる．TwinOS は以下の順番で起動する．

(1) 先行 OS を起動

先行 OS は，通常の Linux と起動方式に関して以下の 2 点が異なっている．

(A) メモリの老番アドレスの部分のみを利用する

(B) あらかじめ指定された I/O 機器のみを認識する

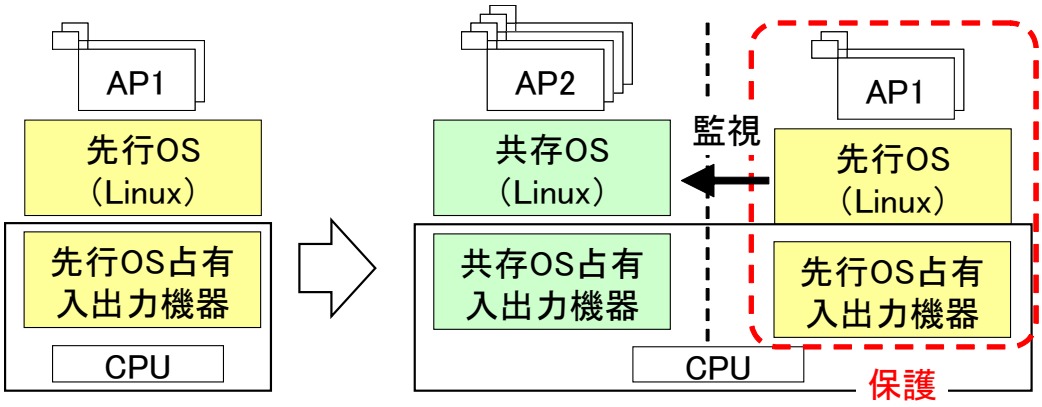

図 2.2 TwinOS の起動方式

このように改変されている先行 OS を通常の OS 起動方式で起動させる．

(2) 共存 OS のカーネルをメモリ上に展開

先行 OS の走行環境を保存し，圧縮された共存 OS のカーネルをメモリ上に展開する．その後，共存 OS セットアップルーチンを起動し，メモリ上にある圧縮された共存 OS カーネルを若番アドレス上に展開する．展開後，先行 OS に処理を戻し，走行環境を復元する．

(3) 共存 OS の起動

共存 OS の起動時には，OS の初期化処理により先行 OS の走行環境が壊されないように保護する必要がある．このため，共存 OS の起動時には，先行 OS の監視下で共存 OS の起動処理を実行する．先行 OS から共存 OS の処理を監視し，先行 OS の環境を破壊しないように保護している．

(4) 共存状態へ移行

先行 OS と共存 OS が独立動作し，共存状態になる．

2.3 現在の TwinOS の問題点

現在の TwinOS には，カーネルのバージョンが低い問題がある．TwinOS は，Linux kernel 2.4.7-10 を改変することで実現している．このカーネルは 2001 年にリリースさ

れたものである．このため，以下の問題がある．

(1) 安全性/安定性が低い

TwinOS 作成時からカーネルのバージョンアップを行っていないため，セキュリティ的に脆弱である．また，同じ理由で安定性も低い．

(2) 近年のハードウェアやソフトウェアに対応していない

近年発売されたマルチコア CPU 等のハードウェアやソフトウェアに対応していない．このため，限られた環境でしか利用できない．

このような背景から，TwinOS のカーネルをバージョンアップしたい要求がある．しかし，TwinOS のカーネルのバージョンアップには，以下の問題点があり，実現は困難である．

(問題点 1) 機能毎にソースコードが役割整理されていない

TwinOS のソースコードは機能毎にソースコードが役割整理されていないため，どのソースコードがどの機能の役割を果たしているかわからない．

(問題点 2) Linux のバージョン間の差が大きい

TwinOS のベースとなっている Linux kernel 2.4.7-10 と，最新の Linux kernel との差が大きいため，TwinOS のソースコードをそのまま最新の Linux カーネルに移植しても動作しない．

(問題点 3) ソースコードが未整理である

ソースコードにコメントが十分には付加されていない．また，関数名や変数名が役割に合った名前ではない．このため，可読性が低く難解なコードとなっている．また，実装途中で破棄されたソースコード等，必要のないコードが存在する．

(問題点 4) ドキュメントが不足している

TwinOS に関するドキュメントはあまり整備されていない．このため，TwinOS の構造 (機能同士の依存関係など) を理解するには，ソースコードを読む以外に方法がない．

TwinOS をバージョンアップするためには，これらの問題点を考慮しなくてはならない．このため，これらの問題点に対処しつつ．TwinOS のカーネルをバージョンアップする必要がある．

第 3 章

バージョンアップ方式

3.1 目的

本研究の目的は，オリジナルコードの改変による独自実装をオリジナルコードの改版に容易に追従させるための効果的な手法を探ることである．とりわけカーネルのコード改変について対象とし，TwinOS を題材として調査する．以降，現在実装されている TwinOS(Linux kernel 2.4.7-10 を基に実装) を TwinOS 2.4，Linux kernel 2.6.25 を基に新たに実装する TwinOS を TwinOS 2.6 と呼ぶ．また，Linux kernel 2.4.7-10 を Linux 2.4， Linux kernel 2.6.25 を Linux 2.6 と呼ぶ．本研究では，TwinOS 2.4 から TwinOS 2.6 へのバージョンアップ方式を検討し，実装する．これを通じてカーネルの種類，およびカーネルのバージョンに依存しないバージョンアップ方式の検討する．

3.2 要求

カーネルのバージョンアップに依存しないバージョンアップ方式を実現するには，以下の要求がある．

(要求 1) Linux と TwinOS の差異を機能毎に分割

TwinOS 2.6 を作成するためには，Linux 2.4 と TwinOS 2.4 の差異を明確化し，この差分を Linux 2.6 に適用する必要がある．ここで，Linux と TwinOS の差異を機能毎に分割し，バージョンアップの際にかかる手間を最小限に抑える必要がある．

(要求 2) Linux のバージョン間差異を明確化

Linux 2.4 と Linux 2.6 のバージョン間差異が大きいため，Linux 2.4 と TwinOS 2.4 の差分をそのまま Linux 2.6 に適用しても，TwinOS 2.6 は動作しない．このため，Linux 2.4 と Linux 2.6 のバージョン間差異を整理し，TwinOS のバージョンアップに関連する箇所を明確化する必要がある．

(要求 3) カーネルバージョンへの依存を最小限に抑えた TwinOS モジュールの作成

Linux と TwinOS の差異を機能毎に分割したもの (以下，TwinOS モジュール) は，カーネルバージョンへの依存を最小限に抑えて作成し，Linux 2.6 だけでなく，今後の Linux カーネルのバージョンアップにも追随できるようする．

3.3 バージョンアップ手順

3.2 節で述べた要求を満たすため，バージョンアップ手順を検討した．検討したバージョンアップ手順を 図 3.1 に示し，以下で説明する．

(手順 1) Linux 2.4 と TwinOS 2.4 の差異の整理

Linux 2.4 と TwinOS 2.4 におけるソースコードの差分を抽出し，整理する．

(手順 2) TwinOS の機能のモジュール化

1 度に大量の改変を行うことはバグの発生率を高め，デバッグの面から見て効率的ではない．このため，TwinOS の機能毎に差異を分割し，TwinOS モジュールを作成する．

(手順 3) TwinOS モジュールの確認

正しくモジュールが作成できたかどうかの確認するため，TwinOS モジュールを Linux 2.4 に適用し，モジュール分割版の TwinOS 2.4 が正常に動作することを確認する．

(手順 4) Linux 2.4 と Linux 2.6 の差異の整理

Linux のバージョン間による差異を明確にするため，ソースコードの差異を抽出し，整理する．

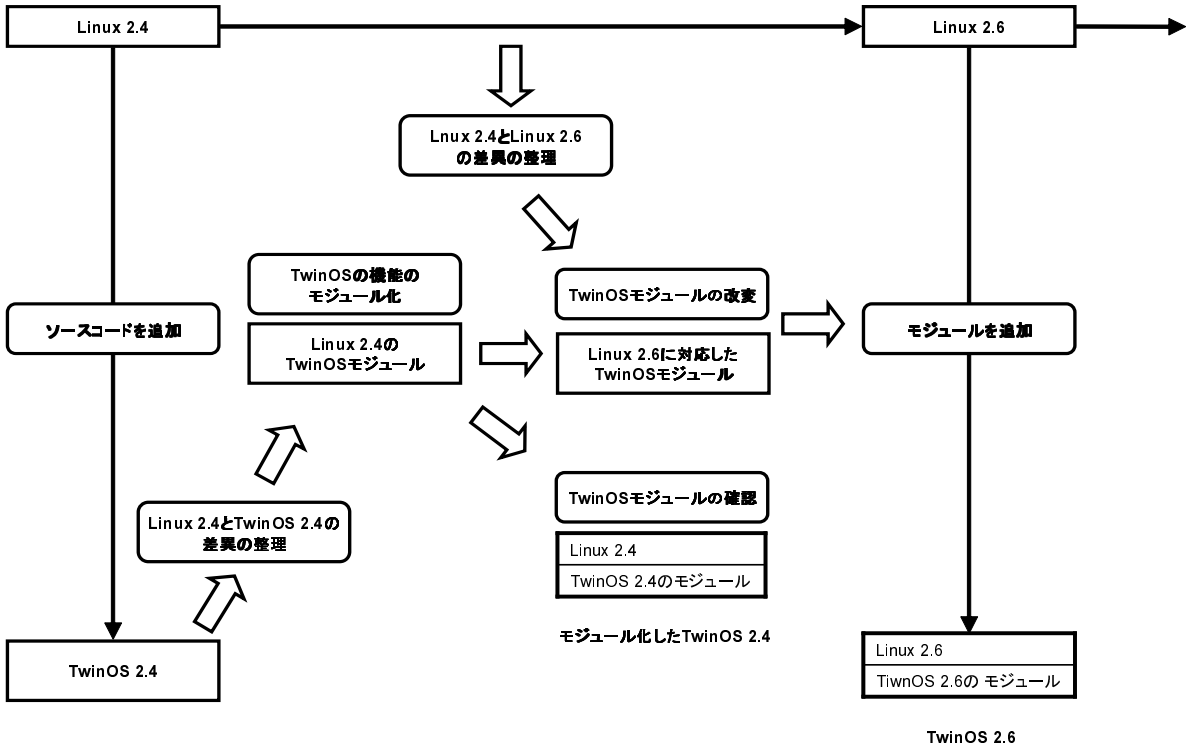

図 3.1 TwinOS のバージョンアップ手順

(手順 5) TwinOS モジュールの改変

TwinOS モジュールをそのまま Linux 2.6 に適用しても，TwinOS 2.6 は動作しない．Linux 2.4 と Linux 2.6 の差異を基に，TwinOS のモジュールを Linux 2.6 に適用できるように改変する．

(手順 6) TwinOS 2.6 の作成

(手順 5) で改変した TwinOS モジュールを用い TwinOS 2.6 を作成する．モジュール毎にソースコードを追加することにより，デバッグ作業を行いやすくし．作業を効率化する．

以上の手順に従い，TwinOS のバージョンアップを行う．本論文の以降では，TwinOS のバージョンアップついて詳細を述べる．

第 4 章

TwinOS のバージョンアップ

4.1 Linux 2.4とTwinOS 2.4の差異の整理

TwinOSをバージョンアップするために，Linux 2.4とTwinOS 2.4の差異を整理する．差異は，以下の2つに分けられる．

(1) Linux 2.4のソースコードを変更

Linux 2.4のソースコードに変更を加え，新たな動作を追加するためのソースコードによる差異である．例えば，メモリマップ取得部分のソースコードの変更が挙げられる．

(2) 新たにソースコードを追加

Linux 2.4に新たに追加されたTwinOS用のソースコードによる差異である．例えば，共存OSをメモリ上に展開するシステムコールが挙げられる．

これらの差異を明確にするため，Linux 2.4とTwinOS 2.4の差異をとり，表にまとめた．変更されたファイルの一覧を表4.1に，追加されたファイルの一覧を表4.2を示し，以下で説明する．

先行OSは変更されたファイルは22個，追加されたファイルは22個であり合計44個であった．共存OSは変更されたファイルは17個，追加されたファイルは8個であり合計25個であった．この結果，TwinOSを5つのモジュールに分割できることが分かった．4.2節で今回分割したモジュールについて説明する．

4.2 TwinOSの機能のモジュール化

4.2.1 TwinOSモジュール

TwinOSの機能毎に5つのモジュールに分割した．以下でTwinOSの各モジュールについて詳細を述べる．

(機能1) メモリ分割機能

メモリを若番アドレスと老番アドレスに分割し，先行OS カーネルをメモリの老番アドレスに展開する機能である．通常のOSの場合，カーネルは若番アドレスに展開される．TwinOSでは，メモリを分割し先行OSを老番アドレスに展開することにより，共存OSをメモリの若番アドレスに置くことが可能となり，同一メモリ上に2つのカーネルを展開することが可能になる．

(機能2) 共存OS メモリ展開機能

メモリの老番アドレスに共存OSのカーネルイメージを展開する機能である．先行OSの走行環境を保存しておき，共存OSカーネルをメモリ上に展開後，保存しておいた走行環境を復元し，先行OSへ復帰する．

(機能3) 共存OS 起動機能

メモリ上に展開された共存OS を起動し，共存状態に移行する機能である．この機能も，共存OS メモリ展開機能と同様に先行OSの環境を保護する必要がある．このため，共存OSの発行する命令に対して監視処理を行う．保護にはCPUのトレースモードを利用し，トレース例外処理において共存OSが次に実行する起動処理の命令を取得し，実行の許可/不許可を判断することで，先行OSを保護する．

(機能4) 入出力機器占有機能

各OS に入出力機器を割当て占有させる機能である．入出力機器の検出の際に各OSが占有する入出力機器のみを検出させることにより，入出力機器を占有させる．また，割り込み/例外処理は，当該割り込みを発生した入出力機器を占有制御しているOSの割り込み処理が呼び出されるように制御する．このため，走行していないOS が占有する入出力機器からの割り込みの場合は，OS を切り替える．

(機能 5) CPU 時分割機能

1 つの CPU を 2 つの OS で利用するために CPU を時分割で割り当てる機能である．一定の周期で OS 切り替えが発生し，OS 切り替えのために OS の走行環境の保存と復元の処理を行う．

4.2.2 TwinOS モジュールの確認

作成した TwinOS モジュールを Linux 2.4 に適用し，TwinOS モジュールを正しく作成できたかどうか確認した．具体的には，TwinOS 2.4 のカーネルイメージと Linux 2.4 に TwinOS モジュールを適用して作成したカーネルイメージが同一であるか比較した．この結果，正しく作成できていることを確認した．

4.3 TwinOS 2.4 の実装における各機能の検討項目

各機能毎の検討項目を示す．

(機能 1) メモリ分割機能

(1) 分割するメモリアドレスがマジックナンバで指定されている

分割するメモリアドレスを 128M と直接指定している．マジックナンバを用いることは，実装上好ましくない．また，分割するメモリアドレスを変更する場合，変更箇所が多く誤りが発生しやすい．

(2) メモリ分割方法が汎用的でない

現実装はメモリアドレスの先頭が 1M のエントリを変更しているが，この実装方法は汎用的ではない．

(3) メモリマップをはじめに 3 分割し，後で 2 分割に変更している

この部分は実装が困難な部分であり，過去に試行錯誤しながら実装された．こういった経緯から，本来必要とされない冗長な手順をふんでメモリ分割が行われていると思われる．このため，メモリマップ分割の手順が複雑である．

(機能 2) 共存 OS メモリ展開機能

(1) 例外処理を発生させ共存OSのメモリ展開処理を行っている

例外処理を用いて，共存OSのメモリ展開処理を行っているが，例外処理で行う必要があるのか検討する必要がある．

(2) 必要ない記述により不具合が発生する可能性がある

前後の処理の流れを把握せずにソースコードを追加しているため，例外処理の途中にTwinOSのソースコードが入ってしまっている．このため，例えば，除算例外が発生したときTwinOSのソースコードが実行されるため不具合が発生する可能性がある．

(3) TwinOSの機能強化により必要のなくなったソースコードが残っている

以前は，共存OSのメモリ展開時に先行OSを再起動する必要があった．しかし，TwinOSの機能強化により，再起動の必要がなくなった．これにより，必要のなくなったソースコードが残っている．

(4) 実行されることのないソースコードが削除されずに残っている

ソースコードの可読性が下がるため，削除するべきである．

(5) 処理の最後にキーボード割り込みを発生させているが，必要性が分からない

共存OSメモリ展開処理の最後にキーボード割り込みを発生させている．制御の関係上この処理を行っていると思われるが，このソースコードの必要性について検討する必要がある．

(機能 3) 共存OS起動機能

(1) アセンブリ言語で書く必要のない部分がアセンブリ言語で書かれており，冗長で無駄が多い

ほとんどの部分をC言語で書けるにもかかわらず，アセンブリ言語を用いて記述している．これは，アセンブリ言語とC言語のインタフェースの整合を取るのが当時の実装者にとって難しかったためだと思われる．このため，低級な言語によってソースコードが長く，難解となっている．

(機能 4) 入出力機器占有機能

(1) 単純にコメントアウトで入出力機器の読み込みを回避しているため，前後の記述と不整合が起きている

コメントアウトしたことにより，スタックの使用に不整合がありメモリリークの可能性がある．

(2) デバイスのポート指定をマジックナンバで行っている

マジックナンバの使用はやめるべきである．

(機能 5) CPU 時分割機能

(1) アセンブリ言語で書く必要のない部分がアセンブリ言語で書かれており，冗長で無駄が多い

ほとんどの部分をC言語で書けるにもかかわらず，アセンブリ言語を用いて記述している．これは，アセンブリ言語とC言語の整合を取るのが難しかったためだと思われる．このため，低級な言語によってソースコードが長く，難解となっている．

これらについてTwinOS モジュールをLinux 2.6 に適用できるように改変する際に検討し，対処する．

4.4 Linux 2.4とLinux 2.6の差異の整理

バージョンアップを行う際には，Linuxのバージョン間での差異が問題となる．この差異のため，TwinOS 2.4のソースコードを流用したのではTwinOS 2.6は実装できない．以下に，TwinOS 2.6を実装する上で問題となるLinux 2.4 とLinux 2.6 での主なバージョン間差異を示す．

(1) 識別子やファイル名の変更

ファイル名や関数名が変更されている場合がある．この場合，TwinOS 2.4 のTwinOSモジュールを流用できない．

(2) 新たな機能の追加および機能の仕様変更

関数の内部動作が変更されていた場合，既存の関数の使用方法では望んだ結果が得られない可能性がある．

これらの差異は大きいため，5章で作成したTwinOSモジュールのLinux 2.6への適用について述べる際に，各機能毎に関連する差異についてのみ述べる．

これまでに，(機能 1) メモリ分割機能についてのみLinux 2.6 に適用した．

表 4.1 変更されたファイル一覧

通番	先行OS変更ファイル	共存OS変更ファイル
1	/arch/i386/boot/compressed/head.S	/arch/i386/boot/compressed/head.S
2	/arch/i386/boot/compressed/misc.c	/arch/i386/boot/compressed/misc.c
3	/arch/i386/boot/setup.S	/arch/i386/boot/setup.S
4	/arch/i386/kernel/entry.S	/arch/i386/kernel/entry.S
5	/arch/i386/kernel/head.S	/arch/i386/kernel/i8259.c
6	/arch/i386/kernel/i8259.c	/arch/i386/kernel/irq.c
7	/arch/i386/kernel/irq.c	/arch/i386/kernel/Makefile
8	/arch/i386/kernel/Makefile	/arch/i386/kernel/process.c
9	/arch/i386/kernel/process.c	/arch/i386/kernel/time.c
10	/arch/i386/kernel/time.c	/kernel/Makefile
11	/arch/i386/kernel/traps.c	/kernel/panic.c
12	/arch/i386/vmlinux.lds.S	/kernel/sys.c
13	/drivers/char/keyboard.c	/drivers/char/tty_io.c
14	/drivers/char/pc_keyb.c	/include/asm-i386/io.h
15	/include/asm-i386/dma.h	/include/asm-i386/unistd.h
16	/include/asm-i386/e820.h	/init/main.c
17	/include/asm-i386/io.h	/Makefile
18	/include/asm-i386/page_offset.h	
19	/include/asm-i386/unistd.h	
20	/init/main.c	
21	/kernel/sched.c	
22	/Makefile	

表 4.2 追加されたファイル一覧

通番	先行OS追加ファイル	共存OS追加ファイル
1	/arch/i386/kerne/netos.c	/arch/i386/kernel/dualos.c
2	/arch/i386/kernel/netos_dump.c	/inlucde/linux.dualos.h
3	/arch/i386/kernel/netos_irq.h	/include/linux/vnic/vnicdebug.h
4	/arch/i386/kernel/netos_trace.h	/include/linux/vnic/vnicdebug2.h
5	/arch/i386/kernel/netos_var.h	/include/linux/vnic/vnic_io_test.h
6	/arch/i386/kernel/os2_trace_os1.c	/include/linux/vnic/vnic_io_type.h
7	/arch/i386/kernel/vnic.c	/include/linux/vnic/vnic_os2.h
8	/arch/i386/kernel/vnic_core.h	/kernel/vnic/debug.c
9	/arch/i386/kernel/vnic_filter.h	
10	/arch/i386/kernel/vnic_io.h	
11	/arch/i386/kernel/vnic_nic.h	
12	/arch/i386/kernel/vnic_ops.c	
13	/arch/i386/kernel/vnic_show.c	
14	/include/linux/netos.h	
15	/include/linux/netos_dump.h	
16	/include/linux/os2_trace_os1.h	
17	/include/linux/vnic/packetlog.h	
18	/include/linux/vnic/vnic_io_type.h	
19	/include/linux/vnic/vnic_ops.h	
20	/include/linux/vnic/vnic_ops2.h	
21	/include/linux/vnic/vnic_OS2.h	
22	/include/linux/vnic/vnic_show.h	

第 5 章

TwinOSのモジュールの改変

5.1 メモリ分割機能

5.1.1 Linux のメモリ管理

Linux は起動時に BIOS のサービスを呼び出してメモリマップを取得し，搭載メモリ量とこの内で利用可能な物理メモリの領域を得る．TwinOS では，この BIOS が返却するメモリマップを加工し，各 OS 毎に変化させることによってそれぞれが利用する物理メモリ領域を分離する．BIOS が返却するメモリマップの構造を 図5.1 に示し，以下で説明する．

メモリマップはページアドレスを表す 8 バイト，領域の長さを表す 8 バイト，領域のタイプ表す 4 バイトの 3 つで構成される．ページアドレスとは領域の先頭アドレスの情報のことである．領域の長さとは領域の大きさのことである．領域のタイプは 1 から 4 までの数字で表され，1 から使用可能なメモリ (usable)，予約領域 (reserved)，コンピュータ内部の各パーツの電力の管理 (ACPI data) 用メモリ，OS の休止状態からの復旧 (ACPI NVS) 用メモリを表す．BIOS が返却したメモリマップを 図5.2 に示し，以下で説明する．

はじめのエントリは usable であり OS が使用可能である．この領域は主に OS 起動時に使用される．2 番目のエントリは reserved である．3 番目のエントリは usable であり，OS が使用可能である領域である．4 番目のエントリは ACPI data，5 番目の領域は ACPI NVS である．

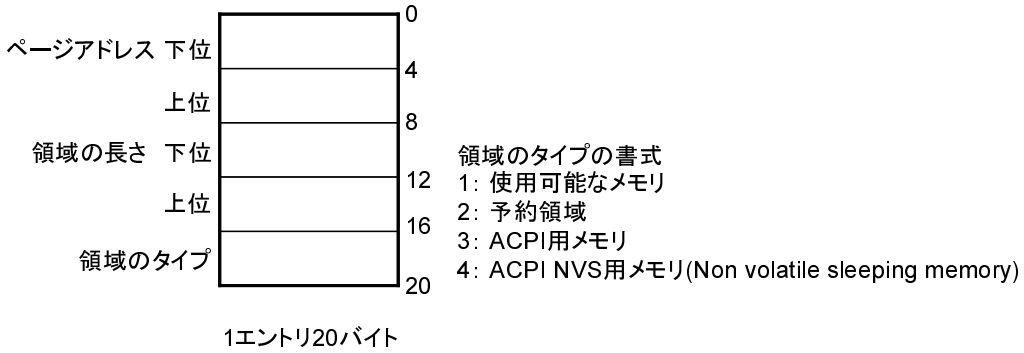

図 5.1 メモリマップの構造

5.1.2 TwinOS 2.4 のメモリ分割機能

TwinOSでは，BIOSの返却するメモリマップを書き変えることでメモリを若番アドレスと老番アドレスに分割し，１つのOSからは一方のみを利用する．具体的には，先行OSはメモリの老番アドレスに，共存OSはメモリの若番アドレスに展開する．しかし，Linuxは通常カーネルプログラムを1MB(Linux 2.6では4MB)以降に配置する．このためTwinOSでは，先行OSカーネルをメモリの老番アドレスに展開するように変更する必要がある．本項ではTwinOS 2.4の実装方法について述べる．

メモリマップの変更

メモリの分割利用を実現するために，メモリマップ取得部分のソースコードを変更する必要がある．Linux 2.4では，arch/i386/boot/setup.Sにアセンブリ言語でメモリマップを取得するソースコードが記述されている．TwinOS 2.4では，このファイルに47行のソースコードを追加することにより，メモリマップをTwinOS用に変更している．TwinOS 2.4 の実装方法を 図5.3 に示す．

このソースコードには以下の問題がある．

(1) 分割するメモリマップエントリのメモリアドレスが1Mに決め打ちになっている

図5.4 の (a) のようなメモリマップがBIOSから返却される．TwinOS 2.4では，メモリマップの1Mからのusableのエントリを変更する．しかし，この実装方法では，BIOSから返却されるメモリマップが変わった場合に対応で

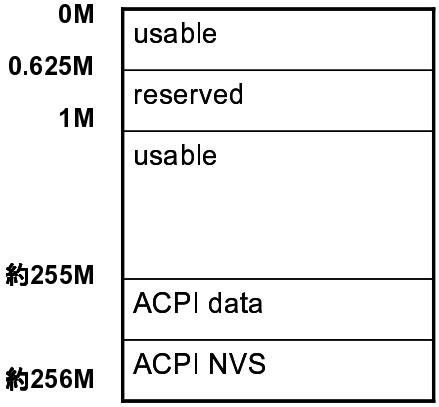

図 5.2 BIOS が返却したメモリマップ

きない．本来は,「使用可能であり，かつページアドレスが 128M 未満であり，ページアドレスの終了アドレスが 128M 以降のメモリマップエントリを変更する」といった方法をとるべきである．

(2) メモリマップエントリを 3 つに分割する．その後，2 つの領域に分割する．

TwinOS 2.4 の実装では 図 5.4 に示すように，本来不要な (b) の状態を経てメモリ分割を実装している．しかし，この操作について，本来は (a) を直接 (c) に変更可能なはずである．

先行 OS 老番アドレス展開方法

カーネルのメモリ展開位置を変更するには，カーネル展開ルーチンを変更する必要がある．TwinOS 2.4 では，展開場所の指定に複数行を変更を必要とし，かつマジックナンバを使用し実装している．

しかし，4.3 節で示した通り，この実装方法には検討すべき点がある．このため，TwinOS のバージョンアップをする際には，実装方法を変更する必要がある．

5.1.3 TwinOS 2.6 のメモリ分割機能

本項では TwinOS 2.6 の実装方法について述べる．

メモリマップの変更

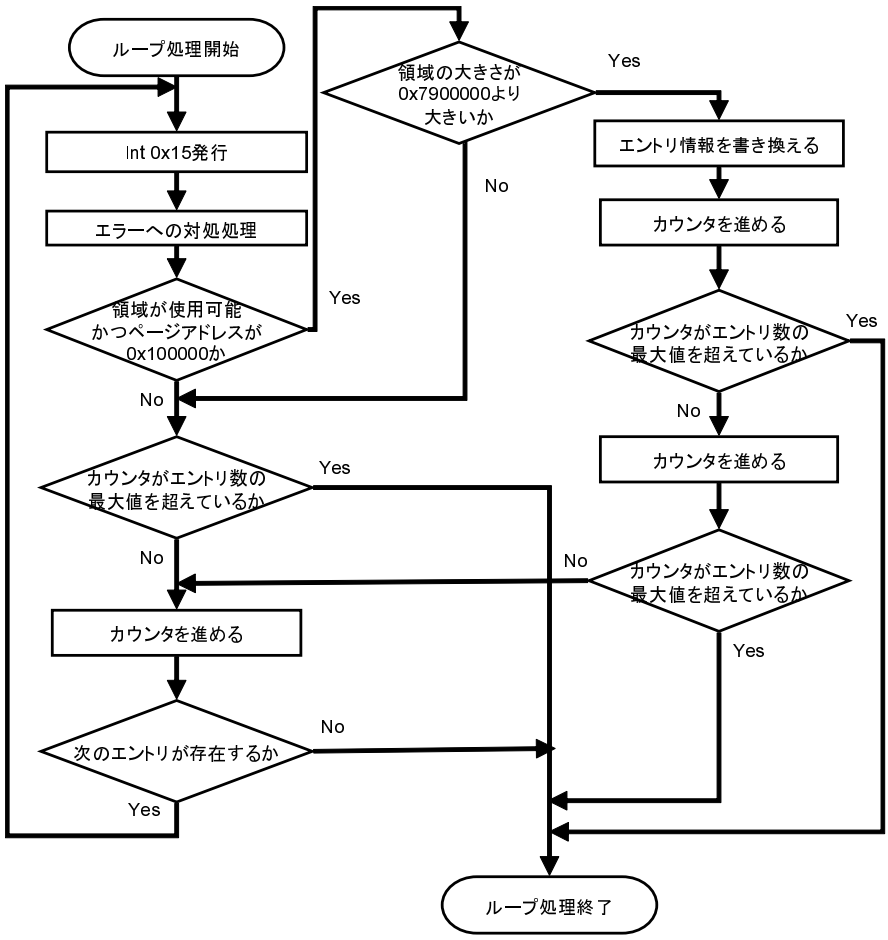

図 5.3 TwinOS 2.4 のメモリマップ変更フローチャート

今回の実装にあたり，arch/x86/boot/memory.c に 23 行のソースコードを追加した．分割するメモリマップエントリの判定方法を汎用的にすることで，カーネルへの依存性を減少させた．また，TwinOS 2.4 の問題点に対処するため，新たに TwinOS 2.6 用メモリマップ変更方法を考察した．TwinOS 2.6 でのメモリマップ変更フローチャートを 図 5.5 に示す.

図 5.3 に比べ処理を簡略化した．マクロを用いることにより変更箇所を最小限に抑え，マジックナンバを使用しないようにした．また，TwinOS 2.4 と比べ分割するメモリマップエントリの判定方法が変わっている．この方法により，メモリアドレスの開始アドレスが 1MB でない場合でも正しく分割できる．また，Linux 2.6 が BIOS から取得したメモリマップは Linux 2.4 のそれと異なる．本来

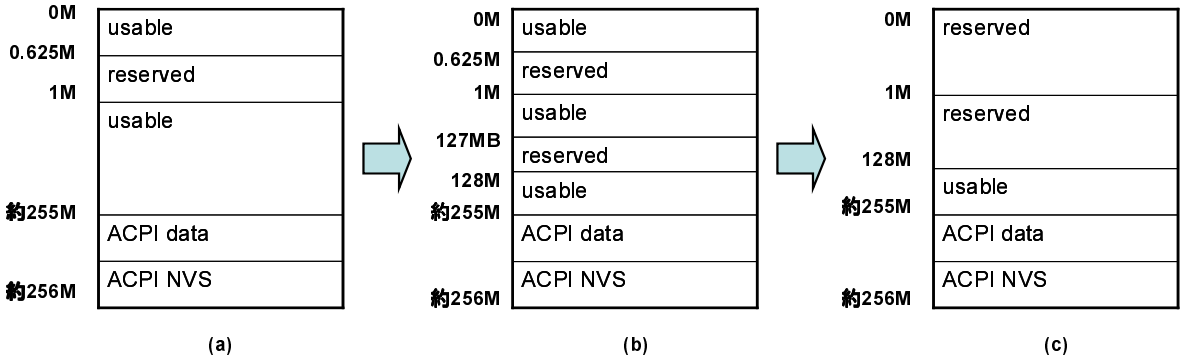

図 5.4 TwinOS 2.4 のメモリマップエントリの分割

BIOS からの返却は同一であるはずであるが，BIOS とのインターフェース部分において何らかの変換が加えられている可能性がある．しかしながら，今回の汎用的な方式の採用により，違いを吸収することができている．このため，今後の Linux のバージョンアップにも対応可能であると思われる．Linux 2.6 のメモリマップを 図5.6 に示す．また，今回実装したソースコードにより変更されたメモリマップを 図5.7 に示す．

この実装では最初のエントリが使用可能のままである．これは，1 番目のメモリ領域は OS 起動時に使用されるため，予約領域にできないからである．

先行 OS 老番アドレス展開方法

TwinOS 2.6 では，先行 OS を展開するアドレスを指定するためのマクロを 1 つ定義することにより，マジックナンバを使用しない実装に変更した．この実装方法により，このマクロの値を変更するだけでカーネルの展開位置を変更できるようなった．

5.1.4 動作確認

メモリマップの変更と先行 OS の老番アドレス展開の各々についての動作確認を行った．以下に詳細を述べる．

メモリマップの確認

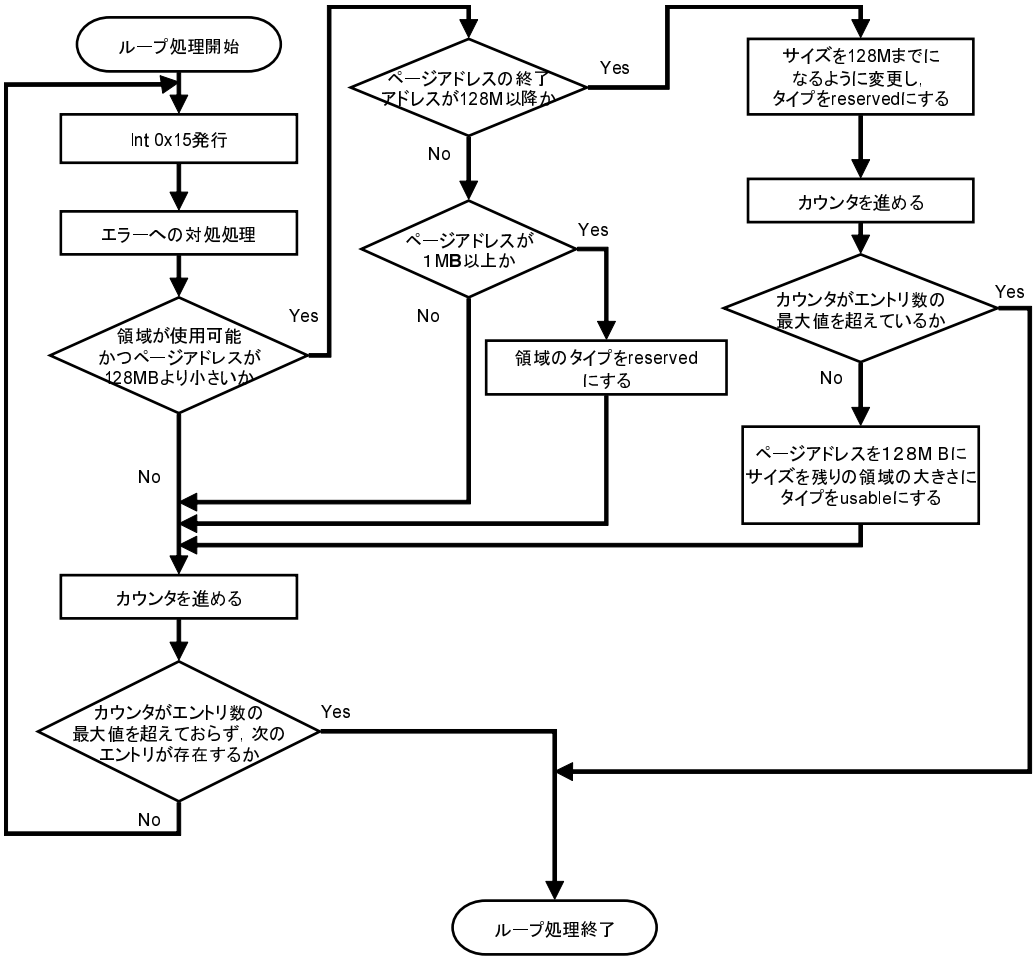

図 5.5 TwinOS 2.6 のメモリマップ変更フローチャート

ブートアップメッセージには，メモリマップの情報も記されている．このため，先行 OS 起動後に dmesg コマンドを実行することにより，メモリマップをメモリの半分で分割できていることを確認できた．

先行 OS の展開位置の確認

Linux 2.6 にソースコードを追加し，メモリ分割機能を実装した．確認のため静的変数の物理アドレスをコンソールに表示するシステムコールをカーネルに追加した．静的変数はカーネルのデータ部に保存されるため，静的変数の保存されているアドレスを調べることでカーネルのソースコードが大体どこにあるか分かる．カーネル起動後このシステムコールを実行したところ．約 131M とい

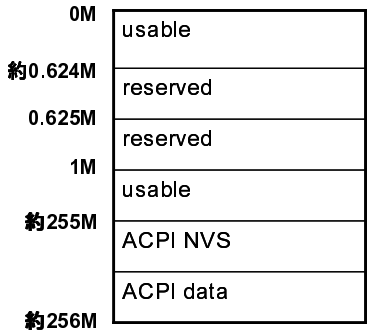

図 5.6 Linux 2.6 のメモリマップ

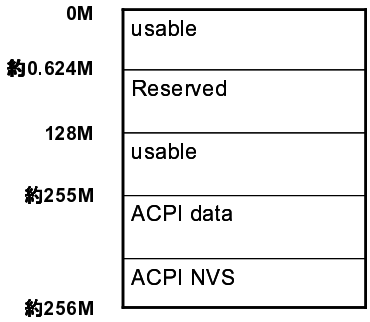

図 5.7 TwinOS 2.6 のメモリマップエントリの分割

う値が表示された．つまり，先行 OS をメモリ老番に展開できたといえる．

第 6 章

おわりに

本論文では，TwinOSのバージョンアップ手法を検討し，実装した結果を述べた．

まず，現在のTwinOSの問題点を述べ，TwinOSをバージョンアップする目的と要求について述べた．次に，TwinOSのバージョンアップへの要求を満たすTwinOSのバージョンアップ手順を検討した．

検討したバージョンアップ手順に基づきTwinOS 2.6に対応したメモリ分割機能のモジュールを実装した．まず，Linux 2.4とTwinOS 2.4の差異を整理し，まとめた．次に，TwinOSの機能毎に差異を5つのTwinOSモジュールに分割した．また，分割した各機能毎に検討項目を示した．その後，TwinOSモジュールの確認を行い，正しくモジュールが作成できていることを確認した．そして，Linux 2.4とLinux 2.6の差異を整理し，バージョンアップ時に問題となるバージョン間差異について示した．実装の完了したメモリ分割機能について，TwinOSモジュールをLinux 2.6に適用できるように改変する方法について述べた．

残された課題として，メモリ分割機能以外のモジュールをLinux 2.6に対応させることとLinux 2.4からLinux 2.6へのTwinOSの適用以外への適用可能性を探ることがある．

謝辞

本研究を進めるにあたり，懇切丁寧なご指導をして頂きました乃村能成准教授に心より感謝の意を表します．また，数々のご助言を頂きました谷口秀夫教授，および田端利宏准教授に厚く御礼申し上げます．最後に，日頃の研究活動において，お世話になりました研究室の皆様ならびに本研究を行うにあたり，経済的，精神的な支えとなった家族に感謝いたします．

参考文献

[1] 田渕正樹，伊藤健一，乃村能成，谷口秀夫，“二つの Linux を共存走行させる機能の設計と評価，” 電子情報通信学会論文誌 (D-I)，vol.J88-D-I，no．2，pp．251-262 (2005．02).