

VxWorks 6オペレーティングシステムへの移行

ウインドリバー・プロフェッショナル・サービス 北米コンサルティング

ひとつのデバイス・オペレーティングシステムから別のオペレーティングシステムへ

移行を行うかどうかの評価にあたり、考慮すべき問題は数多くあります。

移行の試み自体に伴う課題も非常に大きく、デバイスメーカーは移行に関わるあらゆる選択肢をもれなく検討する必要があります。

本資料では、サードパーティデバイスのオペレーティングシステムからウインドリバーのVxWorksプラットフォームへの

移行を想定した場合に、考慮すべき事項について詳しく述べます。

目次

1 はじめに	1
2 ビルド環境	1
3 コードの再利用	1
4 アプリケーションの移植	2
5 おわりに	4

1 はじめに

あるオペレーティングシステムから別のオペレーティングシステムへの移行をデバイスメーカーに決断させる要因は数多くあります。一部のメーカーは、共通開発環境を作り出すことを含め、新たな戦略を確立することによって、コードの共有化、複雑度の緩和、生産性の向上を実現し、利益を大幅に増大させることを望んでいます。また別のメーカーは、市場圧力に対応するためのコスト削減、従来のベンダのオペレーティングシステム(OS)またはリアルタイムオペレーティングシステム(RTOS)における価格体制の変更、あるいはその他の構成コンポーネントの価格変更を必要としています。また、要求の厳しい市場における競争力維持の一助として、より高度なOS機能(リアルタイムプロセスなど)を模索しているデバイスメーカーもあります。

コスト面と技術面の検討に加えて、移行プロセス自体も重要な要素です。移行を実行するか、また、実行するのであればどのOSを選択するかという最終的な決断を下す前に、メーカーは、移行の実現性と容易さに重点を置いて検討する必要があります。本資料は、ウインドリバーのVxWorks 6 RTOSへ移行するための重要な決定事項と問題について、詳しく検討します。

2 ビルド環境

VxWorks 6 リアルタイムOSの最も重要な開発コンポーネントは、Workbenchツール環境とそのプロジェクト指向の開発環境です。

そのコア部分で、ウインドリバーWorkbenchプロジェクトのビルドはMakefileによって動作します。このMakefileは、BSPとアプリケーションを作成するためにGNUやウインドリバー・コンパイラ・ツールチェーンを利用しています。Makefileはフォーマット化されたファイルで、makeユーティリティと連携したプロジェクトの作成と管理が必要です。ウインドリバーWorkbenchは、「未管理」、「柔軟管理」の作成を含め、標準的なMakefileベースのプロジェクトを新しいWorkbenchベースのプロジェクトにインポートする機能を提供することによって、作業時間を短縮するよう設計されています。VxWorks 6には、makeプロセス時にツールチェーンを移植する機能も含まれています。

さらに、ウインドリバーWorkbenchに組み込まれたプロジェクトインポート機能を使用すれば、既存のTornadoプロジェクトをWorkbench対応プロジェクトに容易に移植できます。Workbenchは、従来の「コマンドライン」作成によって開発されたMakefileベースのプロジェクトをインポートする機能も備えています。

3 コードの再利用

ボード・サポート・パッケージ

VxWorksを指定ボード上で実行するには、ボード・サポート・パッケージ(BSP)が必要です。BSPは、アドレスマップやデバイスI/O(タイマ、シリアル、Ethernet等)へのアクセスなど、ボード固有ハードウェアの抽象化レイヤとして機能します。また、BSPはブートプロセス中にハードウェアの初期化も行います。

通常、低水準ブートコードはVxWorksへの移行時に再利用できます。また、スタートアップコードは、低水準CPUレジスタ(特殊用途用CPUレジスタ、キャッシュ、MMUなど)、SDRAMコントローラ(存在する場合)、割り込みコントローラ、およびこれらと同様の要素を初期化します。一般に、この初期化コードはアセンブリ言語で書かれており、対象となるOSにはほとんど依存しません。

BSPは、VxWorksが必要とするいくつかの機能を実装しています。これらのルーチンは、公表されたAPIに従ってハードウェアへのアクセスを実現しています。たとえば、BSPは sysClockRateSet()という関数を実装して、システムクロック周波数をセットします。ボードの機能セット

により、VxWorksを正しくサポートするにはこれらの関数が10~20程度必要です。

BSPによってVxWorksに提供されるルーチンは、単純なものから複雑なものまでさまざまです。たとえば、NVRAMの読み取りや書き込みを行うAPIを提供するには、BSPが必要です。単純なケースではBSPはNVRAMのストレージ機能を持たず、単にエラー結果を返すだけで他は何の動作も行いません。もう少し複雑な例では、I2Cバス上に置かれたシリアルEEPROMデバイスを介してNVRAMストレージを実装するBSPがあります。このようなBSPは、NVRAMデバイスをサポートするアルゴリズムを提供するほか、I2Cプロトコルを扱うルーチンを実装する必要があります。

デバイス・ドライバ

ほとんどのアプリケーションレベル/Oは、ANSIのライブラリ・ファイル・モデルに基づいています(open()、close()、read()、write()、ioctl())。デバイス・ドライバが異なれば、このモデルのサポート方法も異なります。

低水準シリアル・ドライバは、割り込みハンドラ、コールバックインストール機能、OSとの間のキャラクタ送受信機能を提供します。シリアル・デバイス・ドライバは、VxWorks独自のドライバ・モデルに従っています。ここではほとんどの共通サポーティングコードがVxWorksによって提供され、ドライバはシリアル・デバイス固有の機能だけを実装します。

既存のシリアル・ドライバは、少なくとも部分的な再利用が可能です。たとえば、割り込みハンドラとデバイス初期化は、OSが異なっても共通化できる傾向にあります。シリアル・ドライバの一部は、VxWorksのシリアル/Oドライバ・モデルに合わせて変更する必要があります。代表的な変更プロセスはウインドリバーのテクニカルドキュメントに詳しく説明されていますので、ここでは省略します。

同様にネットワーク・デバイス・ドライバも特定のモデルに従っており、その一部はBSDネットワーク・ドライバ・モデルに基づいています。シリアル・ドライバ同様、既存のネットワーク・ドライバ・コードは多くの場合再利用できますが、一般に、VxWorksが提供するネットワーク・ドライバ・モデルに適合させるには、多少の変更が必要になります。

カスタム・デバイス用のドライバは、多くの場合は変更なしで使用できます。たとえば、あるプロジェクトがカスタムインターフェイスを介してカスタム・タッチパッド・タブレットを使用している場合、そのインターフェイスをVxWorksに移行するために、ほとんど、あるいは全く変更の必要はありません(この種のデバイスは既存のVxWorks入力デバイスのフレームワークに組み込むことができますが、その必要はありません)。

実際、「標準」ドライバ(シリアル・ドライバなど)であっても、ある機能を持たせるために、確立されたVxWorksドライバ・モデルを使用する必要はありません。VxWorksはこのようなデバイス・ドライバを認識しませんが、それでも、これらのドライバと直接連携するアプリケーション・コードは正常に機能します。

VxWorks 6ではユーザーアドレス空間でアプリケーション・コードを実行できます。ただし直接ハードウェアをアドレス指定するソフトウェアは、カーネル(スーパーバイザ)空間で実行しなければなりません。6.x以前のVxWorksバージョンでは、すべてのコードがカーネル空間で実行されていました。

4 アプリケーションの移植

VxWorksに移植されるほとんどのコードはアプリケーション・コードです。開発者は、中でも、移植を必要とするコードの量および既存コードのデザインや実装といった要素に基づいて移植方法を選ぶことを求めます。ここでは、OS適合レイヤ、POSIX、ネイティブVxWorksボートの3種類の移植オプションについて検討します。

OS適合レイヤ

VxWorks適合レイヤは大規模なコードを移植する場合の選択肢の1つであり、既存コードの大部分を変更しないうまま実行することができます。アプリケーションは、引き続き移植前と同じOS呼び出しを行います。元のOSへのOS呼び出しは、適合レイヤによってVxWorksのネイティブ呼び出しに変換されます。

この移植方法は、VxWorksと元のOSのAPIがほとんど同じであることを前提としています。ある種のOSはヘビーウェイト・プロセスモデルを使用しますが、VxWorksはこの種のモデルを完全にはサポートしていません(たとえばfork()やexec()はサポートしていません)。このような場合は、アプリケーション・コードを変更する必要があります。あるいは、その機能の重要性を再考することもできます。変更が大がかりなものである場合は、VxWorksを使い、より適切な方法で実装することができないか、あるいは移植後もその機能が果たして必要なかを検討します。

VxWorksの豊富な機能を使用すれば、相互排除などの目的を、より良い方法で実現することができます。そのためアプリケーション・デザイナーは、移行プロセスの際、VxWorksの高度な機能を利用するかどうかを検討する必要があります。

適合レイヤの利点は、ほとんど、あるいは全く変更を加えずに既存コードベースの大部分を再利用できることです。したがって非常に多くの時間と労力を節約できる可能性が高く、移行プロセスが比較的楽なものとなります。反面、OS呼び出しに多少のオーバーヘッドが必要となるため、性能が低下するという短所があります。また、必ずしもすべてのOS呼び出しが他のOSから直接VxWorksにマップされるわけではなく、ある程度のアプリケーションレベルの変更が必要になります。

POSIX

アプリケーション・コードをVxWorksに移植するためのもう1つの方策は、両方のOSでPOSIXインターフェイスを使うことです。POSIXは、異なるOS間のアプリケーションの移植性を保つ助けとなるUNIX的なAPIを提供するIEEE標準です。VxWorksは、OS移行の助けとなるPOSIXをアプリケーション用に提供します。

VxWorks 6リアルタイム・プロセス・モデルによって提供されるアプリケーション環境の全体的な動きはPOSIX 1003.1標準に近いものですが、VxWorks OSの組み込み特性とリアルタイム特性は維持されています。しかし、POSIX標準と異なり、VxWorksで以下の機能を使うことはできません：

- 仮想メモリ・モデルのオーバーラッピング
- fork()およびexec()によるプロセス生成
- メモリマップドファイル
- ファイルの所有権とアクセス許可

非同期I/	aioPxLib
バッファ操作	bLib
クロック機能	clockLib
ディレクトリ処理	dirLib
環境処理	Cライブラリ
環境情報	sysconf、およびuname
ファイル複製	ioLib (ユーザ・モード用)、iosLib (カーネル用)
ファイル管理	sPxLib、およびioLib
I/O機能	ioLib
算術演算	Cライブラリ
メモリ割り当て	memLib
ネットワーク/ソケットAPI	ネットワーク・ライブラリ
オプション処理	getOpt
POSIXメッセージ・キュー	mqPxLib
POSIXセマフォ	semPxLib
POSIXスレッド	pthreadLib
POSIXタイマ	timerLib
標準I/Oおよび一部のANSI	Cライブラリ
文字列操作	Cライブラリ
ワイド・キャラクタのサポート	Cライブラリ

表1: VxWorks用に提供されるPOSIX適合ライブラリ

ユーザ・モード(リアルタイム・プロセス)でのVxWorksのPOSIXサポートは、カーネル環境よりも高いレベルでPOSIXとの互換性を提供することを目的としています。特にCライブラリのサポートは、かなりのレベルでPOSIXの要求事項を満たしています。カーネル・モードでタスクに使われるさまざまなAPIが、ユーザ・モードのプロセスにも使われます(kill()、exit()など)。VxWorksのユーザ・モード・アプリケーション環境はPOSIX 13(IEEE標準1003.13)で記述されたリアルタイム・コントローラ・システム・プロファイル(PSE52)に似たもので、これはPOSIX 1(IEEE標準1003.1)を基本としています。

VxWorks用に提供されるPOSIX適合ライブラリには、表1のものが含まれています。

ネイティブVxWorksポート

アプリケーション開発者には、コードにわずかな変更を加えたり変更なしで使用したりするのではなく、VxWorksのネイティブ呼び出しを行うようにソースコードを変更するという選択肢もあります。この場合、アプリケーション・ソフトウェアは効率が向上し、VxWorksが提供するすべての機能を使用できるという利点があるほか、適合レイヤによるオーバーヘッドや異なるOS間での機能セットのミスマッチを回避することができます。比較的小さいアプリケーション、あるいはVxWorksとの共通点が多いOSからの移植でコード変更がわずかで済むようなアプリケーションの場合、VxWorksのネイティブ呼び出しを行うためのソースコード変更が最も適しています。

VxWorksと共通点の多いOSからVxWorksへの移行は、旧来のOS呼び出し、パラメータ、データタイプをVxWorksのネイティブ定義に変更する、一連の検索/置換セッション程度の作業で容易に行うことができます。しかし通常、移行を完了させるにはこれら以外の編集も必要になります。

VxWorks 6は、ユーザレベル・アプリケーションを保護モードで実行するという新機能を備えています。このモデルでは、アプリケーション・コードはCPUのユーザ・モードで実行されますが、VxWorksのカーネルは引き続きスーパーバイザ・モードで実行されます(開発者は任意の量のアプリケーション・コードをカーネル空間に含めるという選択ができますので、旧バージョンのVxWorksからの移行が容易になります)。

VxWorks 6では、リアルタイム・プロセス(RTP)の概念も導入されています。RTPはユーザ空間で実行されますが、他のRTPによってコードやデータが破壊される恐れがないよう、完全に保護されています。RTPは1個または複数のVxWorksタスクで構成されており、これは、他のOSにおけるヘビーウェイトプロセスが1個または複数のスレッドを持つ場合の概念に似ています。

VxWorks RTPはメモリ・プロテクトされたコードを他のOSから移植するメカニズムを備えています。VxWorksの決定性リアルタイム・スケジューラを利用することができます。従来のVxWorksアプリケーションは従前通りカーネル空間で実行することができ、開発者は、時間の許す限りRTPモデルを利用することができます。

移行作業の見積り

周知のRTOSやデバイスOSからウインドリバーVxWorksへの移行をお考えの場合は、見積りをご依頼ください。弊社のプロフェッショナル・サービス・チームが、そのデバイス・ソフトウェアの移行サービスに関する広範な経験に基づき、移行に必要な時間とリソースの見積りを行います。

移行プロジェクトは1つごとに内容が異なるので個別に見積りを行う必要がありますが、VxWorksと他のOSとの共通性は、プロジェクトの長さを見積もる際の有効な判断材料となります。表2のガイドラインは、ビルド環境の移植にどの程度の期間が必要かを示したものです。評価基準は以下の通りです。

評価基準:

容易 — 数週間単位

中間 — 数週間～数カ月単位

複雑 — 数カ月単位

PORT TO PORT FROM	VxWorks 5.4	VxWorks 5.5	VxWorks 6.1	VxWorks 6.2	VxWorks 6.3
Green Hills	中間	中間	中間	中間	中間
QNX	中間	中間	中間	中間	中間
MontaVista	複雑	複雑	中間	中間	中間
ユーザ作成のLinux	複雑	複雑	中間	中間	中間
カスタムOS	ウインドリバーに 連絡	ウインドリバーに 連絡	ウインドリバーに 連絡	ウインドリバーに 連絡	ウインドリバーに 連絡

表2:ビルド環境移植のガイドライン

5 おわりに

デバイスOS移行の決定は、慎重なリスク評価を必要とする複雑な作業です。しかし、広く使用されている実績あるOSと、経験豊富なプロフェッショナル・サービス・チームによるサポートを提供する安定したベンダを選択すれば、リスクを軽減することができます。さまざまな移行オプションを使用した柔軟な技術が、期限内、予算内でのプロジェクト目標の実現と、デバイス・ソフトウェアへの投資に見合った利益の獲得を支援します。

ウインドリバーはスマートデバイス搭載ソフトウェアの最適化(DSO)をワールドワイドに提供するリーディングカンパニーです。企業がスマートデバイスに搭載するソフトウェアを、品質および信頼性のさらなる向上を実現しつつ、リーズナブルなコストで開発することを可能にし、早期にマーケットへ投入することを支援します。

WIND RIVER ウインドリバー株式会社

東京本社 〒150-0012 東京都渋谷区広尾1-1-39 恵比寿プライムスクエアタワー TEL.03-5778-6001(代表) FAX.03-5778-6002
大阪営業所 〒532-0011 大阪市淀川区西中島7-5-25 新大阪ドイビル TEL.06-6100-5760(代表) FAX.06-6100-5761
E-mail:info-jp@windriver.com http://www.windriver.co.jp

登録商標: Wind River, Wind Riverロゴ, Tornado, VxWorksは、Wind River Systems, Inc.の登録商標または商標です。記載されているすべての名称は、各社の登録商標、商標またはサービスマークです。