マルチインタフェースOSにおけるクロスプロセス間通信

6P-4

遠城秀和
NTTデータ通信(株)

1 はじめに

新規にOSを作成する場合、そのOS(実OS)のインタフェース(実OS界面)以外に既存OSのインタフェースを仮想インタフェース(仮想OS界面)として提供すると既存ソフトウェアが容易に移植、走行可能にでき既存ソフトウェア資産の活用が図れソフトウェア開発効率の向上に有効である[1]。

筆者はマルチマイクロプロセッサをサポートするトランザクション処理(Tr処理)用OSであるDIROS (DIstoributed Real-time Operating System) [2]を新規に開発している。DIROSは、UNIX[1]の豊富なソフトウェア資産を流用することを目的に、UNIXインタフェース(UNIX界面)のシミュレート機能を実現している。

本論文ではDIROSインタフェース(DIROS界面)とUNIX界面の2つのインタフェースを提供するDIROS上において、それぞれのインタフェースを使った2つのプロセス相互で行うプロセス間通信の実現方法について報告する。

2 複数インタフェースのクロスプロセス間通信

仮想OS界面をシミュレートする主な目的は、既存OSインタフェースのみを使ったソフトウェア資産の活用である。さらに、次に実OS機能を活用するため実OSプロセスと協調するソフトウェア機能拡張が要求される。プロセス相互の協調にはプロセス相互のデータ授受機能及び同期機能が必要である。

具体的なプロセス相互の協調方法には以下の2通りが考えられる。

1. ファイル経由によるデータ授受の実現
2. メッセージ通信などのIPC (Inter-Process Communication)によるデータ授受と同期の実現

同一ファイルが実OS界面と仮想OS界面の両方からアクセス可能ならばデータ授受がファイル経由ででき、データ授受のみのプロセス間協調はファイル経由で実現できる。しかし、Tr処理のように複数プロセスが走行しながら協調する場合、速度や効率の向上のためIPC同期機能が必要となる。DIROSのようなTr処理用OSでは、実OS界面を使ったプロセスと仮想OS界面を使ったプロセスとの間でのプロセス間通信(クロスプロセス間通信)が必要となる。

3 UNIXプロセス間通信シミュレート方法

DIROSでは実OS界面であるDIROS界面を使用するプロセスはTr処理を、仮想OS界面であるUNIX界面を使用するプロセスはTSS処理を対象としている。Tr処理とTSS処理は性質が大きく異なるため、DIROS界面を使用するDIROSプロセスとUNIX界面を使用するUNIXプロセスを異なるプロセスとして扱っている。このため、クロスプロセス間通信はインタフェースをまたがるプロセス間通信となる。

DIROSではUNIX界面のシミュレートをカーネル内で実現している[3]。UNIX界面の基本的シミュレートは、引数の個数、型、順序等のインタフェース形式の変換を行い、その後DIROSカーネル内の内部関数を呼び出し実現している(図1)。

しかし、UNIXのプロセス間通信機能[4]はDIROSに比べ高機能であり、DIROSのプロセス間通信機能の内部関数で対応できない。そのため、UNIXのプロセス間通信機能を実現する内部関数を独立に用意しシミュレートしている(図2)。このため、UNIXプロセス間、DIROSプロセス間ではプロセス間通信が可能であるが、DIROSプロセスとUNIXプロセスとのクロスプロセス間通信は不可能であった。

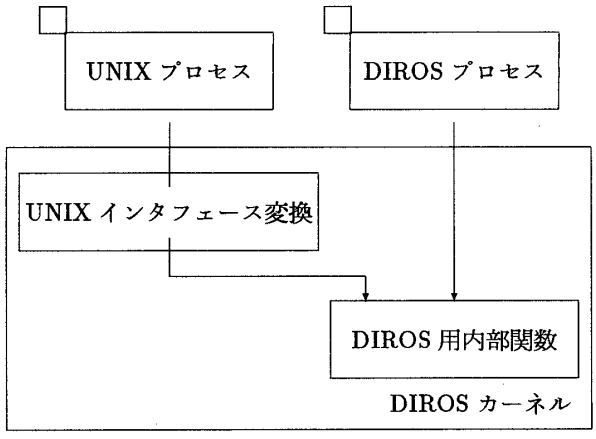

図 1: 基本的シミュレート方法

Mutual Inter-Process Communication
on Multi-Interface Operating System
Hidekazu ENJO
NTT DATA COMMUNICATIONS SYSTEMS CORPORATION

[1]UNIXはAT&Tのベル研究所が開発したOSです。

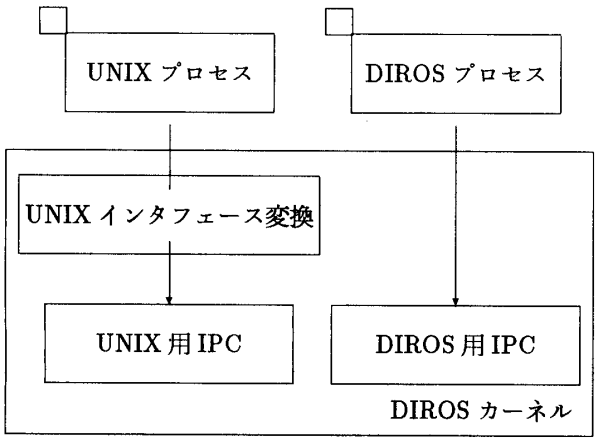

図 2: UNIX プロセス間通信シミュレート方法

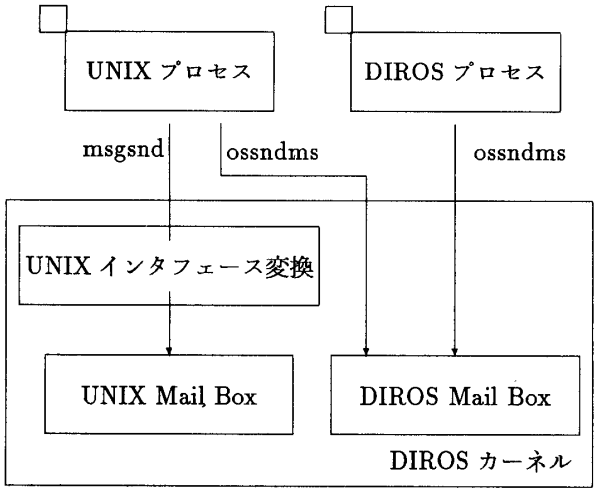

図 3: 複数界面共存方式

4 クロスプロセス間通信の実現法

クロスプロセス間通信を実現する方法としては、以下に示す 2 通りがある。

1. 複数界面共存方式

   1 つのプロセス中で複数インタフェースのシステムコールを混在使用可能とする方法 (例えば UNIX プロセスから DIROS システムコール (ossndms) を使い DIROS のメールボックスに送信する (図3)。)

2. システムコール限定変換方式

   高機能なインタフェースの内、相手内部関数の機能に限定して使用可能とし、相手内部関数にない機能はエラーとするシミュレート方法 (例えば UNIX プロセスが UNIX システムコール (msgsnd) を使い DIROS のメールボックスに送信する (図4)。)

複数界面共存方式は、 Tr 処理用と TSS 処理用のように異なるインタフェース体系が混在すると、矛盾の発生や無意味な組み合わせが生じる。例えば、メールボックス指定の ID は整数値であり同じ値でも UNIX 界面と DIROS 界面では異なるメールボックスの指定になり混同しやすい。

システムコール限定変換方式は、インタフェース変換時にエラー判断が必要なため複数界面共存方式に比べ実現は複雑である。しかし、各プロセス内のインタフェースは一体系に閉じ、複数インタフェース共存方式に比べ使用誤りは発生しにくい。

DIROS では、使用誤りの発生が少なくなるシステムコール限定変換方式を採用した。具体的には UNIX のメールボックス取得システムコール (msgget) の引数フラグを拡張し、 DIROS メールボックス指定 /UNIX メールボックス指定のフラグビットを設けた。DIROS のメッセージ機能に key 付きメッセージ機能がないため key 指定の操作は全てエラーとした。

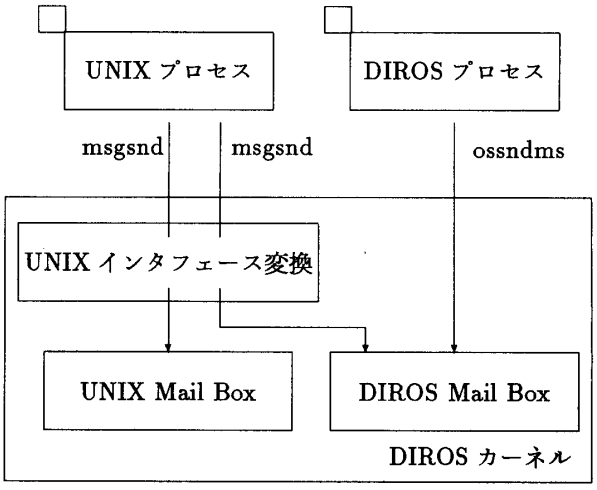

図 4: システムコール限定変換方式

5 まとめ

2 つのインタフェースを提供する DIROS 上においてそれぞれのインタフェースを使った 2 つのプロセス相互で行うプロセス間通信の実現方法について述べた。

参考文献

[1] Richard F. Rashid: "Threads of a New System", UNIX REVIEW, Vol.4, No.8, pp.37-49, 1986.

[2] 谷口: "OS 機能の分散化を可能とする OS 構成法", 信学論, Vol.J72-D-I, No.3, pp.168-174, 1989.

[3] 遠城: "異種 OS 上における UNIX インタフェースの実現法", 情処第 33 回全大, 2V-2, 1986.

[4] SYSTEM V/68 User's Manual, Motorola Inc., 1983.