汎用ＯＳにおけるマルチプロセスサポート方式の提案

5Ｈ－1

池ヶ谷直子　細内昌明　新井利明　木下俊之
（株）日立製作所システム開発研究所

１．はじめに

ワークステーション，パソコンの処理能力向上およびネットワークの利用拡大にともない，価格対性能比の格段の向上を求めるユーザは，異種のハードウェアやソフトウェアを自由に組み合せたオープンシステムを構築するようになった．一方，長年メインフレームを中核に据えた集中処理による定型業務が主流だったことから，メインフレーム上には培われてきたソフトウェア資産が豊富にある．オープンシステムが大規模システムに適用されるにともない，相互運用性を向上させるため，メインフレーム上のソフトウェア資産をオープンシステムから利用する技術が必須である．そこで，汎用ＯＳ上にオープンシステム用基盤ＯＳの標準仕様であるPOSIXインタフェースを設ける方式を提案する．本方式は，オープンシステム上のＯＳに依存した関数を汎用ＯＳ上でサポートし，メインフレームとオープンシステムの連携処理を行うアプリケーションを，機種を意識せずに記述できるようにする．本発表では，特にマルチプロセス用関数を実現するための新機能を報告する．

２．POSIXインタフェース実現方式の課題

汎用ＯＳは，メインフレームのアーキテクチャに依存した独自仕様のＯＳである．豊富なファイルアクセス機能と優れたセキュリティ機能を活かし，多人数による共用の運用形態のもとで性能を考慮しながら発展してきた．これに対してPOSIXインタフェースの基盤ＯＳは，プログラム開発用ＯＳとして使い勝手に優れ，少人数による利用を前提としてきた．両ＯＳは利用目的や発展経緯が異なるため，制御方法にもかなりの違いがある（表１参照）．

POSIXインタフェースでは多くのプロセスを同時に実行できる．fork関数は，新しいプロセスを生成する関数である．新プロセスは生成元プロセスの仮想記憶の内容を継承する．

一方，汎用ＯＳが採用している多重仮想記憶方式では，各ジョブ毎に仮想空間が割り当てられる．このため，ジョブ空間間の記憶保護が容易に達成できる．しかし，汎用ＯＳには，仮想記憶を継承し，かつ独立性を持ったプロセス生成に相当する機能がなかった．

そこで，汎用ＯＳ上でマルチプロセスを実現する

表１．ＯＳ制御方法の比較

NO	項目	汎用ＯＳ	POSIXインタフェース基盤ＯＳ
1	プログラム実行単位	タスク	プロセス
2	仮想記憶レイアウト	ジョブ固有領域と共通領域がある．用途別サブプールに分離される	ユーザ領域には，テキスト，データ，スタックの各連続領域がある
3	プログラム共有	共通領域のみ全ジョブが共有	テキスト領域を複数プロセスが共有
4	空間生成	制御ブロックとシステムタスクを生成後，ウェイト	
5	空間間の連絡	共通領域を使ったポストーウェイト．基本的に他空間とは独立	プロセス間通信機能は豊富
6	ファイル	空間毎にオープン／クローズ	オープン済みファイルを継承し，自動的に共有する
7	ユーザＩＤ	ジョブ単位に単一のユーザＩＤ．ジョブ終了まで不変	複数空間が同一ユーザＩＤを持つ

A Method for Supporting Multiprocess Interface on General Perpose Operating System
Naoko Ikegaya, Masaaki Hosouchi, Toshiaki Arai, Toshiyuki Kinoshita
Systems Development Laboratory, Hitachi, Ltd.

ためには，互いに独立して動作するプログラムが，仮想記憶の内容を共有できる機能が必要である．

3．マルチプロセスサポート機能の提案

（1）プロセス空間生成機能

汎用ＯＳの既存のＴＳＳ空間やジョブ空間とは異なり，親プロセスとテキストを共有するプロセス空間を生成する．プロセス生成後は親プロセスに従属せず，独立対等に動作させる．

（2）メモリ継承機能

仮想記憶の内容を新空間に継承させるとき，汎用ＯＳではクロスメモリコピーを行う．しかし，単純にメモリを複写した場合，領域の大きさに比例してページイン処理等によるＣＰＵオーバヘッドと複写先の領域のメモリオーバヘッドが増大する．また，参照されないまま解放される領域についてもコピーの対象になることから，性能上問題をきたすことが予想される．この問題を解決するメモリ継承機能が必要である．

（3）端末Ｉ／Ｏ機能

汎用ＯＳのＴＳＳ入出力サービスルーチンは，ＴＳＳ空間でのみ動作する．しかし，プロセス空間からの端末Ｉ／Ｏ要求もＴＳＳ端末で受付ける必要がある．

4．プロセス空間マッピング方式

上記マルチプロセスサポート機能により，プロセス空間を汎用ＯＳ上で実現する方法を表２に示す．

セキュリティを考慮すると，プロセスは空間にマッピングするのが良い．性能面では専用空間方式が優るが，汎用ＯＳの既存機能との親和性の点では従来ジョブ空間方式が優っている．そこで，従来空間方式を採用して他制御への影響を抑え，かつメモリ継承機能[1]により性能向上を図ることにした．

複数のプロセス空間を１ユーザが所有するとき，プロセス空間の親子関係およびファイル制御テーブルを一括管理する必要がある．そこで，ファイル及び通信の入出力を一括処理し，プロセスを管理するマスタ空間を設ける（図１参照）．

5．おわりに

メインフレームとオープンシステム連携処理を行うために，プロセスを汎用ＯＳの仮想空間にて実現する方式を提案した．

参考文献

[1]細内，他：汎用ＯＳにおけるマルチプロセスサポート方式のメモリ継承機能，本予稿集（1995）

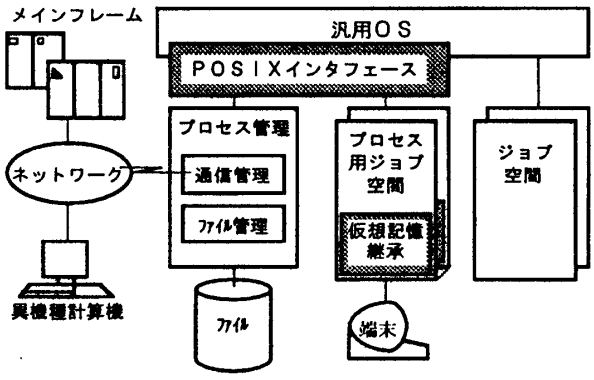

図１．マルチプロセスサポート方式

表２．プロセス空間実現方法比較

NO	項目	専用空間方式	データ空間方式	従来空間方式
1	プロセス空間	POSIXインタフェースに不要な機能を除いた空間	プロセスはタスク、データはデータ空間に	プロセスはジョブ空間
2	テキスト配置	ジョブ固有領域のテキスト用サブプール内	複数プロセス（タスク）が同一空間内で共有	ジョブ固有領域
3	データ配置	ジョブ固有領域のデータ用サブプール内	データ空間	ジョブ固有領域
4	プロセス生成	空間生成およびメモリ管理テーブル更新	タスク生成およびデータ空間コピー	空間生成およびテキスト&データのコピー
5	参照オーバヘッド	領域コピー	なし	なし
6	その他	資源管理等への影響が大きい	独立プロセスを作成できない	空間生成時のメモリーコピーオーバヘッドが大きい

注）POSIXは，Electrical and Electronics Engineers（IEEE）で制定された標準仕様です．