4K−8　CTRONインタフェース準拠OSのインプリメントにおける高リアルタイム性の実現

斎藤喜道[1]，関根明[1]，斉藤賢爾[2]，内記健二[3]
[1](株)日立製作所ソフトウェア工場，[2]日立ソフトウェアエンジニアリング(株)，
[3]日立中部ソフトウェア(株)

1．CTRONインタフェースとVCS/OS

CTRONインタフェースは、1990年代のオペレーティングシステム(以下OS)の開発を目指したTRON(The Realtime Operating system Nucleus)プロジェクトの1つであるCTRONサブプロジェクトが開発したOSインタフェースである。このCTRONインタフェースは情報、通信、交換、ワークステーションに共通なOSインタフェースであり、リアルタイム処理分野への適応、OSの階層化による流通範囲の拡大、及びサブセット化による応用分野への適用性の向上などの特徴を持っている[1)]。

筆者らの開発しているVCS/OS(Versatile Communication Server/OS)は日立製作所汎用コンピュータシステムMシリーズ上で動作するCTRONインタフェースに準拠した通信処理分野向けのOSであり、図1.−1に示すコンポーネントから構成される。VCS/OSの開発にあたっては高リアルタイム性の実現を目標にした。

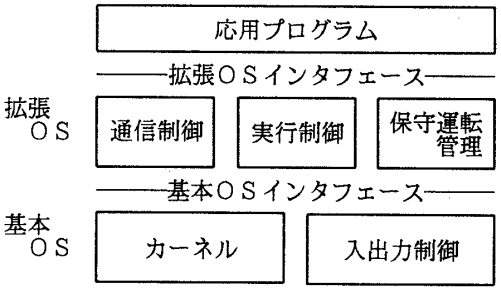

図1.−1 VCS/OSの構成

2．リアルタイムOS

リアルタイム処理とはハードウェアなどから発生した処理要求信号をすばやく受信し、その要求に対しシステムで許される時間内で処理することを言い、このリアルタイム処理用のOSを リアルタイムOSと言う。リアルタイムOSでは、リアルタイム処理を実現するため割り込みに対する高速な応答、高速なシステムコールの提供が必要である。

3．高リアルタイム性の実現

高リアルタイム性を実現するための手段は大きく分けて、システムコールレベル、インプリメントレベルでのリアルタイム処理への適合性の強化がある。システムコールレベルでは、事前処理とリアルタイム処理で使用するシステムコールの分離や、メモリの割り当て単位など処理の高速化につながるパラメタの設定などがあり、これはCTRONインタフェース仕様の設計時に工夫されている。筆者らは、後者にあたるインプリメント上のリアルタイム処理への適合性の強化を図った。インプリメントにあたっては、CTRONインタフェースが基本OS(ハードウェアの差を隠蔽)と拡張OS(応用プログラムに高度なインタフェースを提供)に分かれていること、及び基本OSと拡張OSではリアルタイム性の要求度が違うことから両者で異なるアプローチをとっている。

3.1　基本OSでのアプローチ

今回インプリメントした基本OS(カーネル、入出力制御)は、特にリアルタイム性が要求される機能に重点を置き、内部処理方式、ライブラリ処理方式などの面で工夫を行っている。

(1)オーバヘッドの局所化

システムコールによっては、性能が大きく追求されるものと、さほど追求されないものがある。前者としては送信・受信・確保・解放のシステムコール、後者としては生成・削除・参照系などのシステムコールがある。本システムでは、極力前者の性能が向上するようなインプリメントを行っている。

例1）統計情報の1つとしてタスク毎の使用メモリ量があるがこの算出をメモリ確保・解放時に行

Realization of highly real-time system based on CTRON interface specifications
Yoshinori SAITOH[1], Akira SEKINE[1], Kenji SAITOH[2], Kenji NAIKI[3]
[1]Software Works, HITACHI, Ltd., [2]Hitachi Software Engineering Co., Ltd.,
[3]Hitachi Chubu Software, Ltd.

うと性能劣化となるため参照時に情報収集を行う方式とした。

例２）メッセージボックスへの送受信時、メッセージをキューイングするための管理ブロックが必要となるが、この管理ブロックの確保・解放を送信時に行うと性能劣化となるため、メッセージボックス生成時にあらかじめ行う方式とした(図３．１－１参照)。

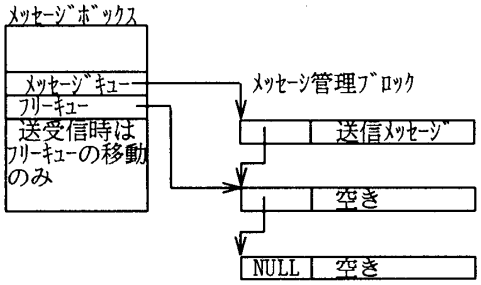

図３．１－１　ﾒｯｾｰｼﾞﾎﾞｯｸｽ管理ﾌﾞﾛｯｸの構成

(２)システムコールライブラリの工夫

例１）コール方法：本システムではシステムコールを実現するためにＳＶＣ(SuperVisor Call)割り込みを利用しているが、発行が非タスクに限定されるシステムコールはシリアライゼーションやプロテクションレベルの変更が不要なため、処理関数へ直接リンクさせることにより性能向上を図っている。

例２）前処理のライブラリ化：ＯＳ域のプロテクションレベルを読み出し可能としており、システムコールの前処理(エラーチェックなど)をできる限りライブラリ内で処理することにより、非タスク走行範囲の削減を行っている。

(３)　システムタスクの設定

タスク生成時、拡張ＯＳ用のタスクに対してのみ、負値整数の実行レベルを許可することにより、システムタスクの優先実行を保証している。

３．２　拡張ＯＳでのアプローチ

今回インプリメントした拡張ＯＳは、割り込みの遅延防止及び他制御との並行処理率の向上のため、非タスク部を極力減らしタスクで処理をすることにした。実際のインプリメント方式は、実行するタスクの種別によってサーバタスク方式とユーザタスク方式に分類される。

(１)　サーバタスク方式

本方式は、各機能ごとに専用のタスク(サーバタスク)を割り当てる方式である。各サーバタスクは自分あてに送られたメッセージを処理することでサービスを提供する。ここで、各サーバタスク間の連絡は直接にメッセージ送信で、更にユーザからの要求はシステムコール処理ルーチンを経由したメッセージ送信によって行う(図３．２－１参照)。

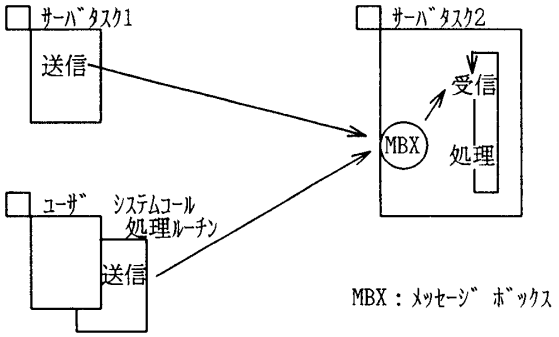

図３．２－１　サーバタスク方式の概要

ただし、複数のサーバタスクを用いた場合、サーバタスク間でのテーブルの排他制御による処理量の増大、サーバタスク間のメッセージの増大、同期処理による並行処理率の低下の問題がある。これに対し、共有エリアをリードオンリのものに限定し 排他制御をなくす、各サーバタスクの独立性を高めメッセージの送受信及び同期処理を減らす事を図った。この方式により高い並行処理率が実現できた。

(２)　ユーザタスク方式

ユーザタスク方式は要求元のタスク(ユーザタスク)上で処理する方式であり、プログラムのローディングなどユーザタスク固有の資源の割り当て、又はアクセスする場合に用いている。本方式はユーザからシステムコールが発行された際、システムコール処理ルーチンに直接分岐しユーザタスク上で処理している。このため、タスク切り替えやプロテクション制御などが不要になり、高速な処理を可能にしている。

４．　おわりに

今後、ますますリアルタイム処理への要求が高まるとともに、 この分野でのソフトウェアの流通性が重要になるであろう。 筆者らはこの１つの解として、ＣＴＲＯＮインタフェースに基づいたＯＳの開発を行った。今後もシステム全体での高リアルタイム性の実現を目指していく。

５．　参考文献

１）社団法人トロン協会編：ＣＴＲＯＮ概説，オーム社，１９８８