高速イベント通知方式

7P-6

宮田 俊介

NTT情報通信処理研究所

1．はじめに

ＯＳを実現する場合に、通信デバイスをはじめとする入出力装置からの割り込み信号の延長で、上位から依頼された処理の完了通知イベントまたは、非同期に発生するイベントを通知する場合が存在する。

本稿では、階層化ＯＳにおける下位モジュールが、上位モジュールタスクに対してこのようなイベントの通知を行う場合の高速な同期インタフェースとして、エントリコール方式と呼ばれる方式を提案する。

2．従来技術と問題点

タスクとの間で非同期に発生するイベントの授受を行うためのインタフェースとしては、イベントフラグ、メッセージボックス、ランデブー等の機構が知られており、これらをＯＳ機能として提供するシステムが多い。しかし、これらの同期機構では、通知先のモジュールがイベント処理を行うのがタスクディスパッチ後となってしまうため、通信処理における入出力チャネルからのデータの転送など、割り込み信号の延長での即時性が要求される処理を起動するために使用するには不適当である。（図１）

図１：メッセージボックス機構による通知処理のタイムチャート

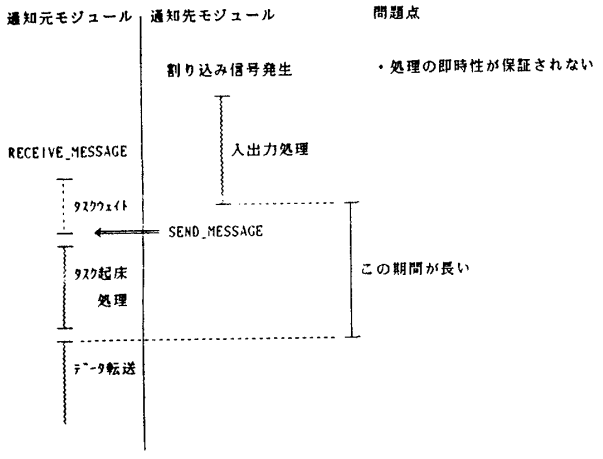
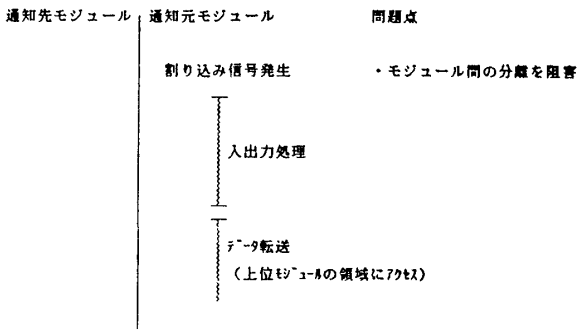


図２：通知元モジュールによる直接処理のタイムチャート



一方、処理の即時性を実現するために、通知先モジュールの管理している領域に対するデータの転送等、通知先モジュール内で行うべき処理を通知元モジュールが直接行ってしまう方式もあるが、モジュール間の分離を阻害することになる。（図２）

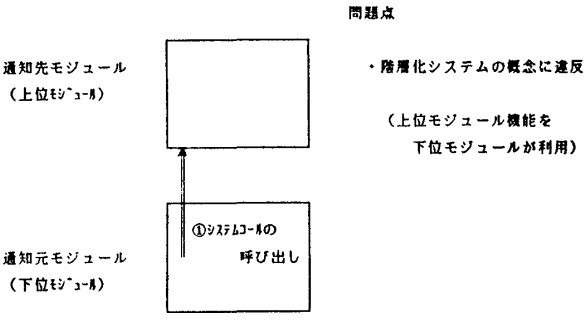
また、通知先のモジュールがデータ転送機能をシステムコールとして提供することは、通信処理の場合には上位モジュールを下位モジュールから呼び出すこととなり、呼び出し関係により階層構造を構成する階層化システムとしての前提を満足できない。（図３）

以上のように従来技術においては

・処理の即時性
・モジュール間の独立性
・階層化システムの概念の遵守

の全ての条件を満たすことが困難であった。

図３：通知先モジュールによるシステムコール提供方式での呼び出し関係



A High-speed Notifing Method of Asynchronous Events

Shunsuke MIYATA

NTT Communications and Information Processing Laboratories

3．エントリコール方式

エントリコール機構はメモリ管理等と同様にシステムが提供する最も基本的なＯＳ機能として実現し、大きく分けて２つの機能から構成する。通知すべきイベントが発生した場合に通知元モジュールが起動するルーチン（以降エントリルーチンと呼ぶ）を通知先モジュールが登録するための機能と、通知先モジュールへ通知すべきイベントが発生した場合に、通知元モジュールが登録されたエントリルーチンを起動することによりイベントを通知するための機能である。

本機構を階層化システムの下位モジュールから上位モジュールへのイベント通知に適用した場合、下位モジュールによるエントリルーチンの起動は、同レベルの機能の利用となり、エントリルーチン相当を上位モジュールがシステムコールとして提供する方式と異り階層化システムの概念に抵触することはない。（図４）

図４：エントリコール方式での呼び出し関係

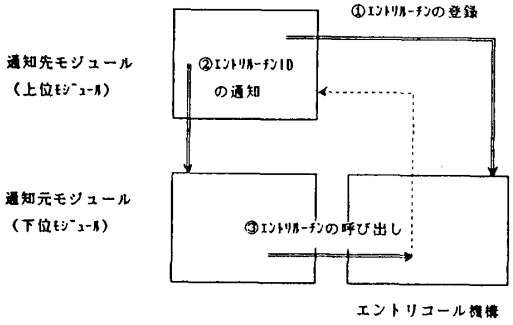
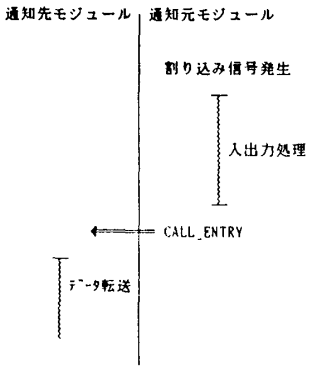


図５：エントリコール方式による処理のタイムチャート



エントリコール機構では、エントリルーチンの起動の延長で処理が行われるため、メッセージボックス等のタスク間同期機構を用いた場合と異なり、タスクディスパッチを介さないので、イベント処理の即時性を確保することができる。（図５）

エントリルーチン自体は通知先モジュールが提供するものであるため、通知元モジュールが直接処理を行う方式と異なり、通知元モジュールが通知先モジュールの領域構成等を意識することは不要なため、モジュール間の独立性が保たれる。

4．実施例

ＯＳ機能のうち通信処理を例にとり、入出力制御を下位のＯＳ機能とし、手順処理を上位のＯＳ機能とした場合を考える。入出力デバイスがデータを受信した場合には、割り込み信号が発生し、下位ＯＳとしての処理が実行されるが、さらに上位のＯＳ機能である手順処理を行うために入出力処理モジュールから手順処理モジュールへのデータの引渡しが必要となる。このデータの引渡しに当たって、処理の即時性、モジュール間の独立性、階層化システムの概念への準拠が要求されるため、エントリコール機構を基本的なＯＳ機能として提供し、データ転送処理をエントリルーチンとすることが有効となる。（図６）

5．まとめ

階層化ＯＳにおける高速な同期インタフェースとしてエントリコール方式を提案した。今後は通信制御などのＯＳ機能のインプリメントを通じて、エントリコール機構の利用方式を確立していきたい。

図６：エントリコールの利用者インタフェース（例）

```
●エントリコール部の提供する機能
    エントリルーチンの登録用システムコールの仕様，
        DEFINE_ENTRY(routine_address, entry_id);
            routine_address(入力パラメタ)：エントリルーチンのアドレス
            entry_id(出力パラメタ)：システムが払出すエントリルーチンの
                                    識別子

    エントリルーチンの呼出し用システムコールの仕様
        CALL_ENTRY(entry_id);
            entry_id(入力パラメタ)：実行するエントリルーチンの識別子
-----------------------------------------------------------------------
●通信入出力制御部が提供する機能
    パス設定システムコールの仕様
        CONNECT_PATH(line_no, entry_id, path_id);
            line_no (入力パラメタ)：通信パスを設定する回線の番号
            entry_id(入力パラメタ)：データ受信時に呼出すエントリルーチン
                                    の識別子
            path_id (出力パラメタ)：システムが払出す通信パスの識別子

    データ受信処理のコーディング（部分）
                :
            CALL_ENTRY(<CONNECT_PATHで指定されたエントリＩＤ>);
                :
-----------------------------------------------------------------------
●手順処理部が提供する機能
    パス設定処理のコーディング（部分）
                :
            DEFINE_ENTRY(<受信通知用ルーチンのアドレス>, ENTRY);
            CONNECT_PATH(<回線番号>, ENTRY, PATH);
                :
```