1Q-3　ＪＳＩ　ＡＩワークステーション（３）
―　ウインドウシステムからみたＯＳの条件

江藤　博明　　白鳥　敏幸　　天明　崇　　戸沢　義夫
日本アイ・ビー・エム株式会社　サイエンス・インスティチュート

１．はじめに

ＡＩワークステーションではプログラム、ルール、データーベース等の情報を同時に見ながらプログラム開発をする必要があるためマルチウインドウ機能は必須である。ここでは我々が開発したウインドウシステムを紹介する。さらにウインドウシステムの効率やオーバヘッドに注目することでマルチプロセスＯＳの構成について考察する。

２．ウインドウシステムの概要

図１は我々が開発したウインドウシステムの表示例である。このウインドウシステムは次の機能を持つことを特長としている。

・ビットマップウインドウ
・マルチ漢字フォント
・グラフィックス

なおウインドウ管理はスクリーンをメッシュ状に分割した単位で行なっている。

３．ウインドウシステムの構成

このウインドウシステムは社内ＯＳの上で開発された。このＯＳは図２のようなプロセス構造を持っている。

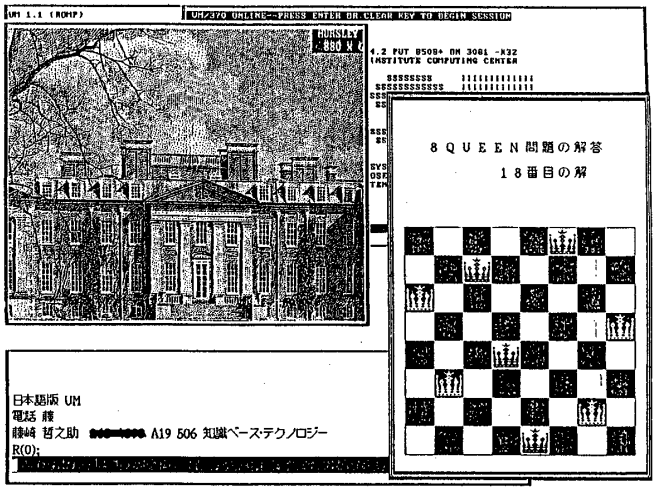


図１　画面表示例

ここで四角の箱はそれぞれプロセス空間を表しており、斜線部はプロセス間の共有部分を表している。このようにＯＳのカーネルはそれぞれのプロセスの共有部分として存在していることが社内ＯＳの特長の一つである。

このＯＳ上でウインドウシステムを実現する場合に次の２つ方法が考えられる。

・ウインドウマネジャーを一つのプロセスとする方法
・カーネルとして共有部分に組み込む方法

今回は共有部分にウインドウマネジャーを組み込むことにした。その理由としてアプリケーション側からのインタフェースとして、ウインドウ管理用の階層を提供するのではなくシステムコールレベルで提供したいこと、もう一つにはウインドウ移動の際に生ずる下のウインドウの再表示のために、移動ウインドウの仮想スクリーン領域と下のウインドウの仮想スクリーン領域は同時にアクセスできるように仮想スクリーン領域は共有部分に置きたいことが挙げられる。

さらにユーザーが作成したプログラムをカーネルに組み込むことができるというＯＳの機能が非常に有効であることも挙げられる。カーネルに組み込んだプログラムでは以下のことが可能である。

・プロセス間のデータ共有
・ダイナミック・メモリ・アロケーション

実際このウインドウシステムは図３－ａのような形でインプリメントされている。
参考のためにＵＮＩＸのようなＯＳにおけるウインドウシステムのプロセス空間を図３－ｂに示す。

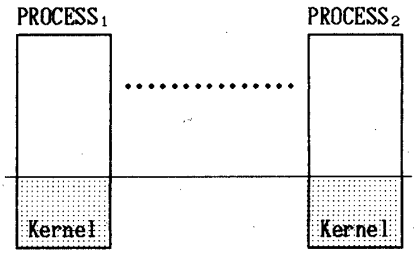


図２　プロセス構造　　:shared

JSI AI Workstation(3) - Requirements of Operating System from the point of view of window system
Hiroaki ETOH, Toshiyuki SHIRATORI, Takashi TENMYO, Yoshio TOZAWA
Science Institute, IBM Japan, Ltd.

４．マルチプロセスＯＳの条件の検討

４．１マルチプロセスＯＳの条件

一般にウインドウシステムの評価基準としてプログラムインターフェース、ユーザーインタフェース等が重要視されているが、ここではマルチプロセスＯＳ上のウインドウシステムとして以下の項目が重要であると考えた。

・応答性が良いこと

・システムオーバヘッドが小さいこと

例えばマウスを操作してウインドウの大きさを変える等の作業での応答性の良さはユーザインタフェースの善し悪しを決定する一つの要素であると考えられる。またウインドウシステムやＯＳカーネル等の下位レベルのプログラムはできるだけオーバヘッドが小さいことがトータルシステムとして優れていると考えられる。

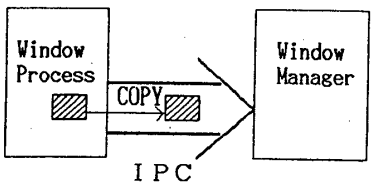
以下では、この２点について例を挙げ、ＯＳの形態（プロセス構造）について検討する。

４．２　マウスセンシティブなウインドウのサポート

ポインティングを目的として使用されるマウスカーソルでは、それが指し示しているものを明確化するためにポイントされたウインドウの配置や形状を変える等の処理が行なわれる。このようなマウスセンシティヴなウインドウを実現する場合を考えてみよう。マウスカーソルがウインドウを横切ったときに、イベントとして割り込みが掛かるようなハードウェアがあればよいのだが、ここではそれをソフト的に処理することを考える。そのためには、カーソル、ウインドウの形や色を制御している Window Manager が常にマウスの動きを監視する必要がある。図３－ａのようなＯＳでは、 Window Manager プロセスが待ち状態にあるような時には、Window Manager プロセスへと切り替わるまでの間、処理が遅れることになる。図３－ａのように Window Manager がすべてのプロセスから共有される形態のＯＳにおいては、 Window Manager が待ち状態になることがないため、マウスに対する応答が良いと言える。つまりこのようなＯＳ構造をとることで、リアルタイム処理にも対応していると言うことができる。

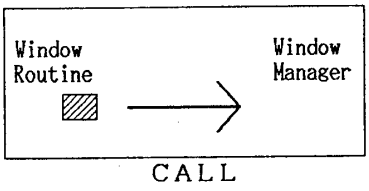
４．３　ウインドウへのデータ出力

ここでは、ウインドウ上へのデータ出力における図３－ａと図３－ｂのＯＳのデータの流れに関して比較検討する。図３－ｂでは次の様な流れとなる。



このように、ウインドウプロセスからの書き込み処理要求およびデータは Inter Process Communication を利用して渡される。この際にＩＰＣでは内部に抱えているバッファにデータをコピーしなければならないというオーバヘッドが存在する。なぜなら図３－ｂのＯＳではプロセス間で共有するようなデータ領域がないからである。もう一つのＯＳオーバヘッドとして、ウインドウプロセスからウインドウマネジャーへとプロセスの切り換えが必要ということが挙げられる。

一方図３－ａでは、次のような同一プロセス内の CALL BY REFERENCE によりデータを渡すため中間バッファへのコピーやプロセスの切り換えの必要がない。したがってこの方式では、図３－ｂの方式に比べてウインドウ処理のオーバヘッドは軽くなる。



５．結び

ウインドウシステムの評価基準として、応答性の良さとオーバヘッドの小ささという二つの観点を取りあげ、それについて検討した。この点において社内ＯＳが優れているのは、

・カーネルの拡張が容易

・プロセス間に共有領域が存在する

というような特徴を持っているからであると言える。

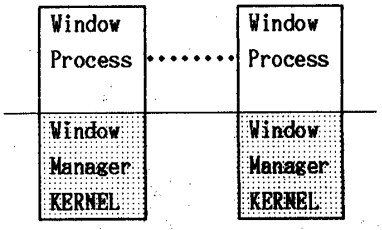


図３－ａ（本構成）

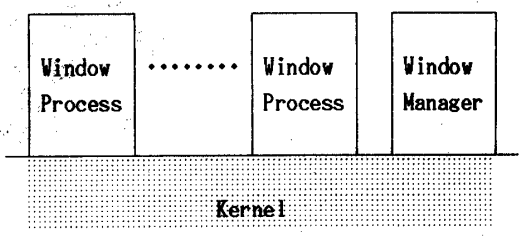


図３－ｂ

図３　ウインドウシステム構成の比較