Vol. 31 No. 3　　情　報　処　理　　Mar. 1990

解　説

OS/2 の現状と展望†

石田晴久††

1. はじめに

OS/2 は MS-DOS の上位 OS として，1987年4月に IBM 社およびマイクロソフト (MS) 社によって発表された高級パソコン用の OS である．ここで“高級”というのは，この OS を使うには，パソコンのプロセッサとして，16 ビット型（では上位）のインテル 80286 あるいは 32 ビット型の 80386 が必要であり，また主記憶（メモリ）は 5 MB 以上，固定ディスクは 20 MB 以上のものを用意しなければならないからである．インテル 8086/8088 や日本電気 V 30 のみを使ったパソコンでは OS/2 は使えない．

さて，その後，表-1 に示すように，この OS/2 は，1987年夏から秋にかけて，開発者向けに IBM PC/AT および PS/2 用の SDK (System Development Kit) が配布され始めた．さらにマルチウィンドウ機能 (PM＝Presentation Manager) の入った V 1.1 は開発者向けが MS（マイクロソフト）や NEC より 1987年末から 1988年夏にかけて配布されている．ユーザ向けの PM 日本語版が日本 IBM から出荷され始めたのは 1989年5月からである．

こうした動きからみて，OS/2 が普及し始めるのは，早くても発表後3年の 1990年4月以後と予想される．この意味で，OS/2 は 1990年代のパソコン OS といってよい．現時点では，OS/2 用の日本語版の応用プログラムはまだほとんどないから，本稿では OS/2 自体に焦点をしぼってのべる．

2. OS/2 登場の背景

現在パソコン用の OS としてもっとも広く使われている MS-DOS は，1981年8月に発表された IBM PC 用に開発されたもので，その後 MS 社より世界中の各メーカに供給され，我が国ではそれが日本語化されてから普及するようになった．

この MS-DOS は，もともとは，8 ビット型のパソコンの標準 OS であった CP/M（ディジタル・リサーチ社製）をお手本にして，インテル 8086/8088 という単純な 16 ビット型プロセッサ用に開発された単一タスキングの簡単な OS であった．MS-DOS の対象機種となった 8086 (8088 はその 16 ビット外部バスを 8 ビットにしたもの) では，アドレスは 20 ビットだから，プロセッサでアクセスできるメモリ容量は 1 MB ということになる．しかし IBM PC や PC9800 では，1 MB 内の上方の領域を BIOS やビデオ RAM などのシステム領域にあて，プログラムで使える領域は 640 kB という設定が行われている．

CP/M ではわずか 64 kB のメモリしか使えなかったことを考えると，MS-DOS で 640 kB ものメモリが使えるということは 1981年当時は画期的なことであった．しかしその後，メモリが安くなって 640 kB 以上のメモリがパソコンに実装可能となったこと，16 MB のメモリをサポートする 80286 が 1982年から登場したこと，応用プログラムが複雑高度になって，640 kB に収まりきれないものが出て，オーバレイが必要になる場合が増えたこと，などが重なって，メモ

表-1　OS/2 の開発経過

1987年4月	OS/2 発表 (IBM, MS)
1987年夏	MS が SDK (1.00＝V1.0) 配布
1987年12月	MS が SDK (1.02＝V1.1, PM 予備版) 配布
1988年3月	日本 IBM が J1.0（日本語版）出荷
1988年7月	NEC が V1.0（日本語版）出荷
1989年2月	MS が SDK (1.1, PM 正式版) 配布
1989年5月	日本 IBM が J1.1（基本版）出荷
1989年6月	日本 IBM が J1.1（拡張版）出荷
1989年7月	NEC が V1.1-SDK（日本語版）出荷
1989年12月	NEC や富士通が 1.1 正式版出荷
1990年3月	日本 IBM が J1.2（拡張版）出荷予定
1990年	V2.0 (80386 用 32 ビット版，仮称) 予定

注) SDK はシステム開発者用キット．日本 IBM 版は 55××シリーズ用，NEC 版は PC 9800 シリーズ用．

† Current Status and Perspective of OS/2 by Haruhisa ISHIDA (Computer Centre, University of Tokyo).

†† 東京大学大型計算機センター

リ領域が 640 kB しかないことは MS-DOS の大きな制約となってきた．かな漢字変換のために数 10 kB 以上の日本語フロントエンドプロセッサ (FEP) をメモリに常駐させなければならない我が国の場合は，この 640 kB の壁は深刻である．

しかし単にこの壁の克服だけであれば，OS/2 でなくても，MS-DOS で EMS（拡張メモリ）を使う手がある．EMS はアメリカでは 1987 年から，我が国では 1989 年から普及し始めたもので，640 kB より上のアドレスのうちの 64 kB を 4 個のブロックに切って，外部のメモリ領域に切り換えて割り当てることにより，メモリの最大 32 MB までの拡大を可能とする．この EMS は 8086 でも使えるという利点はあるものの，16 MB までのメモリが使える 80286 の特徴は生かせないことになる．

MS-DOS に対してもうひとつ出てきた要求はマルチタスキングである．たかがパソコンにマルチタスキングが本当に必要かという疑問もなくはないが，たとえば，ネットワークの中でパソコンをサーバとして使おうとすると，マルチタスキングが不可欠となる．単一タスキングのパソコンだと，だれかがそれを使っているときには，バックグランドでファイル・サーバやプリンタ・サーバとしての機能が果たせないからである．

ところで，マルチタスキングの OS としては周知のように UNIX がある．近年 UNIX は 4 MB 程度以上のメモリと数 10 メガバイトの固定ディスクのついたパソコンで利用可能となり，UNIX の下で MS-DOS を走らせることも可能になりつつある．したがってマルチタスキングの面だけとれば，OS/2 は UNIX と競合することになる．

OS/2 の開発が進められた 1980 年代半ばの時点では，パソコンでの EMS や UNIX の利用はまだ考えられなかった．OS/2 の開発や普及が当初予想あるいは期待されたのよりはやや遅れ気味なのは，OS/2 自体の難しさにもよるが，下からは EMS による MS-DOS の延命，上からは UNIX という両側面からの競合があるためとみてよいのではなかろうか？

3. OS/2 の特徴

アドレスが 32 ビットで，4096 MB までの物理メモリ領域がアクセスできる 80386 の機能までは活かしていないという意味で，現在の OS/2 は 80286 対応の OS である．この OS/2 の特徴をあげてみると，次のようになる．

（1） メモリは 16 MB までアクセスできる．メモリが不足した場合は，メモリと固定ディスクとの間で，セグメント（最大 64 kB）単位のスワッピングを自動的に行わせることが可能である．（ページングは不可）

（2） マルチタスキングができる．〔V 1.0 では異なるタスクに対してキー操作で画面をフルスクリーンで切り換えていたが〕V 1.1 ではマルチウィンドウの上でマルチタスキングが可能である．マルチタスキングのためのプロセス間通信 (IPC) には，シグナル（イベント・フラグ），パイプ，セマフォ，共有メモリが使える．これらはプログラムの中では後述の API という形のシステム・コールにより利用できる．

（3） PM (Presentation Manager) と呼ばれるマルチウィンドウ・システムが標準装備されている．この PM では，タイトル・バー，メニュ・バー，スクロール・バー，プルダウン・メニュ，ダイアログ・ボックス，アイコン，フルスクリーンへの拡大機能，システム・メニュなどがサポートされる．いつでも開けるウィンドウとしては，タスク・マネジャ，ファイル・システム，“プログラムの始動”ウィンドウなどがある．

（4） MS-DOS と基本的に上向きの互換性をもつ．ファイル・システムは完全に同じであり，MS-DOS のコマンドは大部分がそのまま使える．プログラムの実行に関しては，MS-DOS のプログラムを一時にはひとつだけ（しかも他の OS/2 タスクが動いているときは休止という条件で）動かすことができる．ただし，互換ボックスと呼ばれる MS-DOS 用メモリについては，図-1 に示すように，MS-DOS に割り当てられる 640 kB のうちの一部が MS-DOS エミュレータに食われるため*，メモリ不足で動かない MS-DOS 応用プログラムもありうる．また両 OS でデバイス・ドライバの構造が異なるために動かない MS-DOS プログラムもある．

（5） コードページの記号により，二つの国語によるメッセージが切り換えられる．我が国で出荷される OS/2 は日本語モードでも英語モードでも使えるようになっている．

（6） 応用プログラムのインタフェースが API (Application Programming Interface) ライブラリとして規定されており，これのみを使って書いたプログ

* PC 9800 の場合，約 140 kB がエミュレータで食われる．

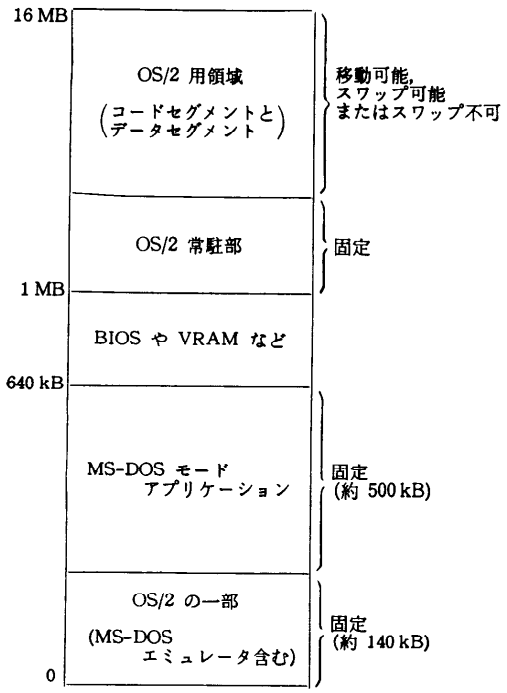

図-1 OS/2のメモリ・マップ

ラムは異機種間で互換性がもたせられる．API のサブセットである Family API を使えば，OS/2 モード（プロテクト・モード）および MS-DOS モード（非プロテクト・モードの互換ボックスの中）の両方と MS-DOS 自体の上で動くプログラムを書くことができる．

（7） プログラム・モジュールのダイナミック・リンクおよびメモリ内での共用が可能である．このため，とくにプログラムが全体として大きい場合，ディスク内およびメモリ内のスペースが節約できる．

（8） 構成制御がかなりキメ細かに指定できる．OS/2 の起動時に参照される config.sys ファイルの中では，MS-DOS にもある break, buffers, device, fcbs, shell のほか，**表-2** のコマンドが指定できる．そのうち，実行制御に関するものは，libpath, protshell, protectonly, run, マルチタスキング関係は iopl, maxwait, priority, threads, timeslice, メモリ管理関係は，diskcache, memman, rmsize, swappath, 文字コード切換え関係は codepage, country, devinfo である．

（9） MS-DOS にないコマンドとして，append, dpath, help, start, spool, またバッチ用に call, setlocal, endlocal の各コマンドがサポートされている．

表-2 OS/2 に特有な構成制御コマンド（＝の後は設定例）

codepage=932, 437	コードページ番号
country=081	国名コード（日本は 081，アメリカは 001）
devinfo=kbd, jp, keyboard.dcp	デバイス情報（コードページ対応）
diskcache=512	キャッシュ・メモリ容量 (kB)
iopl=no|yes	データ入出力特権の付与の有無
libpath=パス名	ダイナミック・リンク用
maxwait=3	タスク実行の最大待時間（秒）
memman=noswap, move	メモリ管理用
pauseonerror=yes|no	config.sys エラーの処理形態
priority=dynamic|absolute	タスクの優先順位
protshell=pmshell.exe	保護モード用シェル
protectonly=no	保護モードのみか否かの指定
rem	コメント
rmsize=640	MS-DOS モード領域のサイズ
run=retract.exe	デーモンの指定
swappath=¥spool	ディスク上のスワップ領域
threads=128	一時に走行可能なスレッドの数
timeslice=32	タイム・スライス値（ミリ秒）

（10） ネットワーク機能が LAN マネジャでサポートされている．

4. PM によるマルチウィンドウ機能

OS/2 の大きな特徴は，マッキントッシュに似たマルチウィンドウ表示が可能で，しかもその上で（マッキントッシュでは本格的にできない）マルチタスキングができることである．**図-2** に OS/2 の PM によるウィンドウの表示例を示す．PM ではこのような重ね合せ表示（カスケード方式）のほかにウィンドウが重ならないタイル方式の表示も可能である．画面は一般にグラフィックになっているから文字と図形を混在表示させることができる．

さて，PM ベースの OS/2 (V 1.1) パソコンの場合，起動直後に“プログラムの始動”というウィンドウが出てくるから，実行したいプログラムをあらかじめ登録しておいたときには，その名前を画面上のリストから選んで実行させることができる．登録していないプログラムは，画面上の“OS/2 ウィンドウ表示コマンド・プロンプト” ①あるいは“OS/2 フルスクリーン コマンド プロンプト” ②を選ぶと，新しいウィンドウが表示されるからその中でコマンド名をキーインすればよい．このとき②を選ぶと，他のウィンドウが一切見えないフルスクリーン・モードになってしまうが，プログラムによってはウィンドウ内で動かないものがあるので，そのときはこのモードで実行させるしかないわけである．

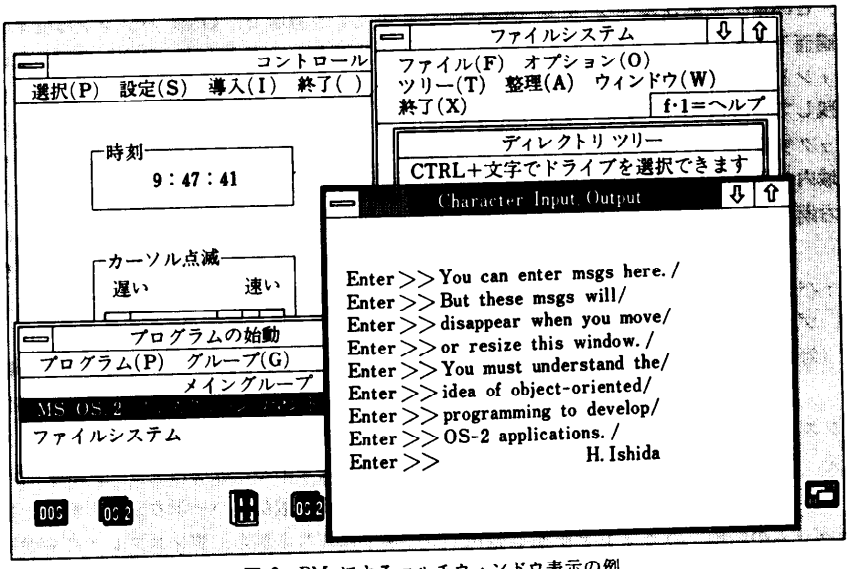

図-2 PM によるマルチウィンドウ表示の例

さて，表示されているウィンドウの中でもっとも上にあるウィンドウをアクティブ（または focus）ウィンドウと呼ぶ．キーボード入力はこのウィンドウに対してのみ可能である．他のウィンドウをアクティブにするには，次の4つのやり方がある．

（1） 目的のウィンドウのどこかにカーソルをもっていって，マウス・ボタンをクリックする．

（2） ESC キーと CTRL キーを同時に押して，タスク・マネジャのウィンドウを呼び，そこにリストされているプログラム（ウィンドウ）を選ぶ．タスク・マネジャの画面には現在実行されているプロセスが常にリストアップされているのである．

（3） ESC キーと Alt (GRPH または前面キー) を押すことを繰り返して目的のウィンドウを一番上にする．

（4） 画面下のアイコンの中から目的のプログラムを選ぶ．

なお，MS-DOS プログラムを走らせる目的で MS-DOS モード (real mode) にしたいときには，画面左下の DOS アイコンをダブル・クリックするか，タスク・マネジャ画面で “MS-DOS（日本語 DOS）コマンド・プロンプト” を選ぶかすればよい．こうして入る MS-DOS モードはフルスクリーンとなる．このとき，同時に走らせることのできる MS-DOS プログラムはひとつのみであるが，バックグランドでは OS/2 プログラムが走らせられる．このモードで MS-DOS プログラムが暴走すると，OS/2 全体がダウンすることがあるので注意がいる．

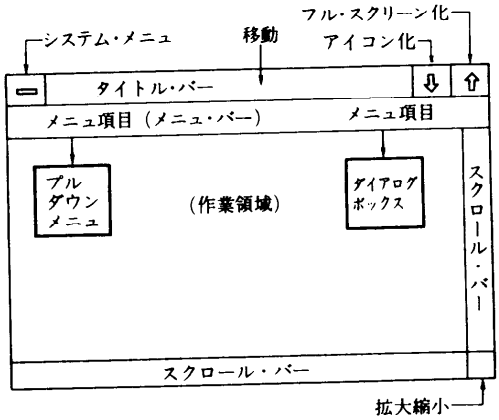

図-3 PM のウィンドウ構成

次に図-3 に，ひとつのウィンドウの構成を示す．各部分は常に存在するとは限らないが，一番上にはタイトル・バー，その下にはメニュー項目の表示されるメニュ・バー，右端および下端にはスクロール・バーが普通表示される．この GUI (Graphical User Interface) をとおして統一的に行えるユーザ操作には次のものがある．

① タイトル・バーにカーソルをあてて，ドラッグすることで，ウィンドウ全体を移動させる．

② ウィンドウの四隅あるいは枠にカーソルをあてて，ドラッグしてウィンドウを拡大縮小する．

③ しばらく不用なときは，右上の下向きの矢印を

クリックして，このウィンドウをアイコン（小さな4角）にし，画面下に置く．

④ このウィンドウを画面いっぱい（下側のアイコン表示領域は残して）にしたいときは，右上の上向きの矢印をクリックする．

⑤ 作業領域内の情報を上下左右にスクロールさせたいときは，右端または下端のスクロール・バーをクリックする．

⑥ メニュ・バーに表示されているメニュ項目をクリックして，プルダウン・メニュまたはダイアログ・ボックス（対話のための箱）をとおして操作を行う．

⑦ 左上のシステム・メニュをクリックして，元のサイズへの復元，移動，拡大縮小，アイコン化，全画面化，タスク・マネジャ呼出しなどを行う．

なお，“プログラムの始動”のウィンドウから常に起動できるユーテリティのひとつに“ファイル・システム”がある．これを使うと，ファイルのツリー構造が調べられる．MS-DOS の dir, copy, erase (del), rename, print, attrib, mkdir などのコマンドの操作はこれのメニュで可能である．

このファイル・システムの面白い機能のひとつに“関連づけ (associate)”がある．これはデータ・ファイル名の拡張子（ピリオドの下の3文字）にそれを処理するプログラムを割り当てる機能である．たとえば，画面エディタ (e.exe) に txt や c という拡張子を関連づけておくと，ファイルのリストで myfile.txt や myprog.c をピックするだけで，画面エディタが起動されて編集がすぐできるようになっている．これはオブジェクト指向的，すなわちデータを指すとそれを処理するプログラムが動くというプログラムの走らせ方である．[マッキントッシュではいちいち associate しなくてもこの機能が使える．]

以上のべた PM の look & feel はユーザの操作性を統一するものとして IBM の SAA (System Application Architecture) でサポートされている．この PM は，MS-DOS の Windows およびOSF (Open Software Foundation) が開発中の Motif というマルチウィンドウ・システムに採用されている操作法でもある．

さて，こうしたウィンドウ指向のユーザ・インタフェースは，ベテラン・ユーザにとってはかえって操作に時間がかかる面もあるが，一般のユーザにとっては，マッキントッシュ（マック）のような感じで使えるという意味で，OS/2 パソコンは従来の一般パソコンより親しみやすく使いやすいに違いない．ウィンドウの大きさが自由に変えられ，アイコン化さえすぐできるのは使ってみると面白い．マックに比べると，ファイル名のアイコン化やファイル名と応用プログラムの関連づけが自動化されていないとか，応用プログラムのインストール（それによる config.sys の書き変え）に問題はありうるが，PM ウィンドウもなかなか使いやすいし，各ウィンドウでマルチタスキングができるのも素晴らしい．

OS/2 パソコンのハードウェアで今後期待したいのは，ディスプレイの解像度が，現在の 640×400 ドットや 640×480 ドットから 1000×1000 ドット近辺に上ることである．後述のように，PM 用のソフトウェアはドット数などハードウェアの特性とは独立になるように作れるから，ディスプレイの解像度は自由に設定できる．それが 1000×1000 ドット近辺になれば，OS/2 パソコンでは現在のワークステーション並みの美しい表示を行うことが可能となる．

5. OS/2 のプログラミング

OS/2 で使える応用プログラムはまだ多くないから，パソコンのソフトウェア関係者にとっては，応用プログラムを増やすことがさしあたりの急務である．そこで本章では OS/2 におけるプログラミングについてとくにのべる．

まず，OS/2 のソフトウェアのシステム構成は図-4 のようになっている．OS/2 で動くプログラムは，PM すなわちウィンドウを使うもの，非 PM すなわちウィンドウを使わない（しかし保護モード用の）プログラム，それに MS-DOS 用のプログラム（フル・スクリーン使用）の三つに大別できる．このうち MS-DOS 用のプログラムには一般に機種依存性があるが，PM および OS/2 のカーネルのシステム・コール (API など) をきちんと使えば，機種によらず，どの OS/2 パソコンでも共通でも動く応用プログラムを作ることが可能である．

(1) ウィンドウ内で動く PM 用プログラム

自分でウィンドウを開いて，その上で動くプログラムを書くのを助けるため，PM には WinCreateStd_Window や WinRegisterClass など 239 種類の関数（システム・コール）が用意されている．これらの利用はオブジェクト指向になっていて，クライアント・ウィンドウ（作業領域）などのオブジェクトへの

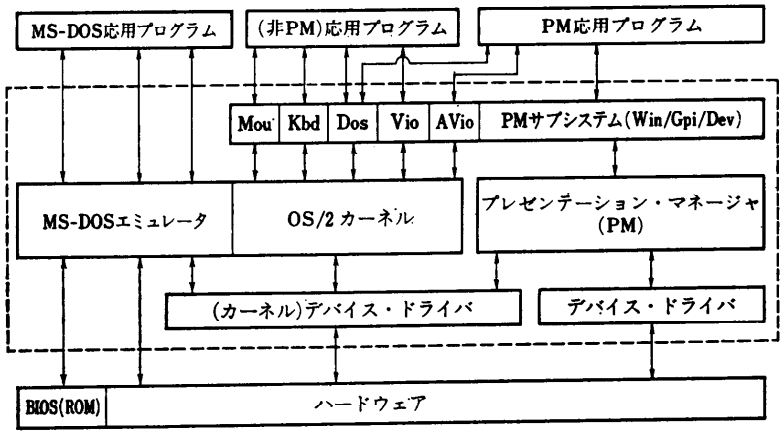

図-4 OS/2 のシステム構成

指示を行うメッセージとしては，WM_COMMAND や WM_PAINT など 94 種類のものがある．

さらに，図-4 にある GPI (Graphics Programming Interface) では，GpiLine や GpiCharStringAt など 224 種類のグラフィック関数，また入出力機器関係では DevOpenDC (DCは Device Context) など7種の関数が定義されている．前者の Gpi 関数を使うと画面（ウィンドウ）の座標は完全に仮想化されるから，ハードウェアの物理的なドット数とは独立なプログラミングが可能となる．しかも GpiCreatePS 関数のオプションを変更すれば，ミリ単位やインチ単位の寸法の指定が可能だから，ドットをまったく意識しないプログラミングも可能である．

ところで，自分の作るウィンドウ内でメニュ・バーやダイアログ・ボックスを使いたいときには，Cプログラムとは別の形で，それらのリソースを記述することが可能になっているが，そうしたリソースをロード・モジュールに組み込む手段として，リソース・コンパイラが用意されている．

またダイアログ・ボックス，文字フォント（今のところ英字のみ），アイコンを表示するためのツールとしては，それぞれを扱うエディタがあるが，それらの出力はすべてリソース文からなる．それらはリソース・コンパイラでコンパイルして始めて，ロード・モジュールに組み込めるようになる．

図-5 に PM 用プログラムの例として，図-2 の一番上の（アクティブ／フォーカス）ウィンドウを表示するプログラムの例を示す．この例では，まず main の中で，メッセージ・キューを作り，ウィンドウ・オブジェクトのクラスを定義して，標準ウィンドウを表示している．このとき style の指定で，図-3 で解説したシステム・メニュ，タイトル・バー，アイコン化・フルスクリーン化の矢印 (MINMAX) などが付加される．このあと main の中の while ループでやっているのは，このウィンドウ・オブジェクトへのメッセージを待ち，メッセージがきたらこのオブジェクト・クラスを処理する frame 関数に制御を渡すことである．

次にフレーム関数では，（最初およびウィンドウを移動させるとき出る）WM_PAINT メッセージを受けたときに，グラフィック画面である作業領域に Enter>>なる文字列を表示する．次に WM_CHAR はキーボード入力があると送られるメッセージで，このときは (x, y) 座標をしかるべくセットして入力文字を表示する．（ただしこのままでは，ウィンドウを移動させると，入力文字の表示が消えてしまうから，本来は表示文字をバッファに入れておいて，WM_PAINT 時に再表示させる必要がある．またこのウィンドウのサイズを変えると，表示はすべて消えてしまうから，WM_SIZE メッセージがきたときには再表示をするようにすべきである．）

この例で分かるように，PM 用のプログラミングは，オブジェクト／メッセージ指向かつグラフィック指向である．したがって OS/2 用の応用プログラムを書くにはオブジェクト指向の考え方を十分理解しておかなければならない．

（2） カーネル用ライブラリ (API)

API は Application Programming Interface の略で，これに属するライブラリ関数は，ウィンドウを意識しない（しかし普通はウィンドウの上でも動く）フルスクリーン用の応用プログラムを機械独立な形で書

```
/*·····example.c·····H. Ishida·····*/
#define INCL_PM
#include <os2.h>
#include <stdio.h>
int cdecl main ( void );
MRESULT CALLBACK frame ( HWND, USHORT, MPARAM, MPARAM);
HAB     whandle;
HMQ     msgque;
HWND    client;
HWND    fhandle;
CHAR    class[ ]="myclass";
CHAR    title[ ]="Character Input/Output";
/*·····*/
int cdecl main ( ) { QMSG     qmsg;
  static ULONG style=FCF_SYSMENU | FCF_TITLEBAR |
                     FCF_MINMAX  | FCF_SIZEBORDER |
                     FCF_SHELLPOSITION;
    /* initialize a window & create a msg queue */
  whandle=WinInitialize ( NULL );
  msgque=WinCreateMsgQueue ( whandle, 0 );
  if (! WinRegisterClass (whandle, class, frame,
            CS_SIZEREDRAW, 0 )) return 0L;
    /* create the frame window */
  fhandle=WinCreateStdWindow ( HWND_DESKTOP, WS_VISIBLE
            &style, class, title, 0L, NULL, 1, &client );
    /* loop to poll messages from event queue */
  while ( WinGetMsg ( whandle, &qmsg, NULL, 0, 0 ))
     WinDispatchMsg ( whandle, &qmsg );
    /* Clean up */
  WinDestroyWindow ( fhandle );
  WinDestroyMsgQueue ( msgque );
  WinTerminate ( whandle ); }
/*·····frame·····H. Ishida·····*/
#define   XSTART   60L
#define   YSTART   200L
#define   XDELTA   10L
#define   YDELTA   17L   /* for PC 9800 */
MRESULT CALLBACK frame( HWND hwnd, USHORT msg,
                  MPARAM mp1, MPARAM mp2 )
{               HPS      hps;
                RECTL    rectangle;
    static      POINTL   origin;          /* stored in memory */
                USHORT   fs;
                CHAR     ch[2];
  switch ( msg ) {
    case WM_PAINT:
      hps=WinBeginPaint( hwnd, NULL, &rectangle );
      WinFillRect( hps, &rectangle, CLR_DARKBLUE );
      origin.y=YSTART;
       origin.x=0L;
       GpiSetColor ( hps, CLR_RED );
       GpiCharStringAt ( hps, &origin, 7L, "Enter>>");
       origin.x=XSTART;
      WinEndPaint ( hps );     break;
    case WM_CHAR:
      hps=WinGetPS ( hwnd );
      GpiSetColor ( hps, CLR_YELLOW );
      fs=(USHORT) SHOR1FROMMP ( mp1 );
      if (fs & KC_CHAR) {
         ch[0]=(CHAR) SHOR1FROMMP ( mp2 );
         GpiCharStringAt ( hps, &origin, 1L, ch );
         origin.x+=XDELTA; }
      if (ch[0]=='/') {
         origin.y-=YDELTA;
         origin.x=0L;
         GpiSetColor ( hps, CLR_RED );
         GpiCharStringAt ( hps, &origin, 7L, "Enter>>");
         origin.x=XSTART; }
      WinReleasePS (hps);   break;
    default;
      return WinDefWindowProc ( hwnd, msg, mp1, mp2 );
      break;     } return 0L; }
/*·····*/
```

図-5 ウィンドウ表示（図-2 の1番上）を行うCプログラムの例

くのに使われる．これらは次の4グループに分かれている．

Dos 関数（152 種）…ディスク入出力，プロセス間通信（セマフォ，パイプ，シグナルなど使用），メモリ管理など．PM 用にも使用．

Kbd 関数（18種）…キーボード入力．PM ではメッセージ WM_CHAR を使うため不要．

Mou 関数（23種）…マウス入力．PM では不要．

Vio 関数（55 種）…画面表示用．そのうちの一部は PM でも使えるが，PM では Gpi 関数を使うのが普通．

なお，全部で248種ある API 関数のうちの92種は MS-DOS モードでも使えるところから，FAPI (Family API) と呼ばれる．FAPI 関数のみを使うプログラムは，保護モードでも MS-DOS モード（実モードおよび単独の MS-DOS）でも動く両モード・プログラムとなるが，それには bind というユーテリティを使って標準ライブラリ api.lib (MS-DOS モードで API をシミュレートするパッケージ）とリンクを行う必要がある．

以上のほか，我が国では，メーカ連合によって，かな漢字変換モジュール（日本語フロントエンド）へのインタフェースを含む日本語処理用標準 API の開発が進められている．

（3） ダイナミック・リンク・ライブラリ (DLL)

OS/2 には，ロード（ディスクからメモリへの）ときではなく，プログラムの実行中に必要なモジュールを動的にリンクする機能がある．しかしこうしてリンクできるのは，普通のプログラムではなく，エントリ・ポイントが外から見えるようになっている特別な構造のプログラム

で，それらを集めたものを DLL と呼ぶ．

この DLL は OS/2 自体でも応用プログラムでも広く使われる．DLL を使うと，実行時間はロードとリンクのために若干長くなることがあるが，ディスク内で同じプログラムが異なるロード・モジュールに重複してリンクされているという状態がなくなって，スペースが節約され，メモリ内でもそのとき呼ばれないモジュール（エラー処理ルーチンなど）はリンクされないところから，やはりスペースの節約になる．

DLL（複数使用可）の所在場所は config.sys ファイル内で libpath=α；β；γ の形でそのパス名を指定するが，OS/2 自体が使用する DLL ルーチン（全部で約 60 種）には次のものが含まれる．

doscall1.dll API の中の Dos 関数
kbdcalls.dll API の中の Kbd 関数
moucalls.dll API の中の Mou 関数
msg.dll メッセージ・システム・コール
xviocall.dll 拡張ビデオ・サブシステム
pmwin.dll PM 用のウィンドウ関数

こうした DLL ルーチンを使うプログラムを自分で作るには，implib（インポート・ライブラリ・マネジャ）という名のツールを使う必要がある．

以上みたように，OS/2 でのプログラミングは，①機械独立，すなわち移植性・互換性の高いプログラムを作る，②ウィンドウを使う，③MS-DOS でも動く両モード用を可能にする，④ダイナミック・リンクを使う，といった幅広い要求を満たすために，全体としては非常に複雑になっている．とくに②には，すでにのべたように，オブジェクト指向のプログラミングが必要である．

UNIX や MS-DOS の場合と同じく，OS/2 のプログラミングにはC言語がよく使われるが，Cで呼べるシステム・コール用の関数は，C言語自体のライブラリを入れないで，すでにのべたように 700 種以上あり，ウィンドウ用のメッセージは 90 以上もある．

またそれらを使うためのソフトウェア・ツールとしては，最低限次のコマンドの使い方をマスタしなければならない．

e 画面エディタ
cl Cコンパイラ（& リンカ）
link リンカ
cvp CodeView デバッガ
rc リソース・コンパイラ
dlgbox ダイアログ・ボックス・エディタ
iconedit アイコン作成用エディタ
fontedit フォント作成用エディタ
bind 両モード・プログラム用バインダ
implib ダイナミック・リンク用ライブラリアン
make コンパイル・リンクの自動化

6. ネットワーク機能と SAA

シングルタスキングの MS-DOS パソコンがサーバになりにくい（それを使用している間に，背景で他のコンピュータと交信できないため）のに対して，OS/2 ベースのパソコンは，UNIX システムと同じように，サーバになりうる．この機能を生かすべく開発されているのが，OS/2 の LAN マネジャである．

LAN マネジャでは，ネットワークの管理者を指定し，そのためのパスワードを登録して，ファイルや入出力機器の共用の指示やユーザの登録を行うなどの特権をもたせることができる．しかし，UNIX と違って OS/2 にはユーザ ID の概念がないから，一般ユーザが（管理者にはなれないものの）サーバ・パソコンを直接使えば，管理者が設定したファイルを消すことができるなど，セキュリティは十分ではない．

この LAN マネジャのもとで，サーバ（その名を server とする）上のあるディレクトリ（たとえば ourdir）の下のファイルをシェア名 ourshare で共用したいときには，まず管理者がサーバ上で

net share ourshare=a：¥ourdir

のようにコマンドを入れておく．このリソースには net admin コマンドで共用可能なユーザ名などを指定することも可能である．

このリソースの共用を許可されたユーザが，他のパソコン（ワークステーションという）で

net use d: ¥¥server¥ourshare

のようなコマンドにより，これを（実在しない）ドライブ d に割り当てれば，あとは d が手元のディスク・ドライブであるかのように扱えるようになる．同様にして，ファイル・ディレクトリのみならず，プリンタや通信ポートの共用も可能である．ワークステーションとしては，MS-DOS ベースのパソコンも使える．

この LAN マネジャでサポートできる物理的なネットワークは，メーカのサポートがあれば何でもよいが，たとえばイーサネットが選ばれ，使われるプロトコルは TCP/IP あるいは XNS であることが多い．

ところで，OS/2 の推進メーカのひとつである IBM 社では，SAA の一部として，その OS/2 の拡張版で，

LAN マネジャを通信マネジャと呼び，その下に LU6.2 プロトコル，APPC (Advanced Program to Program Communication) および NetBios を組み込んで，データベース・マネジャ（リレーショナル分散データベース機能）とともにサポートしている．ついでにいえば，SAA は，OS/2，(AS/400 シリーズの) OS/400 および大型機の MVS/VM の間で，ユーザからみた操作性およびソフトウェアの互換性を高めるための体系であり，その中には次のものが含まれる．LAN マネジャは下記の ③ に相当する．

① CUA＝Common User Application インタフェース

② CPI＝Common Programming Interface (すでにのべた API, FAPI, PM 中の GPI が C, COBOL, FORTRAN の各言語*で共通に使われ，GPI ではハードウェアの仮想化がはかられる．)

③ CCS＝Common Communication Support (これには OCA＝Object Content Architecture も含まれる)

表-3 OS/2 と MS-DOS および UNIX との比較

項目＼OS	MS-DOS	OS/2 V1.1	OS/2 V2.0 (仮称)	UNIX
対象プロセッサ	8086	80286	80386	多種多様
物理メモリ	1 MB	16 MB	4,096 MB	(システムによる)
仮想メモリ	なし	1,024 MB	64 TB	(システムによる)
マルチタスク	×	○		○
マルチユーザ	×	×		○
ユーザ ID/パスワード	×	×		○
マルチウィンドウ	(Windows)	PM		ベースは X
ネットワーク機能	(標準なし)	LAN マネジャ		TCP/IP など
日本語機能	◎	○		△
応用プログラムの価格	安い	(多分中間)		高い

7. おわりに

現在，ソフトウェア開発者およびユーザに出荷されつつある OS/2 の V1.1 はまだ 16 ビット型プロセッサ対応のものであり，またダイアログ・マネジャ（詳細不明）を含む V1.2 への拡張が計画されている版である．1990 年中には，32 ビット型プロセッサ 80386 対応の V2.0 (仮称) が出ると期待されるが，この版になると，(16 MB でなく) 4096 MB までの物理メモリ，4 kB 程度のページ単位のページング，64,000 MB (64 TB, テラバイト) までの仮想記憶，(今の MS-DOS モードに代わる) 複数の仮想 8086 モードなどがサポートされるものと思われる．そうなれば，80386 とともに OS/2 は 2000 年までは使用に耐える OS/2 になることであろう．

1990 年代ということで考えると，OS/2 と競合すると考えられる OS は UNIX である．そこで，表-3 に MS-DOS, OS/2, UNIX の比較を示す．OS/2 と対比して UNIX の特徴をあげると次のようになる．

(1) マルチユーザ (TSS) である．ユーザ ID およびパスワードでユーザは識別される．

(2) OS/2 が 20〜40 MB のディスクで動くのに対し，UNIX は数 10 ないし 100 MB 前後のディスクを必要とする複雑な OS である．ユーザによる環境設定も当然 OS/2 より複雑である．

(3) マルチウィンドウ機能が統一されていない．ベースはほぼ X ウィンドウだが，UII (UNIX International) グループが Open Look を，OSF グループは Motif を推進している．また一方で各メーカは独自のウィンドウ・システムをもつ．

(4) 日本語機能がまだ十分高くない．日本語のコードすら統一されていないのが現状である．

(5) 応用プログラムが高価である．OS/2 のものはまだ出ていないから比較しにくいが，MS-DOS のに比べると UNIX 用はほぼ 1 桁高い（それだけ高機能だが）．このため大学などでは無料ソフトウェア (Public Domain Software) が広く使われている．

さて，こうした背景のもとで，OS/2 と UNIX の競合がどうなるかは予想が困難であるが，筆者は，我が国の場合，ビジネス用のパソコンでは OS/2 が，エンジニアリング用のパソコン（ワークステーションと区別しにくい）では UNIX が多用されるものと予想する．それは，OS/2 のほうが現在の MS-DOS パソコンとの親和性がよく，より小さくて，ビジネス・ユーザにはより扱いやすいと思われるからである．

しかし OS/2 を普及させるには，PM すなわちマルチウィンドウ対応の応用プログラムの種類と数を増やし，安くしなければならない．この点で今後心配されるのは，オブジェクト／メッセージ指向のプログラミングが現在のプログラマにとってはかなり難しく，プログラマの再教育が必要なことである．Whitewater 社の Actor や NeXT 社のインタフェース・ビルダの

* NEC では OS/2 特有のマルチタスキング文を組み込んだ N$_{88}$ BASIC を供給している．

ような開発ツールの充実も望まれる．

一方，エンド・ユーザに対しては，PM という形で操作性の統一およびソフトウェアの互換性（ハードウェア独立性）がもたらされるのはよいが，環境設定などをいかに容易にするか，LAN につながれた OS/2 パソコンのセキュリティをいかに高めるかなどの問題が残っている．表-2 に示した config.sys は一見すると柔軟性のためにはよさそうに見えるが，個々の応用プログラムを登録するときに，その内容変更をユーザまかせにするのでは危い．デバイス・ドライバなどは，config.sys によらずに，各応用プログラムごとに着脱可能にする必要がある．

こうして OS/2 には問題はあるものの，1990 年代が進むにしたがって，さらに改良・強化され，パソコン・ユーザに快適な利用環境を提供してくれる OS に成長することが期待される．

参 考 文 献

レトウィン，G.（三浦訳）：OS/2 システム・アーキテクチャ，アスキー（1988）．

クランツ，J.（千田訳）：OS/2—特徴・機能・アプリケーション，近代科学社（1989）

富士ソフトウェア：OS/2 プレゼンテーション・マネジャ，富士ソフトウェア（1989）．

Urlocker, Z.: Whitewater's Actor: An Introduction to Object-Oriented Programming Concepts, Microsoft Systems Journal, Vol. 4, No. 2, pp. 33-44 (1989).

北郷：90年代に向けて加速するオブジェクト指向言語，日経バイト，No. 64, pp. 182-195 (1989).

石田・鷹野：OS/2，共立出版（1990）．

マ ニ ュ ア ル

＜IBM OS/2 J 1.1（基本版）＞
（フロッピーは 13 枚）
導入と基本操作の手引き
ご使用の手引き
システム・エディタご使用の手引き
プログラマツール・キット
　プログラミング概説
　実行可能プログラムの作成
　プログラミングの手引き
Command Reference

＜Microsoft OS/2 SDK（1.06 の次の最終版）＞
（所要メモリ＝2.5 MB；フロッピーは 13 枚）
User's Guide
Programmer's Reference (Volume 1, 2, 3)
QuickHelp User Guide
Programming the OS/2 Presentation Manager (by Charles Petzold)
Presentation Manager Softset (link, bind, implib, lib, rc, dlgbox, fontedit, iconedit)

＜日本電気 OS/2 1.1＞
（所要メモリ＝4.6 MB；フロッピーは 11 枚）
スタート・ガイド
コマンド・リファレンス
システム・エディタ・リファレンス
トレーニング・ガイド
ユーザーズ・リファレンス（1, 2）
プログラマーズ・ガイド（1, 2）
プログラマーズ・リファレンス（1〜5）
プログラミング・ツールズ
デバイス・ドライバ・ガイド
N88 日本語 BASIC (86) ユーザズ・マニュアル
日本語入力ガイド

＜MS-C コンパイラ（C 5.1）＞（フロッピーは 5 枚）
ユーザズ・ガイド
ランゲージ・リファレンス
ランタイム・ライブラリ・リファレンス（1, 2）
グラフィック・ライブラリ・リファレンス
CodeView（デバッガ）& ユーテリティ（link, lib, make）ユーザズ・ガイド
ミックスド・ランゲージ・プログラミング・ガイド

（平成元年 10 月 3 日受付）