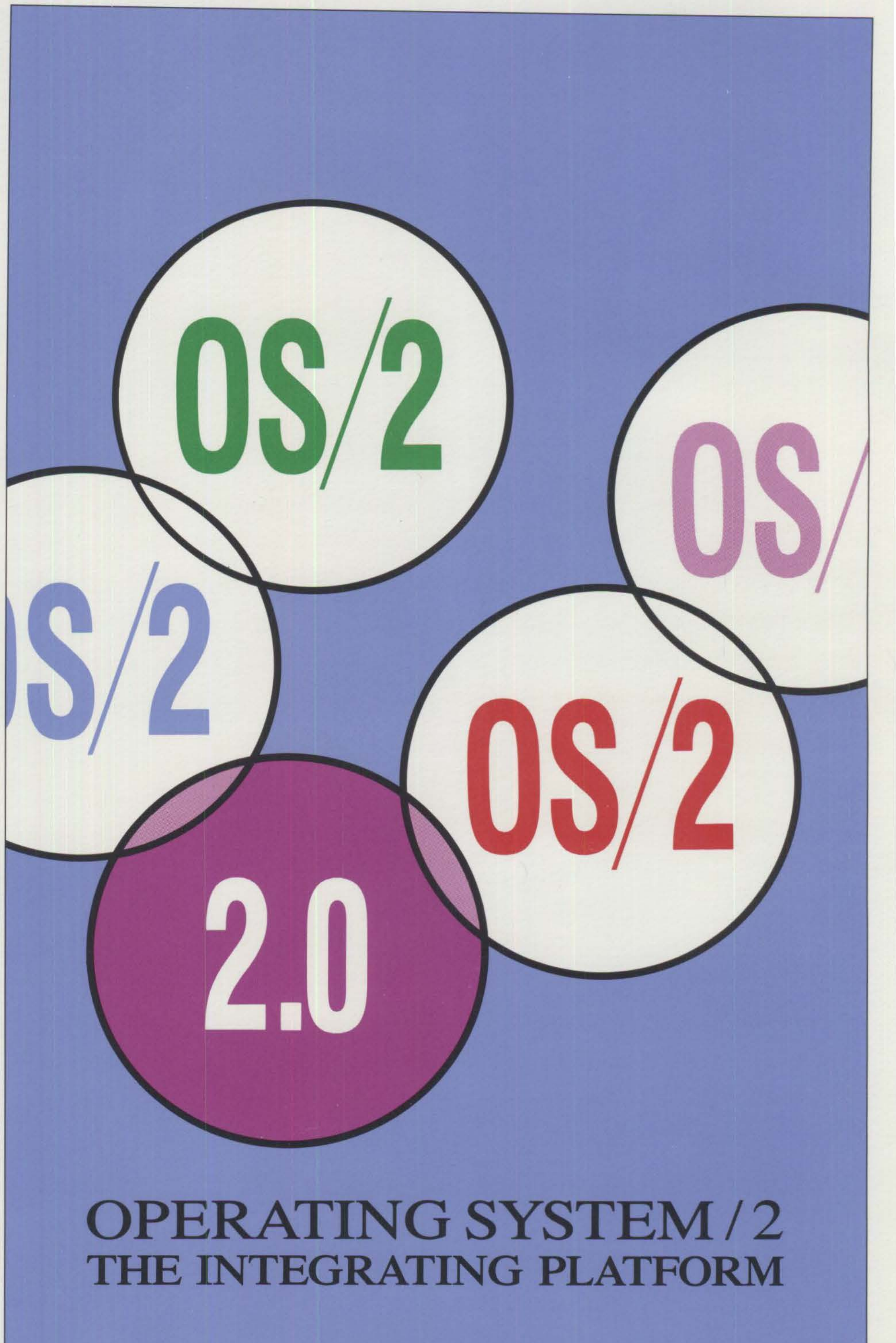
Personal Systems

January 1992



OPERATING SYSTEM / 2
THE INTEGRATING PLATFORM

IBM Personal Systems Technical Solutions

IBM®

$10.00

IBM Personal Systems Technical Solutions is published quarterly by IBM United States technical support center, International Business Machines Corporation, Roanoke, Texas, U.S.A.

Editor	Libby Boyd
Communications	Elisa Davis
Cover Design	Corporate Graphics
Illustrator	Bill Carr
Publication Services	The TDA Group
Typesetting	Genesis Publications
Manager	Bill Hawkins

To correspond with *IBM Personal Systems Technical Solutions*, please write the editor at:

IBM Corporation
Internal Zip 40-A2-04
One East Kirkwood Blvd.
Roanoke, TX 76299-0015

To subscribe to this publication, request IBM System Library Subscription Service (SLSS) from an IBM branch, and specify form number GBOF-1229.

Printed on recycled paper

Contents

Hardware

1 Additions to the IBM PS/1 Family

7 IBM LaserPrinter 4029 Series Print Quality Enhancements

Software

13 OS/2 2.0: The Integrating Platform

19 Multiple Virtual DOS Machines

30 IBM OS/2 LAN Server 2.0

41 OS/2 2.0 Memory Management

45 Coding for Performance Under OS/2 Version 2.0

49 Extending the Functions of OS/2 REXX

54 Protecting User Exits Under OS/2 1.X

62 GDDM-OS/2 Link

65 IBM Upgrade Enhanced Install Utility/DOS 5.0

67 Advanced Peer-to-Peer Networking: An Overview

73 Using IBM SAA Networking Services/2

80 The AAI Family of Products

83 Securing the Enterprise Workstation

Random Data

87 Little Solutions

89 New Products

Trademarks

Additions to the IBM PS/1 Family

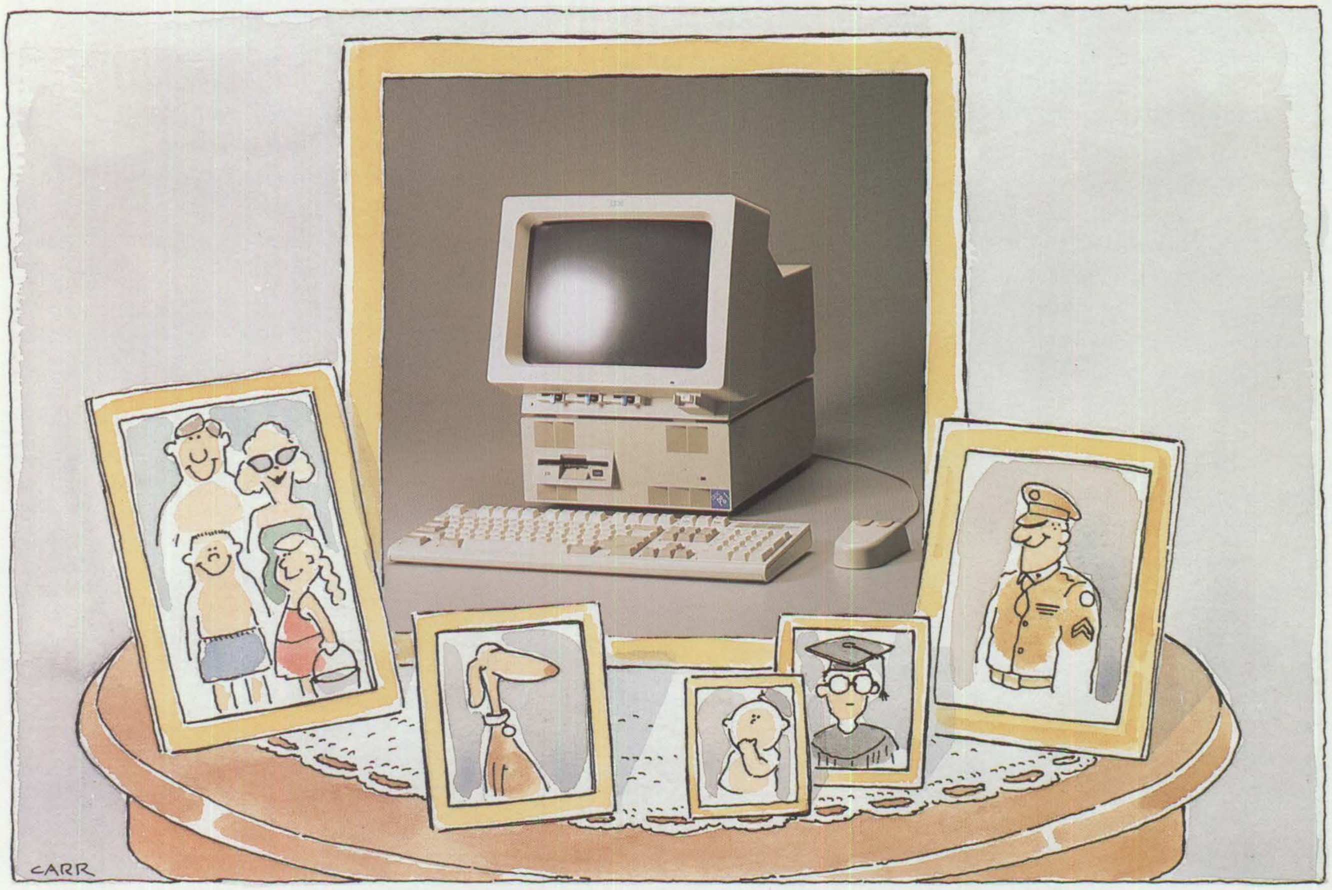
Brian Dalgetty
IBM Corporation
Lexington, Kentucky

IBM has recently announced several new PS/1™ 386 SX models with expanded hardware and software capabilities that should appeal to more technical personal computer users. Features such as faster processing, increased memory, hard disk storage up to 250 MB, and support of networking cards make the 386 SX models ideal for small business owners. In this article, we'll take a detailed look at the hardware included, and discuss the software changes and improvements.

PS/1 History

In 1988, market research indicated that the personal computer was positioned to become the household product of the 1990s. Several other consumer electronics products (such as the VCR, color TV, and CD player) were studied to determine what criteria needed to be met before they became prevalent in the home.

PS/1 Development: A small team consisting of marketing personnel, development engineers, product planners, and manufacturing representatives was pulled together in Lexington, Kentucky, to design a serious personal computer for the home. Before any hardware was modeled, this team surveyed thousands of users through focus groups and phone questionnaires and assembled a list of the core design requirements.



Following is a summary of the "shopping list" used to develop the PS/1:

- Users are looking for a "total solution" – they do not want to be computer experts.
- The computer must be easy for the whole family to use and accommodate different experience levels.
- It must be easy to buy, readily available, and there should not be too many decisions to make.
- The product must be powerful and expandable, run all the common business software, and be able to grow as users' needs change.
- Support from the manufacturer is also important; many users said they wanted to have a way to ask IBM questions and have a service plan that can support a busy schedule.

The initial PS/1 offering took 18 months to develop and was announced worldwide in the third quarter of 1990. Surveys conducted after the announcement indicated that user satisfaction was extremely high: above 95 percent. Most purchasers were novices or new users, confirming that the target market was reached. The data also showed that to appeal to more technical users, PS/1 models with more hard disk storage, greater system memory, and faster processor speeds were needed.

Development of new PS/1 models with an 80386 processor became an immediate priority. These new models were developed in only eight months, and are described as follows.

PS/1 386 SX Models

The PS/1 386 SX is a worldwide product with many different models. All models come with a color IBM VGA Photographic display, IBM mouse, keyboard, a system unit containing an 80386SX processor with a 40, 80, or 129 MB hard disk drive, as well as the core IBM PS/1 software. Some models are available with two internal AT® adapter card slots and Microsoft® Windows® 3.0. IBM has also announced that it intends to have an IBM PS/1 386 SX model preloaded with OS/2® 2.0 available in the first half of 1992. The models currently available in the U.S. are shown in Figure 1.

PS/1 386 SX Hardware

SIB Pack: Being part of the PS/1 family, the 386 SX models provide a complete solution of hardware, software, service, and support. Everything you need comes in a single box – referred to as a "Solution In a Box" or the SIB pack. The same SIB pack is used for all PS/1 models and accommodates the different system units, displays, and various software packages included. The box is designed to fit into most car trunks, and a Kraft carton was chosen because it is easier to recycle.

Inside the SIB pack is a top tray that holds all the software and user documentation. It includes the *Getting Started* booklet already open to the page that illustrates how to set up the system. Under the top tray are the four basic system components: the mouse, keyboard, display, and system unit.

Many users ask why a printer is not included. All PS/1s come equipped with a parallel port and printer drivers to support almost any printer on the market. With the wide range of printers available, IBM decided to leave this decision up to the individual. Besides, our packaging engineers would never speak to us again if we tried to squeeze any more into the SIB pack. Customers can also purchase a PS/1 printer specifically designed to work with PS/1 systems.

Setup: It's extremely easy to set up the PS/1 386 SX. Most users will be up and running within 15 minutes. Icons on the rear of the system unit indicate where to plug in the display, mouse, and keyboard. There is only one power cord to be plugged in, and there is a single power switch on the display, both of which minimize getting started. The hard disk is already formatted and software customized

Model Number	Machine Type	Description
C42	2121	40 MB Hard Disk 2 MB System Memory (RAM)
B82	2121	80 MB Hard Disk 2 MB System Memory (RAM) Internal AT Expansion Slots Microsoft Windows 3.0
C92 (Sears model)	2121	129 MB Hard Disk 2 MB System Memory (RAM) Microsoft Windows 3.0

Figure 1. PS/1 386 SX Models Available

for the PS/1 is already installed. Upon being powered-up, all PS/1 systems are self-configuring and recognize all attached options without having to run a setup routine.

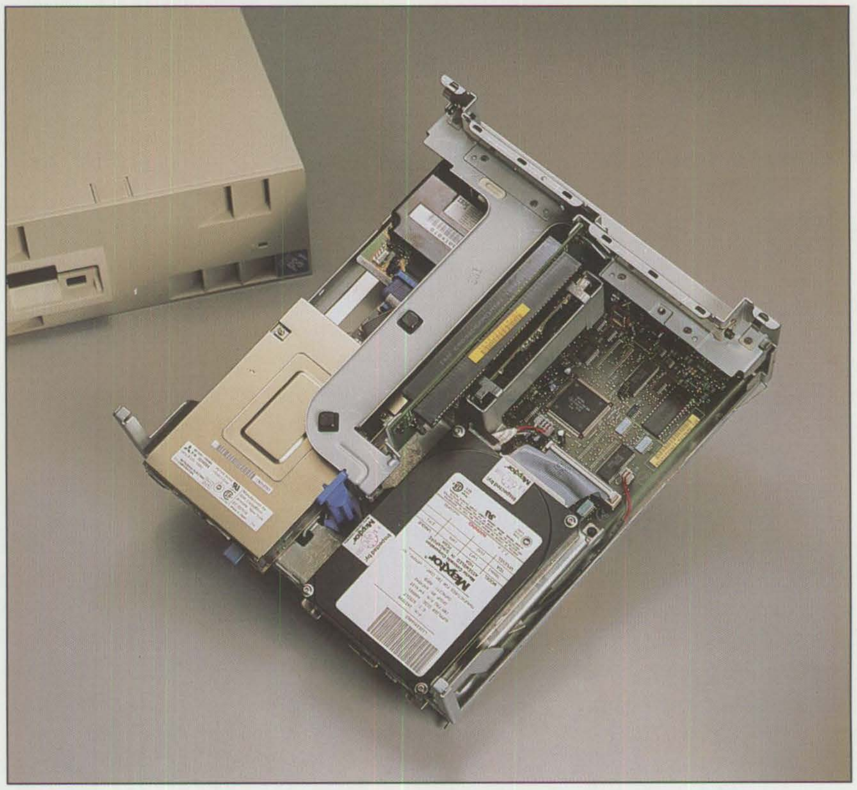
PS/1 386 SX System Unit: This section contains most of the hardware differences between the new 386 SX models and the original PS/1 offering.

One of the features users appreciated most about the PS/1 was its small footprint. The 386 SX system unit is about the same size as a CD player, and like the initial PS/1, it takes up 10.8 by 13.8 inches of a desktop and is 3.3 inches high. The system unit with AT expansion slots is 4.7 inches high and weighs slightly more than 11 pounds.

System Unit Access: Getting inside the system is necessary to add PS/1 options or plug-in internal expansion cards. Like the original PS/1 where the front bezel snaps off and the top cover slides forward, access into the system unit is easy and requires no tools. On the taller 386 SX models, system unit access is even easier. The cover and bezel have been integrated into one piece and can be removed by pressing the cover release button.

Covers Off: Inside the system unit, the major components can be identified to get to the heart of the personal computer. At first glance, the system unit appears to be missing the power supply. The supply actually resides in the display and only low voltage (36 volts) is provided to the system. Power is distributed to internal components by a power card mounted vertically on the center of the system board. Using distributed power helped the PS/1 designers achieve the small footprint.

DASD: The Direct Access Storage Devices (DASDs) are located on the front shelf inside the system unit. All PS/1 models come standard with a 1.44 MB, 3.5-inch diskette drive. The 386 SX models are designed to accept diskette drives that utilize "media sense," and have up to 2.88 MB.



The hard disk sits to the right of the diskette drive and comes with 40, 80, or 129 MB of storage. This hard disk interface has also changed. The initial PS/1 used family 1-type drives; the 386 SX models have a standard AT interface, compatible with off-the-shelf hardware. A second hard disk can be mounted above the first in the taller 386 SX model (see the section "Adding Options"). All the DASDs are mounted on plastic rails that slide into place. Combined with the cover removal, this makes inserting and removing hard disks as easy as swapping diskettes.

At the back left corner inside the system unit is an internal 2400 bps modem, standard on all U.S. models. With the modem, users can take advantage of the IBM PS/1 Users Club and "talk" directly with IBM or other PS/1 owners. Non-U.S. models come with a serial card, and country-specific modems can be attached externally if desired.

Adding Expansion Cards: On the 386 SX models accepting internal expansion cards, two 11-inch-long AT cards can be attached horizontally in the top of the system unit. The adapter card slots inside the 386 SX models utilize a pure AT bus running at 8 MHz; this too increases the compatibility with existing hardware. An optional PS/1 expansion unit provides three AT slots to those models without internal capabilities.

Networking Support: In addition to providing internal AT slots, compatibility testing for the 386 SX models has been conducted on the following IBM networking cards:

- IBM 3278/79 Emulation Adapter
- IBM Enhanced 5250 Emulation Adapter
- IBM Token-Ring Network 16/4 Adapter
- IBM Token-Ring Network 4 Adapter II
- IBM PC Network Adapter II
- IBM PC Network Baseband Adapter

Novell certification has also been obtained using IBM and Novell® networking cards running under an Ethernet™ and Token-Ring environment. Networking support on the PS/1 386 SX systems provides small businesses an opportunity to assemble a very affordable office solution.

System Board: Attached to the bottom of the system unit is the main system board. All the new PS/1 models use an 80386 SX microprocessor running at 16 MHz. These models come standard with 2 MB of Random Access Memory (RAM), upgradable to 6 MB on the system board using a 4 MB PS/1 memory card option. U.S. models have 256 KB of Read-Only Memory (ROM) that contains the BIOS and PS/1-specific system programs. Space for an additional 256 KB of ROM is available and used on non-U.S. models for country-specific code.

A 16-bit VGA controller provides display signals and can support up to 640 x 480 pixels with 256 colors. As an enhancement to the initial PS/1, four modules are now contained in a single VLSI chip, resulting in a significant cost savings. The 386 SX systems have switched to a real-time clock with a replaceable lithium battery.

Other System Components: All 386 SX models come with a 12-inch Color Photographic VGA display. The term "photographic" was selected so those unfamiliar with VGA could relate to the picture-like quality displayed. All systems also come standard with a two-button mouse and a PS/1 keyboard. In order to fit better in a space-constrained environment, the keyboard size and weight have been reduced without sacrificing function. The PS/1 keyboard uses the same 101-key layout that has become standard on most personal computers.

All the new PS/1 models use an 80386 SX microprocessor running at 16 MHz.

Adding Options: In order for users to expand their PS/1 systems as they become more experienced or as their needs change, the following options are available for all PS/1 386 SX models:

Memory Expansion Cards: PS/1 memory cards, 2 MB and 4 MB (about the size of a credit card), can be plugged into the front of the system unit directly into the system board. The 512 KB memory card (sold with the original PS/1) can also be used.

Hard Disk Options: The 80 MB or 129 MB PS/1 hard disk option is used when additional hard disk storage is needed. For PS/1 386 SX models with internal slots, these options can be used to add a second hard disk. The second drive is mounted on top of the first and is connected with a dual power and signal cable. With dual drives installed, users can access up to 258 MB of storage.

Expansion Unit Option: For models without internal slots, an expansion unit can be added that provides three AT slots. This option attaches to the top of the system unit, increasing the overall height 2.5 inches, but maintaining the small footprint. Two cards up to 11 inches and one up to 9.5 inches can be accommodated with the expansion unit installed.

5.25-Inch Diskette Option: Like the original PS/1, a 360 KB or 1.2 MB 5.25-inch diskette drive can be added to the bottom of the system unit. The overall system unit height increases to just over 5 inches.

Audio Card and Joystick: An audio card capable of producing four voice-synthesized sounds plus digital audio and supporting up to two joysticks can be added to all PS/1 models. The audio card fits inside the system unit and mounts above the internal modem. Besides the joystick connector, the audio card contains a Musical Instrument Digital Interface (MIDI), which supports the attachment of musical instruments. Music and entertainment software is also included with this option, enabling these new features to be used immediately after setup.

System Unit Upgrade: As part of a continuing effort to offer complete expansion, owners of the original PS/1 can upgrade their systems to the 386 SX system unit with 80 MB hard disk and AT expansion slots. The upgrade is purchased directly from IBM, and owners must return their existing hard disk system unit for credit.

Software Overview

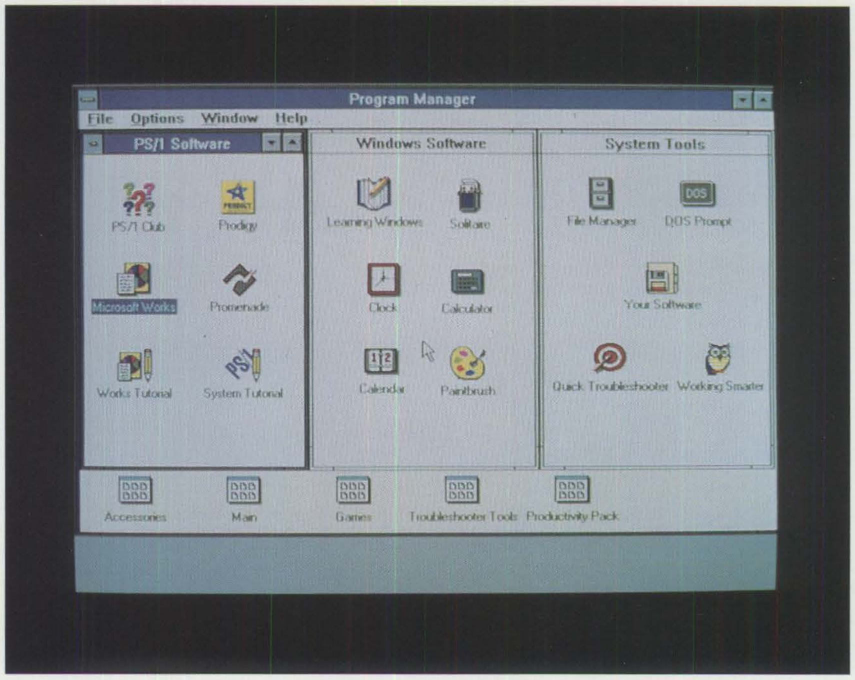
If hardware is the heart of a system, it's the software that makes it come to life. All PS/1 systems come with many preloaded software applications that have been customized to work together. Each application has been integrated, tested, and guaranteed to perform its stated function. This provides a significant advantage over personal computers that just include an off-the-shelf release with the system. All 386 SX models have the following core software.

System Menu: After the PS/1 is powered up, the system menu displays four selectable quadrants. The system menu is a Graphical User Interface (GUI) designed so users can access all applications simply by using the mouse or keyboard to select a picture. The quadrants shown on the initial system menu are Information, Microsoft Works, IBM DOS, and Your Software.

Information: This quadrant contains information on how to use all the capabilities of the PS/1. The programs contained in this section are online tutorials or services that can be accessed using the internal modem.

System Tutorial: Surveys indicated that few users like using personal computer manuals; most prefer to learn by using the system. With the online tutorials and help screens, users can have their wish. The system tutorial covers the basics of personal computing, identifies system components, and defines commonly used terms.

Works Tutorial: Lessons on the word processor, spreadsheet, database, and communications programs within Microsoft Works are contained in the Works Tutorial. These lessons are



topic-specific and can be completed in any order.

Users' Club: Personal support was one of the key requirements users requested. The Users' Club is an online service accessed via the internal modem for PS/1 users to communicate directly with IBM. The Answer Bank, Info Exchange, and Write To Us contain answers to questions about PS/1 hardware and software.

PRODIGY™ and Promenade: Additional online services can be accessed via modem. PRODIGY contains information on numerous areas, from sports and investments, to shopping and planning vacations. Promenade is an education and entertainment service offering online computer classes and downloadable software.

Microsoft Works: This productivity tool contains a word processor, spreadsheet, database, and communications program. These tools are easy to use and are integrated so data can be freely interchanged.

IBM DOS: This quadrant provides access to a DOS shell from which most common DOS commands can be executed. This shell offers unique options including "Install Software on Your Fixed Disk" and "Customize How System Starts." The latter option allows users to alter how the system boots up, thus revealing an interesting aspect of the PS/1 design.

A DOS kernel, COMMAND.COM, AUTOEXEC.BAT, CONFIG.SYS, and other files controlling the System Menu have been placed in ROM. The customization screen allows users to specify where these files are located; the system default is the hard disk. If the AUTOEXEC.BAT or system files ever get corrupted or deleted, the system can be powered off and on with the two mouse buttons pressed, and the system will boot from ROM. This allows users to recover the system, returning it to the initial factory settings.

Your Software: This final quadrant presents each subdirectory as a file folder and displays the executable files of an open folder. Microsoft Windows (available on some models) can be accessed from the "Your Software" root directory file folder.

Microsoft Windows: Microsoft Windows is a GUI with an icon-based Program Manager for running software applications. It contains multitasking capabilities so applications can be run concurrently. Like the previously mentioned software, Microsoft Windows on the 386 SX models has been customized to work specifically on the PS/1. Following is a summary of the enhancements:

- All PS/1 applications are grouped under the Program Manager and are identified by icons.
- A "Your Software" area similar to that contained in the system menu has been added to the Program Manager and provides a simple method to create additional groups.
- A Win Reset Restore feature was added to the Program Manager screen to restore icons back to their original settings.
- The Windows productivity pack has been included. It contains an online tutorial for using Windows, a troubleshooting guide for diagnosing problems, and an application that demonstrates techniques for working smarter.
- Windows is configured to start in 386 enhanced mode to provide multiple DOS environments for non-Windows applications.

The 386 SX models come with a well-integrated package of software applications. The PS/1 system menu ties these applications together, and makes them easy to access and ready to use. Like the hardware, the PS/1 software system can be expanded to run almost any DOS application.

Like the hardware, the PS/1 software system can be expanded to run almost any DOS application.

System Support

All PS/1 systems come with a one-year warranty on everything contained in the SIB pack. That includes not only hardware and software, but also every claim made in the PS/1 documentation. PS/1 owners use the PS/1 Express Maintenance service to correct problems arising while the system is in warranty. Express Maintenance delivers replacement components, usually within 48 hours, directly to the owner. This service sets a new standard for user support and has been very well received.

Expanded Retail Outlets

IBM is increasing its PS/1 outlets to three times more mass-merchandise business partners, including national and regional consumer electronic stores, catalog showrooms, and office equipment retailers. The new models are marketed by IBM PC superstore business partners as well as the more retail-oriented IBM authorized personal computer dealers.

Summary

The new PS/1 386 SX models directly address the requests of PS/1 buyers and potential users for more power, greater expandability, and compatibility. With its improved hardware and software capabilities, more technical personal computer users will soon become PS/1 owners.

ABOUT THE AUTHOR

Brian Dalgetty is a development manager for the PS/1 organization at IBM in Lexington, Kentucky. In 1984, he joined IBM in Charlotte, North Carolina, and worked in serial printer development designing dot-matrix print heads. In 1989, Brian transferred to Lexington to work on the initial PS/1. During the development of the 386 SX product line, Brian served as the lead technical coordinator. He is also the PS/1 User Group interface and manages all PS/1 User Group activities. Brian received a BS and MS in mechanical engineering from the Massachusetts Institute of Technology.

IBM LaserPrinter 4029 Series Print Quality Enhancements

Gary Overall and Phil Wright
Lexmark International Inc.
Lexington, Kentucky

This article describes the technology behind the print quality of the LaserPrinter 4029 Series.

Printing characters and images may be considered an art form; therefore, the perception of quality is in the eye of the beholder. Some of the factors that affect this perception are the darkness of the characters, the amount of raggedness or variation in the edges of characters, the integrity of the digitized character compared to the intended outline, and for images, the number of gray levels. The IBM LaserPrinter 4029 Series sets a new print quality standard for desktop laser printers by incorporating the following technology advances to optimize these factors.

- Print Darkness control: Allows adjustment of overall stroke thickness to meet individual preferences.
- Print Quality Enhancement Technology (PQET): Eliminates the digitization stairsteps and enhances character serifs in 300 dot-per-inch (dpi) mode.
- 600 dpi: Provides true high resolution 600 x 600 dpi printing to more accurately represent intended character features, and increases the number of representable gray levels in image printing.

These three components are part of the advanced printing solution developed by Lexmark International, providing unparalleled print quality to meet customer demands, from the entry-level LaserPrinter 5E up to the LAN-capable LaserPrinter 10L.

Laser Printing Primer

To understand the terms covered in this article, it is helpful to understand the concept behind the operation of a laser printer such as the IBM 4029 Series. This explanation is not intended to cover all the specifics involved in the electrophotographic printing process; discussions of toner charging, development, photoconductors, and coronas will therefore be avoided.

The printer takes information from a computer – in a printer language such as PostScript®, IBM Personal Printer Data Stream (PPDS), or Hewlett-Packard® Graphics Language (GL) – and builds a bitmap, a logical representation of the page in digital memory. The bitmap represents a large matrix of printer dots or pels. A 300 dpi printer has 90,000 of these pels per physical inch on the page. The pels are placed on the page by a laser that sweeps from the left to the right of the paper (conceptually, not physically). The information necessary to turn the laser on and off during each sweep is sent from the bitmap so that each bit represents a pel on the page. With each sweep of the laser, the paper is moved in the printer by a fixed amount (1⁄300 inch for 300 dpi printers). This paper movement allows the next laser sweep to create a new scan of information. The accumulated scans create the completed page. Toner particles are then placed on the page where the laser was turned on and are melted, or fused, to the paper.

Print Darkness

The LaserPrinter 4029 Series features a control panel adjustment for Print Darkness, permitting user selection of stroke thickness. The Print Darkness setting is simply an adjustment of the printed pel size. As the scan of the laser passes within the boundaries of a pel, the laser is turned on for only a fraction of the pel time. This fraction is adjustable and is determined by the Print Darkness setting. The longer the laser remains on for a pel, the larger the printed dot. Figure 1 gives the relative widths of single-pel-wide lines for a typical LaserPrinter 4029. Note that both horizontal and vertical lines have equal widths.

PQET Theory

The PQET feature is included with all models of the 4029 and requires no additional memory upgrades. PQET intercepts the 300 x 300 dpi data as it is being sent from the bit-

Resolution	Darkness Setting	Laser Activation Time/Pel Time	Line Width
300x300 dpi	Light	202ns/539ns (3⁄8)	132µm
300x300 dpi	Normal	337ns/539ns (5⁄8)	142µm
300x300 dpi	Dark	472ns/539ns (7⁄8)	172µm
600x600 dpi	Light	34ns/168ns (1⁄5)	98µm
600x600 dpi	Normal	67ns/168ns (2⁄5)	105µm
600x600 dpi	Dark	101ns/168ns (3⁄5)	121µm

Figure 1. Widths of Single-Pel-Wide Lines

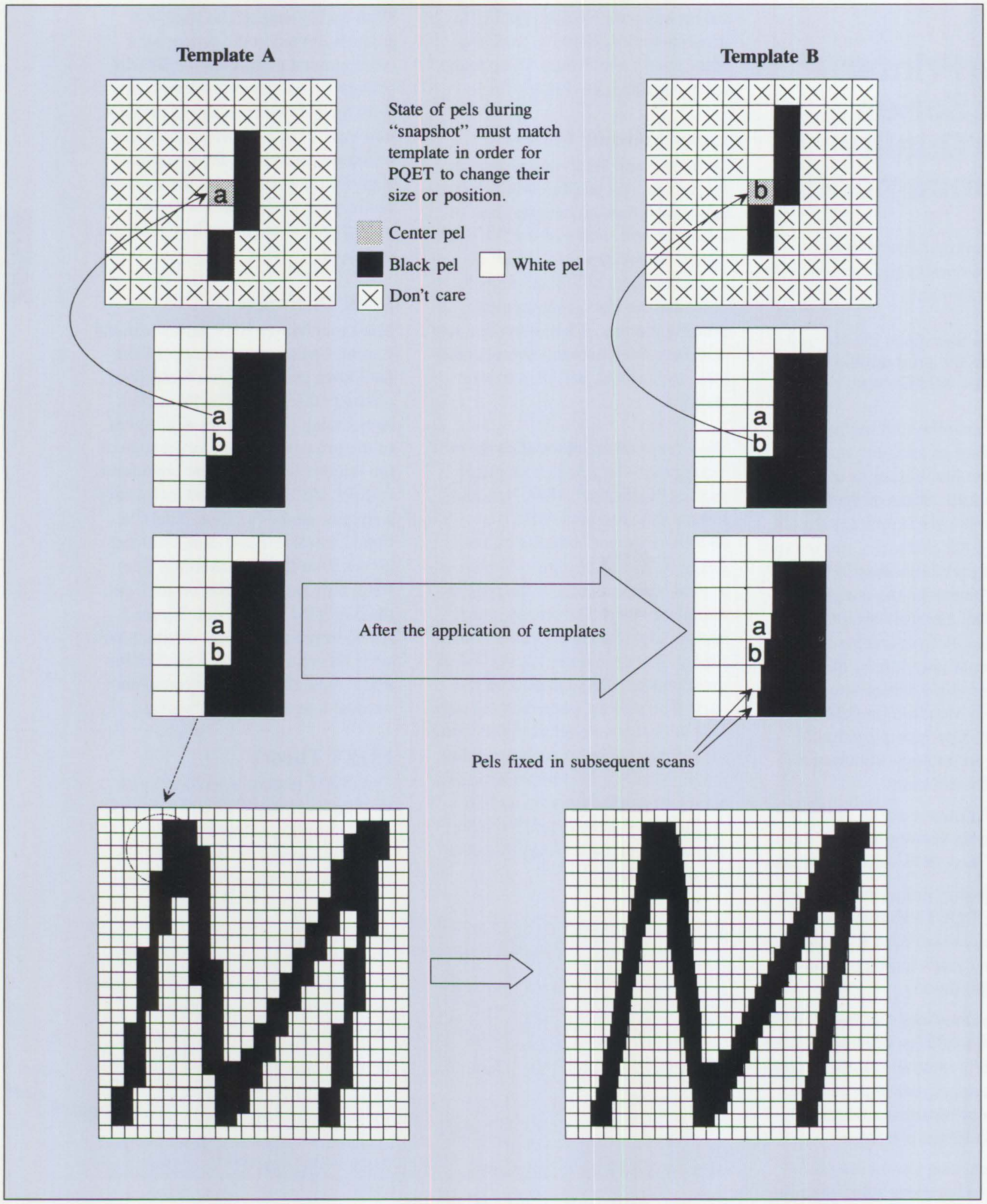

Template A
Template B
State of pels during "snapshot" must match template in order for PQET to change their size or position.
Center pel
Black pel
White pel
Don't care
a
b
After the application of templates
Pels fixed in subsequent scans

Figure 2. Template Matching

map memory to the laser. An electronic window is passed over the bit representation of the page to be printed, and for each pel time a "snapshot" of this window is taken and compared to more than two hundred matching "templates" stored in the 4029 PQET logic. If any of the templates match the state of the pels in this snapshot, the center pel in the window has its on-time or position altered to more closely approximate the intended edge. Figure 2 illustrates this process as the letter "M" is corrected by the PQET logic. The matching templates contained in the PQET logic are designed to detect angled lines, serifs, and curves without adversely affecting image printing (images are printed using digital halftone techniques, creating near circular dots, rather than angled lines).

PQET Implementation

PQET is implemented in an Application Specific Integrated Circuit (ASIC) on the 4029 electronic circuit card. As mentioned previously, it intercepts the 300 x 300 dpi data as it is sent to the laser from the 300 x 300 bitmap of the page to be printed. PQET divides each of the 300 x 300 dpi source pels into "slices," which are individually controlled to achieve the smoothing effect described earlier. As each pel reaches the center of the window, it is categorized by the PQET logic. Since the 4029 allows the control of Print Darkness while PQET is enabled, the type of modulation applied to this center pel is determined by the Print Darkness setting. Figure 3 shows the first scans of the letter "M" when using the light, normal, and dark modes.

Since PQET can only change the print addressability in the horizontal direction (the direction scanned by the laser), smoothing of near-horizontal lines and near-vertical lines utilizes two different types of pel modulation techniques. Near-vertical lines can be smoothed by simply starting a pel early or delaying a pel by a number of "slices" (depending on the location of the pel in relation to the step being repaired). However, near-horizontal lines are smoothed by modulating the laser, and combining the energy of two adjacent scans to create a smaller dot positioned between scans. Figure 4 illustrates control of the laser to repair near-horizontal and near-vertical lines.

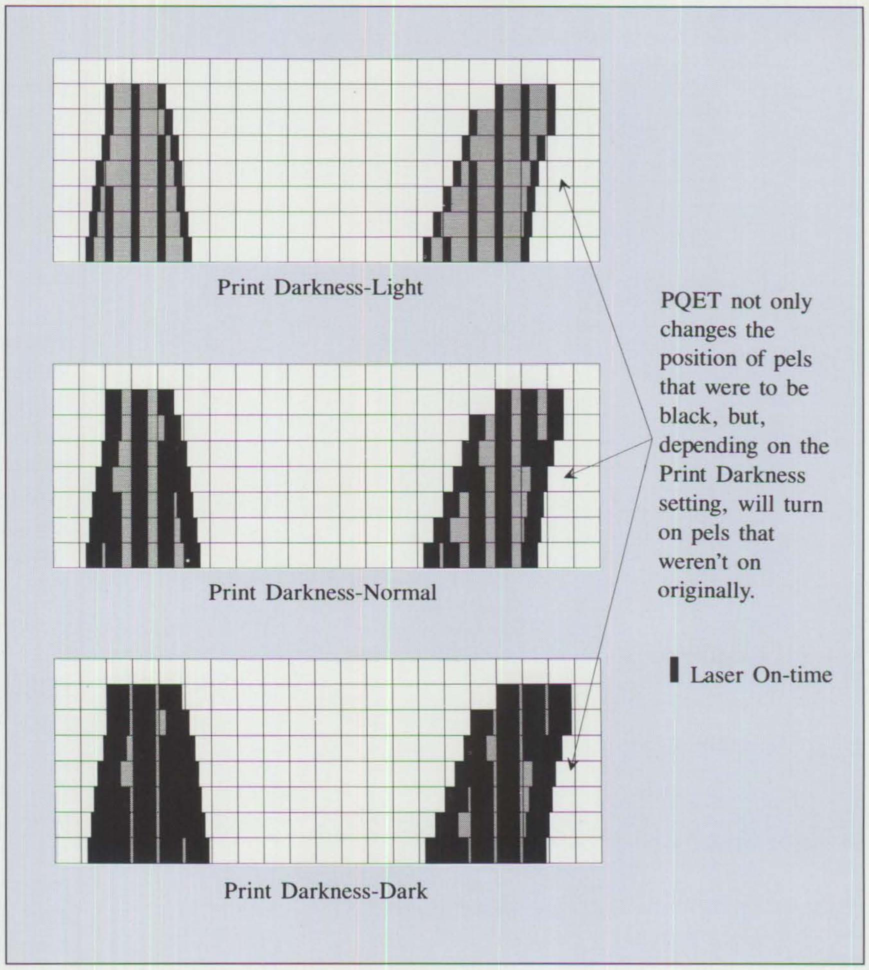

Print Darkness-Light
Print Darkness-Normal
Print Darkness-Dark
PQET not only changes the position of pels that were to be black, but, depending on the Print Darkness setting, will turn on pels that weren't on originally.
▌ Laser On-time

Figure 3. PQET Interaction with Print Darkness

Mechanics of 600 dpi

As an alternative to PQET, the 4029 provides the capability of true 600 x 600 dpi resolution. With the PostScript option and a 4 MB memory upgrade, 600 dpi can be selected. Compared to 300 dpi, 600 dpi requires 4 times the number of pels per page. Since the PostScript interpreter must build a bit image of the entire page before starting the printing process, the bitmap memory requirement for 600 dpi is 4 times greater. In 300 dpi mode, the laser scans horizontally across the page and begins a new scan every 1⁄300 inch. Increasing mirror motor speed (which controls laser scanning speed) and decreasing the paper transport speed results in a scan-to-scan spacing of 1⁄600 inch, producing 600 dpi vertical resolution.

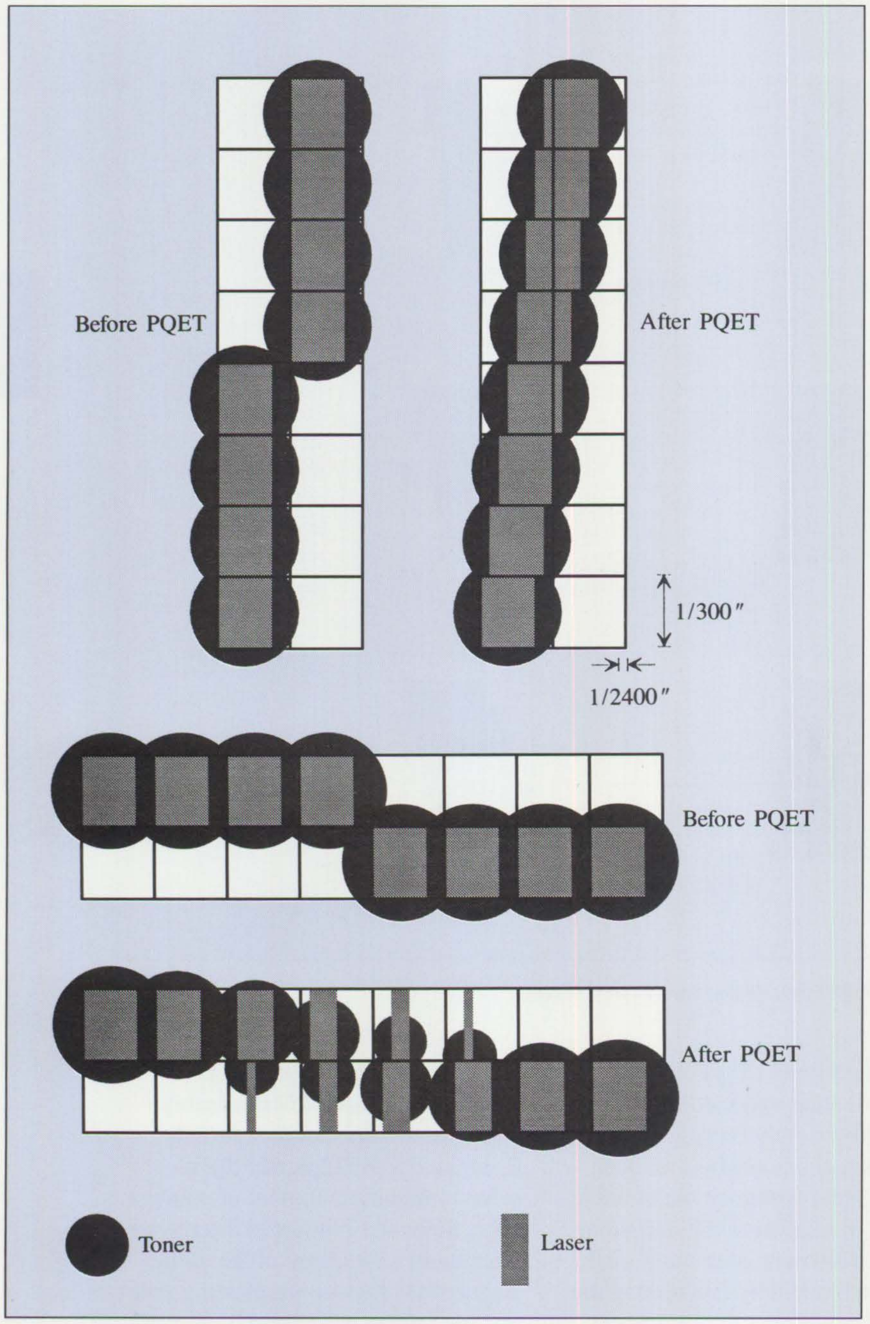


Figure 4. PQET Edge-shifting Techniques

Resolution	Mirror Motor Speed	Paper Transport Speed	Pel Time
300 dpi	6000 RPM	2 inches per second	539 ns
600 dpi	9600 RPM	1.6 inches per second	168 ns

Figure 5. Summary of 300 dpi versus 600 dpi Timings

A 600 dpi horizontal resolution is achieved by adjusting the pel time so that the laser travels 1⁄600 inch horizontally for each pel. Figure 5 summarizes the timings of 300 dpi versus 600 dpi. Note that *pel time* is the amount of time required for the laser to travel between consecutive pels.

The LaserPrinter 4029 differs from some competitive offerings by outputting at 600 dpi horizontally as well as vertically. By contrast, several competitive "high-resolution" printers give higher resolution horizontally but lower resolution vertically. Ownership of the print technology gives Lexmark the flexibility to control all parameters affecting print quality.

600 dpi Characters

Fonts used by the PostScript interpreter are defined by mathemathical descriptions of their outlines. When processing a page and creating a bitmap from the outlines, the interpreter uses either a 600 dpi or a 300 dpi bitmap, depending on the current resolution setting of the printer. Having 4 times the number of available pels in 600 dpi, the interpreter has the ability to better represent desired character outlines. Figure 6 illustrates the bit representations of 4-point characters in both 300 and 600 dpi. Notice the dramatic improvement in 600 dpi.

600 dpi Halftones

Because laser printers print only solid tones – in this case, black – continuous tone copy, such as photographs, must be converted to regularly spaced patterns of small dots that simulate gray tones. A digital raster printer such as the 4029 must simulate traditional halftone dots with clusters built of smaller printer pels. Each printer pel within the cluster is turned on or off to produce perceived gray values, from completely black (all dots turned on) to completely white (all dots turned off). When combined, the clusters produce a pattern that creates a halftone. At 600 dpi, cluster size can be adjusted with greater precision than at 300 dpi, resulting in four times the number of gray levels. Figure 7 shows examples of halftone dot clusters at both 300 and 600 dpi. With a 4X increase in gray levels, the quality of image printing rises to a new plateau.

PQET versus 600 dpi

PQET provides an effective means of smoothing the visible stairsteps of 300 dpi printing, with no performance or monetary impact. The improvement can be seen in the enlarged type shown in Figure 8. If the character is large, PQET can make an intelligent decision regarding the original intent of the edges, thereby rivaling, and often exceeding, the quality of 600 dpi. In the case of smaller characters (typically less than 14 point), PQET provides dramatic improvement; but 600 dpi is necessary for representation of the subtle stroke-weight variations and fine details found in these characters. Nothing compares to true high resolution. For images, PQET neither disturbs nor enhances halftones. However, 600 dpi offers a dramatic improvement in the representation of halftone images.

Summary

All models of the LaserPrinter 4029 Series are packaged with Print Darkness control and PQET. The 600 dpi option, available for the 6, 10, and 10L models, is intended for the discriminating user desiring print quality approaching that of professional typesetting. With these print quality enhancements, the 4029 brings inexpensive desktop printers into areas previously controlled by more expensive printers and typesetters. The

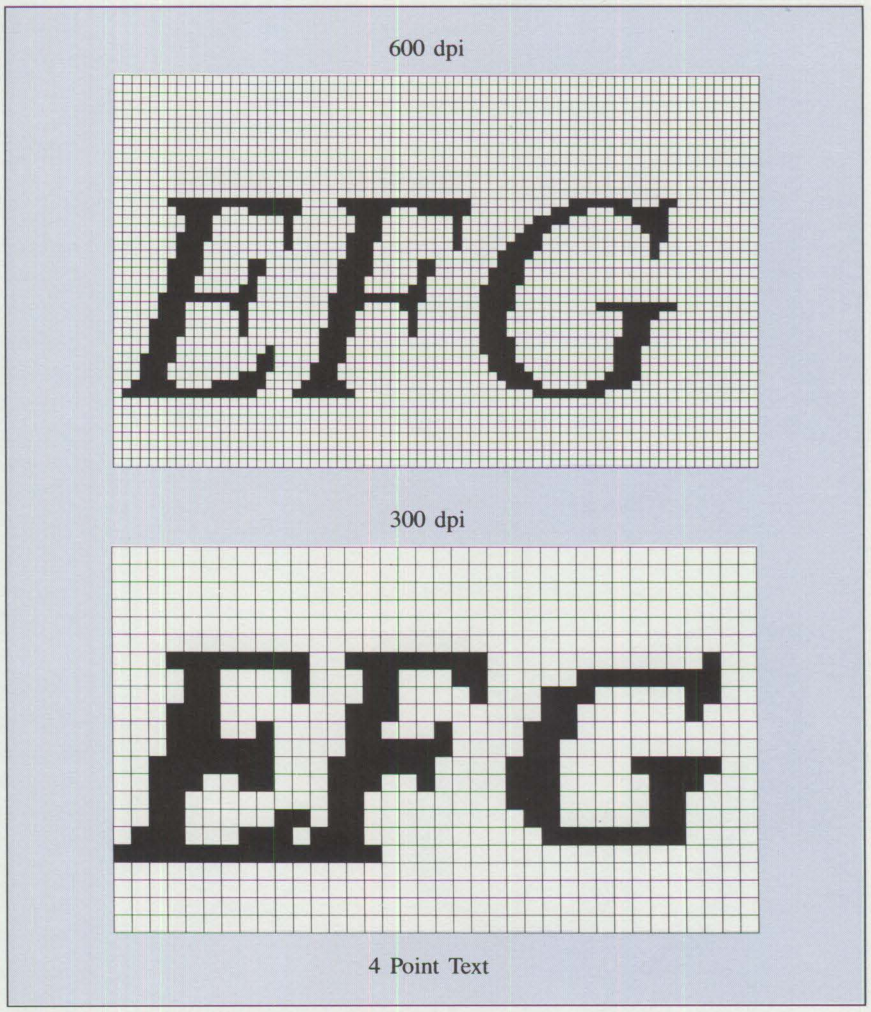


Figure 6. 600 dpi versus 300 dpi Character Bitmaps

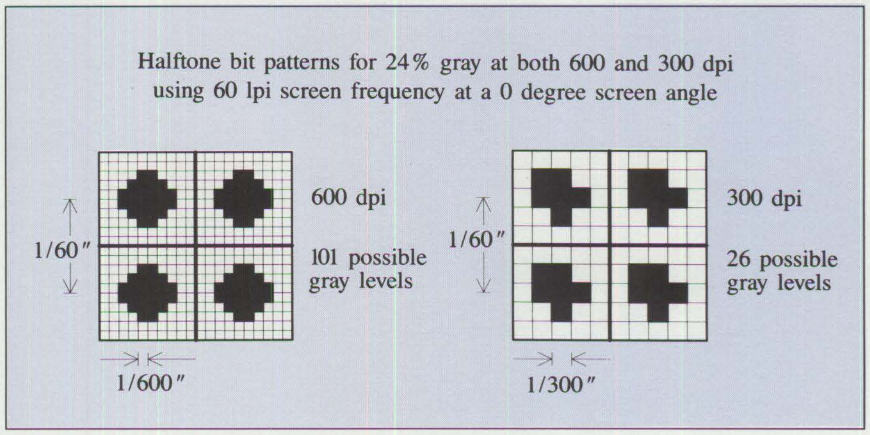
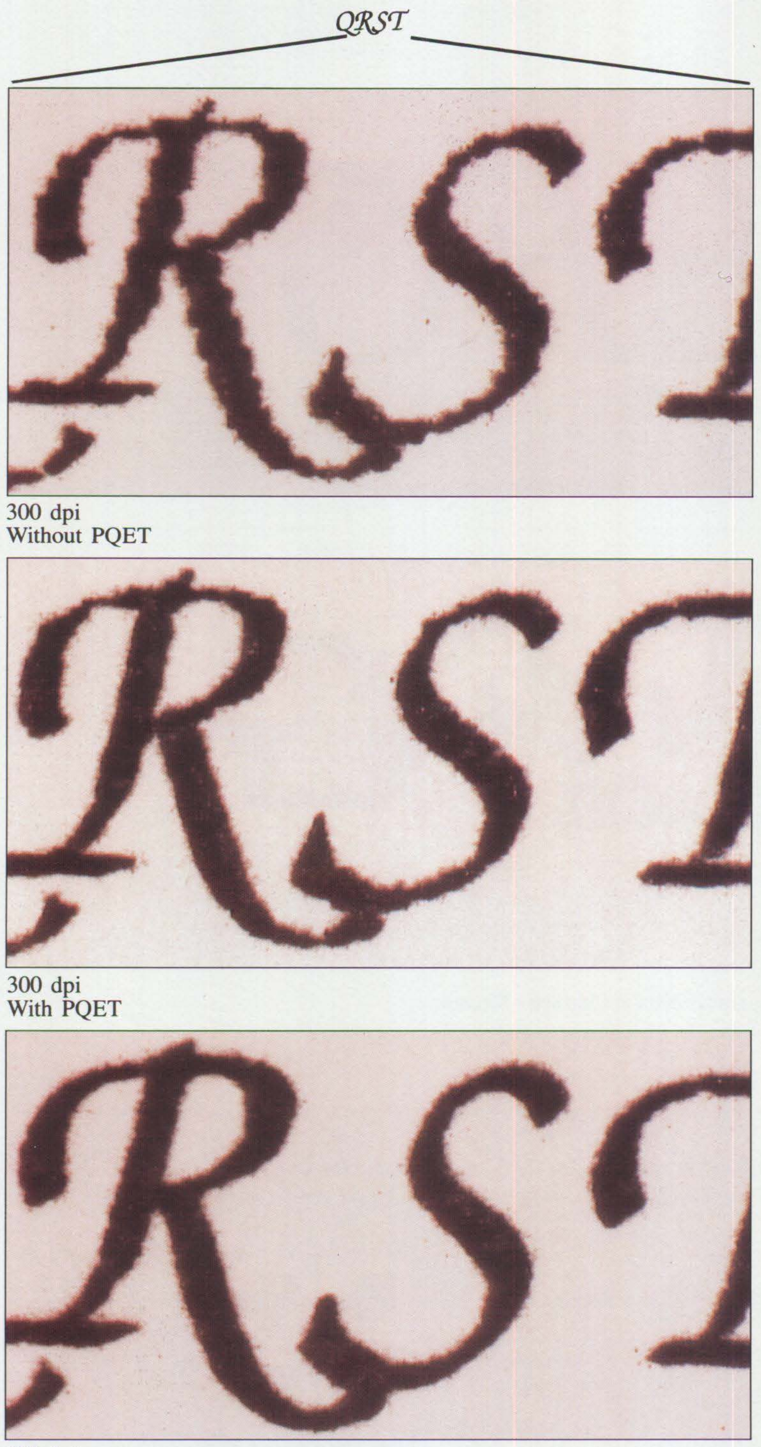

Halftone bit patterns for 24% gray at both 600 and 300 dpi using 60 lpi screen frequency at a 0 degree screen angle
1/60″
600 dpi
101 possible gray levels
1/600″
1/60″
300 dpi
26 possible gray levels
1/300″

Figure 7. Halftones at 300 and 600 dpi



Enlarged 12-Point Text

Figure 8. Enlarged 12-Point Text

IBM LaserPrinter 4029 Series has set a new standard in print quality.

ABOUT THE AUTHORS

Gary Overall is an advisory engineer at Lexmark International. Since joining IBM in 1982, he has had responsibilities in electrical system architecture and mechanism control for several IBM printers. His most recent activity was the definition of the logic and development of the pattern recognition algorithms for the IBM LaserPrinter 4029's PQET feature. He holds a BS in computer science and an MS in electrical engineering, both from the University of Kentucky.

Phil Wright is an advisory engineer at Lexmark. He joined IBM in 1984 as a member of the Print Technology group working on the IBM Quietwriter® Printers. In 1986 he joined the LaserPrinter development team leading the design of the 4019 engine control ASIC. He continued his engine control ASIC responsibility as a member of the 4029 development team, along with co-architecting and implementing the PQET hardware design. He holds a BS and MS in electrical engineering from the University of Louisville.

OS/2 2.0: The Integrating Platform

Kevin Maier
IBM Corporation
Boca Raton, Florida

The announcement of OS/2® 2.0 has generated many questions relating to the function, capability, and compatibility provided. Users need to know more about OS/2 2.0 because it is the first Intel 386/486 32-bit operating system that integrates many existing PC-based applications. This new operating system allows users to run existing DOS, Windows, and OS/2 applications without modification. It is a preemptive multitasking operating system that provides protection of all tasks and improves performance. When you add to this the capability of communications, database, LAN, and the new Workplace Shell, you have an operating system that provides a better environment than DOS, Windows, or previous versions of OS/2.

Operating System/2® (OS/2) 2.0 is a 32-bit multitasking, multithreaded operating system. It consists of a kernel and several code layers that allow concurrent execution of multiple applications that can be based on DOS, Windows, OS/2 1.X, or 32-bit OS/2 2.0. The operating system protects each application from other applications. This preserves the integrity of the system itself and that of each application. The operating system requires an Intel Complex Instruction Set Computer (CISC) processor type 80386SX and higher. These are 32-bit processors that can address up to 4 GB of physical memory and have 32-bit registers, with the exception of the 80386SX being limited to 16 MB of physical memory.

The processor is run in its native 32-bit flat memory mode of operation, which allows greater flexibility, performance, and protection. All features were carried over from OS/2 1.3 and have been improved to take advantage of the 32-bit processors. New features have also been added to improve usability, performance, and compatibility with existing applications.

The memory segment swapping mechanism from OS/2 1.3 has been replaced by a memory paging mechanism made possible by the 80386/80486 architecture. This allows an improved memory management subsystem where memory is paged to the hard disk in 4 KB blocks or pages, as opposed to the 64 KB block size used in OS/2 1.3. This significantly increases the performance of the memory management subsystem. Under normal operating conditions, memory paging to the hard disk is virtually undetectable by the user.

Another distinct advantage in the new memory management subsystem is the dynamic allocation of the swap file. Under OS/2 1.3, the swap file grew as physical memory was exhausted. It would not, however, decrease in size as physical memory was released, thus eliminating the need to use hard disk space. Under the memory paging mechanism in OS/2 2.0, the size of the swap file is dynamically managed and will grow and shrink as needed. This makes more efficient use of hard disk swap space by the operating system.

The installation process for OS/2 has been completely rewritten. Users who have installed OS/2 1.X will remember it as a fairly involved, lengthy process that prompts with questions that are not very clear. The new installation procedure gives the user some very basic prompts with good defaults. It then finishes the installation from a graphic screen that allows four choices: learning to use a mouse, a full installation, a preselected installation that helps save disk space, and a fully selectable installation.

DOS Environment

OS/2 2.0 supports multiple DOS sessions, with each session completely isolated from the others. The 80386/80486 processor architecture makes this possible. Intel has provided the ability in hardware to emulate multiple 8086 processors. The 8086 sessions are referred to as Virtual DOS Machines (VDMs). OS/2 2.0 exploits this hardware feature. As a result, the VDMs are completely compatible with the actual 8088/8086 processors at an architectural level. The support of the VDMs is so robust, you can actually boot a DOS diskette from within a VDM. This capability allows applications that depend on a specific DOS version to execute.

The VDM has several advantages over DOS. First, you have the ability to start up to 240 VDM sessions concurrently. Also, each VDM is protected so if you have a DOS application that crashes, only that specific VDM is brought down. All remaining applications continue to execute unaffected by the crash. The Basic Input/Output System (BIOS) is provided by the OS/2 kernel along with the compatible DOS function available in PC DOS 5.0. This allows high-memory usage for Terminate-and-Stay-Resident (TSR) programs and device drivers that are unique. With OS/2 providing the DOS function, most of the 640 KB memory area is left free for user programs. Even though DOS 5.0 allows much of its code to be moved into the High-Memory Area (HMA), a heavily loaded configuration reduces the available memory. OS/2 VDMs contain more free memory because the operating system manages the device drivers and other software extensions.

With the improved function that VDM offers over DOS, additional configuration parameters exist. The operating system provides help for each parameter as it is selected. This allows the user to adjust the VDM parameters quickly and easily. The user can also reset the parameters to their default setting with a single mouse click.

When DOS is booted on a system, there are numerous hooks for the hardware and software interfaces. Many of these communicate directly with the hardware and do not have alterable parameters. When you have an operating system that allows coexistence of multiple virtual operating systems, something must manage the virtual operating systems. With this in mind, there are several alterable parameters provided for the VDMs. They vary from hardware timer ticks to EMS/XMS memory specifications to exclusive mouse access. These settings allow each DOS session to be configured for efficient operation with the application it will support. All these parameters have basic default values that are compatible with most DOS applications and will probably not need alteration. The basic design of DOS has led many programmers to do some creative coding in the past, and there are many DOS applications under this heading. The flexibility to alter the parameters under the VDM allows support of most DOS applications.

Some DOS applications require processor function beyond that provided by the 8088/8086 architecture. When the Intel 80386 first became popular, many programming shops that had applications requiring more power than DOS could provide started implementing the 80386 instructions and running the processor in 32-bit mode. These applications still use DOS for file I/O, but run the bulk of their code in 32-bit mode.

These applications may conform to one of two DOS extenders: the DOS Protect Mode Interface (DPMI) or the Virtual Control Program Interface (VCPI). OS/2 2.0 supports the DPMI interface so applications that use DPMI can execute. OS/2 2.0 does not support the VCPI interface, so applications that require VCPI will not. Applications that require VCPI are good candidates for migration to the native 32-bit OS/2 mode.

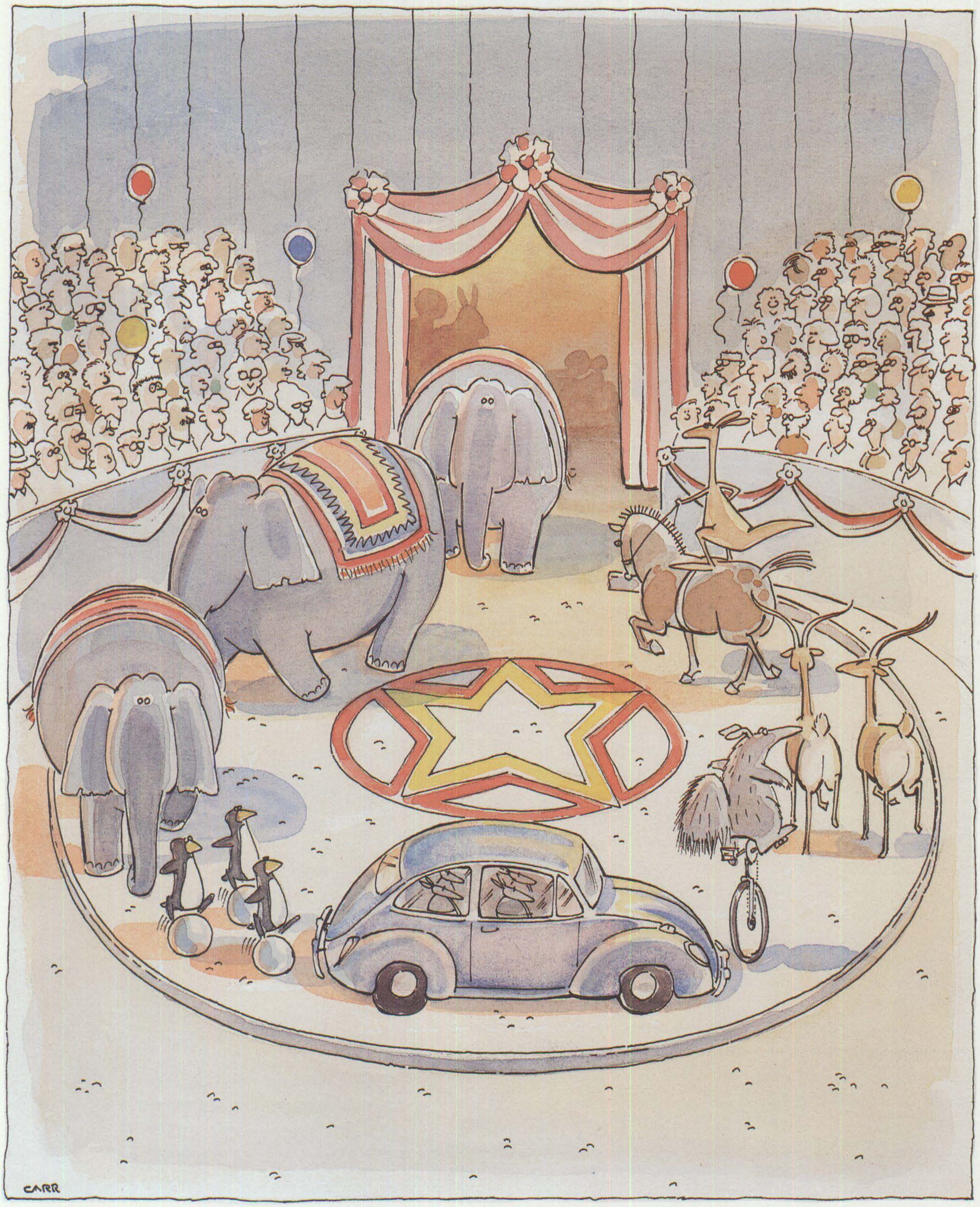
The VDM has other features that set it apart from DOS. The protection is the first and one of the most important. Other features are in the area of file I/O. OS/2 supports both the File Allocation Table (FAT) and High Performance File System (HPFS) formats. Any DOS application can perform file I/O to these file systems. The actual file system is maintained by OS/2 and the DOS INT 21h interface is provided for access. This is necessary because multiple applications will be performing file I/O concurrently. It also provides a higher level of performance for file I/O to the VDM because a 32-bit file system can have certain performance enhancements over that possible with a 16-bit operating system. Access to the hardware is generally faster by means of the OS/2 kernel than by DOS directly. This can give the VDM a performance edge over DOS.

Other detailed areas of the VDM could be addressed, but that would be beyond the scope of this article. This provides the basic overview of the VDM. When the VDM function is compared to DOS, the VDM emerges as the overall winner. Not only do you get several performance improvements, you get protection from other applications, the ability to take advantage of clipboard functions (cut and paste), and the capability to run multiple DOS applications concurrently.

Windows Compatibility

Microsoft Windows is a DOS extender that has a large growing application base. Many customers have standardized on the function available within several applications that use DOS/Windows as the application platform. To preserve customers' investments in software, OS/2 2.0 has the capability to run Windows applications. Microsoft Windows has been here for several years during which it has evolved. Most Windows applications are either Windows 2.X-type applications or the newer Windows 3.0 applications. Microsoft Windows has different modes of operation depending upon the type of hardware being used and the amount of memory available.

Real Mode: Windows 2.X could only operate in what is called *real mode*. Real mode is the equivalent of Intel 8088/8086 processors that can address a total of 1 MB running


CARR

DOS. As such, only 640 KB of memory is available, of which DOS and Windows occupy part and then the application gets the rest. This mode also allows the use of Expanded Memory Specification (EMS), which uses a pool of memory that is not addressable by the processor. This memory pool is banked into a segment area that is located in the C0000 to DFFFF (hex) address range. The size of the actual transfer is usually around 16 to 64 KB. This allows larger programs to run more efficiently because a memory-to-memory transfer is significantly faster than disk-to-memory.

New Modes: Microsoft Windows 3.0 has improved the way it uses processor resources. Windows 3.0 can run in real mode like Windows 2.X, but can also run in two new modes of operation: *standard mode* and *enhanced mode*. These two new modes of operation provide the application with more memory and improved function.

Standard Mode: In standard mode, the processor requirement is an Intel 80286. This processor can support real mode as well as standard mode. In standard mode, the 80286 is run in *protected mode*. This mode allows the 80286 to address up to 16 MB of physical memory. All this memory is available to Windows and Windows applications. As such, newer applications can take advantage of this mode of operation allowing better use of memory and processor resources and having an overall improvement in performance. When running in standard mode, DOS still provides the file system, so the processor must be "switched" back to real mode for all file I/O. Windows has a DOS box for running DOS programs in this mode, which also requires that the processor be running as an 8088/8086 (real mode). The 80286 does not have any logic or instruction to switch from protected mode to real mode (other than a cold start, or processor reset), so system services are provided to make this switch possible.

Enhanced Mode: For enhanced mode, an Intel 32-bit processor is required. Enhanced mode also runs in a protected mode environment, but uses the available 32-bit mode. This has two distinct advantages. The first is that paging is enabled for memory overcommit. For example, if you have only 4 MB of memory installed and are running Windows 3.0 configured for enhanced mode, the Help dropdown can show several times more free memory than physical memory. This number can vary as a result of software configuration and the amount of free space on the hard disk (enhanced mode swaps memory to the hard disk). When using multiple DOS sessions, the 8086 emulation feature of the 80386 hardware architecture is used. The disadvantage to these DOS sessions is that each one has the same software configuration as the base DOS before Windows is invoked. If you have already used more than 100 KB of memory (for device drivers, TSRs, and so on) before loading Windows, then every DOS session will have its 640 KB reduced by the same amount.

OS/2 2.0 was designed to allow Windows applications to run unmodified.

OS/2 2.0 Windows Support

OS/2 2.0 was designed to allow Windows applications to run unmodified; therefore, OS/2 supports real mode as well as standard mode. OS/2 can support Windows real mode by using a VDM (discussed earlier). All the same advantages apply to running Windows real mode. More memory is available within the 640 KB area and EMS/XMS memory is available as well. The overall performance improves for the same reasons DOS performance improves when using the VDM.

Since the introduction of Windows 3.0, many Windows 2.X applications have been migrated over to 3.0. In the process, many have been rewritten to take advantage of standard mode. Many new applications have appeared as well. The dependency on standard mode requires an 80286 or higher processor and runs in protected mode. Many new Windows applications require standard mode. Applications like Aldus® PageMaker® 4.0, along with many others, require this mode to execute. By supporting standard mode for Windows, OS/2 2.0 adds another large base of Windows applications to an already impressive list of DOS applications.

OS/2 2.0 supports standard mode by using a DOS VDM and the DOS Protect Mode Interface (DPMI) feature. OS/2 2.0 has all the required Windows code elements to support Windows applications without the need for Windows itself. The support of Windows applications is transparent and seamless to the user.

From the Workplace Shell desktop, a user need not be concerned with the actual application platform such as Windows or OS/2 Presentation Manager (PM). The desktop contains program icons for all applications. As applications are started from the desktop, they appear in their respective windows. The user does not see any difference between a Windows application and an OS/2 PM application.

To provide the full function of Windows applications running under OS/2 2.0, the operating system supports clipboard functions as well as Dynamic Data Exchange (DDE) functions. There are four possible combinations of clipboard and DDE support:

- Windows application to Windows application
- Windows application to OS/2 PM application
- OS/2 PM application to Windows application
- OS/2 PM application to OS/2 PM application

These functions are required to support the seamless integration of Windows applications into the OS/2 environment.

OS/2 2.0 does provide certain advantages over a native DOS/Windows environment. Most users of Windows have experienced the Unrecoverable Application Error (UAE) that occurs when a Windows application has overwritten its memory space. This has the possibility of corrupting other Windows applications and Windows itself, and results in a loss of loaded applications and the unsaved data. After the occurrence of a UAE, Windows must be terminated and the machine rebooted. This scenario is perceived to be a basic weakness of the DOS/Windows environment. Any application running under DOS/Windows has the capability to take the entire system down, resulting in a possible loss of data.

OS/2 2.0 prevents this type of failure in the Windows environment. When running any application under OS/2 2.0, each application is protected from the others regardless of whether it is a DOS, Windows, or OS/2 application. If a Windows application attempts to write beyond its memory area, the operating system prevents the write and terminates the offending application. This provides a level of safety and system integrity not possible in a DOS or DOS/Windows operating environment.

When you examine the differences between the two environments, OS/2 2.0 emerges as the better operating environment. Figure 1 shows the native DOS/Windows support versus Windows support provided by OS/2 2.0.

Presentation Manager

The Presentation Manager remains the base graphics engine within OS/2 2.0. Much work has been done to improve the overall performance of the graphics engine. The new graphics engine is written as a 32-bit engine and provides a higher level of performance than that achieved under OS/2 1.X, which used a 16-bit graphics engine. The new graphics engine provides full compatibility with existing 16-bit applications, and provides an enhanced 32-bit API for new applications and migration of existing 16-bit applications.

16-bit OS/2 Applications: The graphics engine has been designed to fully support existing PM applications. This is done by an internal translation layer, which provides a 16-bit API to the application. When a 16-bit application is executed, all the API calls are translated from 16-bit to 32-bit. The new 32-bit calls are passed through to the 32-bit graphics-rendering engine and executed. The return code must be translated to a 16-bit return code or the calling application cannot handle the return. The translation layer also provides this function. As a result, the translation

Available Function	DOS/Windows			OS/2 2.0
CPU	8088/8086	80286	80386/80486	80386/80486
Real	Yes	Yes	Yes	Yes
Standard	No	Yes	Yes	Yes
Enhanced	No	No	Yes	No[2]
EMS Memory	Yes	Yes	Yes	Yes
XMS Memory	No	Yes	Yes	Yes
DOS Session	Yes	Yes	N/A[1]	Yes[3]
VDM Session	No	No	Yes	Yes[4]
Virtual Memory	No	No	Yes	Yes[5]

Notes:

1 Windows enhanced mode uses VDM feature.
2 Features of enhanced mode provided by OS/2 outside of Windows.
3 OS/2 uses VDM feature, which has improved performance, more memory, and better protection.
4 OS/2 VDM feature has improved memory, performance, and protection.
5 OS/2 virtual memory management is done natively without Windows.

Figure 1. DOS/Windows versus OS/2 2.0 Windows Support

layer is transparent to the application that is running. This same function is provided for all APIs for 16-bit application support. This provides compatibility with existing 16-bit applications.

32-bit OS/2 Applications: The new 32-bit APIs allow application developers to write applications that can address greater than 16 MB of memory, and provide a flat memory model for application code and data space. This relieves programmers of having to manage memory segments within their programs and allows data areas to grow dynamically without segmentation problems. The newer 32-bit APIs allow for faster and more robust applications. It is expected that new applications will be coded to use the new 32-bit APIs and that older applications will be migrated over time to the 32-bit native environment.

High-Performance Porting Layer: The graphics engine has a new layer that was not in previous releases of OS/2 1.X. This layer, known as the High Performance Porting Layer (HPPL), is the code layer responsible for support of Windows applications under OS/2 2.0. HPPL is the translation layer that accepts Windows graphics calls made by a Windows application and uses the OS/2 2.0 32-bit graphics engine to perform the actual function. HPPL also handles the return code to the Windows application. This ensures compatibility of existing Windows applications.

The Workplace Shell: IBM has developed and implemented an updated user shell for OS/2 2.0 known as the Workplace Shell. It replaces the older PM Shell, which has been in OS/2 since Version 1.1. The new shell provides an improved "look and feel" to the operating system and increases the users' flexibility and control over the machine.

The Workplace Shell allows a user to manage all the system resources from a single interface. System resources are presented to the user as objects of multiple types, such as files, programs, printers, and drives. The new Workplace Shell has the necessary function to integrate DOS, Windows, and OS/2 applications on a single desktop.

A set of templates makes it easy to configure the desktop. These templates provide a simple and fast method of creating program entries and data entries, and managing resource objects and folders. The new Workplace Shell also provides a "drag and drop" feature that allows objects to be managed easily. For example, a user could drag a text file from a folder, drop it onto the printer object, and generate a print job. The new Workplace Shell also saves the desktop organization at shutdown and restores it during the next boot of the machine.

OS/2 2.0 has several productivity aids to help users do basic tasks that, under most operating systems, require additional programs. These aids, called applets, include a clipboard view utility, calculator, notepad, mini-spreadsheet and database, calendar, alarms, file searchers, an enhanced editor, and many others. The applets work together to provide a useful set of tools for the user. One of these applets is a terminal emulator that allows asynchronous communications to a bulletin board service or mainframe. Games are also provided: Solitaire, Reversi, Chess, a jigsaw puzzle, and more.

OS/2 2.0 has a detailed help facility. There are command references for the basic operating system and the REXX procedure language, a Master Index for the system, and a Glossary of terms and their meanings. Users can get to the information they need quickly and easily from menus. There is even an online tutorial on using the mouse and operating the desktop.

Putting It All Together

Considering the complexity of an operating system such as the one described, installation and usability are greatly simplified. The level of function in OS/2 2.0 is beyond that of previous desktop operating systems. It allows a user to take advantage of 32-bit processors, run existing 16-bit applications whether they are based on DOS, Windows, or OS/2 1.X, and provides a solid platform for future applications. Networking, database, and communications functions are available for the system and provide additional resources to the user.

In summary, OS/2 2.0 is an excellent integrating platform. It allows users to run their existing DOS, Windows, and OS/2 applications from a common desktop, and has new levels of protection and integrity unmatched in previous desktop operating systems. OS/2 2.0 has succeeded in providing a better operating environment than DOS, Windows, and previous versions of OS/2. And with the new Workplace Shell, the user has better control of the system and the functions it provides.

ABOUT THE AUTHOR

Kevin Maier is a market support administrator with IBM's Personal Systems Line of Business. Kevin joined IBM in 1977. His experience includes mid-range and personal computers. During the last seven years, Kevin has provided technical support for personal computer hardware and operating systems, and desktop publishing under OS/2 and DOS/Windows environments.

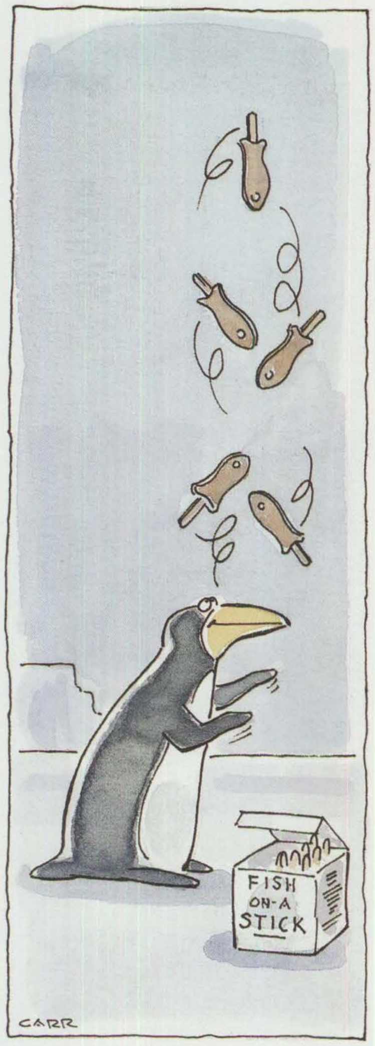
Multiple Virtual DOS Machines

Hans Goetz
IBM Corporation
Boca Raton, Florida

An important new feature of OS/2 is the ability to execute multiple DOS applications concurrently, with full preemptive multitasking and memory protection for each application. Microsoft Windows applications are also supported in the same way. These capabilities allow the use of OS/2 Version 2.0 as an integration platform for DOS applications, Windows applications, and OS/2 applications in a seamless, fully functional environment. This article looks at Multiple Virtual DOS Machines (MVDM) and the DOS Emulation provided to programs running within MVDMs.

OS/2 Version 2.0 gives the user the ability to run multiple concurrent DOS applications, and to multitask these applications with OS/2 applications. In previous versions of OS/2, support for DOS applications was limited to a single DOS session, known as the DOS Compatibility Box, in which the amount of memory available to that DOS session was restricted. Applications running in the DOS Compatibility Box could operate in full-screen mode only, and were suspended when switched to the background.

Support for DOS applications has been completely redesigned in OS/2 2.0, which now allows the execution and management of multiple concurrent DOS applications, where each application is executed as a single-threaded, protected mode OS/2 program. This capability is provided by a component of OS/2 2.0 known as Multiple Virtual DOS Machines (MVDMs).



MVDM introduces powerful DOS application support to OS/2 by exploiting the Virtual 8086 (V86) mode of the Intel® 80386 processor. This mode of operation allows emulation of an Intel 8086 processor and associated hardware devices within a protected mode 80386 task. OS/2 2.0 uses the V86 mode to allow creation of multiple instances of independent Virtual DOS Machines (VDMs).

Through this technique, an interface is provided to each VDM that gives the impression the application running in that machine owns all the required resources, both hardware and software.

Each VDM runs as a protected mode process, in a manner similar to an OS/2 application. The use of protected mode allows preemptive multitasking of DOS applications and provides a protected system environment in which DOS applications can execute. This means that system memory as well as other DOS and OS/2 applications are protected from ill-behaved applications, allowing users to terminate "hung" DOS applications without affecting other applications. An errant DOS application can affect only its own VDM; other applications in the system will not be affected.

Each VDM has a great deal more available memory than did the DOS Compatibility Box in previous versions of OS/2. Depending on the use of DOS device drivers and Terminate-and-Stay-Resident (TSR) programs, it is possible to have as much as 630 KB of available memory for application execution. In addition, OS/2 2.0 supports the use of the Lotus®-Intel-Microsoft (LIM) Expanded Memory Specification (EMS) and the Lotus-Intel-Microsoft-AST (LIMA) Extended Memory Specification (XMS), as well as the DOS Protected Mode Interface (DPMI) APIs to provide additional memory for DOS applications capable of using such memory extenders. OS/2 2.0 maps this expanded or extended memory into the system's linear memory address space, and manages it in the same manner as any other memory.

Each VDM can run either in full-screen mode or within a Presentation Manager window. A DOS window can be sized and manipulated in the same manner as an OS/2 window, and other Presentation Manager® desktop features are readily available (such as the ability to cut/copy/paste information between applications using the clipboard, and the ability to change fonts).

From the user's perspective, DOS applications behave exactly like Virtual Input/Output (VIO) applications. DOS applications have the following characteristics:

- They can run in either full-screen mode or in window mode.
- They can run in the background.
- Windowed DOS applications have all the same system menu controls as windowed OS/2 applications, including font adjustment and clipboard functions such as cut, copy, and paste.

Furthermore, DOS applications under OS/2 2.0 have advantages over VIO applications:

- They can be switched between windowed and full screen while running.
- A full-screen graphics-mode DOS application can be switched into a window and the graphics bitmap will be rendered in the window, giving the viewer more context when running multiple applications.

Application compatibility in the VDM is also enhanced over previous versions of OS/2. A VDM can be used to execute applications that address hardware I/O devices, such as DOS-based communications applications. Through the use of virtual device drivers, device driver calls from DOS applications are mapped to the appropriate physical device driver within the operating system. Applications using hardware devices that do not have to be shared with DOS applications in the same system can access these devices using the standard DOS device drivers, without the need for a virtual device driver. Certain restrictions still apply regarding communications line speed and time-critical interrupt handling.

A powerful new feature called DOS Settings allows an individual to use Presentation Manager dialogs to easily tailor the resources, such as video and memory, available to an application running in a VDM. This optimizes the way a DOS application runs.

The Virtual DOS Machine Manager contains the mechanism to start and interact with DOS applications.

The sole restriction for DOS applications running in a VDM when compared with VIO applications is that they cannot be used in process subtrees. That is, VDMs cannot be run as child processes of either an OS/2 session or another VDM session.

There are some DOS applications and products that cannot be supported by DOS emulation, due to the nature of the emulation code and the multitasking and protection demands of OS/2 2.0. Unsupported products/functions include:

- DOS applications that have internal DOS structure dependencies, such as Windows 1.0X and MS/PC Net
- DOS applications that do not work in a multitasking environment, such as Norton Disk Utilities, DOS block device drivers, and Fastback
- Some DOS network drivers, because DOS emulation uses a different implementation to control its I/O. However, DOS applications running in VDMs can access network services through the normal OS/2 network driver.

The user can still get around some of these limitations by booting a "real" DOS kernel under OS/2 2.0. This feature is called virtual machine boot and loads any kernel that would run on an 8086/8088 family of processors, such as IBM DOS 5.0 or DR DOS 6.0 from Digital Research®.

MVDM Architecture

MVDM is designed to exploit the V86 mode of the 80386 processor, which allows operating systems such as OS/2 2.0 to execute multiple DOS applications within the 80386 protected mode environment. Under OS/2 2.0, each DOS application executes in a VDM, which runs as a single-threaded, protected mode process. The operating system scheduler controls task switching of VDMs in the same way as it does OS/2 application processes.

The MVDM kernel controls the state and operation of concurrent VDMs, and is composed of four major components as shown in Figure 1.

The **Virtual DOS Machine Manager** contains the mechanism to start and interact with DOS applications. It creates, initializes, and terminates VDMs. The Virtual DOS Machine Manager manages system resources such as memory, timers, semaphores, and files for all VDMs active in the system. It is also responsible for loading and initializing all virtual device drivers, in conjunction with the Virtual Device Driver Manager (Figure 2).

The **8086 Emulation** manages communication between 8086 instruction streams and virtual device drivers. This emulation performs 8086 instruction decoding, controls the 80386 processor's I/O Privilege Map for each VDM, manages the reflection of software interrupts for each VDM, routes I/O instruction traps to the appropriate virtual device driver, and manages various control structures required by each virtual device driver.

DOS Emulation emulates the function and operation of the DOS operating system on a per-VDM basis. Each VDM emulates an entirely independent instance of DOS. DOS services are emulated within the MVDM kernel, or by invoking protected mode services provided by the OS/2 kernel. For example, most DOS file I/O functions are provided by the OS/2 file system. DOS 5.0 compatibility is provided.

The **Virtual Device Driver Manager** loads, initializes, and communicates with virtual device drivers. Virtual device drivers are required to virtualize the hardware and ROM BIOS, thereby allowing DOS applications to access hardware devices and BIOS without affecting other VDMs or other non-V86 mode processes in the system. The Virtual Device Driver Manager provides various system services, known as Virtual Device Helper services, to virtual device drivers.

These four components interact with one another and with the OS/2 Version 2.0 operating system kernel to provide requested services to DOS applications executing in VDMs.

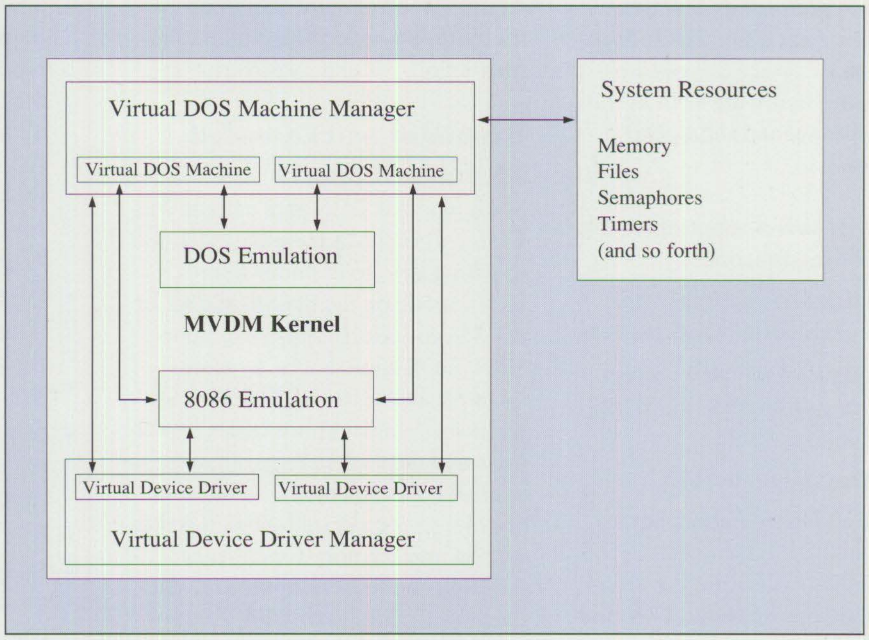


Figure 1. MVDM Kernel

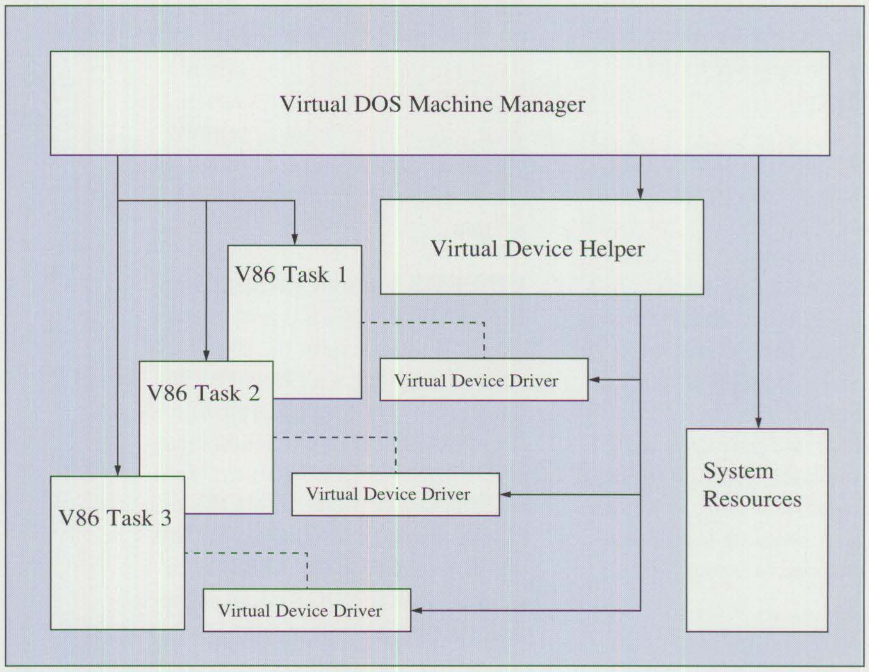


Figure 2. MVDM System Structure Overview

Virtual Device Drivers

These installable modules support the hardware and ROM BIOS aspects of the DOS environment for virtual DOS machines. A virtual device driver manages shared access to hardware I/O devices for multiple VDMs, allowing an application running in a VDM to act as though it exercised sole control over I/O devices. Virtual device drivers are implemented whenever possible, allowing the BIOS in the system to perform its functions without interference from DOS applications. Virtual device drivers are used to control hardware such as the keyboard, mouse, and serial and parallel ports.

Virtual device drivers are responsible for the following functions:

- Maintaining a virtual hardware state for each virtual DOS machine
- Preventing a VDM from corrupting the state of another VDM, or the system as a whole
- Supporting fast screen I/O
- Supporting fast communications I/O

The virtual device driver architecture implemented in OS/2 2.0 provides support for all standard hardware utilized by DOS applications. Through device driver upgrades, this architecture will support new hardware without requiring an upgrade to the operating system.

Virtual device drivers obtain and release system resources via the Virtual Device Helper services provided by the MVDM kernel. A virtual device driver typically performs I/O through a physical device driver using a direct call interface. However, a virtual device driver can directly access an I/O control device. The virtual video device driver performs such direct access under OS/2 2.0 for performance purposes. A virtual device driver can also simulate hardware interrupts into one or many VDM processes.

Under OS/2 2.0, a virtual device driver is inherently protected from a VDM because it is not visible in the VDM address space. However, a virtual device driver must be careful to check all parameters coming in from a VDM to ensure that it does not damage itself or some other part of the system by executing an invalid instruction.

Expanded and Extended Memory Support

Many popular DOS applications use EMS and/or XMS memory extenders to gain access to memory in protected mode on the 80286, 80386, or 80486 processors. These extenders allow DOS applications to access memory above the 1 MB real-mode addressing limit. This enables a total code and data space larger than the available base memory, and allows very large code or data objects to be loaded into memory for enhanced function and performance. The standard configuration of OS/2 2.0 provides both EMS and XMS functions as part of MVDM.

Under MVDM, EMS and XMS memory allocations are managed as OS/2 pageable virtual memory, not as fixed physical memory. As such, the total expanded/extended memory allocated can exceed the amount of physical memory installed in the system.

LIM EMS Version 4.0 Support: LIM EMS Version 4.0 provides a standard interface for the use of expanded memory with Intel 80X86 computers. The specification allows for up to 32 MB of expanded memory, with up to 255 expanded memory objects. Regions within these objects can be mapped into the 8086 address space (below 1 MB) as required, allowing DOS applications to access large address spaces.

Under OS/2 2.0, EMS emulation provides the following functions:

- Implements all the required functions in LIM 4.0 EMS.
- Provides each VDM with a logically separate EMS emulation. Each VDM has its own set of expanded objects so that features like interprocess communications work as they would if each VDM were running on a different physical 8086. A VDM cannot affect the availability of objects in other VDMs or access expanded memory "owned" by other VDMs.
- Provides for remapping of conventional memory (below 640 KB) for use by programs such as Windows 2.0.
- Provides configurable limits for the total amount of expanded memory available to all VDMs, as well as a limit per VDM.
- Supports mapping of multiple physical addresses to a single logical address. Different 8086 addresses can map to the same expanded memory object address. This is required by programs like Lotus 1-2-3®.

EMS emulation is provided in MVDM by the Virtual Expanded Memory Manager (VEMM). This is a virtual device driver that offers a separate EMS emulation for each VDM. It accomplishes this by placing most EMS control structures for a VDM in a per-VDM instance data area outside the V86 application's address space.

Unlike most virtual device drivers, VEMM does not have a corresponding physical device driver; rather, it traps software interrupts for EMS services using a system call and manages the appropriate memory. VEMM is dependent on the memory management capabilities of the OS/2 Version 2.0 operating system kernel. Allocation, reallocation, or deallocation of EMS memory objects is accomplished by requesting corresponding services from Virtual Device Helper services.

LIMA XMS Version 2.0 Support: LIMA XMS Version 2.0 provides a standard interface for the use of extended memory on Intel 80X86 computers. XMS specifications allow for up to 16 MB of extended memory. XMS functions allow the moving of code and data objects from base memory into extended memory, and from extended memory to base memory. Two of the best known types of programs using XMS functions are print spoolers and virtual disks (RAM disks).

The XMS specifications manage three distinct regions of memory:

- The High Memory Area (HMA) is located immediately above 1 MB and is 65520 bytes in size.
- The Upper Memory Area (UMA) consisting of Upper Memory Blocks (UMBs) is located between 640 KB and 1 MB.
- The Extended Memory Blocks (EMBs) are used only for data storage.

Under OS/2 2.0, LIMA XMS emulation provides the following functions:

- Implements all LIMA XMS Version 2.0 functions.
- Provides each VDM with a separate XMS emulation. Each VDM has its own high-memory area, upper memory area, and extended memory blocks, so that features like interprocess communications work as they would if each VDM were running on a different physical 8086 processor. A VDM cannot affect the availability of objects in other VDMs or access memory "owned" by other VDMs.
- Provides configurable limits for the total amount of extended memory available to all VDMs, as well as a limit per VDM.

The Virtual Extended Memory Manager (VXMS) is a virtual device driver that provides a separate XMS emulation for each VDM. As with VEMM, this is accomplished by placing most VXMS control structures for a VDM in a per-VDM instance data area outside the V86 application's address space. The amount of memory available to a VDM, the number of handles, and the existence of upper memory blocks are all configurable parameters that can be altered on a per-VDM basis.

The DOS Settings feature gives the user more control over the consumption of system resources by a DOS application.

Like the VEMM virtual device driver, VXMS does not have a corresponding physical device driver, and utilizes the memory management capabilities of the operating system kernel. XMS object allocation, reallocation, and deallocation are accomplished by requesting corresponding services from the memory manager.

DOS Settings

MVDM lets the user customize the operation of DOS applications via a feature called DOS Settings. This feature allows the user to control special properties that affect the behavior of DOS applications running in a VDM.

The DOS Settings feature further enhances the DOS compatibility of a VDM because it allows a user to configure the VDM for DOS applications that might otherwise not work well (or at all) with the default settings of the VDM task. Online help is provided for each setting to assist users in tuning their applications' operation. The DOS Settings feature gives the user more control over the consumption of system resources by a DOS application. At the same time, this level of control is not obvious to a novice user.

OS/2 2.0 will be shipped with a "Certified Application Database." The installation process allows a search of the user's hard disk for existing DOS and Windows applications and installs them under the Workplace Shell. If an application is also registered in this database, the appropriate DOS settings will automatically be configured for such a program entry.

DOS sessions have many more customizable properties than OS/2 sessions. MVDM provides a common mechanism that both supports a standard complement of settings, and allows virtual device drivers to register custom settings. The standard settings are a subset of the configuration settings in the CONFIG.SYS file, plus some additional settings required for MVDM. The primary reason for altering these settings is that DOS applications are typically not careful about consuming system resources, such as memory and processor time. Hence, MVDM itself must provide a flexible environment for these applications in order to preserve the integrity and performance of the system as a whole.

DOS settings are managed on a per-VDM basis and are accessed through Presentation Manager dialogs. The dialog boxes presented allow the user to change a setting while the VDM is running. Only those settings that can be changed for that VDM are shown.

Many settings can be tuned. For example, the Idle Detection Threshold detects idle DOS applications and allows the user to configure the system so that processor time is not wasted.

Microsoft Windows Application Support

OS/2 2.0 allows Microsoft Windows applications to run under OS/2 2.0. With this support, applications written for Windows Version 3.0 and earlier (except Version 1.X) can coexist with OS/2 and DOS applications under OS/2.

Each Windows application executes as a protected mode process and is therefore subject to the same application protection facilities provided to other OS/2 or MVDM applications. Windows applications are protected from other Windows applications and from DOS and OS/2 applications executing in the system. This is in contrast to the native Windows 3.0 environment where complete protection is limited to Windows 3.0 applications only.

The execution of Windows applications as protected mode tasks also allows them to take full advantage of the preemptive multitasking capabilities of OS/2 2.0, with full preemptive multitasking among Windows, OS/2, and DOS applications. This is again in contrast to the native Windows 3.0 environment where preemptive multitasking is available only when Windows 3.0 is running in enhanced mode, thereby impacting performance and preventing many applications written for previous versions of Windows from executing. OS/2 2.0 has no such restriction. Windows applications running under OS/2 2.0 will run in a mode equivalent to the real or standard mode of Windows 3.0; the enhanced mode of Windows 3.0 is not required since the OS/2 2.0 operating system itself provides equivalent function.

MVDM DOS Emulation

MVDM DOS Emulation provides DOS services to DOS applications running in MVDMs to achieve maximum compatibility with native DOS 5.0. DOS Emulation addresses only DOS software compatibility aspects; processor and other hardware aspects of the DOS environment are addressed by the 8086 Emulation component and virtual device drivers.

In OS/2 2.0, physical device drivers are loaded above 1 MB and only the DOS Emulation kernel resides below 1 MB.

DOS Emulation Overview

DOS Emulation is implemented by running a very small portion of the DOS Emulation kernel in V86 mode and a much larger portion of this code in protected mode outside the VDM. In OS/2 2.0, physical device drivers are loaded above 1 MB and only the DOS Emulation kernel resides below 1 MB. User-installed OS/2 device drivers do not affect the amount of memory available to a DOS application running in a VDM. Similarly, adding LAN drivers to the OS/2 configuration to support network server or redirector functions does not consume DOS application space, even though DOS applications can make use of these OS/2 network devices. Virtual device drivers are also loaded outside the VDM address space and therefore do not reduce the amount of memory available to a DOS application.

In this way, the MVDM architecture makes the maximum amount of memory available to DOS applications. Up to 630 KB is free for use by DOS applications, reduced by any DOS device drivers or TSR programs loaded into the VDM. This represents an increase of approximately 100 KB over memory available in the DOS Compatibility Box under OS/2 Version 1.3. The following features are implemented by DOS Emulation:

- DOS console device driver
- DOS device driver loading/support
- DOS program loading and execution
- DOS File Control Block (FCB) I/O support
- DOS memory manager
- DOS National Language Support (NLS) support
- DOS PDB process support
- VDM entering and exiting kernel mode support
- Special DOS compatibility mechanisms

DOS Emulation supports all documented DOS interrupts and features. In addition, some undocumented aspects of these functions (especially INT 21h) are incorporated in the DOS Emulation component to support a large number of DOS applications that rely on these interfaces.

The following list shows some of the documented DOS system interrupt services that are implemented by DOS Emulation and are available to DOS applications running in VDMs. Other DOS compatibility functions are implemented by the 8086 Emulation component of MVDM and by virtual device drivers.

INT 20h – Program terminate interface

INT 21h – System call interface

INT 22h/INT 23h – Terminate and Ctrl-Break address interfaces

INT 24h – Critical error-handler interface

INT 25h/INT 26h – Absolute disk read/write interfaces

INT 27h – TSR interface

INT 28h – Idle loop interface

INT 2Fh – Print spool interface

A primary design goal for DOS Emulation was to provide a DOS-compatible environment in which a VDM could not negatively affect other VDMs or OS/2 protected mode processes, while at the same time providing the greatest amount of free memory for DOS applications. This goal was achieved by allowing as much of the DOS Emulation code as possible to execute in protected mode, outside the VDM address space. This implementation provides significantly more memory available for DOS application use, enhancing overall performance.

Initialization and VDM Creation

Initialization of the DOS Emulation component is divided into two stages. The first occurs during OS/2 system initialization. The second stage occurs during creation of each virtual DOS machine.

OS/2 Initialization: OS/2 system initialization involves DOS Emulation because it requires access to information contained in CONFIG.SYS. As the OS/2 initialization procedure processes the CONFIG.SYS file, it records parameters related to DOS Emulation. These parameters include those specified in the FCBS and RMSIZE statements and those DEVICE statements that specify DOS device drivers. These parameters become the default DOS settings for all VDMs.

Virtual device drivers are not loaded or initialized at this stage. DOS settings are initialized before loading virtual device drivers because these default settings may be required by the device drivers during VDM initialization.

VDM Creation: Upon creation of a VDM, the Virtual DOS Machine Manager calls the creation-time initialization routines for virtual device drivers and then passes control to the DOS Emulation kernel. At this point, the V86 memory organization appears as shown in Figure 3.

During VDM creation, DOS Emulation performs the following steps:

1. Initialize VDM-related kernel structures.

Certain structures in the OS/2 Version 2.0 kernel are initialized in preparation for processing VDM requests. For example, the System File Table (SFT) structures, which are used for FCB I/O, are allocated and initialized.

2. Load the DOS Emulation kernel.

The portion of the DOS Emulation kernel that runs in V86 virtual memory is loaded at the high end of the VDM memory address space.

3. Start virtual processor mode.

The protected mode initialized routine returns control to the Virtual DOS Machine Manager, which then invokes the initialization code within

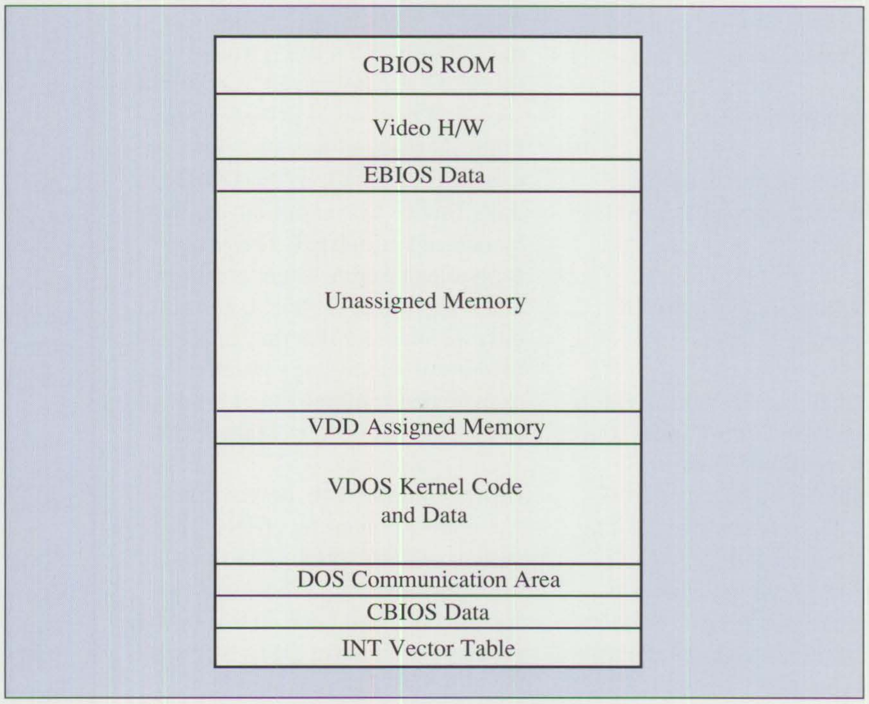


Figure 3. V86 Memory Map Prior to DOS Emulation Initialization

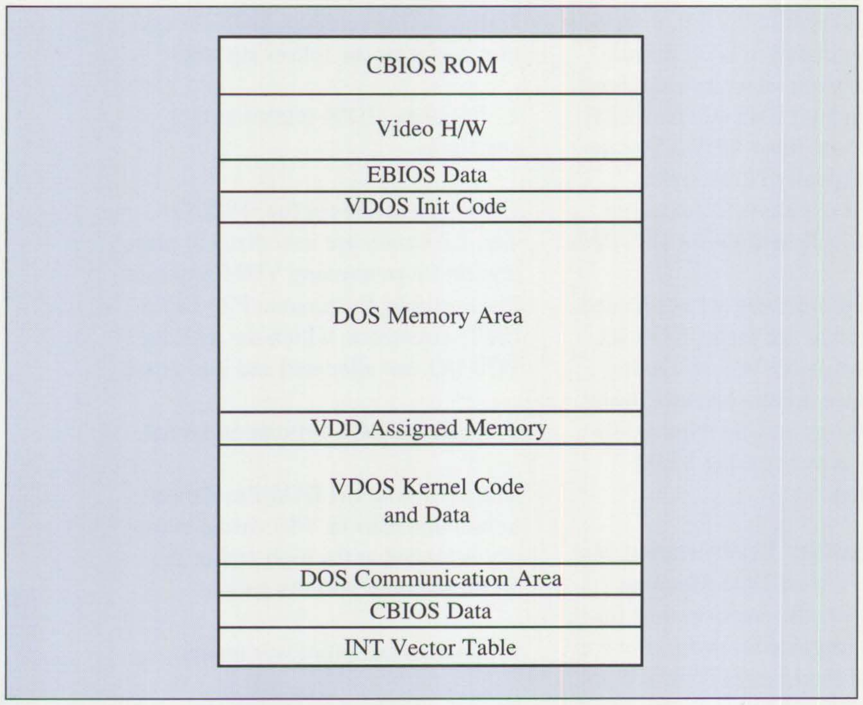


Figure 4. V86 Memory Map at Initial V86 Mode Entry. This diagram shows the VDM's memory map when the V86 mode DOS Emulation kernel is first invoked.

the V86 mode DOS Emulation kernel. This represents the first transition to V86 mode; at this point, memory is organized as in Figure 4.

4. Open standard devices.

The initial five file handles (stdin, stdout, stderr, stdaux, and stdprn) are opened.

5. Initialize virtual device driver DOS device driver "stubs."

Some virtual device drivers provide a DOS device driver "stub." These stubs are inserted into the V86 address space prior to initialization of DOS Emulation. This step involves calling the inserted initialization code and linking the devices into the device chain. Unlike real DOS device drivers, however, the return from the initialization does not allow reducing the size of the driver code.

6. Initialize DOS device drivers.

Each DOS device driver specified in the CONFIG.SYS file is loaded into the VDM and initialized. Any VDM-specific DOS device drivers passed in the DosCreateProcess( ) function call, or configured via the DOS Settings Device Driver option, are then loaded and initialized. This is performed one device driver at a time to allow the device drivers to consume only the memory that they require or to deinstall themselves entirely. As each device is initialized, it is added to the chain of devices in the VDM.

During initialization, device drivers can issue a limited set of INT 21h system calls (functions 01h through 0Ch, 30h, and 35h). This restores function that had been removed from previous versions of OS/2. These are the only INT 21h functions supported. Issuing an unsupported INT 21h system call will crash the VDM.

After all device drivers have been initialized, the initialization code is discarded.

7. Load and execute the DOS shell.

The shell specified in the SHELL command in CONFIG.SYS is loaded; the initialization code in the V86 address space is discarded; and control is passed to the shell program. The SHELL specified in CONFIG.SYS can be overridden in the DosCreateProcess call. This is a useful feature if, for example, a software developer wants to allow different versions of COMMAND.COM for alternative national language support.

When the shell program is invoked, the VDM's memory map is organized as shown in Figure 5.

Before passing control, the PDB of the shell is initialized with the command line parameters as specified in the CONFIG.SYS file. As an extension to the DOS Version 4.0 environment, an additional string is appended after these parameters, separated from the command line string by a NULL byte. This string specifies the drive and directory of the virtual DOS environment after the shell completes its initialization. This extension provides the default working drive and directory for real mode applications as is provided for protected mode applications using the Presentation Manger.

VDM Termination: When a VDM is terminated, DOS Emulation closes all handles that have been assigned to all PDB processes within the VDM. All resources associated with open FCBs are also closed.

Because the VDM may have been terminated due to an error in the VDM's DOS application, no data within the VDM is relied upon when closing resources and cleaning up. DOS Emulation explicitly closes both the files and the OS/2 devices opened by the VDM.

Requesting System Services: As previously mentioned, MVDM isolates some VDM-specific code and places it into the DOS Emulation kernel in the VDM's address space. This code provides those DOS services that can be supported in V86 mode, such as memory management services. Other services that require protected mode execution are provided by additional code that runs in protected mode.

When the DOS Emulation kernel requires protected mode services, it specifies the kernel procedure via an index, and executes a processor instruction that causes a general protection exception. The OS/2 2.0 general protection exception handler traps the exception and invokes the 8086 Emulation component, which in turn transfers control to the specified kernel procedure.

System Service Call Behavior: DOS system services provided within VDMs are generally compatible with their implementation under DOS 5.0. Some differences do exist, however, and are described here.

INT 20h – This service forwards the request to the INT 21h function 00h service to abort the application.

INT 21h – All INT 21h services provided in DOS 5.0 are provided in VDMs. However, the internal behavior and error processing of some functions may be different.

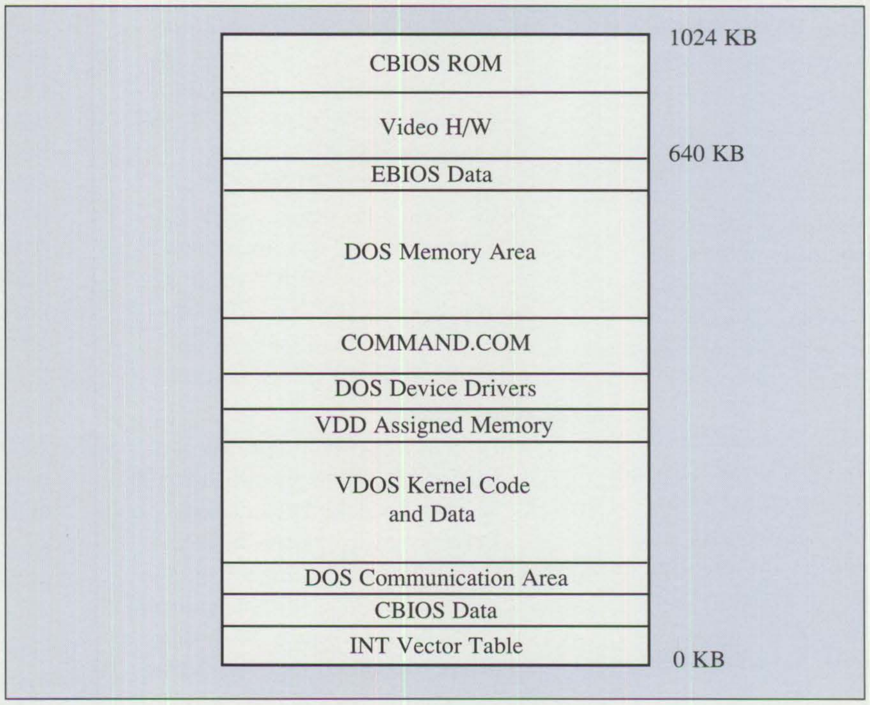


Figure 5. V86 Memory Map after Initialization. This diagram shows the VDM's memory map after initialization of the DOS environment and prior to loading a DOS application.

INT 25h/INT26h – The absolute read function operates the same way it does in a DOS 5.0 system. The absolute write function, however, is restricted to removable media only, and reports a hard error on non-removable media.

Standard Devices: As in DOS, DOS Emulation includes device drivers for the three "standard" devices: CON, AUX, and PRN. While DOS packages these device drivers in IBMBIO.COM, DOS Emulation links these into the DOS Emulation kernel to avoid the need for another file.

Unlike DOS, the AUX and PRN drivers do not include support for COM1, COM2, LPT1, LPT2, and LPT3. These latter devices are supported directly by the OS/2 asynchronous (COMOX.SYS) and printer (PRINTOX.SYS) device drivers.

This approach allows the CON, AUX, and PRN drivers to behave in a highly compatible manner. These drivers issue ROM BIOS calls (INT 10, 14, and 17, respectively) in order to perform their required tasks. At the same time, using the OS/2 device drivers directly for the numbered I/O devices provides higher performance than is possible through the ROM BIOS interfaces. This also allows a numbered I/O device to be easily redirected to a remote device on a network, using the underlying OS/2 mechanisms.

Note that only PRN redirection is supported. This is achieved by the printer virtual device driver VLPT.SYS, which routes INT 17 and directs hardware printing to LPT1, LPT2, or LPT3 using the OS/2 file system.

Maximizing VDM Memory

The OS/2 2.0 CONFIG.SYS file specifies the operating system configuration, and installs device drivers and other memory-resident programs. The OS/2 AUTOEXEC.BAT file is specific to the functioning of the VDM environment. More base memory can be made available to programs running in a VDM by removing unnecessary commands from these files.

CONFIG.SYS: Virtual device drivers utilized by VDMs consume very little or no memory below the 640 KB limit. These device drivers reside outside the V86 mode address space. However, a user can install device drivers that are required by and are specific to certain applications that will run in a VDM. If the commands to load these device drivers (or other memory-resident programs) are added to the CONFIG.SYS file, these device drivers (or programs) will be loaded into every VDM when it is created, and will reduce the amount of conventional memory available to DOS applications in every VDM. To ensure the maximum amount of memory is available in each VDM, it is recommended that:

- DOS device drivers specific to a particular DOS application should be loaded via the DOS Device Drivers option of DOS Settings. DEVICE = statements for DOS devices should be eliminated from CONFIG.SYS unless the device driver is required for every VDM.
- The number of buffers specified in the BUFFERS statement in CONFIG.SYS should be minimized. Each buffer consumes about 500 bytes. Be careful to not reduce this number too much or some programs might not run properly. The default number of buffers is 30; the number should not be reduced to fewer than 10 or 15 buffers.
- If CONFIG.SYS includes the LASTDRIVE statement, this should be set to a letter such as J or K, rather than Z. Each additional drive uses about 100 bytes.
- If the CONFIG.SYS file contains an FCBS statement, set FCBS to 1.

All these DOS-specific parameters can also be applied on a per-VDM base by defining them through the DOS Settings.

The order of the DEVICE and DEVICEHIGH statements in CONFIG.SYS is important because it can affect both the efficient use of memory and the proper operation of the various programs started from CONFIG.SYS.

The CONFIG.SYS statements and options related to VDM operation that can be selected during installation are shown in Figure 6.

The following list shows the order in which device drivers should be specified in CONFIG.SYS:

1. VEMM.SYS

2. Any device drivers that use expanded memory

3. VXMS.SYS

4. Any device drivers that use extended memory

Load the DOS Command Interpreter into memory
```
SHELL = C:\OS2\MDOS\COMMAND.COM C:\OS2\MDOS/P
```

Where to load DOS in memory
```
DOS = HIGH, UMB
```

Optional: extended keyboard and display features
```
DEVICE = C:\OS2\MDOS\ANSI.SYS
```

Expanded Memory Manager
```
DEVICE = C:\OS2\MDOS\VEMM.SYS
```

Extended Memory Manager
```
DEVICE = C:\OS2\MDOS\VXMS.SYS/UMB
```

Mouse support
```
DEVICE = C:\OS2\MDOS\VMOUSE.SYS
```

DOS graphics support
```
DEVICE = C:\OS2\MDOS\VXXX.SYS
```

Where XXX depends on video adapter:

VCGA.SYS	**(if CGA or EGA with 64 KB)**
VEGA.SYS	**(if EGA with 128 KB)**
VVGA.SYS	**(if VGA or 8514)**
V8514.SYS	**(if 8514)**
VXGA.SYS	**(if XGA)**

Direct I/O serial communication support
```
DEVICE = C:\OS2\MDOS\VCOM.SYS
```

Figure 6. CONFIG.SYS Statements and Options Related to VDM Operation

5. Any device drivers that use upper memory blocks

This is the optimal order in which the CONFIG.SYS file should load device drivers. Whether these statements are included depends on the amount of memory installed, the hardware configuration, and the DOS applications to be run.

AUTOEXEC.BAT: The AUTOEXEC.BAT file is specific to the virtual DOS machine environment and has no effect on the OS/2 2.0 operating system. AUTOEXEC.BAT starts memory-resident programs such as networks programs, sets up environment variables, and can define the command prompt. In the following AUTOEXEC.BAT file, the LOADHIGH command loads the APPEND TSR into the High Memory Area, thus making available more memory to all DOS applications running in every VDM. The SET COMSPEC statement that you might expect to find in AUTOEXEC.BAT has been moved to the CONFIG.SYS file.

The default AUTOEXEC.BAT file for all VDMs in OS/2 2.0 is as follows:

```
PATH=C:\OS2;C:\OS2\MDOS;C:\;
LOADHIGH APPEND
C:\OS2;C:\OS2\SYSTEM
PROMPT $I$P$G
```

To maximize the amount of base memory available to applications, any unnecessary commands should be removed from AUTOEXEC.BAT. Commands should only be included in AUTOEXEC.BAT if they are required for every VDM. Commands that are required only for a specific DOS application to be run in a VDM should be placed into a batch file. This batch file should be explicitly entered into the path and file name field of the Program Properties dialog box for that specific application.

Summary

OS/2 2.0 provides significantly enhanced DOS application support when compared to previous versions of OS/2, using a component known as Multiple Virtual DOS Machines. MVDM offers many significant functions and features, including:

- Ability to run multiple DOS sessions concurrently, with full preemptive multitasking and memory protection
- Ability to run DOS applications in Presentation Manager windows
- Ability to switch between DOS applications via Presentation Manager
- High amount of available free memory for DOS applications (up to 630 KB)
- EMS and XMS memory emulation support
- DPMI support for Windows and DOS applications
- Clipboard support

OS/2 2.0 provides DOS 5.0 compatibility within the virtual 8086 mode of the 80386 processor, and allows execution of multiple concurrent DOS applications, each operating in its own 1 MB virtual address space. This brings true multitasking to the DOS environment under OS/2. A user can run multiple DOS applications in much the same way as multiple OS/2 applications. DOS and OS/2 applications can be started in the same manner: from the Workplace Shell, from a command prompt, or via the OS/2 START command.

DOS applications can be run full-screen, windowed, or iconized in the background. In addition to better protection, better compatibility, and more concurrent DOS applications, the OS/2 2.0 MVDM environment provides applications with up to 630 KB of memory in which to execute.

Support for both the EMS and XMS is provided. DOS asynchronous communications applications can support communication speeds greater than 19.2 Kbps, depending on the hardware available. Since the DOS environments are swappable, starting many DOS sessions will not increase requirements for more physical (real) memory.

OS/2 2.0 also provides support for Microsoft Windows applications (written for Windows Version 3.0 and earlier, except Version 1.0X). Each Windows application executes as a separate protected mode task, and is therefore provided with the same preemptive multitasking and memory protection as other protected mode tasks under OS/2 2.0.

This article is condensed from material in the "Red Book," OS/2 Version 2.0, Volume 2: DOS and Windows Environment *(GG24-3731). The book is published by the International Technical Support Center, Boca Raton, Florida, and can be ordered when OS/2 2.0 is available. The Red Book contains the work of many contributors, such as systems engineers from around the world and developers from the Boca Raton laboratory.*

IBM OS/2 LAN Server 2.0

Barbara Barker, Carolyn Easter, and John McCarty
IBM Corporation
Austin, Texas

IBM OS/2 Local Area Network (LAN) Server Version 2.0 is an addition to the LAN Server technology. This article describes the features, functions, and future of the LAN Server.

IBM personal computers were first used as stand-alone systems primarily for personal productivity. To further increase cost effectiveness and performance, this arrangement evolved into the single-server LAN environment, providing basic file and printer sharing for small workgroups.

These single-server workgroups then expanded to environments that included several servers grouped together (domains) capable of providing sophisticated resource sharing in more complex situations.

Servers are continuing to evolve. They will become providers of distributed services and tools, such as distributed file systems, a global naming service, security, and a Remote Procedure Call (RPC) mechanism for the Distributed Computing Environment (DCE) as defined by the Open Software Foundation (OSF)®.

IBM has been an active participant in this evolution, first with the PC LAN Program (PCLP) and today with the OS/2 LAN Server. IBM's strategy is to upgrade, migrate, and gracefully evolve the existing IBM LAN solutions to become participants in DCE, thereby protecting customers' investments in hardware and software.

Requirements

This release of LAN Server 2.0 addresses several major requirements specified by IBM's customers. These requirements are:

- Support for mixed operating system environments

  This provides the ability to choose among DOS, DOS with Windows, or OS/2 for the desktop client environment. Now that IBM is moving toward a 32-bit operating system, businesses want the choices extended to include 16- or 32-bit OS/2. The message is that the future OS/2 LAN Server must support all these environments in any configuration that the user chooses.

- Compatibility with other software products and releases

  Businesses want IBM to ensure that each new release of OS/2 LAN Server can interoperate and coexist with previous releases. They want the freedom to upgrade clients and servers in any sequence or to maintain some set of clients and servers at the existing release level, while upgrading a second set to the new release level. Later in the article, we will describe how the next release of OS/2 LAN Server will interoperate and coexist with previous releases of IBM LAN products from PCLP 1.3 and OS/2 LAN Server 1.0 through 1.3.

  Businesses also want IBM to provide interoperability and coexistence with non-IBM LAN products. IBM's relationship with Novell is an example of how IBM is beginning to address this requirement.

- Support for non-IBM hardware

  Business environments include hardware from other hardware providers. These businesses are looking to IBM for LAN solutions that include support for PCs – IBM or compatibles – that have non-IBM versions of DOS or OS/2 installed. These same businesses want software to work with non-IBM LAN adapters that provide device drivers compliant with the Network Driver Interface Specifications (NDIS). IBM is addressing both aspects of this requirement in LAN Server 2.0.

- Support for standard interfaces

  Businesses are looking to IBM to support all programming interfaces explicitly defined by a standards organization like Open Systems Interconnection (OSI) and International Standards Organization (ISO)® or those interfaces such as NDIS and Microsoft LAN Manager Application Programming Interfaces (APIs) generally accepted as a standard by most network vendors. Continued support and incorporation of IBM-defined interfaces for Systems Application Architecture® (SAA™) and Common User Access™ (CUA™) is also expected. IBM's LAN Server 2.0 addresses all of these standards requirements.

- Exploitation of new technologies

  Businesses want timely availability of software that exploits new hardware and emerging technologies. Examples are 80386/80486 optimizations, dual-processor support, RISC System/6000™ as a server platform, larger media, and multimedia capabilities. Providing for the availability of new hardware, such as the Token-Ring Busmaster LAN Adapter card, without the concurrent availability of software from IBM is not acceptable. Support for the NDIS interface will enable the exploitation of new

LAN adapters within an existing OS/2 LAN Server 2.0 environment.

- Provision for additional system management facilities

  Since the availability of OS/2 LAN Server 1.0, businesses have demanded that IBM provide additional tools and services to simplify administration and management of the LAN. IBM has chosen to address this requirement by incorporating additional function within the LAN Server 2.0 product and by making available separate products to provide a specific system management function. Examples of products available today to provide this additional function are IBM System Performance Monitor/2 (SPM/2) for monitoring memory, and disk and processor utilization; Distributed Console Access Facility (DCAF) for remote console capabilities; and the NetView Distribution Manager (DM)/2 LAN Download Utility (LDU) for the distribution and installation of software from an OS/2 distributor workstation to LAN-connected OS/2 and DOS workstations. The implementation of First Failure Support Technology (FFST/2) within LAN Server 2.0 will enable the capture and reporting of first failure data and the provision of a generic alerter service that can send alerts to either IBM's NetView® or LAN Network Manager product.

- Support for a worldwide product

  Businesses with locations other than in the U.S. – Japan and Germany, for example – require a product that can be used worldwide. This includes country-specific considerations, tools, and service personnel necessary for worldwide support of OS/2 LAN Server. The goal is to provide concurrent general availability of the product. This means upon availability of the product in the U.S., it will be concurrently available in Europe and the Far East. The next release of OS/2 LAN Server starts to address this requirement.

LAN Server 2.0 Highlights

LAN Server 2.0 continues to provide local-area-network function supporting OS/2 and DOS clients, using OS/2 multitasking and application protection, and expanding the architecture to provide better performance, higher reliability, and broader security.

The architecture of OS/2 LAN Server 2.0 permits implementation on a co-processor.

386 High-Performance File Server: The OS/2 LAN Server 2.0-Advanced version provides a 386 High-Performance File Server, referred to as the 386 HPFS (Figure 1). It is an enhanced version of the HPFS for 386/486 computers, replacing the HPFS provided with OS/2. This file system runs at ring 0 with the same priority as the OS/2 kernel. The ring 0 implementation includes 80386-specific kernel extensions and provides a 32-bit data path. It also includes a special network-aware Installable File System (IFS) optimized for performance using common cache and contiguous file allocation mechanisms, supporting fault tolerance, and providing access control for files, print queues, directories, and serial device queues. The net result of this implementation is a fast file server competing very favorably with NetWare 3.11.

IBM intends to enhance LAN Server 2.0-Advanced to run on a 32-bit OS/2 operating system.

When a file request is received, the ring 0 server determines that if data is in the cache, the request can be satisfied without requiring any disk I/O; if not in the cache, the 386 HPFS calls the disk device driver to access the disk. In either case, the requested data is returned very quickly, requiring no ring transitions (going through the ring 3 server) or accessing the OS/2 kernel.

If a request is directed to the FAT file system or to an attached printer, the ring 0 server sends the request to the ring 3 server, which must interface with the OS/2 kernel to access the FAT file system or printer device driver. The transitions from ring 0 to ring 3 and from ring 3 to ring 0 cause processing of these types of requests to be much slower than requests directed to the 386 HPFS.

Local applications are running at ring 3, and must always go through the OS/2 kernel when accessing either the 386 HPFS or the FAT.

Co-processor Support: IBM has made a statement of direction for enhancement of the OS/2 LAN Server to support a co-processor environment. The architecture of OS/2 LAN Server 2.0 permits implementation on a co-processor. By installing an additional processor, some of the system code can be off-loaded to the second processor so that it runs in parallel with the code left on the main processor. The main processor is dedicated to running server-based applications, such as IBM's Database Manager product for OS/2. The intent is to provide high-speed file sharing and LAN transport functions, while significantly improving the performance of server-based applications by eliminating processor contention on heavily loaded network servers.

Local Security: Local security is a function provided by the 386 High-Performance File Server (Advanced Server). Local security is user-level security extended to the server console. When a user logs on at the server console, the user has access to those resources allowed remotely or to resources allowed by the local group. The local server disk is protected from unauthorized access because the access permissions are stored as extended attributes in the file system (386 HPFS) and not in a separate file as they are with the current release (OS/2 LAN Server 1.3). This protection remains in effect even if the Server service is not started, because the 386 HPFS is the file system that is running when the computer is rebooted or the Server service is stopped.

Operator Privileges: Operator privileges (or rights) provide a way for the administrative function to be divided into sublevels. This permits the administrator to delegate management of users/groups, services, and shared resources to another user by specifying the appropriate sublevel of authority. This user is then called an operator. Operators can be designated to manage accounts, server-based services, printers, or communication queues. Directory or file operators are not necessary because the Change Permission Access Control can be used to grant administrative privileges for a specific file or directory to a specific user.

Fault Tolerance: A fault tolerant subsystem (Advanced Server) is included in this release to provide protection from disk failure. This subsystem (Figure 2) provides for continuous fault monitoring (mirrored or unmirrored partitions), dynamic error

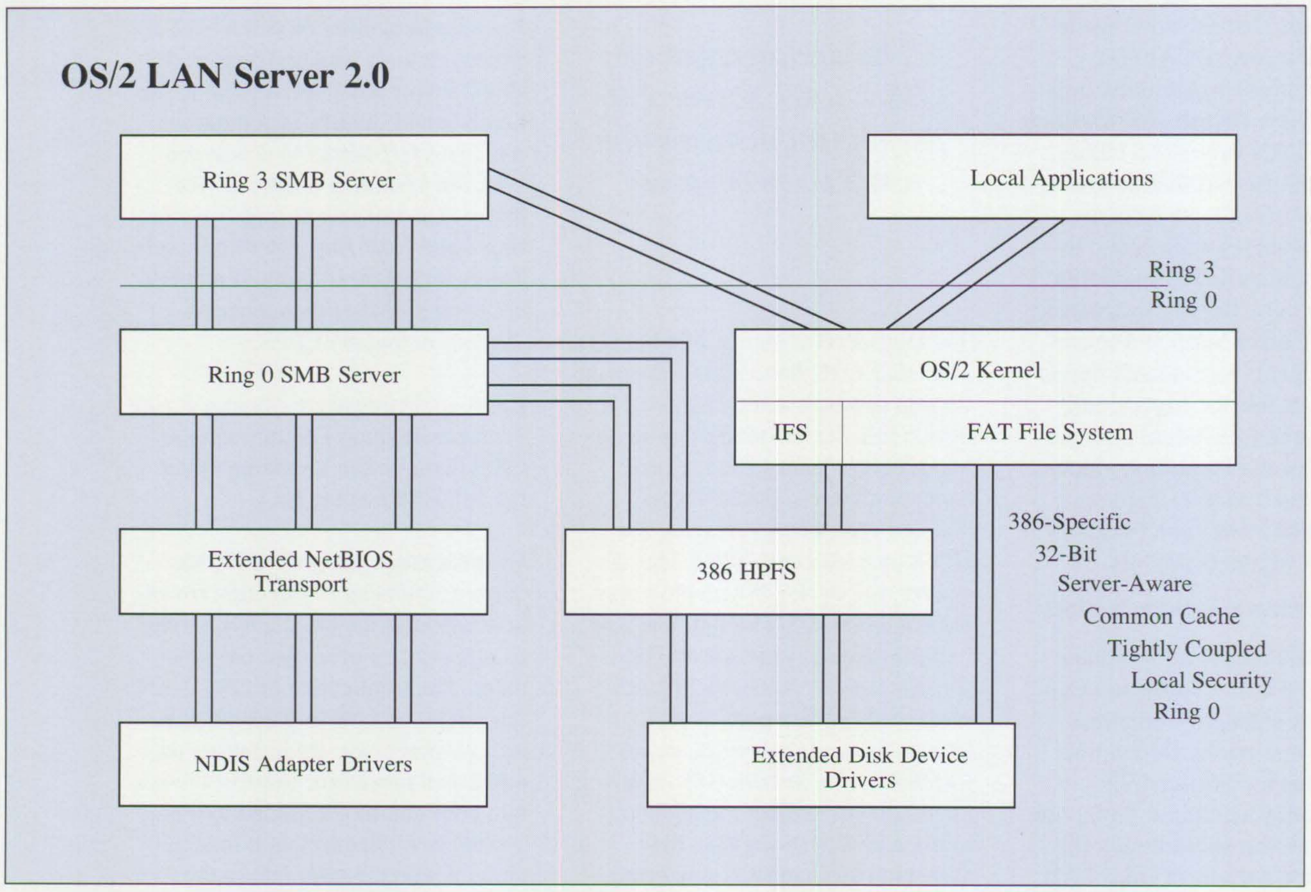


Figure 1. 386 High Performance File Server

correction (administrator can correct software errors, such as cracked mirror, without shutting down the server), disk mirroring for concurrent online backup of data, and disk duplexing, for extending fault tolerance to the drive controller. (In disk mirroring, a single drive controller is writing to two physical disks; in disk duplexing, two drive controllers are writing to two physical disks.)

Additional fault tolerant capabilities are provided via integration of support for an Uninterruptible Power Supply (UPS) with the OS/2 LAN Server alerter service. A standard signal is emitted when the UPS comes online. This signal is detected by the LAN Server 2.0. At this point, the alerter service notifies all users and administrators that the server is running on backup power. Users and administrators are also notified when the UPS is low on power. If power is not restored, a controlled, orderly shutdown of the connected server is performed. The data cache is flushed, the event is logged, and all open files are closed.

Fault tolerance has been extended to the logon function by allowing an additional server to validate logon requests. If the domain controller fails to respond to a logon request, the request is automatically routed to the backup, which assumes the role of the domain controller for user logon. After the logon request has been processed by the backup, the user's logon assignments are retrieved from the domain controller, if available.

When a user is attempting to logon and the domain controller fails to respond after a specified number of tries (three tries, for example), the logon request is routed to the backup. Logon is then accomplished by the backup logon server. If the domain controller is down, the user can still logon but would not be able to receive any logon assignments. Shared resources on other servers in the domain could be accessed by the user by explicitly issuing a NET USE to the desired resource.

Additional Alert Support: With this release, a new service is being provided that enables local LAN

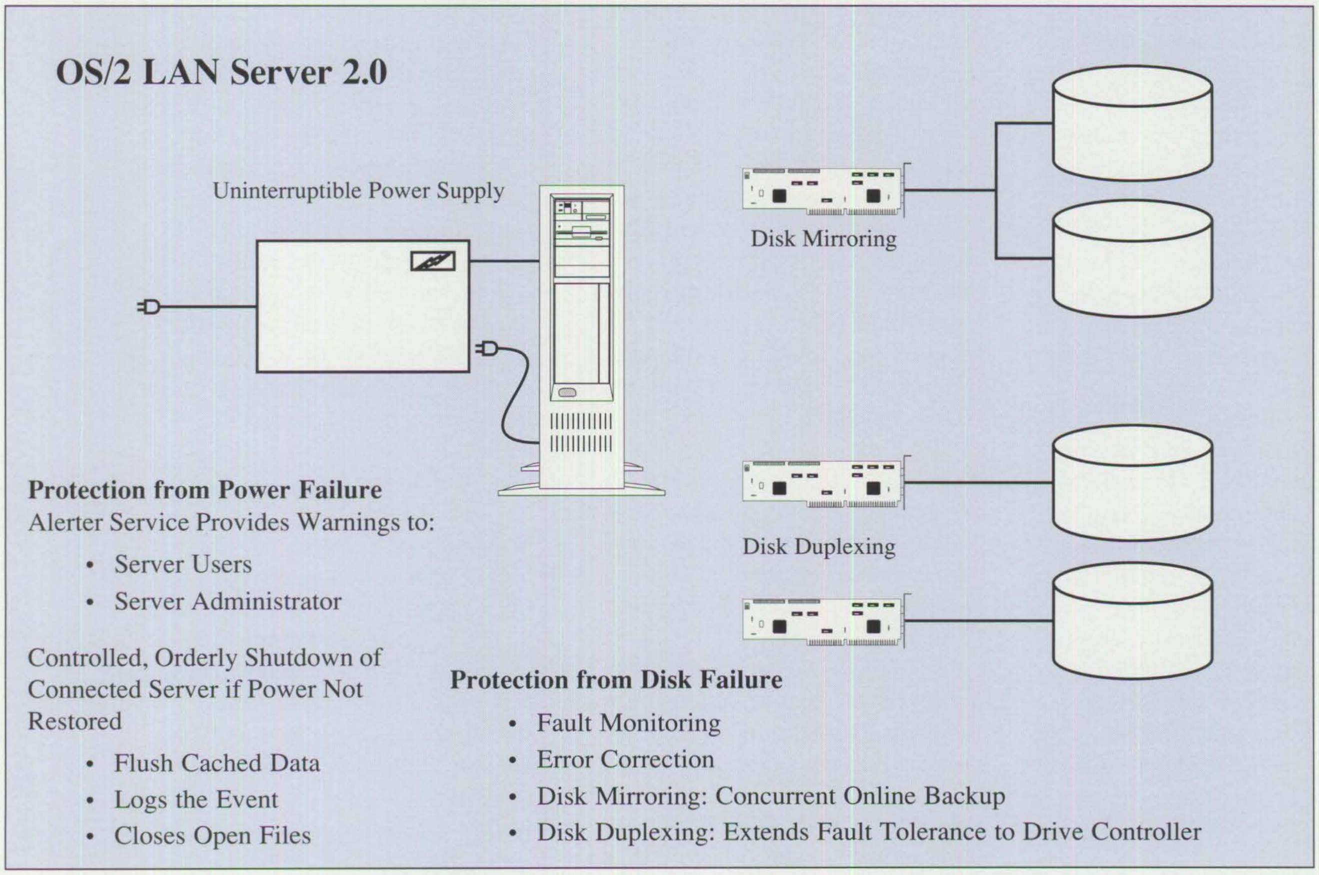

OS/2 LAN Server 2.0
Uninterruptible Power Supply
Disk Mirroring
Disk Duplexing
Protection from Power Failure
Alerter Service Provides Warnings to:
• Server Users
• Server Administrator
Controlled, Orderly Shutdown of Connected Server if Power Not Restored
• Flush Cached Data
• Logs the Event
• Closes Open Files
Protection from Disk Failure
• Fault Monitoring
• Error Correction
• Disk Mirroring: Concurrent Online Backup
• Disk Duplexing: Extends Fault Tolerance to Drive Controller

Figure 2. Fault Tolerant Subsystem

Server alerts to be sent as generic alerts, either to NetView or to the IBM LAN Network Manager. These alerts can be passed directly to NetView via an LU 6.2 session with the host or via an 802.2 session to the IBM LAN Network Manager (the alert can then be passed on to NetView). The generic alert is presented to the NetView operator as textual information that clearly describes exactly what has occurred on the LAN Server; for example, UPS commercial power failure, user's maximum storage exceeded, disk read failure, or excessive failures on a mirrored or unmirrored drive.

Diagnostic and Install Aids: Additional diagnostic capabilities are being provided through the implementation of FFST/2 in OS/2, LAN Server 2.0, and other system components. FFST/2 is an IBM-developed technology and operates similarly to a technology incorporated into a VTAM™ product called FAST-Service™. This technology enables the capture of data upon detection of a system failure (First Failure Data Capture, or FFDC). Problem management is then initiated at failure detection. This information can be dumped, printed, and forwarded to IBM for quicker resolution of problem conditions.

FFST/2 is a common set of tools for OS/2, Communications Manager, Database Manager, and LAN Server. FFST/2 provides error and message logging, data dumping, and the generation of generic alerts. The data dumping facility includes an output formatter so the data can easily be read and interpreted; selected portions of the data (rather than all of the data) can be dumped. The local LAN Server 2.0 alert is formatted into a generic alert by FFST/2.

To enhance the usability of OS/2 LAN Server, a Presentation Manager application is provided for graphical installation. This will make LAN Server 2.0 components easier to install, configure, and remove.

To enhance the usability of OS/2 LAN Server, a Presentation Manager application is provided for graphical installation.

LAN Transport Enhancements: LAN transport services are being enhanced for this release. IBM is externalizing the NDIS information, and is providing NDIS-compliant protocol managers for NetBIOS and 802.2. Support of the NDIS interface will permit release-independent configuration and installation of LAN adapters and device drivers. LAN Server 2.0 will be capable of supporting any new IBM or non-IBM LAN adapters that provide a Media Access Control (MAC) driver certified to be NDIS-compliant.

The LAN transport services are designed to support multiple protocols, and have been optimized for performance. Performance of NetBIOS, 802.2, and Advanced Program-to-Program Communications (APPC) has been significantly improved. The architecture defines standards for the future addition of protocol managers or device drivers.

Multiple Adapter Support: Multiple adapter support permits both server and OS/2 requester machines to connect to up to four logical networks concurrently. LAN Server 2.0 can support 1016 concurrent NetBIOS sessions with four adapters. The maximum number of NetBIOS sessions equals the total sessions available for all installed adapters per OS/2 device (254 x 4 = 1016). A domain controller or an additional server must be direct connected (not bridged) in order to share resources with OS/2 requesters in one or more logical networks. The four adapters can be any combination of supported topologies (Token Ring, Ethernet, or PCNet). No bridging function is provided by the OS/2 device.

Multiple adapters are not supported in bridged LANs. A problem occurs because the NetBIOS name associated with each adapter in an OS/2 device is the name of the server or requester workstation. In disconnected LAN segments, no problems arise when each of the adapters is known by the same name, because each adapter is on a separate physical LAN. But in bridged LANs, all the adapters are on the same physical LAN, so when the server or OS/2 workstation identifies itself, all the adapters attempt to initialize using the same name, causing a NetBIOS duplicate name error. The problem could be eliminated if internal bridging were supported on the OS/2 device or if the duplicate NetBIOS names could be filtered.

Time Source Service: The Time Source Service is a new service for OS/2 LAN Server. A domain controller can be designated to provide a reliable clock. The domain controller acts as the network time server; synchronization of time within the domain operates as it has in the previous version of the product. This facility will be useful primarily for applications that use APIs to synchronize time in a client-server environment.

Remote Initial Program Load: LAN Server 2.0 has additional Remote Initial Program Load (RIPL) facilities for OS/2. RIPL is provided for DOS and OS/2 workstations with the appropriate adapter. RIPLing of workstations with or without local media is supported. RIPL information for DOS is stored in an image file that must be built with specific image-build routines. The entire image must be downloaded to the DOS workstation before the IPL process can begin. RIPL information for OS/2 is contained in an OS/2 File Index Table (FIT) file, which can be edited with any ASCII text editor. The RIPL process for an OS/2 workstation is a multiphase process. During the first phase, OS/2 downloads a mini File System Driver (FSD) to the workstation. This mini FSD starts the OS/2 booting process and loads the base device drivers, Dynamic Link Libraries (DLLs), CONFIG.SYS, network device drivers, and the RIPL redirector.

Each OS/2 workstation can be uniquely configured or can use the default configuration. Figure 3 shows one OS/2 workstation doing IPL with the default configuration, while the other OS/2 workstation is doing an IPL with a unique configuration. Both workstations share a common set of OS/2 system files. At the end of the OS/2 RIPL process, the mini FSD is replaced with the FSD for the booted file system (FAT or HPFS). Writable OS/2 files, such as SWAPPER.DAT and OS2.INI, should be located on local media (if available on the workstation) for best performance. Each OS/2 workstation without local media must have its own set of writable files on the OS/2 LAN Server. Swapping across the LAN will not significantly degrade performance, but should be minimized when possible.

DOS LAN Requester Usability Enhancements: Usability of the DOS LAN Requester has been enhanced. Cooperation between DOS LAN Requester and Windows 3.0 is improved. A DOS LAN Requester user can log on or off via the Windows interface. The Files Browse function supports aliases and displays only those resources accessible to the user. The DOS LAN Requester

OS/2 LAN Server 2.0

- RIPL of both DOS and OS/2 workstations
- Support for workstations with appropriate adapter
- Multiple RIPL servers image maintenance via Replicater service
- Support for systems with or without local media
- Image build and image migration utilities for DOS workstations (OS/2 FIT files can be created with any editor.)

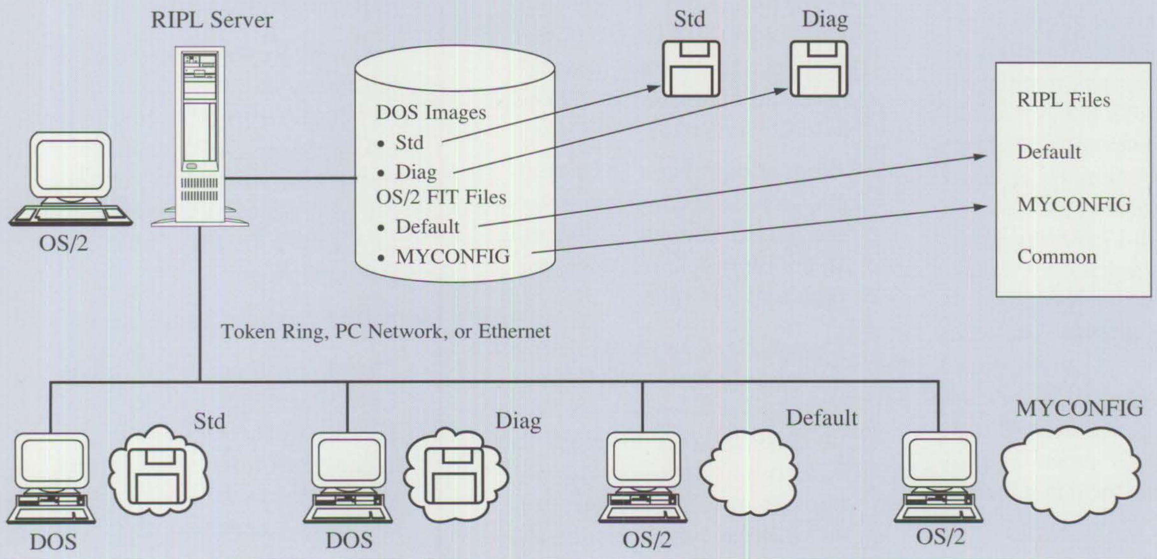


Figure 3. Remote Initial Program Load

user's resources can be automatically assigned at logon, and public applications can be selected from the Windows interface. This function is unique to DOS LAN Requester and LAN Server 2.0 – it is not provided by Windows or Microsoft LAN Manager. The function becomes available after installation of DOS LAN Requester with Windows 3.0. During installation, a module shipped with LAN Server 2.0 is used to replace a Windows 3.0 program module.

Double-Byte Character Set (DBCS) enabling is being provided by the DOS LAN Requester for the command line and for redirecting print jobs. This capability will enhance the usability of the product in countries such as Japan and Korea, where IBM has had only OS/2 client support available. Menu interface for DBCS users of the DOS LAN Requester is not yet available.

Online Publications and Helps: In this release of OS/2 LAN Server, the online reference information is a Presentation Manager application. This means that online reference information can be viewed in a window and the user can hot-key between the reference information and the LAN Requester interface as needed. In certain cases, complex definitions are presented graphically for faster comprehension of concepts and problem resolution. Hypertext words and phrases are linked to definitions; due to a very fast index search, complete articles can be displayed quickly in a Presentation Manager pop-up window. The online reference information can be printed, copied to a separate file, or copied between applications via the clipboard, facilitating the development of user-customized online documentation.

Helps are also implemented with hypertext links. A help panel, dialog box, or pop-up can remain on the screen while the user interacts with the applications.

A help panel, dialog box, or pop-up can remain on the screen while the user interacts with the applications.

The API documentation can be purchased separately and is provided as both hard copy documentation and online indexed reference.

Utilities and Administration Tools: Various utilities and administration tools are provided with LAN Server 2.0. These include:

- The DCDB migration utility for migrating the DCDB from PCLP 1.3 or OS/2 LAN Server 1.0 to LAN Server 2.0. DCDB migration is not necessary when moving from OS/2 LAN Server 1.2 or 1.3 to LAN Server 2.0.
- Access Control List (ACL) utilities to back up and restore the user/group accounts information, access control permissions, and audit trail information

  These utilities are useful in migrating from LAN Server 2.0-Entry, LAN Server 1.2, and LAN Server 1.3 versions to the 386 High-Performance Server so that an administrator is not required to rekey access control permissions. (Remember that access control is stored in the 386 HPFS for the LAN Server 2.0-Advanced, but is stored in separate NET.ACC files for the LAN Server 2.0-Entry server.)
- RIPL utility for migrating DOS RIPL environments from LAN Server 1.0, 1.2, or 1.3 to LAN Server 2.0

  The format for DOS images in LAN Server 2.0 is slightly different. The utility is used so that existing DOS images do not have to be re-created.
- Home directory utility for converting aliases from the old LAN Server 1.0, 1.2, or 1.3 format to the new LAN Server 2.0 format

  The utility can also be used to convert from the new format back to the old. (*Note:* With the refresh to LAN Server 1.3, home directories can now be located on any drive, not just C, and can be shared between users.)
- Complete set of fault tolerance tools to assist the administrator in configuring, managing, and monitoring errors on a server with disk mirroring or disk duplexing

  From any workstation in the LAN, the administrator can remotely correct many errors that occur in fault-tolerant systems. For example, a mirror can become cracked and can be repaired by the administrator via the FTADMIN utility, a PM application.
- A DASD check storage utility

  The administrator can set limits for the users' home directories. This utility checks a user or group against their limits and sends the information to the screen. Alerts can also be generated, and if the generic alerter service is running, this alert can be transmitted to NetView or the LAN Network Manager.

- Two utilities to change the domain controller or server name

Clients: The LAN Server 2.0 package includes a DOS client (DOS LAN Requester) and two OS/2 clients, one for OS/2 Standard Edition (SE) Version 1.3 and one for OS/2 2.0. The OS/2 2.0 client is provided so that users can start taking advantage of OS/2 2.0 on the desktop, while maintaining connectivity to an OS/2 LAN Server domain. The clients are provided via a LAN Requester's Distributed Feature.

Product Packaging

LAN Server 2.0 is available in two separately sold installation packages: OS/2 LAN Server Version 2.0-Entry (LAN Server 2.0-Entry) and OS/2 LAN Server Version 2.0-Advanced (LAN Server 2.0-Advanced).

LAN Server 2.0 has been packaged so that the server, OS/2 and DOS clients, and OS/2 and DOS LAN transport code are available via a single package. Previously, the LAN Server package contained only the server, the DOS client, and the DOS LAN transport code components. The OS/2 client and the OS/2 LAN transport code were provided with the IBM OS/2 Extended Edition (EE) products.

The LAN Server 2.0 package contains the server component, the DOS LAN Requester (DLR), the LAN transport code for DOS and OS/2, and OS/2 client code for OS/2 SE 1.3 and OS/2 2.0. The SE 1.3 client installs on OS/2 SE 1.3 (latest refresh code level) and the OS/2 2.0 client installs on OS/2 2.0.

LAN Server 2.0-Entry can be installed on either OS/2 SE 1.3 or OS/2 2.0. However, in order to use the 386 HPFS, the LAN Server 2.0-Advanced must be installed on OS/2 SE 1.3. IBM intends to provide a version of 386 HPFS that will support OS/2 2.0, enabling the LAN Server to provide disk mirroring and duplexing and local security to an OS/2 2.0 server environment. LAN Server 2.0-Entry is an application and runs at ring 3 (priority level), whereas LAN Server 2.0-Advanced is a system process that runs at ring 0 (a higher priority). The ring 0 implementation is a rewrite of much of the 386 HPFS and ring 0 server code for OS/2 2.0.

LAN Server 2.0 has been packaged so that the server, OS/2 and DOS clients, and OS/2 and DOS LAN transport code are available via a single package.

The LAN Requester's Distributed Feature is offered to enable DOS 3.3, DOS 4.01, DOS 5.0, Windows 3.0, OS/2 SE 1.3, and OS/2 2.0 workstations to access the OS/2 LAN Server in a client/server relationship. The function offered in the Distributed Feature is a subset of that provided by LAN Server 2.0. The Distributed Feature may be purchased in any number appropriate for client support of the above mentioned packages, and must be purchased as a license for each workstation upon which the client code is installed.

Extended Services (EXTD/2) is installed on either OS/2 SE 1.3 or OS/2 2.0. Businesses with combination gateway, database, and LAN servers will still be able to have a combination server with LAN Server 2.0.

Non-IBM Hardware Compatibility

This release of LAN Server 2.0 provides support for selected systems running non-IBM versions of OS/2 SE or DOS. The vendors – Compaq, Tandy, Olivetti, Seimens/Nixdorf NCR, AST, Compuadd, and others – have contractually agreed to test IBM's software on their hardware, using IBM's test suite and following IBM's guidelines. As a result, IBM has agreed to provide service and support to customers who install IBM's software on these vendors' hardware.

Support is also provided for non-IBM LAN adapters whose device drivers are certified to be NDIS-compliant. IBM uses an independent testing lab to provide non-IBM vendor certification.

Server/Requester Compatibility

The new LAN Server 2.0 and DOS and OS/2 clients are supersets of the same components in previous releases of LAN Server. The new OS/2 clients have complete administrative capabilities when logged on to an OS/2 LAN Server 2.0 domain and can access and administrate OS/2 1.3 domains. OS/2 EE 1.2 and 1.3 LAN Requesters can also administrate the LAN Server 2.0 domain controller, but are limited to the functions supported in the 1.2 or 1.3 releases. Any supported level of DOS LAN Requesters can logon and access resources on the 2.0 server.

A LAN Server 1.3 server can exist in the same domain with a LAN Server 2.0 server and each will provide the capabilities that are described for the respective release.

An OS/2 EE 1.1 LAN Requester and a PCLP 1.3 Requester can access resources on a higher-level server after logging on to the domain controller appropriate to each.

LAN Manager Interoperability

LAN Server is based on Microsoft's LAN Manager. All implementations of LAN Manager are based on a standard software library, which means they are compatible and interoperable. LAN Server is tested to ensure a high degree of interoperability with Microsoft's product.

Comparison of LAN Server 2.0 and LAN Manager 2.0

This release of LAN Server supports the LAN Manager 2.0 APIs and, with few exceptions, all the function of LAN Manager 2.0. Both products contain the following:

- 386 Server Option
- Disk Fault Tolerance
- Uninterruptible Power Supply Support
- NDIS Support
- OS/2 RIPL
- Domains
- DOS Windows Support
- LAN API Support
- Named Pipes Support
- Original Equipment Manufacturer (OEM) Support

In addition, the IBM LAN Server 2.0 product provides:

- Dynamic Resource Sharing
- Served Applications Support
- RIPL over Ethernet
- Alias Support
- User Profile Manager Functions
- Online Publications

Through this level of support, IBM LAN Server 2.0 provides near functional parity with LAN Manager 2.0. In addition, by providing the IBM-unique features, the LAN Server 2.0 product achieves an advantage in many situations.

More details of the specific items supported in the product are available in the announcement materials and in

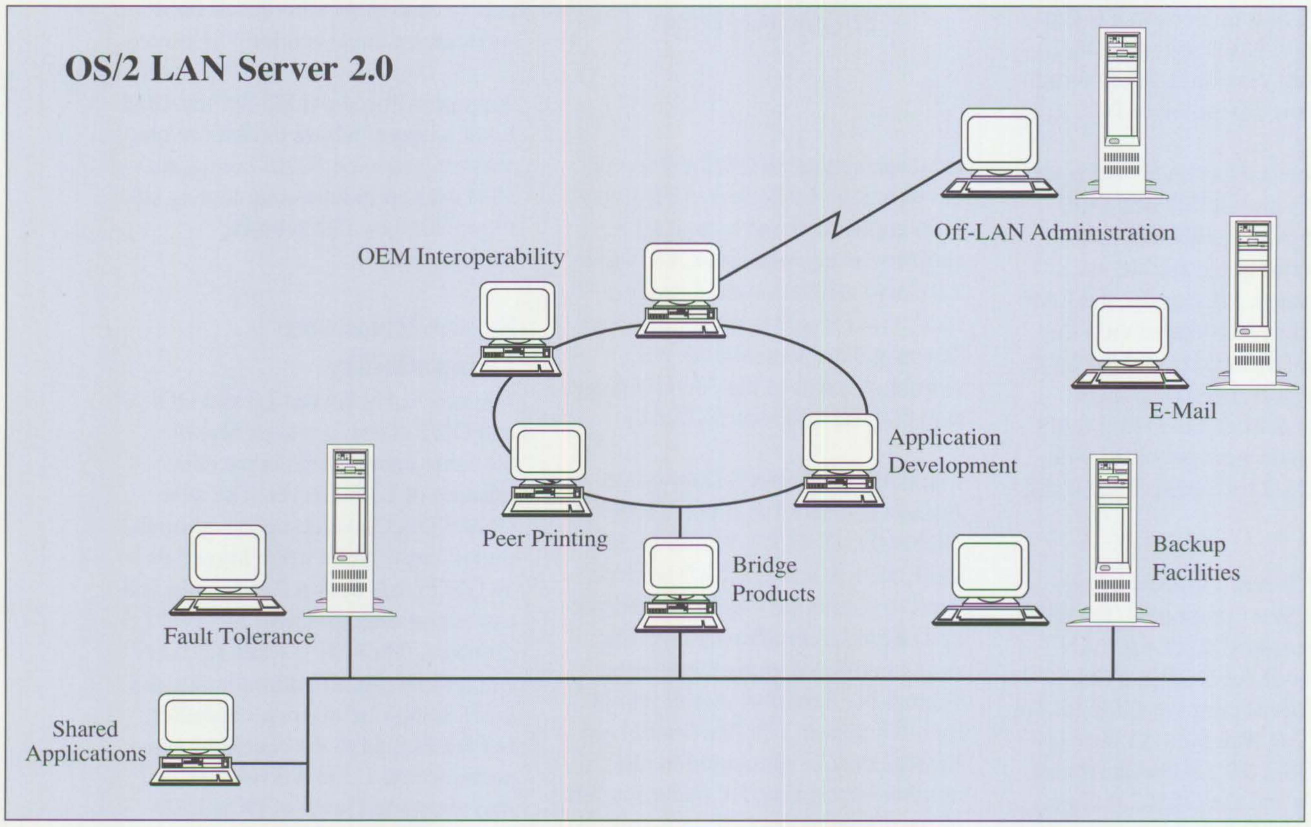


Figure 4. Expanding the LAN Server Environment

the OS/2 LAN Server Version 2.0 Information and Planning Guide.

Complementary Solutions

IBM seeks to provide total network solutions to its customers. Therefore, IBM encourages the development of products that enhance the LAN Server environment. Products are available from both IBM and OEM vendors that complement the capabilities of the OS/2 LAN Server in the areas of backup, peer printing, off-LAN administration, and interoperability (Figure 4). Some of the vendor products are Ellipse, a transaction processing and development system, Spool+ and LANSpool to provide peer printing capabilities, and Filesafe OS/2 and Sytos Plus® for automatic file backup to tape.

A complete listing of OS/2 program products that work with OS/2 LAN Server can be found in the OS/2 Application Solutions cross-referenced directory of applications published by IBM (G362-0002). This directory contains more than 1,400 OS/2 application listings.

Through the Developer Assistance Program (DAP) and the Software Developer Program (SDP), IBM provides assistance to vendors working on complementary solutions products for the OS/2 and DOS operating system environments. Recently, IBM announced a LAN Application Certification program to encourage OEM vendors to enable applications that run on the OS/2 LAN Server.

LAN Server and NetWare Interoperability

IBM's relationship with Novell is an example of IBM providing interoperability with another vendor. LAN Server 2.0 interoperability with NetWare will be a phased process.

Phase 1 is client co-existence, and is available today. This phase provides concurrent access to LAN Server 2.0 and NetWare from either an OS/2 or a DOS client workstation.

Phase 2 is client-server interoperability. IBM plans to provide function that will allow a LAN Server 2.0 client to access either a NetWare server or a LAN Server 2.0 domain via a single logon.

IBM will provide distributed system leadership, and Novell plans to participate in the IBM distributed system strategy.

This level of interoperability is planned to provide:

- Single logon for the OS/2 LAN Server 2.0 client to servers on multiple platforms
- Single view of a mixed server environment

In phase 3, advanced technology is used for distributed processing. IBM and Novell plan to improve seamlessness by supporting common extensions and by developing standards.

By this phase, transparent network integration of IBM LAN systems, non-IBM systems, and SAA systems should be achieved. Both LAN Server 2.0 and NetWare are to be included in the evolution from today's environment to the IBM distributed system. IBM will provide distributed system leadership, and Novell plans to participate in the IBM distributed system strategy.

The LAN client will be provided a single view of the system and transparent access to resources in a heterogeneous distributed environment. The LAN client can be IBM's, Novell's, or any other vendor that chooses to participate. IBM plans to base its distributed system on the DCE defined by OSF in May 1990. (Information on DCE is generally available.)

Product Direction

The following details IBM's directions for OS/2 LAN Server:

- OS/2 LAN Server on OS/2 SE 2.0 (ring 0 implementation of the 386 HPFS); 32-bit APIs; exploitation of OS/2 SE 2.0 capabilities; support of a co-processor by the server.
- Server function on other platforms; for example, resource sharing for OS/2 and DOS clients as OS/2 LAN Server does today but extended to other systems, such as IBM hosts that might have performance and capacity advantages over that available today.
- Heterogeneous workgroup interoperability – beginning of enterprise-wide DS; incorporate dissimilar clients, such as Macintosh®, into OS/2 LAN Server network solutions.
- Evolve OS/2 LAN Server to become an integral component in a distributed LAN system environment.

Distributed LAN System Evolution

IBM strategy is to upgrade, migrate, and evolve IBM LAN solutions to a distributed LAN system environment planned to be based on DCE and Distributed Management Environment

(DME) technologies. Additional distributed services are planned to be provided so that both user and administrator will have transparent access through a single logon to resources in a heterogeneous environment.

Object-oriented interfaces are desirable and, where applicable, will be exploited.

The Future

In the future, a majority of IBM's customers will still be engaged in simple workgroup computing – sharing LAN resources, accessing host data and applications, and transferring files between desktop computers and hosts.

Some businesses are currently implementing LAN distributed solutions based on the client-server computing model. (OS/2 LAN Server 2.0 is built on this model and enables client-server programming today.) Some of these businesses will move toward implementing more sophisticated solutions in a distributed computing environment. They will want to distribute data (applications, files, and objects) to be shared in this environment. They will want to do enterprise and wide-area networking on a peer-to-peer basis. They will need basic services and tools, such as directory, remote procedure call, print, and communications, to take advantage of the distributed system. System and network management tools that provide performance, configuration, problem, accounting, and security management services will also be required for LAN distributed solutions as well as for the distributed LAN system. (IBM is actively participating in the definition of the technology base for DME.)

IBM is actively participating in the definition of the technology base for Distributed Management Environment.

There are three issues that affect this movement:

1. Increased reliance on PC LANs will require LAN distributed solutions to be available sooner.

2. The downsizing of host-based applications to LAN implementations will require that centralized management and host-based tools be provided in the LAN environment.

3. A desire to take advantage of the LAN's capabilities and the power of the PC to improve productivity will require that additional functionality be provided for workstation-based LAN solutions.

IBM will be an active participant in all phases of this movement, with OS/2 LAN Server 2.0 as the strategic product for achieving the desired results.

ABOUT THE AUTHORS

Barbara Barker is an advisory programmer in the LAN System Products group providing planning and technical support for the OS/2 LAN Server Product. Previous experience includes work on an advanced technology editor and work in Datastream architecture. Barbara holds a BA in mathematics and a MS in computer science from the University of Texas at Austin.

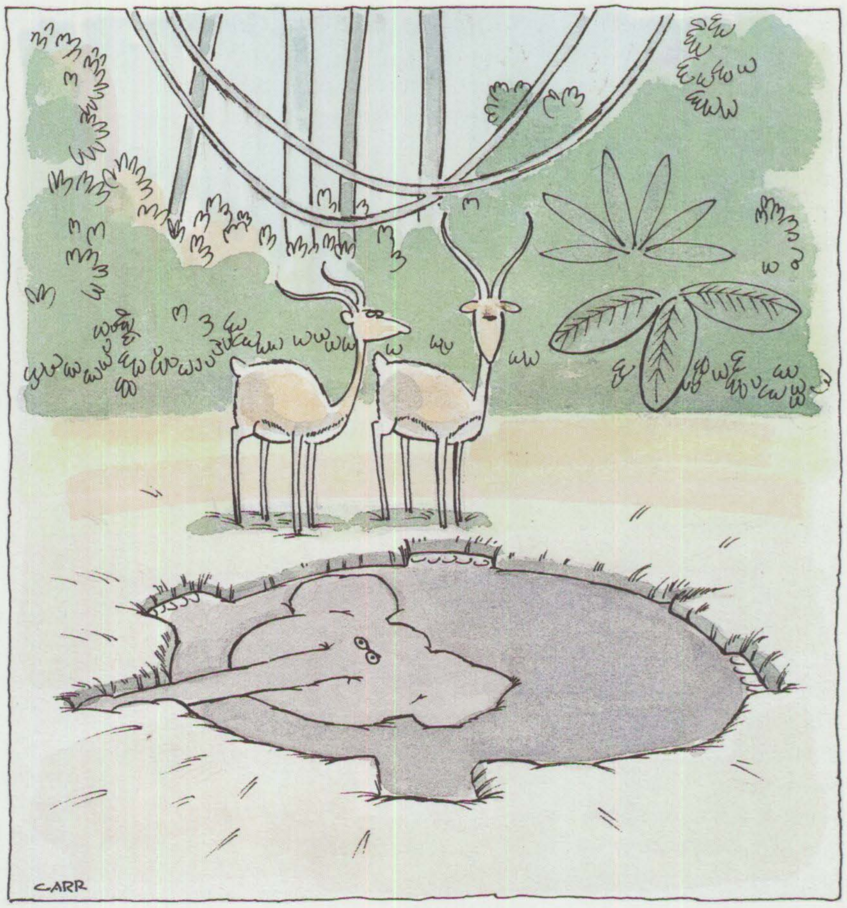
Carolyn Easter is an advisory programmer in IBM's Entry Systems Division in Austin, Texas in the LAN System Products planning organization. Carolyn provides technical marketing support to both internal and external organizations and individuals. She has a BA in computer science from the University of Texas at Austin.

John McCarty is a planner in the LAN System Products organization where he handles the relationship between the development group and IBM marketing organization. Previously John was a program manager for the Asia/Pacific Group representing Austin's hardware products and software offerings to the Asia/Pacific countries.

OS/2 2.0 Memory Management

Jeff Beardsley and Scott Sorensen
Southlake, Texas

This article explains the OS/2 2.0 32-bit flat memory model and shows the benefits of the flat memory model over the segmented memory model used in earlier versions of OS/2. Included are discussions of the segmented memory model, the flat memory model, paging, swapping, and OS/2 2.0 memory objects.



One of the new features of OS/2 2.0 is a simpler and more portable memory management facility targeted at both higher performance and an eventual port to other hardware platforms. This facility is based on a 32-bit flat memory model, where virtual memory is handled by paging rather than by the segment swapping and memory compaction scheme used in Version 1.X. New Application Programming Interfaces (APIs) have been defined to allocate and handle OS/2 2.0 memory objects and control page attributes.

Segmented Memory Model versus Flat Memory Model

Previous versions of OS/2 are based on a segmented memory model. In this model, memory is referenced by a selector and an offset. The selector points to the segment where the data is located, and the offset specifies the exact position of the data within the given segment. When memory is allocated and used this way, each item in memory must be referenced with a combination of two 16-bit registers: one each for the segment and the offset. This is known as the *16:16 memory model*. Because the offset into each segment is stored in a 16-bit register, each segment is limited to 64 KB. The most obvious disadvantage of the 16:16 model is the difficulty incurred in allocating arrays larger than 64 KB. This is known as a *huge memory allocation*, and involves the allocation of multiple consecutive entries in the appropriate Local Descriptor Table (LDT) and occasional calls to special API functions to adjust pointers to shared huge allocations.

OS/2 2.0 uses a 32-bit flat memory model and is conceptually much easier to understand and use. This is accomplished by mapping the stack, code, and data segments throughout the machine into a single large linear system address space, allowing the programmer to view the machine as such.

Because segmentation within the 80386 processor cannot be disabled, OS/2 implements linear addressing by defining a single large code segment and a single large data segment, both with starting addresses at zero. Offsets into these segments are therefore equivalent to linear addresses with 32-bit offsets providing 4 GB of address space. (Currently, each process is limited to 512 MB of address space to remain compatible with previous versions of OS/2, but this restriction will change in some future release.) This system is known as the *0:32 memory model*.

Benefits of using the 0:32 memory model include not only simplicity and flexibility, but a significant performance increase over previous OS/2 versions. This performance increase is due to the near elimination of segment register reloads. In OS/2 Version 1.X, memory protection is handled at segment register load time by invoking the 80286 hardware memory protection scheme to verify access rights and segment limits. A different scheme is used by 32-bit OS/2, which is shown later (see "Paging Particulars").

Paging versus Swapping

In 16-bit OS/2, memory is allocated in segments. Logically, a segment can be any size from a single byte to 64 KB. This memory is allocated on paragraph bounds, that is, multiples of 16 bytes. When memory is full and additional allocation is requested, a Least Recently Used (LRU) algorithm chooses one or more segments to be swapped to disk. After these segments are copied to the swap file, memory compaction can take place. This consists of moving blocks around in physical memory to consolidate free memory blocks into a larger contiguous space.

Under 32-bit OS/2, memory is divided into fixed, non-movable 4 KB blocks called *pages*. When a new page is needed, the LRU algorithm chooses a page to write to disk, and the new page is brought into memory in its place. Memory never needs to be shuffled, because all pages are the same size and are logically ordered within a page table for each process. This is a much more efficient system than segment swapping.

Paging Particulars

Each process in 32-bit OS/2 has its own page table whose entries specify individual pages. The upper 20 bits of each page table entry contain the base address of the page, while the lower 12 bits contain its attributes. These lower bits can be used for page attributes, because all pages are aligned on 4 KB boundaries. This means that the lower 12 bits of the base address for each page will always be zeros, and storage for these bits can therefore be used for page attributes. Memory sharing is accomplished by defining the page to be shared in the page table of any sharing process. This scheme is illustrated in Figure 1.

Under 32-bit OS/2, memory is divided into fixed, non-movable 4 KB blocks called pages.

Page attributes are used to determine which memory accesses are allowed. When allocating memory objects in the flat memory model, page attributes can be controlled by the application program. The following page attributes can be specified by the program:

COMMIT	The page is committed to physical storage. This can be either physical memory or the swap file.
DECOMMIT	The page is not committed to physical storage, and attempts to access it will cause an access fault.
EXECUTE	Execute access to the committed page is allowed.
READ	Read access to the committed page is allowed.
WRITE	Write access to the committed page is allowed.
GUARD	The committed page is a guard page. Any other marked access is allowed, but such accesses generate a guard page fault.

When an application tries to access a page in a manner that is inconsistent with the page attributes, an access fault is generated.

The main purpose of the guard page is to allow for automatic stack growth. This is accomplished by committing a guard page at the top of the stack. When the stack grows into the guard page, a guard page exception is generated. The operating system then tries to commit another guard page, and, after removing the GUARD flag, uses the existing guard page to increase the size of the stack.

Applications can use guard pages in the same manner. A guard page can be committed at the end of a range of memory. When the range of memory is exceeded and the application tries to access the guard page, a guard page exception is generated. The application can then trap this guard page exception if it has registered an exception handler with the system. When the application's exception handler has received the guard page exception, it can handle the situation accordingly by committing the guard page for use or by handling the exception in another manner.

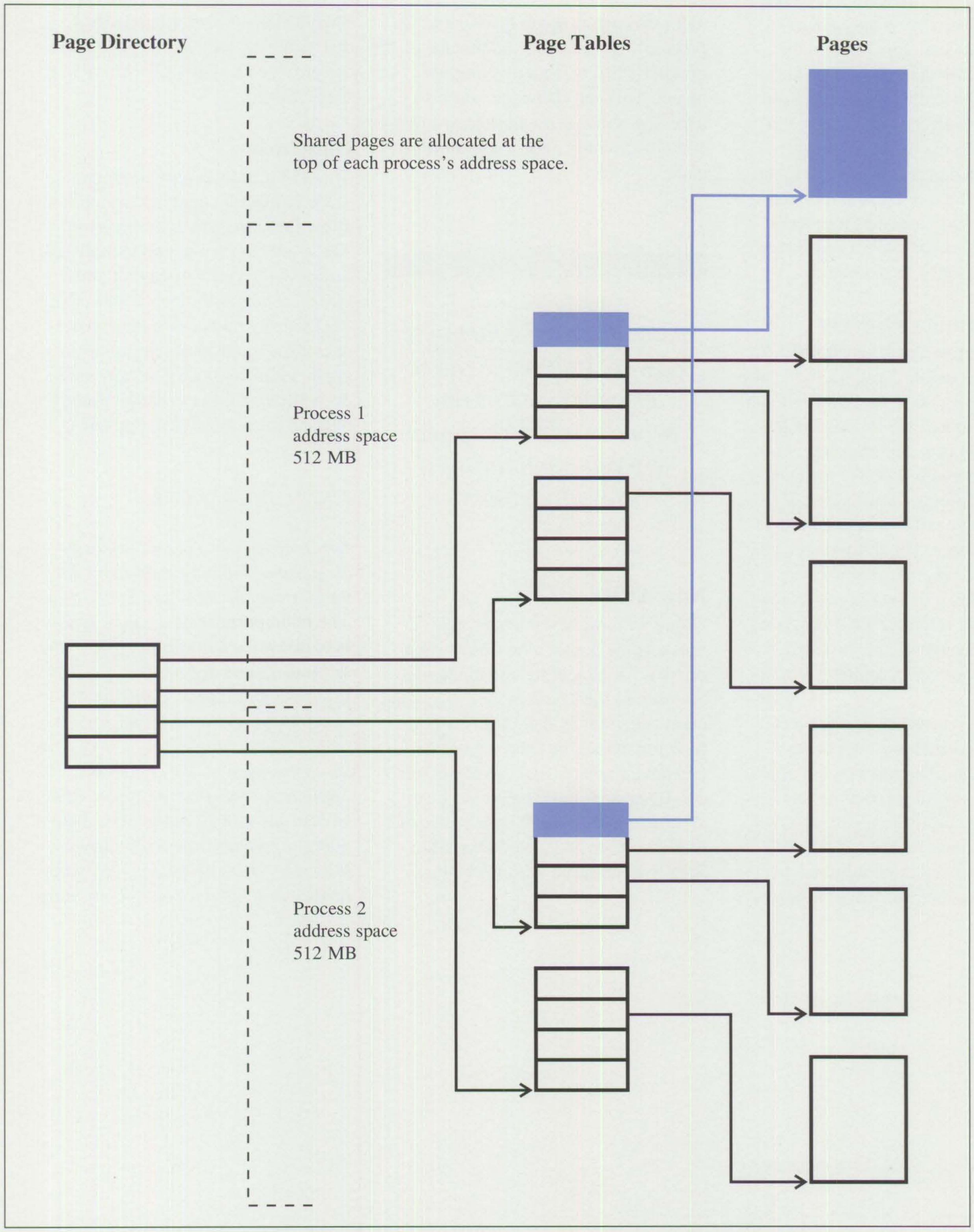

Page Directory
Page Tables
Pages
Shared pages are allocated at the top of each process's address space.
Process 1 address space 512 MB
Process 2 address space 512 MB

Figure 1. Illustration of Paging Scheme. Page table entries pointing to shared memory are indicated in dark blue.

The 80386 also provides a cache buffer that is contained within the paging unit on the microprocessor. This buffer is called the Translation Lookaside Buffer (TLB) and it contains the address of the 32 most recently used pages. This buffer can satisfy most requests for pages and eliminate many accesses to system memory. This can have a significant impact on paging performance.

OS/2 Memory Objects

Using memory objects is another new feature of OS/2 2.0. Memory objects between 1 byte and 512 MB can be requested and are satisfied with one or more pages of memory. Allocation of a memory object reserves a range of addresses in the process address space without committing it to real memory. The individual pages of the memory object can then be committed to real memory as needed, thus making efficient use of physical memory resources. This is called *sparse memory allocation.*

The size of a memory object is set when it is created and cannot be changed. It is also important to note that even if the object is only one byte in size, the system still allocates 4 KB. For these reasons, it is suggested that when large numbers of small objects are needed, programmers allocate as much memory as the process will need and suballocate individual objects from this memory object. This can eliminate waste by allowing multiple small objects to be contained within a 4 KB committed page.

Allocation of a memory object reserves a range of addresses in the process address space without committing it to real memory.

New APIs

Along with the new memory allocation scheme comes a new collection of APIs for allocating and managing memory in the 32-bit model. As with other functions in the 32-bit API, the function names are more consistent, and the parameter lists are more logical. Because segmentation has been eliminated, DosAllocSeg is replaced with DosAllocMem, and DosAllocShrSeg is replaced with DosAllocSharedMem. New functions for querying and setting page attributes are provided as DosQueryMem and DosSetMem.

Conclusion

The 32-bit flat memory model provides immediate benefits, which will allow OS/2 to grow into the future. The model is easy to understand and brings the benefits of paging and OS/2 2.0 memory objects. The 32-bit flat memory model also overcomes one of the major obstacles in porting applications to other platforms and even allows for the eventual port of OS/2 to other hardware platforms.

ABOUT THE AUTHORS

Jeff Beardsley is a fourth-semester cooperative education student with the IBM Application Assistance Center in Southlake, Texas, and a senior in computer science at the University of North Texas. Jeff's area of expertise is the OS/2 base operating system and Presentation Manager.

Scott Sorensen works at the OS/2 Application Assistance Center where he also specializes in the OS/2 base operating system and Presentation Manager. He is a senior in computer science at Brigham Young University.

Coding for Performance Under OS/2 Version 2.0

Ron Morrill
IBM Corporation
Dallas, Texas

OS/2 Version 2.0 is designed to exploit the 32-bit environment while providing upward compatibility for 16-bit applications. This article discusses several ways programmers coding 32-bit, 16-bit, or mixed-mode applications can significantly improve application performance by exploiting the new features and functions of OS/2 2.0.

OS/2 Version 2.0 requires an 80386 or 80486 processor (or their functional equivalents, the 80386SX and 80486SX) and was written to exploit the features of these 32-bit processors. It uses additional registers and arithmetic instructions. It uses doubleword move instructions. It employs scatter/gather for SCSI hard drives (and simulates it for ESDI drives). And it uses a flat memory model with paging architecture rather than segment architecture.

OS/2 2.0 allows you to run a wide variety of application programs. It supports the 16-bit Application Programming Interfaces (APIs) you've been using, so existing 16-bit applications will run unchanged under Version 2.0. However, if you change existing applications to take advantage of the high-performance 32-bit APIs provided in Version 2.0, you can expect significant performance improvements. You can use both 16- and 32-bit APIs in the same program; but by coding new applications exclusively for the 32-bit platform, you can achieve far better performance than was possible under Version 1.X.

OS/2 2.0 supports a wider variety of DOS applications as well. In addition to supporting Microsoft Windows 2.X and 3.0 applications, it supports the Expanded Memory Specification (EMS) standard with up to 32 MB of memory per DOS Mode session, it supports the Lotus-Intel-Microsoft (LIM) and Expanded Memory Specification (XMS) standards with up to 16 MB of memory, and it supports DOS Protected Mode Interface (DPMI) with up to 512 MB of memory.

In addition, asynchronous communications support for DOS applications has been improved. Asynchronous applications can now support line speeds up to 9600 bps.

The DOS Compatibility Box in Version 1.X permitted you to run one DOS application in real mode. Version 2.0 uses protected mode, allowing you to run up to 240 sessions concurrently and safely. Applications are protected from each other, so a catastrophic error in one program will not bring the entire system down. System performance is improved because switching between

real and protected mode has been eliminated, and execution of DOS applications is no longer suspended when another application is in the foreground. DOS applications can run windowed, and execution continues in the background.

OS/2 32-Bit Memory Management

OS/2 2.0 is a virtual memory system that uses a flat memory model. It uses a 32-bit near pointer, so far calls, which have the overhead of a segment register load, are not required.

The 32-bit operating system can address 4 GB of memory, but because of the need to convert addresses between 16- and 32-bit programs, a process can allocate only the memory below 512 MB. This area, known as the *process address space*, includes a 64 MB system area, leaving 448 MB of memory available to each process.

OS/2 2.0 divides memory into *pages* of 4 KB each. Instead of swapping segments of various sizes, it moves pages in and out of physical memory. Because pages are all the same size, paging is inherently simpler to manage than swapping segments, and therefore results in better performance.

In addition, 32-bit applications use a new call, DosAllocMem, to request memory in *objects* rather than in the segments used by 16-bit applications. Although an object may be any size from 1 byte to 448 MB, DosAllocMem reserves address space in 4 KB (one page) increments. DosSubAlloc (and the C language statement malloc) can be used to allocate memory for objects smaller than 4 KB.

Memory objects cannot be changed in size, but the amount of memory allocated to an object can be changed in one-page increments. If an application uses DosAllocMem to allocate a large, sparse, memory object, OS/2 reserves the address space, but does not actually assign physical memory to it. The actual assignment occurs when the application commits to using specific pages of the memory object by issuing a DosSetMem call. The physical memory is released (but remains reserved) when the application uses DosSetMem to decommit specific pages.

OS/2 2.0 is a virtual memory system that uses a flat memory model.

The segments allocated by 16-bit applications (using DosAllocSeg and DosAllocHuge) are translated into objects, and their sizes are rounded up to the next 4 KB so that each segment uses one or more pages. However, segments can be reallocated dynamically up to a maximum size of 64 KB, and OS/2 2.0 must reserve the space against possible segment size changes. Hence, in OS/2 2.0, DosAllocSeg reserves a minimum of 64 KB of the address space for each object, no matter how small.

Technique: The optimum memory management strategies in the 16- and 32-bit environment are very different. In the 32-bit environment, allocate large memory objects and then commit pages as the memory is required. When the memory is no longer required, decommit the pages or free the memory object.

You can improve the performance of mixed-mode applications (applications that use both 16-bit and 32-bit APIs) and applications designed to run in both environments by defining segments in 32 KB and 60 KB sizes.

A 32 KB segment is a good size for OS/2 to handle in either environment. It's large enough to be an efficient size to write to disk in both paging and swapping systems, and it's a good size for code and data handling.

A 64 KB segment is the obvious next size. However, when it swaps a segment to disk, OS/2 must add information to enable it to manage the segment. Since 64 KB is the maximum for a single write to an ESDI drive, a 64 KB segment, with the additional data, would require two writes. For this reason we recommended allocating segments of 64 KB minus 512 bytes under Version 1.X; in the paging environment, however, OS/2 allocates an entire page for the additional data, so we recommend 64 KB minus one page, or 60 KB.

Memory protection violations can occur in both environments, but they happen in slightly different circumstances. In the 16-bit environment, an error condition occurs every time a program addresses a memory location beyond the segment, even if it goes only one byte beyond. In the 32-bit environment, a program can address storage beyond its segment (or object) boundary without getting an error as long as it doesn't go beyond the page boundary.

Technique: Testing a new routine in such circumstances can be a problem. Try replacing your usual memory allocation routine with one that uses DosAllocMem to get enough memory to hold the requested object plus one page. Your test routine should commit the allocated memory except for the last page. DosAllocMem returns a pointer to the first byte of the object; align the last byte of the object with the last byte of the committed memory by adding the amount of unused, committed space to the pointer (Figure 1).

If the program addresses a memory location beyond the last byte of the object, a memory protection violation will occur because it has gone beyond the page boundary to an uncommitted page.

Good memory management is important to good performance under OS/2 2.0. In general, keep items in memory only if they are needed; when they are no longer needed, decommit the pages they occupy or free the memory objects. Otherwise, extra paging – and slower performance – will result. However, if the amount of memory involved is small, use DosSubAlloc or malloc (which commit the requested memory) and keep the memory rather than repeatedly allocating and freeing memory.

Tasking/Thread Management

OS/2 2.0 increases the number of threads allowed from 255 to 4096 and contains two new thread-related APIs that can help you improve application performance: DosWaitThread and DosGetThreadInfo. DosCreateThread has been enhanced.

DosCreateThread allows you to create threads in either an active or suspended state. Threads in a suspended state can be activated with DosResumeThread, and you can use DosWaitThread to block a thread until other threads in a process terminate. DosGetThreadInfo gives you access to the Thread Information Block, which in OS/2 2.0 contains thread and process variables.

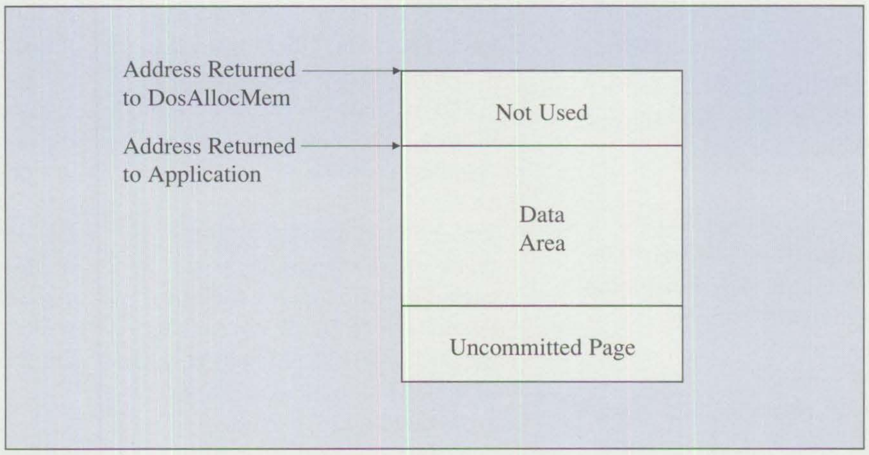

Address Returned to DosAllocMem
Address Returned to Application
Not Used
Data Area
Uncommitted Page

Figure 1. New Allocation Routine

Tip: Because the overhead associated with starting and destroying threads is greater than that of keeping them in a suspended state, you should start groups of threads, activating them as necessary, rather than starting and destroying individual threads frequently.

Under OS/2 2.0, each thread requires a minimum of 4 KB for a stack, even though it may not use the whole page. The process is similar to that used for allocating memory objects. DosCreateThread allocates the initial pages, and as the stack size increases, DosCreateThread uses the 80386/80486 guard page feature to commit additional memory for the stack as required. When the thread terminates, DosCreateThread frees the stack.

Tip: Memory for stacks is allocated in pages, so you can help performance by designing the program to use a stack size that is slightly smaller than a multiple of 4 KB, rather than slightly more.

Interprocess Communications

Like Version 1.X, OS/2 Version 2.0 includes several ways for threads and processes to communicate with each other. The best performer, of course, is shared memory; the others (semaphores, queues, signals, and pipes) have been improved in both performance and syntax.

Semaphores have been simplified in function, implementation, and syntax. Each type of semaphore has its own code path in OS/2 2.0, and this results in a significant performance gain over the single code path used for all types of semaphores in the 16-bit environment.

OS/2 2.0 offers two classes of semaphores. Private semaphores communicate among the threads in a single process and public semaphores can communicate among processes.

There are three types within each class of semaphores:

- Event semaphores signal events.
- Mutual Exclusion (Mutex) semaphores provide mutually exclusive access to a resource. If two or more threads request access to a

resource, the first one gets access, while the others wait their turn.

- Multiple Wait (MuxWait) semaphores allow a thread to wait on several Mutex or event semaphores.

OS/2 2.0 allows a total of 64,000 public semaphores. In addition to the public semaphores, each process can have a total of 64,000 private semaphores.

Pipes provide communication similar to files, but reside in memory rather than on disk. There are two types of pipes, each with a maximum buffer size of 64 KB:

- Unnamed pipes are used for communication between closely related processes, such as a parent and child.
- Named pipes are used for communication among both related and unrelated processes.

Pipes can be defined so that processes have read, write, or read/write access to the pipe.

Tip: Semaphores and pipes occupy storage and must be managed by the operating system. Private semaphores and unnamed pipes require less overhead, so you should use them whenever possible. And like any other resource, shared memory, queues, semaphores, and pipes should be released when they are no longer needed.

Using Dynamic Link Libraries

During program development it is usually productive to define a large number of Dynamic Link Libraries (DLLs). However, OS/2 must use real storage to keep track of each one, and may need to execute the initialization code in each DLL when the application starts.

Tip: You can help the performance of your application if, before releasing it, you minimize the number of DLLs and package the DLLs by related function, such as initialization, graphics, and error handling.

Private semaphores and unnamed pipes require less overhead, so you should use them whenever possible.

Video, Mouse, Keyboard, and Monitor Calls

OS/2 2.0 does not implement 32-bit replacements for the 16-bit video, mouse, keyboard, and monitor calls. The 16-bit calls are still available.

Tip: You will find that your users will appreciate the consistent appearance and ease of use of the user interface if you use the equivalent Presentation Manager calls.

Whether you are coding a 32-bit application, a 16-bit application that will run under OS/2 2.0 as well as under Version 1.X, or an application that includes both 16- and 32-bit calls, you can significantly improve application performance by careful memory and task management, by designing interprocess communications properly, and by packaging DLLs efficiently. OS/2 2.0 exploits the 32-bit hardware; the programmer can use the tips and techniques presented here to exploit the features of OS/2 2.0.

ABOUT THE AUTHOR

Ron Morrill joined IBM in New York as a systems engineer on the City College team in 1968. He moved in 1969 to a marketing support group where he supported TSO, APL, CP67 and other time-sharing systems. When HONE consolidated operations in Palo Alto in 1976, Ron moved with them as a system support representative. In 1981, he became a systems engineering manager supporting Texas Instruments in Dallas. After several development management positions, Ron resumed a staff position supporting the IBM Contracts and Commissions applications. At present he is a senior marketing support representative supporting OS/2.

Prepared with the assistance of
Bornn Communications
2205 Carmel Drive
Carrollton, Texas 75006
(214) 418-1882

Extending the Functions of OS/2 REXX

Patrick Steil
Dallas, Texas

Can REXX access OS/2 Presentation Manager (PM) calls? How about the rich set of OS/2 APIs? REXX is so flexible and easy to use that both programmers and casual users of OS/2 often want to extend the set of functions available in REXX to include some of these functions. This article shows how easy it is to extend the functionality of REXX.

REXX for OS/2, although a powerful procedural and string manipulation language in itself, lacks many functions available to OS/2 programmers using C, COBOL, or other high-level languages. Using the methods shown here, you'll need to spend only a short time writing code for your REXX program to access any function available in another high-level language.

The methods illustrated in this article can be used for:

- Adding groups and programs to Desktop Manager
- Accessing information in the OS/2 INI files
- Creating quick and customizable file I/O routines
- Displaying PM dialog boxes to obtain user input
- Controlling a PM application using REXX

Basically, REXX can call any function available in another high-level language such as C, COBOL, or Pascal. The interface between REXX and functions written in other languages is easy to use. Your favorite piece of code could be compiled as a Dynamic Link Library (DLL) (see Figure 1). Once the DLL is created, your function can be registered with REXX using a provided REXX call. The function is now callable as though it were built into REXX. Where REXX leaves off in function, you can pick up with your own.

Why Extend REXX?

Perhaps you need to write a software installation procedure using OS/2 REXX that:

- Copies your application's executable and data files from diskette to a user-specified drive and directory
- Modifies the CONFIG.SYS file and, while specifying the application's path inserts any DEVICE= statements or other statements needed for your program
- Adds a new group to the OS/2 Desktop Manager and adds the programs that should go into this group

All the preceding functions can be implemented with REXX. The first two requirements, which can be handled by REXX built-in functions, are easy to code. The last requirement needs some extra coding in another language, because these functions are not normally available using REXX.

What Do I Need to Start?

The bare minimum needed is a language compiler such as IBM C/2™ or Microsoft C Version 6.0, along with OS/2 Version 1.3 or higher. The IBM C/2 compiler and linker were used for the examples in this article. The Microsoft C compiler can also be used without any modification to the code provided. If you want to use any OS/2 Application Programming Interface (API) calls, you'll need the IBM OS/2 Programming Tools and Information package, commonly known as the OS/2 Toolkit.

A Dynamic Link Library (DLL) is a segment of compiled code, usually containing functions that will be used by more than one program. A DLL is not linked to an application's executable file at compile time, but is linked when a program calls it. DLLs can be used to make executable code smaller by placing redundant code in a central module.

When a DLL is used, only one copy of the module is loaded into memory, even if many programs are accessing and using this piece of code. OS/2 itself uses DLLs extensively for repetitious code and data. For example, the resource definitions of many PM controls (buttons, icons, and so on) are contained in a file called DISPLAY.DLL.

Figure 1. What Is a DLL?

How Does it Work?

Figure 2 contains a sample C DLL callable by REXX. This DLL contains two functions. The first one – RxAddGroup – calls an OS/2 API called PrfCreateGroup, which adds a new group to the OS/2 Desktop Manager. This function is normally available only to conventional OS/2 programs. By placing this call in a DLL, REXX can access the same functionality. Any function placed in a DLL can do one or more of the following:

- Receive a string passed from the REXX procedure
- Call any C or OS/2 function
- Call other user-defined functions or DLLs
- Return a string to the REXX procedure

```
*****************************************************************************************
*  RxSample.C                                                                           *
*  This C code will be compiled as a DLL and registered in a REXX procedure. Once registered, the func-  *
*  tion will be used as if it were a built-in REXX function.                            *
*****************************************************************************************

#define INCL_WINPROGRAMLIST          //   Needed for Prf Calls
#define BUFFSIZE 256                 //   Allocate 64 bytes for buffer
#include <memory.h>                  //   Needed for mem functions
#include <rexxsaa.h>                 //   Needed for RX variables

*****************************************************************************************
*  Setup function entrypoint for AddGroup.                                              *
*****************************************************************************************

SHORT APIENTRY AddGroup  (PSZ RxFuncName,
                          USHORT NumArgs,
                          PRXSTRING Args,
                          PSZ Session,
                          PRXSTRING RetString)
{

*****************************************************************************************
* AddProgram          -->     Third parameter in RxFuncAdd (see Figure 3)               *
* RxFuncName          -->     Name of REXX invoking function                            *
* NumArgs             -->     Number of arguments passed from REXX proc                 *
* Args                -->     Pointer to array of RxStrings received                    *
* Session             -->     Name of current data queue                                *
* RetString           -->     String returned to REXX proc                              *
*****************************************************************************************

HINI   hini=HINI_PROFILE;                                  //Store in the OS2.INI file
UCHAR chVisibility=SHE_VISIBLE|SHE_UNPROTECTED;          //Flags for the group

*****************************************************************************************
* Define buffer for later use with received Args.                                       *
*****************************************************************************************

char buffer[BUFFSIZE];
int i;

*****************************************************************************************
* Check number of arguments passed in.                                                  *
*****************************************************************************************

if (NumArgs < 1)                     //if no arguments were passed in
     return 40;                      //then return Incorrect Call to Routine

*****************************************************************************************
* Loop i times copying incoming Args to buffer and then adding a new group using 'buffer'.   *
*****************************************************************************************

for (i=0;i<NumArgs;i++)
     {
       memset(buffer,'\0',BUFFSIZE);
       memcpy(buffer,Args-strptr,(USHORT)Args->strlength);
       printf("Adding Group %s to Desktop Manager...\n",buffer);
       PrfCreateGroup(hini,buffer,chVisibility);
       Args++;
     }

*****************************************************************************************
* Set RetString to return a string or number back to REXX.                              *
* You must give a length for the string!                                                *
*****************************************************************************************

strcpy(RetString->strptr,"Returned from AddGroupFunction");
RetString->strlength=strlen("Returned from AddGroupFunction");
```

Figure 2. Sample C Code to Create a DLL (continued)

```
*****************************************************************************************
* Return to REXX procedure.                                                             *
*****************************************************************************************
return(0);
}
*****************************************************************************************
* Setup function entrypoint for KillDll function.                                       *
*****************************************************************************************

SHORT APIENTRY KillDll(PSZ RxFuncName,USHORT NumArgs,
                       pRXSTRING Args,PSZ Session,PRXSTRING Retstr)
{char *x;*x=10;}
```

Figure 2. Sample C Code to Create a DLL

The second function – KillDll – is used primarily as a debugging aid when writing a DLL. It causes an OS/2 protection violation, or trap D, as it is commonly called. KillDll illustrates one method that can be used to unload the DLL from OS/2's memory. Once the DLL is loaded into memory, it is not unloaded easily. When you attempt to recompile your DLL, the linker tells you that it cannot write to the file because it is still open (by OS/2). This function, by causing a trap D, forces OS/2 to end the program (the DLL), removing it from memory. The file is closed and compiling can continue again.

As shown in Figure 2, ARGS is an array that holds a string passed to the DLL when invoking the function from REXX. NUMARGS is an integer that tells how many arguments were passed. (An argument is any string. Multiple arguments are separated by spaces.) The C structure holding this information is called RXSTRING. This structure consists of a pointer to a string and a length of that string. These are referenced by ARGS->STRPTR and ARGS->STRLENGTH, respectively.

Let's take a look at the main loop of our program. Any time a string is passed from REXX, the memset and memcpy functions must be used to get the string from REXX. This is necessary to provide data-type independence. C declares every variable to be a certain data type. REXX handles all variables as strings. Furthermore, REXX and C differ in how they handle strings in that C places a null terminator at the end of all strings. So when a REXX string is passed to a C function, the string must be formatted in a way that C understands. This is what the memset() and memcpy() functions do. Once this formality is out of the way, these strings can be used.

Our DLL now prints a message to the screen stating that it is adding a new group using the string passed in. By issuing the PrfAddGroup call using ARGS, the group is actually created. After all ARGS are processed, a string is prepared for return to REXX.

Once out of our loop, we would like to return some data. A string is copied into RETSTRING->STRPTR and the length is stored in RETSTRING->STRLENGTH. When this return string is received back in the REXX procedure, it can be handled as a text error message or simply a numeric error/return code, depending on the success of the function.

Registering the Function: The REXX procedure in Figure 3 begins with a RxFuncAdd( ) function call. RxFuncAdd is a built-in REXX function provided to register an externally defined function. This function has three parameters that must be included:

- **RexxAddGroup** is the name of the function as called in REXX. This parameter can be any valid REXX name.
- **RxSample** specifies the actual file name of the DLL where this external function is found. OS/2 assumes that the location of the DLL is included in the LIBPATH statement of the CONFIG.SYS file. If it is not, then the complete path must be specified.
- **AddGroup** specifies the name of the function as it appears in the DLL (Figure 2). The reason for this last parameter is that multiple functions may reside in a single DLL.

Why Register the Function?

What actually happens during the RxFuncAdd call? The function name is added to a list of available REXX

```
*************************************************************************************************
* RxSample.CMD                                                                                  *
* Sample Rexx Program to call a C DLL.                                                          *
*************************************************************************************************

Call RxFuncAdd "RexxAddGroup","RxSample","AddGroup"
say rc
Call RxFuncAdd "RexxKillDll","RxSample","KillDll"
say rc

say "This REXX procedure will call two functions residing"
say "in a DLL."
say ''
*************************************************************************************************
* Call the function as if it were a REXX function.                                              *
*************************************************************************************************

group1="REXX Group"
group2="Another Group"
Say 'Adding Groups...'
rc=RexxAddGroup(group1,group2)
*************************************************************************************************
* RC is the string returned by the C DLL.  This will print "Returned from AddGroup Function."   *
*************************************************************************************************

say rc
*************************************************************************************************
* Cause the DLL to end so that modifying the DLL will be possible. You only need to call this function *
* before recompiling your source 'C' code.                                                      *
*************************************************************************************************

call RexxKillDll
*************************************************************************************************
* Drop the registration of the REXX function. This is only necessary if another REXX procedure  *
* will be using a different function with the same name as the one you are using in this procedure. It is *
* provided here to illustrate the usage of the function.                                        *
*************************************************************************************************

Call RxFuncDrop "REXXADDGROUP"
Call RxFuncDrop "REXXKILLDLL"

return
*****************End of REXX procedure*****************
```

Figure 3. Sample REXX Program to Call a C DLL

functions. The DLL is not actually loaded into memory until the REXX procedure issues that function call. Since REXX does not check for the existence of the DLL until it calls the function, the RxFuncAdd may complete successfully even if the DLL does not exist.

Conversely, calling the complementary function RxFuncDrop drops only the name of the function from the list of available functions. The DLL is not automatically unloaded from memory. If another procedure wanted to access the DLL, the code would not have to be reloaded into memory from disk. If RxFuncDrop is used, the memory where the DLL resides is eventually discarded when OS/2 needs the memory for another application. Using the RexxKillDll function removes the DLL from memory immediately.

This "killing" of the DLL is necessary because the DLL file is held open by OS/2 while it is in use. If the file is in use, you cannot successfully compile your C code. By calling the KillDll function, you will resolve this problem.

Once the function is registered, it can be called as if it had always been available in REXX. A single REXX program can be executed at machine boot-up time that registers all functions you might use in your REXX programs when the system is up. This REXX procedure would not issue a RxFuncDrop for these functions. All functions can then be called without calling RxFuncAdd in each REXX program.

What Makes It a DLL?

The trick in getting the preceding C code compiled as a DLL involves the parameters used in compiling and linking the code. Two files, the MAKE file and DEF file, are created by the programmer. These files tell the compiler and linker what parameters to use to build the DLL. The files also help automate the compilation process. Note that these two files were successfully tested with IBM C/2 Version 1.1 and Microsoft C Version 6.0. Similar files can be used with your favorite compiler.

The MAKE File: The MAKE file is used by the MAKE program to tell the compiler which options to use during the compile. Any and all source files that make up the compile can be specified. In this case, only the DLL source code is involved. Figure 4 shows the MAKE file.

The DEF File: The definition file, or DEF as it is more commonly called, specifies the options to use during the link process. This file tells the linker what functions will be accessed by external programs (our REXX procedure) and how the DLL will be loaded and handled by the operating system. The information in this file is made available to the linker by including its name in the preceding MAKE file. Ensure that the DEF file contains the options shown in Figure 5.

```
CFLAGS=/Alfu G2s /W1

rxsample.dll: rxsample.obj rxsample.de
   link /co rxsample, rxsample.dll /align:16, NUL, \
     /NOD OS2 LLIBCDLL, rxsample.def

rxsample.obj: rxsample.c
```

Figure 4. MAKE file

```
LIBRARY   RXSAMPLE
DESCRIPTION 'Sample DLL to be called from REXX'
PROTMODE

;The following two statements are for reentrant code
;Use these two so more than one REXX procedure can use the DLL
CODE LOADONCALL SHARED
DATA LOADONCALL MULTIPLE NONSHARED

EXPORTS ADDGROUP
        KILLDLL
```

Figure 5. Sample DEF File

Compiling: The compile and link process is started by issuing the following command at an OS/2 command line:

```
MAKE <filename[.MAK]>
```

The <filename> is any valid name (usually the name of the program, in this case RXSAMPLE); the extension is optional. Once the compile has completed and the code is successfully linked, a file with the DLL extension is created. The only thing left to do is call your REXX procedure by typing its name at the OS/2 command line. You can call RexxKillDll from your REXX program when ready to recompile the C DLL.

REXX, a Flexible Language

REXX has many powerful and easy-to-use features. The flexibility of REXX also allows it to use functions defined in other languages. This article has demonstrated the technique for extending the functions of REXX so that, where OS/2 REXX may be lacking, external functions can be easily created to enhance REXX.

ABOUT THE AUTHOR

Patrick Steil works with the IBM Application Assistance Center in Dallas, Texas. He has also served as team leader for the OS/2 Standard Edition technical support team. Patrick is working through the cooperative education program and is currently a senior studying computer science at East Texas State University.

Protecting User Exits Under OS/2 1.X

Derek Williams
IBM Corporation
Charlotte, North Carolina

Several OS/2 programs allow user exit extensions in the form of Dynamic Link Library (DLL) functions. Programs supporting these user exits must prevent the exit from terminating the program or suspending it indefinitely. This article presents some techniques to protect a program from its user exits and includes source code to introduce OS/2's DosPtrace and related functions.

Dynamic Link Libraries (DLLs) have become a tremendously popular feature of OS/2. One of the benefits of DLLs and the functions they contain is the ability to provide tightly coupled code as a completely separate package. This helps remove requirements that an application provide every function for every user, and lets an end user or third-party software developer augment and customize the application. For example, a database application might call a user function to validate data. Or, a communications program might call a user exit to process data received, perhaps to capture this data to disk. This frees the main application from knowing every type of validation rule or every possible use of received data. As imbedded applications grow from idea to reality, DLL user exits will likely become even more popular.

REXX function calls and Presentation Manager system hooks are just two examples of places where DLL user exits are supported. If all goes well, the functions running at these and other user exits perform their required tasks and successfully return to the calling program. Unfortunately, much could go wrong in a user exit routine. Such a routine could accidentally or maliciously:

- Exit and terminate the calling program
- Cause a general protection fault or other trap and end the calling program
- Deadlock or hang the calling program
- Modify data areas in the calling program

The following sections explore why such problems can occur, and how to avoid them.

DLL Functions: Separate Code, Same Resources

Figure 1 illustrates how a DLL entry point can affect the application that calls it. Although DLL routines have separate pieces of code and data, they become logical extensions of the programs (EXE files) that call them. DLL functions run in the same process as the calling program and therefore have access to all the segments, file handles, queues, semaphores, exception handlers, and other resources owned by the calling program. For isolating and protecting code, OS/2 does not distinguish between DLL code and the calling program's code. This means if a DLL function exits or causes a trap, OS/2 terminates the calling program.

In a closed system, a development team might develop and test the main program as well as any DLL functions it calls. This team would be responsible for any problems, whether caused by the main program or by a DLL function. However, in an open system, a group writing a user exit DLL function may be completely separate from the group writing the

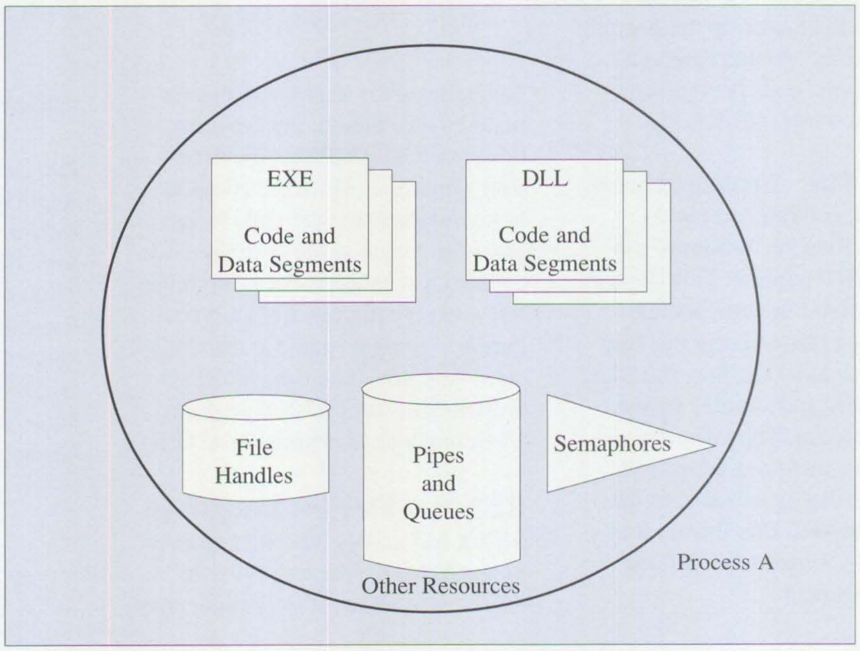


Figure 1. DLL Functions as Extensions of the Calling Program

main program. In this environment, it's important to isolate the main program from its user exits. That way, if a problem occurs, the source of the problem (main program or DLL user exit) can be easily determined.

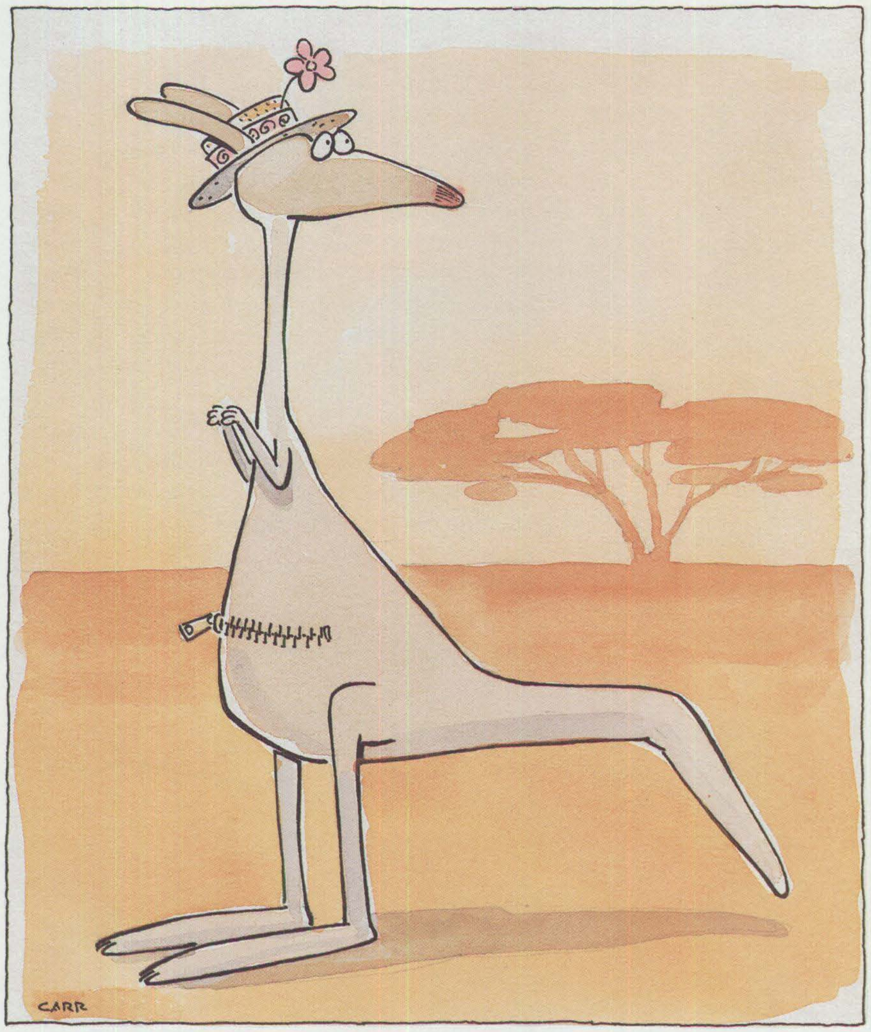
The next section explores ways of isolating the calling program from DLL user exits, along with possible side effects.

Building a Two-Process Wall

Figure 2 illustrates a simple means of preventing side effects caused by DLL functions. By moving the DLL's code and data segments into a second process, we build a wall between the resources owned by the main program and the resources accessed by the DLL. The DLL can access only shared resources granted to it by the main program.

This two-process strategy requires two OS/2 executables (EXE files): the main program and a secondary "shell" program to call the DLL user exits. The second process cooperates with the main program to access shared resources and call the DLL functions when requested by the main program. If a DLL function causes the second process to trap, exit, or hang, the main program remains unaffected.

A variety of two-process architectures for isolating user exits are possible, but we'll focus on one in particular: the technique depicted in Figure 3. Three functions are provided to the main program to allow easy access to user exits. OpenExit and CloseExit are used to begin and end access to a DLL function in the second process. The RunExit function causes the second process to call the DLL function and handle any error conditions the function may have caused.



The OpenExit, CloseExit, and RunExit functions make use of the OS/2 DosPtrace function to control the user exit process. Modeled after the UNIX® PTRACE function, DosPtrace allows one program to act as a debugger for another process, and is used by CodeView® and other OS/2 debuggers. We use DosPtrace because it allows the main process to detect and handle traps without terminating the second process. This is a benefit for frequently-called user exits, because we save the time required to restart the second process if a trap occurs.

There is one restriction imposed by DosPtrace worth noting: a process can have, at most, one debugger (DosPtrace user) at a time. Because the main process is using DosPtrace to control the user exit process, no other debugger can control it. If your main program allows users to use a debugger on their user exits, you might design the program with two modes. A test mode could run the DLL user exits as part of the main process; this would allow debugging but not ensure protection of the main program. A production mode would use the two-process architecture when protection, not debugging, is required.

Figures 5a and 5b contains the code for OpenExit, CloseExit, and RunExit. Figure 6 contains some of the code for the second process user exit "shell." The next section explores this sample code in detail.

Building Blocks for a Two-Process System

From the sample code and from Figure 3, you can see the following ingredients in this protected user exit environment:

- A shared data segment
- A DosPtrace engine
- An asynchronous timer and time-out semaphore
- Three types of threads in the second process

The shared data segment is used for communication between the main process and user exit process. The SHAREDSEGDATA structure shown in Figure 4 defines the shared data elements. Among other things, this area includes the name of the DLL and entry point to call and any parameters to pass to the entry point. In this simple example, we pass only a single word parameter; we could have passed entire lists of parameters.

The DosPtrace engine is simply the DoDosPtrace routine that uses DosPtrace to control the state of the second process. To run user exits in the second process, OpenExit starts the second process shell using DosExecPgm with the EXEC_TRACE option. This allows the main process to control the second process using DosPtrace. Whenever DoDosPtrace needs to run the second process, it sets up necessary parameters in the shared data segment and uses the DosPtrace TRC_C_Go request to start the second process running. DosPtrace retains control until an event such as a trap, exit, or breakpoint occurs in the second process. When DosPtrace returns, OS/2 temporarily freezes all threads in the second process until the next DosPtrace request is made.

The asynchronous timer (DosTimerAsync call in RunExit) and time-out semaphore are used to regain control if a user exit runs too long. We dedicated a thread in the user exit process (the Time-out/Thread routine) to help

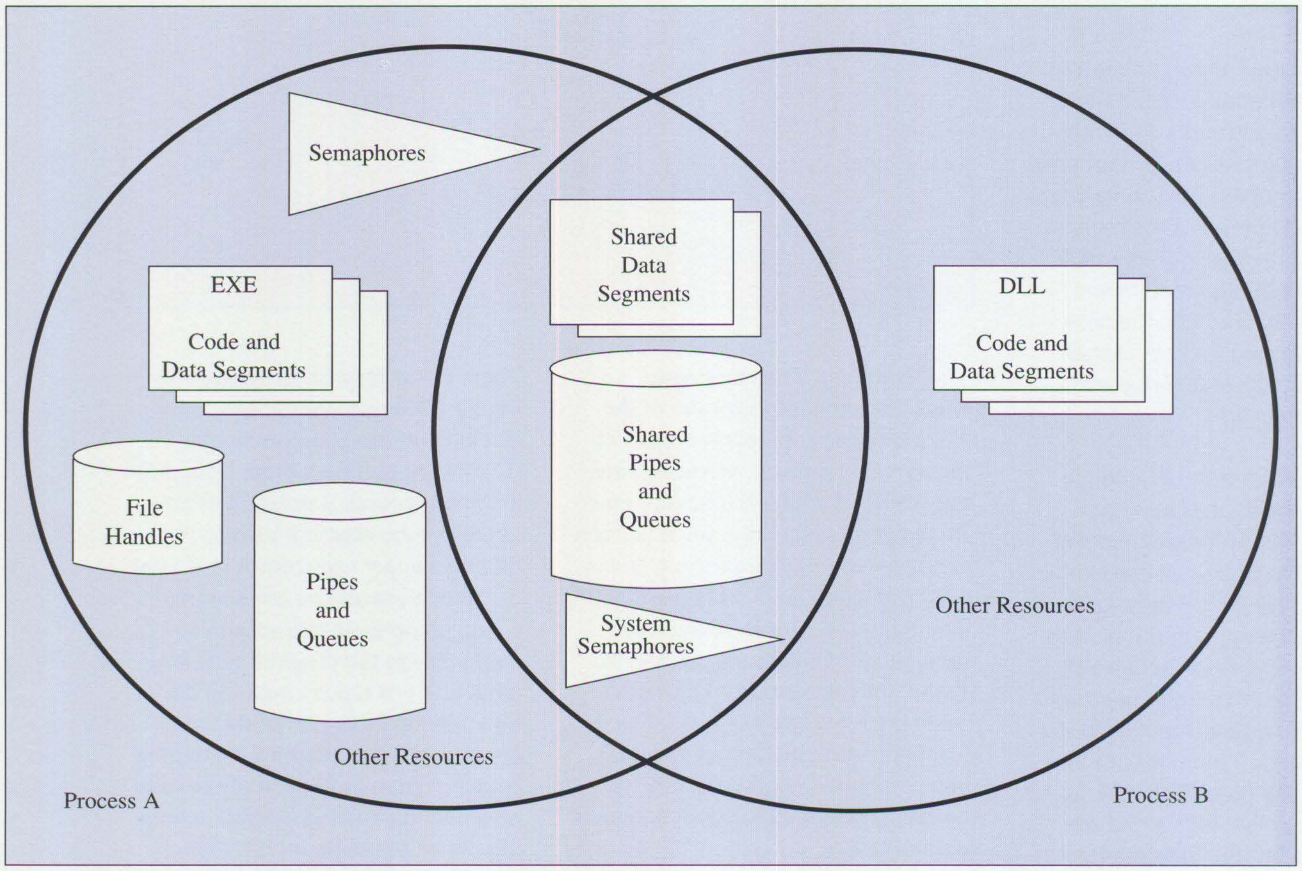


Figure 2. DLL Functions in a Separate Process

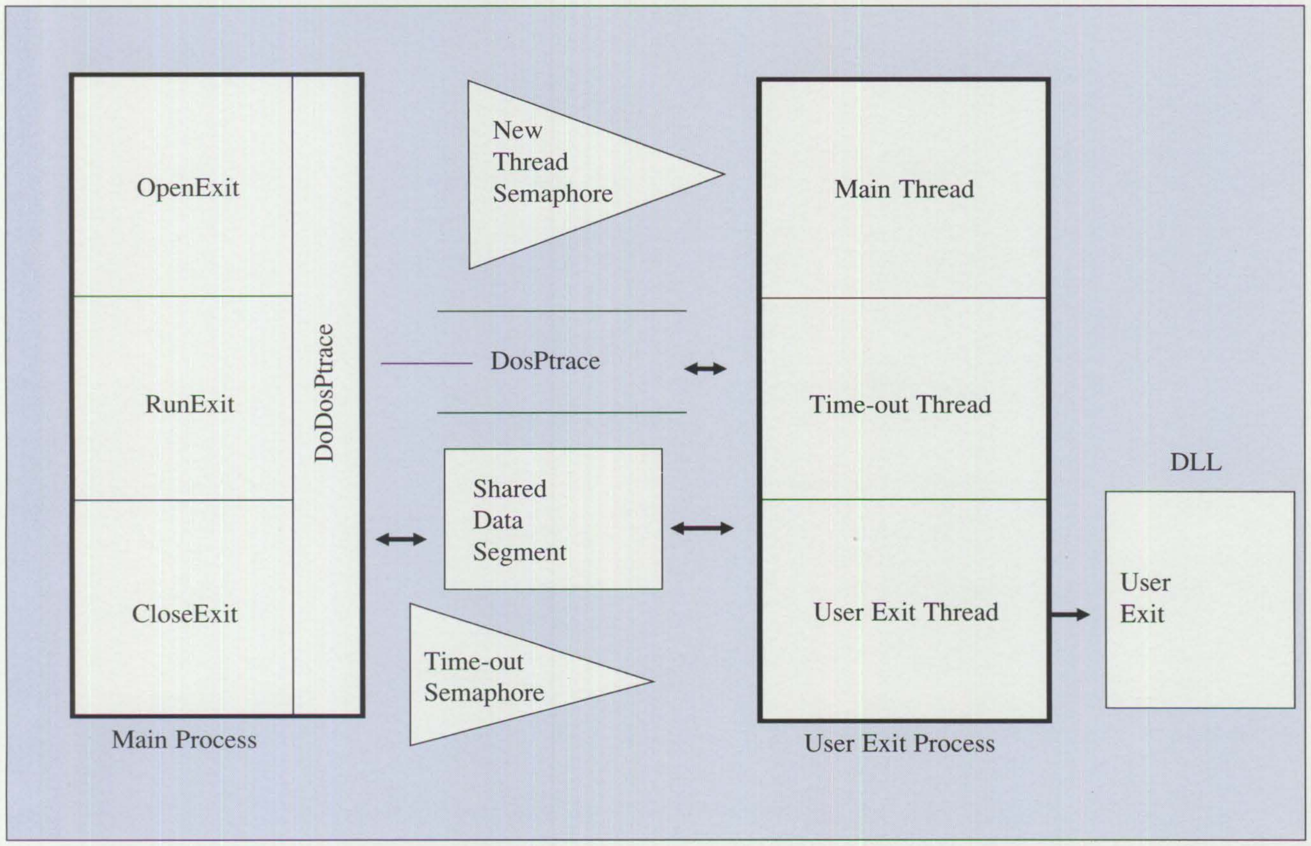


Figure 3. A Two-Process User Exit Architecture Using DosPtrace

```
typedef struct {
     USHORT   usRequest, usReturnCode, usExitReason;
     BOOL     flTimingInProgress;          /* Are we timing?                        */
     USHORT   BreakType;                   /* NONE, READY, or TIMEOUT               */
     USHORT   UserTID;                     /* Thread running user exit              */
     CHAR     chDLLname[32], chEPname[32], chFailNameBuf[32];
     USHORT   Parm;                        /* Parameter passed to user exit         */
} SHAREDSEGDATA, * PSHAREDSEGDATA;
```

Figure 4. Data Shared between the Main and User Exit Processes

us recover from a long-running or deadlocked user exit. If the optional time limit expires before the user exit routine returns, OS/2 clears the time-out semaphore, causing the time-out thread to wake up and issue a breakpoint instruction (interrupt 3). This breakpoint causes DosPtrace to return control to the main process.

The following three types of threads are used to control and run user exits:

- Main thread
- Time-out thread
- User exit thread

The main thread initializes the second process, starts the time-out thread and first user exit thread, and then waits for the main process to request a new user exit thread. If a user exit runs late or traps, the main process permanently freezes the thread and then requests a new user exit thread. This prevents the offending thread from interfering with future user exit calls. At any one time, we have one main thread, one time-

```
PSHAREDSEGDATA pData;

USHORT OpenExit( pchDLLname, pchEPname )    /* Start the user exit process */
PCHAR   pchDLLname, pchEPname;        /* DLL and entry point to initialize */
{
USHORT       usRC, Selector;      CHAR chFailName[32];
RESULTCODES ExecResultCodes;                  /* Return codes from DosExecPgm*/

      /* Create & Initialize shared segment & semaphores used by second process*/
      DosAllocShrSeg( sizeof(SHAREDSEGDATA), chSharedSegName, &Selector );
      pData = MAKEP( Selector, 0 );
      memset( pData, 0, sizeof(SHAREDSEGDATA) );
      strncpy( pData->chDLLname, pchDLLname, sizeof(pData->chDLLname) );
      strncpy( pData->chEPname, pchEPname, sizeof(pData->chEPname) );
      DosCreateSem( CSEM_PUBLIC, &hsemNewThread, chNewThreadSemaName );
      DosCreateSem( CSEM_PUBLIC, &hsemTimeout, chTimeoutSemaName );
      DosSemSet( hsemTimeout );

      /*    Start the second process and prepare it to run user exits:         */
      DosExecPgm( chFailName, sizeof(chFailName), EXEC_TRACE,
                    NULL, NULL, &ExecResultCodes, "exitproc.exe" );
      ExitProcPID = ExecResultCodes.codeTerminate;
      return DoDosPtrace( REQUEST_START, 0 );
   }

   USHORT   RunExit( TimeLimit, Parm )       /* Run the DLL function in the second process*/
   USHORT   TimeLimit, Parm;                 /* # of seconds allowed, Parameter to pass*/
   {
      pData->Parm = Parm;       return DoDosPtrace( REQUEST_RUN, TimeLimit );
   }

   USHORT CloseExit()                  /* Cleanup and terminate the user exit process */
   {
   USHORT   usRC;
      if    (usRC = DosSemClear( hsemNewThread ))          return usRC;
      if    (usRC = DosSemClear( hsemTimeout ))            return usRC;
      if    (usRC = DoDosPtrace( REQUEST_STOP, 0 ))        return usRC;
      if    (usRC = DosFreeSeg( SELECTOROF(pData) ))       return usRC;
      if    (usRC = DosCloseSem( hsemNewThread ))          return usRC;
      return DosCloseSem( hsemTimeout );
   }

PTRACEBUFFER PtraceBuf;                      /* Parameters to and results from DosPtrace    */

   USHORT   DoDosPtrace( Request, TimeLimit )/* Use DosPtrace to process request            */
   USHORT   Request, TimeLimit;              /* Request code, number of seconds allowed     */
   {
   USHORT   usRC, usWarning=0, PtraceCmd;
   BOOL     flDone = NO;                     /* Are we done processing this request?        */
   HTIMER   TimerHandle;                     /* Timer handle used by DosTimerAsync          */

      if ((pData->usRequest = Request) == REQUEST_START)
               PtraceCmd = TRC_C_Stop;       /* Used to initialize the Ptrace Buffer        */
      else     PtraceCmd = TRC_C_Go;         /* Lets the second process run                 */

      while    (!flDone) {
         if    (Request == REQUEST_RUN && TimeLimit) {
               DosTimerAsync((ULONG) (TimeLimit * 1000), hsemTimeout, &TimerHandle);
               pData->flTimingInProgress=YES;/* Tells second process we're timing           */
         }
```

Figure 5a. Source Code for OpenExit, CloseExit, and RunExit

```
      PtraceBuf.Cmd = PtraceCmd;        PtraceBuf.PID = ExitProcPID;
      if    (usRC = DosPtrace( (PBYTE) &PtraceBuf ))     return usRC;
      if    (pData->flTimingInProgress) {
            if (usRC = DosTimerStop( TimerHandle ))      return usRC;
            pData->flTimingInProgress=NO; /* No longer timing                        */
      }
      switch (PtraceBuf.Cmd) {
         case TRC_C_SUC_ret :   if (Request == REQUEST_START &&
                                    PtraceCmd == TRC_C_Stop)
                                   PtraceCmd = TRC_C_Go;
                                else if (Request == REQUEST_STOP)  flDone = YES;
                                else return ERROR_UNEXPECTED_NOTIFICATION;
                                break;
         case TRC_C_BPT_ret :   PtraceBuf.rIP++; /* Point beyond the INT 3           */
                                PtraceBuf.Cmd = TRC_C_WriteReg;
                                DosPtrace( (PBYTE) &PtraceBuf );
                                if (pData->BreakType == BREAK_TYPE_TIMEOUT)
                                   if (usRC = StopUserThread()) return usRC;
                                   else          usWarning = WARNING_TIMEOUT;
                                else if (pData->BreakType == BREAK_TYPE_READY)
                                   flDone = YES;
                                break;
         case TRC_C_THD_ret : if (Request != REQUEST_STOP) {
                                        /* Make sure it's the exit thread...         */
                                        DosSemClear( hsemNewThread );
                                        usWarning = WARNING_THREAD_EXIT;
                             }
                             break;
         case TRC_C_KIL_ret : if (Request != REQUEST_STOP)
                                        return pData->usExitReason ?
                                             pData->usExitReason : ERROR_PROCESS_EXIT;
                             break;
         case TRC_C_GPF_ret : if (usRC = StopUserThread()) return usRC;
                             else             usWarning = WARNING_FAULT;
                             break;
         case TRC_C_SIG_ret :
         case TRC_C_LIB_ret : break;
         default            : return ERROR_UNEXPECTED_NOTIFICATION;
                             break;
      }
   }
   return usWarning;
}
USHORT StopUserThread()            /* Freeze current user exit thread & get a new one  */
{
USHORT usRC;
   PtraceBuf.Cmd = TRC_C_Freeze;        PtraceBuf.TID = pData->UserTID;
   if (usRC = DosPtrace( (PBYTE) &PtraceBuf ))         return usRC;
   usRC = DosSemClear( hsemNewThread );  return usRC;
}
```

Figure 5b. Source Code for OpenExit, CloseExit, and RunExit

out thread, and one active user exit thread. We may have several frozen user exit threads. If too many user exit errors occur, we eventually run out of available threads and must stop and restart the second process.

All threads in the second process depend on the main process to initiate their action. The main thread waits on the main process to clear the "new thread" semaphore before launching a new user exit thread. As described earlier, the time-out thread waits for the time-out semaphore to be cleared by OS/2's DosTimer-Async function before flagging a runaway user exit. Finally, the user exit thread "rests" on a breakpoint instruction (while being temporarily frozen by OS/2) and waits for the main proc-

```
USHORT main()            /* The main thread - initialize & start user exit threads     */
{
USHORT  usRC, Selector;   HSEM hsemNewThread;     HMODULE ModHandle;

   if (usRC = DosGetShrSeg( chSharedSegName, &Selector ))
      return pData->usExitReason = usRC;
   pData = MAKEP( Selector, 0 );
   DosOpenSem( &hsemNewThread, chNewThreadSemaName );
   DosOpenSem( &hsemTimeout, chTimeoutSemaName );
   /*          Try to load the requested DLL and entry point:                          */
   if (usRC = DosLoadModule( pData->chFailNameBuf, sizeof(
                   pData->chFailNameBuf), pData->chDLLname, &ModHandle ))
      return pData->usExitReason = usRC;
   if (usRC = DosGetProcAddr( ModHandle, pData->chEPname, &pUserExit ))
      return pData->usExitReason = usRC;

   if (-1 == _beginthread( TimeoutThread, NULL, 4 * 1024, NULL ))
      return pData->usExitReason = ERROR_CANT_START_TIMEOUT_THREAD;
   /*    Wait for the main process to ask for new threads and start them:              */
   while (pData->usRequest != REQUEST_STOP) {
      if (-1 == (pData->UserTID = _beginthread( UserExitThread, NULL,
                                                   4   * 1024, NULL )))
         return pData->usExitReason = ERROR_CANT_START_EXIT_THREAD;
      if (usRC = DosSemSetWait( hsemNewThread, SEM_INDEFINITE_WAIT ))
         return pData->usExitReason = usRC;
   }
}

VOID TimeoutThread()              /* This thread wakes up if DosTimerAsync expires     */
{
USHORT usRC;
   if (usRC = DosSetPrty( PRTYS_THREAD, PRTYC_TIMECRITICAL, 0, 0 ))
      exit(pData->usExitReason = usRC);
   while (pData->usRequest != REQUEST_STOP) {
      DosSemWait( hsemTimeout, SEM_INDEFINITE_WAIT );
      if (pData->usRequest != REQUEST_STOP)
         DosSemSet( hsemTimeout );
      if (pData->flTimingInProgress) {
         pData->flTimingInProgress = NO;  BreakPoint( BREAK_TYPE_TIMEOUT );
      }
   }
}

VOID UserExitThread()             /* This thread calls the user exit DLL function      */
{
   BreakPoint( BREAK_TYPE_READY ); /* Tell the main proc we're ready                   */
   while (pData->usRequest != REQUEST_STOP) {
     pData->usReturnCode = (*pUserExit)(pData->Parm); /* Call the user exit            */
     BreakPoint( BREAK_TYPE_READY );           /* Tell the main proc we're ready       */
   }
   _endthread();                                      /* Let C runtime cleanup         */
}

VOID BreakPoint( Type )           /* Issue an INT 3 instruction. Requires MSC 6.0.     */
USHORT Type;                      /* Write in assembly if you don't have MSC 6.0       */
{
   pData->BreakType = Type;               _asm { int 3 }
   pData->BreakType = BREAK_TYPE_NONE;
}
```

Figure 6. Source Code for the Second Process Shell

ess to resume its execution for each call to a user exit DLL function.

To use this two-process user exit system, include the OpenExit, RunExit, and CloseExit routines in your main program. Build a second process EXE file using the code shown for the user exit shell (in this example, the EXE file is named "exitproc.exe"). Call OpenExit once to start the second process and call RunExit each time you want to run your DLL function. RunExit returns zero if it ran the DLL function without error. The warning codes returned by RunExit (WARNING_TIMEOUT, WARNING_FAULT, and WARNING_THREAD_EXIT) indicate that the DLL function ran too long, caused a protection fault, or exited. For these return codes, you can continue to call RunExit as needed. Any other return code indicates an error condition that requires stopping and restarting the second process using CloseExit and then OpenExit. Use CloseExit at any time to end the second process and clean up the resources it uses.

The sample code shown is a somewhat simplified version of what your application may require. For one thing, some error checking (such as tests of return codes from DosAllocShrSeg and DosExecPgm) is omitted to keep the examples simple. You can customize the sample code as needed to support different parameters passed to a DLL function and to support several different DLL functions at once.

While there is a definite cost in protecting programs from DLL user exits, the benefit is a robust and more fault-tolerant program.

Summary

If you are writing a program that supports user extensions in the form of DLL functions, consider isolating these user exits from the main program. While there is a definite cost in protecting programs from DLL user exits, the benefit is a robust and more fault-tolerant program. Many types of programs, such as server programs, need such robustness.

We've explored possible hazards when using DLL functions for user exits and methods to overcome these hazards. We've examined a particular solution for OS/2 1.1 or higher. These concepts also apply to OS/2 2.0, because DLL functions run in the context of the calling process. While OS/2 2.0 provides some solutions to help overcome side effects from DLL functions (such as the ability to install exception handlers for general protection faults), complete protection can only come from isolating the user exits in a second process.

ABOUT THE AUTHOR

Derek Williams is a senior associate programmer in the Finance Services Industry development laboratory in Charlotte, North Carolina. Since joining IBM in 1988, he has designed and developed OS/2 applications for bank check processing. Derek received a BS in information and computer science from Georgia Institute of Technology and is currently pursuing an MS in computer science from the University of North Carolina at Charlotte.

GDDM-OS/2 Link

David A. Kerr and John E. Kinchen
IBM Corporation
Hursley, England

This article describes GDDM-OS/2 Link, which enables a workstation running the OS/2 Extended Edition (EE) 3270 Emulator feature to operate as an interactive graphics terminal and access the wealth of host graphics function available from GDDM.

Graphical Data Display Manager (GDDM) is IBM's primary series of mainframe software products that generate and display graphics, images, and text on selected IBM graphics workstations. A workstation running OS/2 EE Version 1.2 or later together with GDDM-OS/2 Link Version 1.0 can be used as an interactive graphics terminal, giving users access to many mainframe GDDM application programs and host functions (Figure 1).

Developed by a small development team at the Hursley UK Development Laboratories, GDDM-OS/2 Link provides the following functions:

- Combines the functions of a personal computer and a host graphics terminal in a single workstation. It runs in a Systems Application Architecture (SAA) environment under OS/2 Presentation Manager (PM) windows.
- Is maintained in the same way that it is delivered to the workstation – through downloading from a host computer.
- Is tightly coupled to the OS/2 EE 3270 Emulator, providing an integrated user environment, and supports host graphics on all configured 3270 host sessions.
- Can support any display device for which a PM device driver with graphics support is available.
- Uses PM printer and plotter support to access a wide variety of hardcopy devices.
- Lets you create Picture Interchange Format (PIF) files and Presentation Manager metafiles on the workstation from host graphics pictures.
- Supports the Presentation Manager clipboard copy function with graphics, and cut, copy, and paste functions with text.
- Supports various screen sizes, including, but not limited to 32x80, 43x80; the number of screen sizes available is dependent on the attached display device.
- Makes use of GPI retained mode graphics to increase the performance of GDDM-OS/2 Link.

This means that any picture redraws are handled locally without asking GDDM to retransmit the data. When this is undesirable (such as with low system memory or complex pictures), retransmission can be deconfigured by deselecting an option on the GDDM-OS/2 Link HOST GRAPHICS OPTIONS dialog.

Getting Started

Prerequisites: GDDM-OS/2 Link Version 1.0 runs on any system unit that OS/2 EE will run on, subject to storage availability. The storage requirements for GDDM-OS/2 Link are 350 KB of hard disk storage and 270 KB of RAM.

It supports all the display attachments supported by the OS/2 Presentation Manager, as well as all printers/plotters with graphics support. It requires GDDM Version 2.3 or GDDM Version 2.2 (with appropriate APAR fixes) and OS/2 EE Version 1.2 or later.

Installing GDDM-OS/2 Link: GDDM-OS/2 Link is a GDDM family product and is distributed on a host tape to mainframe sites. There are no diskettes to install on the workstation.

Figure 1. Complex Graphics Pictures Displayed in Multiple LT Sessions

A small section of GDDM-OS/2 Link is shipped with OS/2 EE. This is simply a stub that enables the installation of the product from the host. Installation is then performed manually by invoking the HGINST command file from either an OS/2 windowed or full-screen session. The command file installs host graphics support by downloading a number of files from the host. After the files have been transferred, the OS/2 EE 3270 Terminal Emulator must be stopped and restarted to enable host graphics support. Unlike GDDM-PCLK (support for GDDM on DOS workstations), which requires modifications to the GDDM defaults file, there is no extra configuration required to get started.

Removing GDDM-OS/2 Link: To safely remove the GDDM-OS/2 Link product from your workstation after installation, invoke the HGREMOVE command file from either an OS/2 windowed or full-screen session.

Features

Interactive Host Graphics: GDDM-OS/2 Link is a GDDM application that can make use of all the facilities of a normal graphics terminal plus some unique features. Users interact with GDDM under OS/2 using both the mouse and the keyboard.

Using the Mouse: A GDDM application can use the mouse in two ways: as a locator selection device and as a simple choice device. When GDDM-OS/2 Link takes over control of the mouse, it changes the pointer style to give the user visual feedback from the usual arrow pointer to one of the following four graphics cursor styles:

- Black and White Target
- Black and White Cross
- Exclusive OR (XOR) Target
- XOR Cross

The user can select between each of these styles with the Alternate Cursor (AltCr) key.

Using the Keyboard: GDDM-OS/2 Link can be used without a mouse because there is a keystroke for each function. The actual keystroke for each action is configurable using the Communications Manager keyboard remap facility. (Keys given in this article are for the U.S. Enhanced Keyboard.)

The graphics cursor key cycles through the available choices for mouse and keyboard action when a host GDDM application enables the graphics cursor. The default key assignment is Alt+F12.

The AltCr key cycles through the available graphics cursor styles when the +Cr symbol is visible in the Operator Information Area (OIA). The default key assignment is Alt+F11.

When the graphics cursor is visible and the +Cr symbol is visible in the OIA, the cursor keys move the graphics cursor by one screen pixel in each direction. Fast cursor keys are available that move the graphics cursor by ten pixels in each direction.

The graphics refresh key is used whenever the user wants to have the graphics data redrawn in the logical terminal window. The default key assignment is Alt+Del. This will work only if the Retained Graphics option is selected on the HOST GRAPHICS OPTIONS dialog.

Printing/Plotting GDDM Graphics: In addition to outputting to any host-attached hardcopy device, GDDM-OS/2 Link can print or plot to any suitable hardcopy device attached either locally to the workstation or across the LAN. It uses the services provided by OS/2 Presentation Manager for hardcopy output. It can use any of the graphics device drivers currently installed and can direct output to any graphics device supported by OS/2 Presentation Manager. This means that GDDM-OS/2 Link can use any new devices as they become available and are supported by OS/2 Presentation Manager.

For a host application to access the OS/2 Presentation Manager hardcopy devices, a nickname must be setup in the GDDM defaults file. One entry must be made for each device to be used. It is the TONAME field that determines which OS/2 hardcopy device to use. This field contains the printer name, as defined to OS/2, truncated to eight characters. However, there is one reserved name, ADMPMOP, that is used to address the current default printer/plotter selected in the Print Manager (Figure 2).

When the user has elected to output to an OS/2-attached hardcopy device, a dialog appears to give a visual indication that the print job is being transferred to the workstation. It also gives the user an opportunity to cancel the job if desired.

By default, the print job is given a system-generated name that indicates

```
ADNMNICK FAM=0, NAME=DEFPRINT, TOFAM=1, TONAME=(*, ADMPMOP)

ADNMNICK FAM=0, NAME=PRINTER1, TOFAM=1, TONAME=(*, PRINTER1)
```

Figure 2. Sample Code in GDDM Defaults File to Define Two Devices to GDDM Applications

the source of the print request (that is, which LT and position in sequence). However, if the user wants to manually name the print job to follow its progress through a print queue, an option on the HOST GRAPHICS OPTIONS dialog can be selected. A dialog will appear requesting a print job name. In most cases this would not be required; however, there may be circumstances when it would be useful to give a meaningful name to a particular print job, such as when it is one of many.

Using the Clipboard: GDDM-OS/2 Link provides limited support for the PM clipboard through the use of the 3270 Terminal Emulator's mark and copy functions. Any area of the host presentation space can be marked and copied into the clipboard in bitmap format. However, the following restrictions apply when copying a bitmap representation of graphics or text to the clipboard:

- The marked area is always an exact multiple of character cells in size.
- Only the visible portion of a marked area is placed into the clipboard as a bitmap.

National Language Support: When GDDM-OS/2 Link is installed, the national-language-specific files to download are determined wherever possible by the language in which the Communications Manager is installed. This ensures that GDDM-OS/2 Link, in most cases, will appear seamless with the 3270 Emulator. Currently GDDM-OS/2 Link supports Canadian French, Danish, Dutch, Finnish, French, German, Italian, Japanese, Norwegian, Portuguese, Spanish, and Swedish.

Automatic Service Update: Service fixes for GDDM-OS/2 Link are shipped on a host tape and are loaded on the host system like other GDDM fixes. They are downloaded automatically to the workstation the next time a GDDM application is started; however, they will have no effect until the OS/2 EE 3270 Emulator is stopped and restarted. This means that all workstations connected to a particular host site always have the same level of product code installed without any requirement for user intervention.

During this process, a dialog visually indicates that the updated modules are being downloaded to the workstation, and allows users to cancel the operation. This can be particularly useful if the user has some urgent task to perform. In this event, GDDM will attempt to download the files again when a GDDM application is started after the next time the OS/2 EE 3270 Emulator is restarted.

Limitations

GDDM-OS/2 Link enables a workstation running OS/2 EE to operate as an interactive graphics terminal. It gives the workstation user access to a wealth of graphics functions available under GDDM. However, it has the following limitations:

- Inability to fully exploit the power of the workstation
- Some deviations from Common User Access (CUA) 2 guidelines
- GDDM-OS/2 Link supports the PM clipboard bitmap format for copying graphics and text to other applications, and does not support the PM metafile format.
- GDDM-OS/2 Link product-related documentation is not consolidated in a single library; it can be found in both the GDDM and OS/2 libraries.

The following sources contain more information on GDDM-OS/2 Link.

IBM OS/2 Extended Edition User's Guide Volume 2: Communications Manager and LAN Requester (70F0160)

IBM OS/2 Extended Edition System Administrator's Guide for Communications (01F0261)

GDDM Installation and System Management Guide (MVS: GC33-0321, VM: GC33-0322, VSE: GC33-0323)

GDDM Guide for Users (SC33-0327)

GDDM Diagnosis and Problem Determination Guide (SC33-0326)

GDDM Messages (SC33-0325)

ABOUT THE AUTHORS

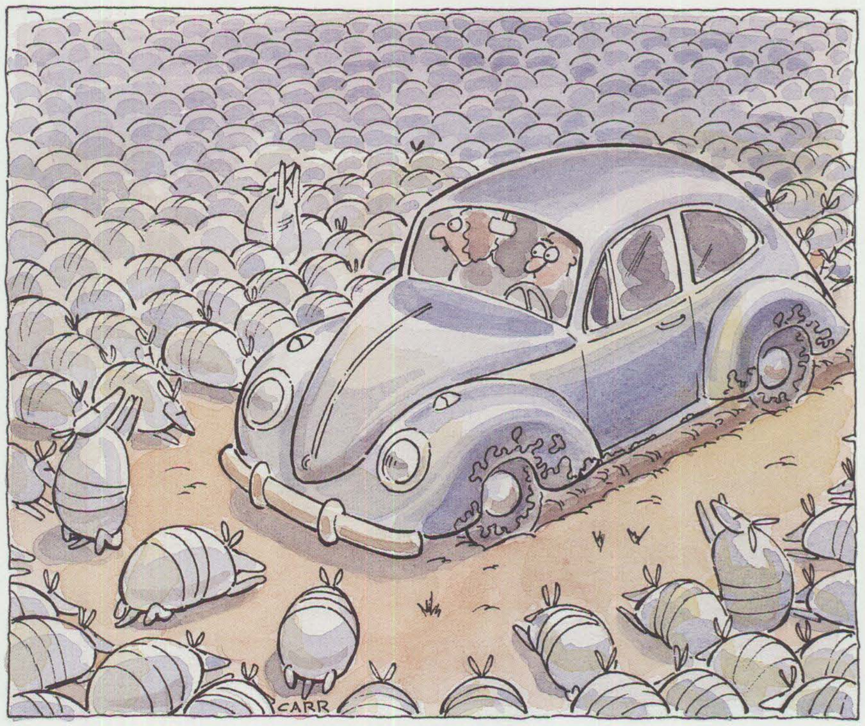
David A. Kerr is a project programmer at IBM's Entry Systems Division in Boca Raton, Florida, and is currently on assignment from IBM's Development Laboratories Ltd., Hursley, England. Presently, he is working on the architecture and design of Presentation Manager for OS/2 2.0. Since joining IBM in 1985 in Hursley, David has worked on GDDM-PCLK product assurance and GDDM-OS/2 Link design and development. He received a BSc with honors in computer science and electronics from the University of Edinburgh, United Kingdom.

John E. Kinchen is a team leader for GDDM Link development at IBM's Hursley UK Development Laboratories Ltd., Hursley, England. He joined IBM in 1984 and has spent much of his time working on OS/2. John developed the File System application for OS/2 1.1 and has a BSc Hon (first-class) in computer science from Portsmouth Polytechnic.

IBM Upgrade Enhanced Install Utility/DOS 5.0

Midge Portney
IBM Corporation
Boca Raton, Florida

The Upgrade Enhanced Install Utility/DOS 5.0 (Enhanced Install) offering allows the DOS 5.0 Upgrade product to be installed in a number of new environments.



This utility requires a licensed copy of the IBM DOS 5.0 Upgrade product (it will not work with the DOS 5.0 base product).

The Enhanced Install replaces the installation program on the DOS 5.0 Upgrade product.

System Requirements for Enhanced Install

The Enhanced Install requires 256 KB of memory. Memory used by Terminate-and-Stay-Resident (TSR) programs, disk caches, multitaskers, and task switchers will have to be considered before the Enhanced Install is started to ensure enough memory. The Enhanced Install requires that your C drive has at least 2.8 MB of free disk space. If you are using the LAN Server Administrator feature, there must be 2.8 MB of free disk space on the server (C drive is not required). You can also use this Enhanced Install if you have OS/2 with dual boot on the workstation.

Features

The following compares the features of the DOS 5.0 Upgrade installation program to Enhanced Install.

- Enhanced Install runs as an application from any drive.

  The existing installation program on the IBM DOS 5.0 Upgrade product works as a bootable program by placing diskette #1 in drive A and restarting your system. The Enhanced Install program works as an application, meaning it can be invoked from the command prompt and can be installed from drives other than A.

- Enhanced Install provides the ability to install the DOS 5.0 Upgrade to many brands of DOS.

  Currently, the install program on the IBM DOS 5.0 Upgrade product will stop if it detects an operating system other than IBM DOS. New function has been added so that the Enhanced Install will continue if it detects any DOS operating system between DOS 3.0 and 5.0.

Network Install

The Enhanced Install will install DOS 5.0 across a network to any workstation running DOS 3.0 or higher. The network administrator should place the DOS 5.0 files in a directory on the server, using either the server or a workstation. Each workstation can then share that directory with the server and run the Enhanced Install.

There are two parts to a successful network installation:

- LAN Server Administrator install
- Workstation/Client install

LAN Server Administrator Install: DOS 5.0 files are placed in a directory on the server (using SETUP/A) to be shared with workstations.

The LAN Server Administrator install places the DOS 5.0 code in a subdirectory on the server machine, but will not install DOS 5.0 as the operating system of the server. The Network Administrator installation can be done at the server or at a workstation that has read/write access to the server.

Insert the Enhanced Install diskette into a diskette drive (A, for example) and type:

A:SETUP /A

Note: If the server is running OS/2 in protect mode only (no DOS Compatibility Box), the Network Administrator install must be run from a DOS workstation that has read/write access to the server.

Workstation/Client Install: DOS 5.0 files are downloaded from the server.

Once the DOS 5.0 files are on the server, workstation users must access the shared server DOS 5.0 subdirectory. To install DOS 5.0 on their workstations, users must point to the shared subdirectory (G, for example) and type:

G:SETUP

Note: In a network environment, it is very important that the network redirector code is upgraded on the workstation before SETUP is run and DOS 5.0 is installed on that workstation. Otherwise, when the workstation is started with DOS 5.0, it may not have the appropriate network support and the network will not start.

Stand-alone Install

To use the Enhanced Install, insert the Enhanced Install diskette into a diskette drive (B, for example) and type:

B:SETUP

The Enhanced Install program, when invoked, looks exactly like the current Install program.

Distribution

The Upgrade Enhanced Install Utility/DOS 5.0 is distributed as one 720 KB or two 360 KB diskettes. There is a README file that explains function and usage. The README is very complete and addresses many issues, including servers that are running in an OS/2 environment, Remote Initial Program Load (RIPL) workstations, and installation on non-IBM hardware.

The Enhanced Install program, when invoked, looks exactly like the current Install program, complete with the help function (type SETUP /?).

How Do I Receive the Upgrade Enhanced Install Utility/DOS 5.0?

The utility is available to DOS 5.0 licensees at no charge through the following channels:

- Your IBM dealer
- The SE at your account
- IBM service at (800) 237-5511
- Your local branch office (through Mechanicsburg as part #G10G-6314-00)

If you have a modem, you can download the utility from the Atlanta Bulletin Board at (404) 835-6600, although this may be a toll call.

ABOUT THE AUTHOR

Midge Portney is the external compatibility planner for entry operating systems and extensions. She has worked in DOS/FORTRAN development and system software compatibility testing for DOS at IBM's Entry Systems Division. Midge has a BA in mathematics from Antioch College, a BA in computer systems from Florida Atlantic University, and an MA in mathematics education from Harvard University.

Advanced Peer-to-Peer Networking: An Overview

Steven T. Joyce and John Q. Walker
IBM Corporation
Research Triangle Park,
North Carolina

For the large base of Advanced Program-to-Program Communications (APPC) application users, a primary problem has been the amount of system configuration required. Using IBM's new Advanced Peer-to-Peer Networking (APPN), computers dynamically exchange almost all the information that had to be configured manually. APPN makes it simple to configure and maintain a Systems Network Architecture (SNA) network.

APPC versus APPN

As their names imply, Advanced Program-to-Program Communications (APPC) deals with *programs*, while Advanced Peer-to-Peer Networking (APPN) deals with *networks*. APPC defines the rules of how programs exchange information. These rules do not deal with the details of network setup and routing. It is APPN that defines how APPC traffic gets from one point to another in a network.
A reasonable comparison between APPC and APPN is the difference between a person using the telephone, and the services a telephone company offers.

APPC: When a person wants to call someone, he or she looks up the telephone number and dials the telephone. Both parties identify themselves, and the exchange of information begins. When the conversation is over, both parties say "good bye" and hang up. This protocol, although informal, is generally accepted and makes it much easier to communicate. APPC provides the same functions and rules, only between application programs instead of people. An application program tells APPC with whom it needs a conversation. APPC starts a conversation between the programs so data can be exchanged. When all the data has been exchanged, APPC provides a way for the programs to end the conversation.

APPN: Networking functions similar to those provided by the telephone companies are provided by APPN. After dialing a telephone number, the network routes the call through trunks, switches, branches, and so on. To make the connection, the network takes into consideration what it knows about available routes and current problems. This happens without the caller understanding the details of the network. A person can talk on the telephone in the same way no matter where the other person is. APPN provides these functions for APPC applications and their data. It computes routes for APPC communication through the network, dynamically calculating the optimal route. Like the telephone company, APPN's routing is done transparently. APPC applications cannot tell whether the communications partner in the APPN network is located in the same computer, one office away, or in another country.

Network Topology

Systems Network Architecture (SNA) has evolved from a heritage of mainframe computers, communications controllers, and terminals. For many years, customers were required to configure networks in a hierarchical design. This hierarchical topology often lacks the flexibility to address varying network geographies, sizes, and workgroup relationships. With the increasing power of workstations and midrange computers, it has become more important to involve those computers in SNA networking.

APPN meets modern flexibility requirements by allowing any network topology that an enterprise wants to create. For example, each networked computer can be directly connected to every other computer (known as a "mesh"), or it can connect through a single routing "hub." Alternatively, some customers will choose to continue to use a hierarchical network design. Mesh, hub, and hierarchical networks, as well as mixtures of these, are all possible using APPN.

Key Concepts

APPC is usually provided as system software. The APPC software provides two interfaces. The first, a programming interface "at the top," responds to requests from application programs that need to communicate. The second interface, "at the bottom," exchanges data with communications hardware.

A connection between the communications hardware on two computers is called a *link*. When an application wants to start communicating with another program, it issues an Allocate call to the APPC programming interface. This call includes the name of the destination, a *Logical Unit (LU) name* in APPC parlance. The LU name is a way to distinguish between different computers in the network. No two computers in an SNA network have the same LU name. This is similar to people in the United States having unique Social Security numbers as a way to identify themselves. One difference is that a computer can have more than one LU name at a time.

When an application issues an Allocate call, APPC sets up a *session* with the named partner LU. A session can be thought of as a pipe used to carry data between a pair of LUs. An LU can have more than one session with a partner LU and can talk to many different partner LUs at once. The new session uses the link already established between the communications hardware in the two computers.

To determine where partner LUs are located in the network, the computers in an APPN network, called *nodes*, exchange different types of requests and responses. We'll refer to these requests and responses as *APPN control information.* At each node in an APPN network, one LU is selected to be the *control point LU*. The control point LU is used by APPN to exchange its control information. Normal APPC applications can also use the control point LU.

Dynamic Configuration

APPN networks include three types of computers: Low-Entry Networking (LEN) nodes, End Nodes (ENs), and Network Nodes (NNs).

LEN nodes, also known as Type 2.1 nodes, have been available since the mid-1980s. The LEN architecture was the first to allow computers in an SNA network to communicate with each other as peers. Examples of IBM platforms currently providing the LEN functions are APPC/PC, VTAM, and the RISC System/6000.

End nodes provide all the functions of LEN nodes but also know how to use the services offered by APPN networks. For example, when end nodes connect to an APPN network, they identify themselves; but LEN nodes do not. Also, when you start an APPC application, the end node works with the APPN network to find the application's partner. This makes setting up a network using end nodes easier than with LEN nodes.

Network nodes provide all the functions of end nodes and add two important services. First, network nodes work together to route information from one node to another. Network nodes providing this intermediate routing form the backbone of a network. The second service is to help LEN and end nodes locate partner LUs in the network. By finding the LUs dynamically, little system definition is required at each node in the network.

Example 1

The easiest way to learn how APPN works is with several examples of how to build and use a network. Figure 1 shows a simple APPN network.

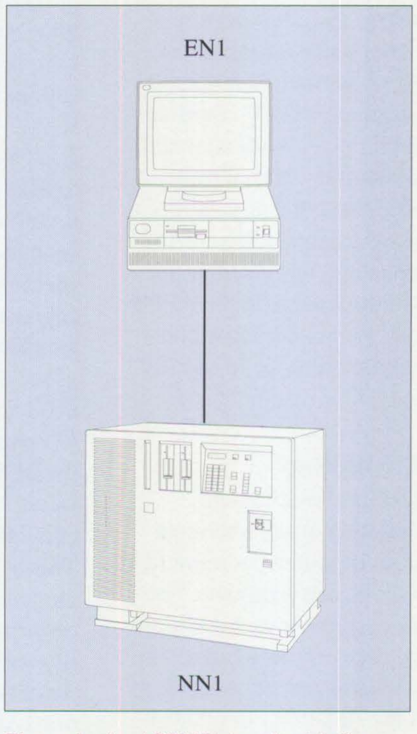


Figure 1. An APPN Network with One End Node and One Network Node

The lines that connect the computers are communications links.

When the link is activated between end node 1 (EN1) and network node 1 (NN1), several things happen automatically:

1. The computers tell each other they can support APPN and tell which type of node they are – end node or network node.

2. NN1 asks EN1 if it needs a *network node server*. When an end node needs to find an LU in the network, it sends its request to its network node server. Because EN1 does not have one yet, it answers "yes." Although an end node can have connections to more than one network node, it can have only one network node server at a time.

3. Because NN1 will be serving EN1, it establishes a pair of control point sessions. These are APPC sessions that will be used to exchange APPN control information. Two sessions are required because the control point uses each session as a one-way pipe, combined to emulate full-duplex.

4. EN1 registers any other APPC LUs defined at its node. It does this by sending a formatted record to NN1 on the control point sessions.

Once these steps are complete, NN1 now knows how to get to EN1 and also knows what LUs are located there. This same set of exchanges occurs every time a network node and an end node are joined by a communications link and agree to set up the EN-NN server relation. The accumulation of this information by network nodes is crucial for locating LUs and calculating routes through the network.

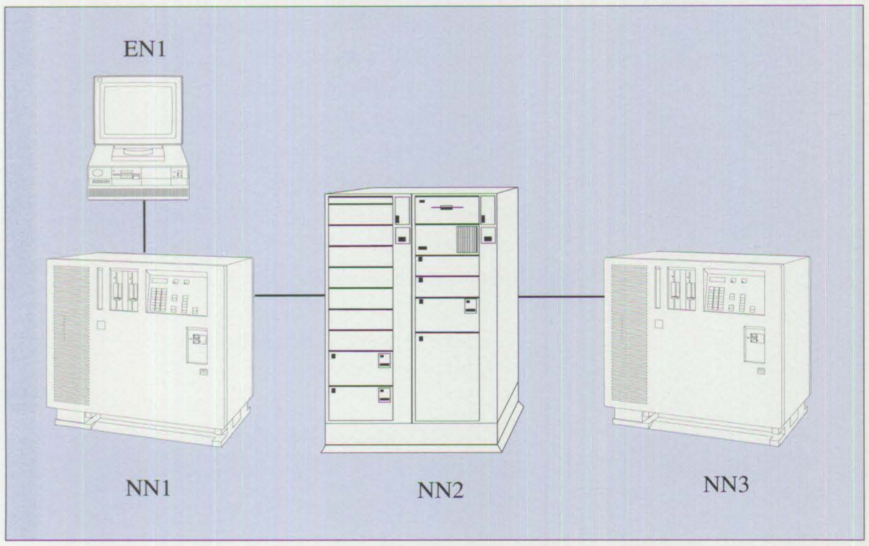


Figure 2. An APPN Network with Three Network Nodes and an End Node

Example 2

Different types of information are exchanged between a pair of network nodes, as shown in Figure 2.

Consider what happens when NN1 activates a link to NN2:

1. The computers tell each other they can support APPN and are both network nodes.

2. Network nodes bring up control point sessions between them to exchange APPN control information.

3. Each network node in an APPN network keeps track of all the other network nodes and the links that connect them. This information is called the *network node topology*. Once NN1 and NN2 have control point sessions, they begin exchanging what they each know about the current network node topology. In this example, NN1 knows only about itself and its link to NN2. NN2 knows about NN3 and the link that joins them.

4. Once they have exchanged topology information, both can construct a complete view of the network node topology. However, NN3 must also have a complete network node topology. When a network node learns new topology information, it spreads the word to any other network nodes to which it has links. NN1 does not have any other network node links, so its job is complete. NN2 passes the new information it learned from NN1 onto NN3, but does not have any other links to network nodes, so its job is also complete.

To summarize, once all the links are activated in the APPN network, each end node knows about itself and its network node server. Each network node knows about itself, all the end nodes it serves, and the full network node topology.

Before we leave this example, note that network nodes can also bridge different types of links. For example, the connection between end nodes and network nodes is frequently on a Local Area Network (LAN). Links between network nodes are often Wide Area Network (WAN) connections such as X.25 or SDLC. Network nodes provide this bridging for APPC efficiently and transparently.

Locating Resources

LEN nodes must have definitions for each partner LU with which they will exchange data. With today's growing and changing networks, keeping those definitions up-to-date can require a significant effort. In an APPN network, end nodes avoid requiring partner definitions by asking a network node to find the partner and the optimal route to get there. As described previously, each end node tells its network node what LUs it has defined. Therefore, by combining the information known by all the network nodes in a network, the location of any LU can be determined. Let's look at some examples.

Example 3

As shown in Figure 3, we'll look at how LUs are found when both end

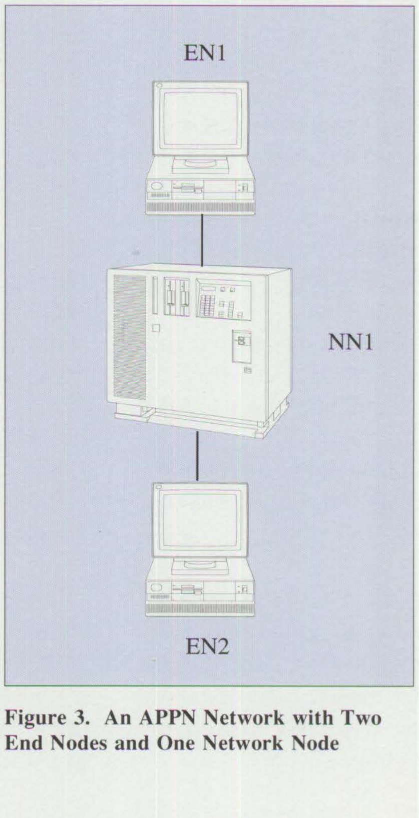


Figure 3. An APPN Network with Two End Nodes and One Network Node

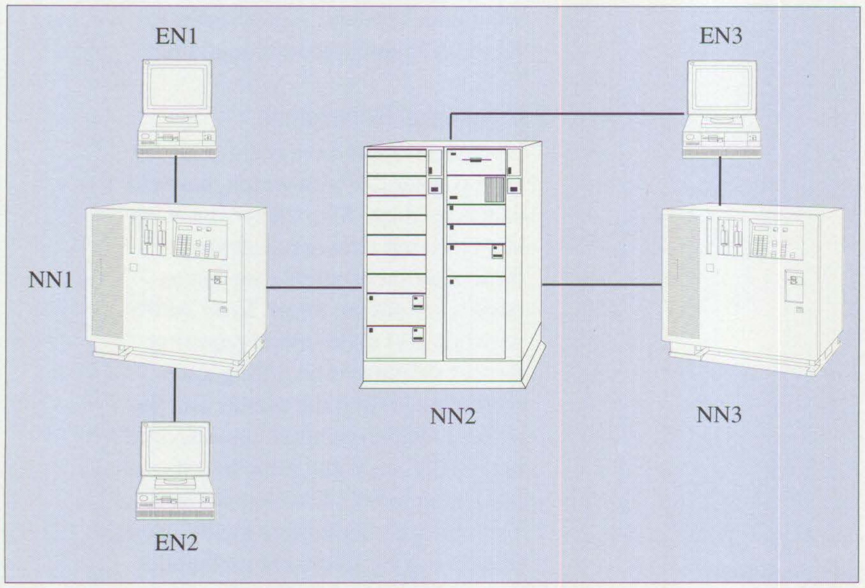


Figure 4. An APPN Network with Three Network Nodes and Three End Nodes

nodes have the same network node server.

The following occurs when an APPC application on EN1 wants to start a conversation with an application on EN2:

1. EN1 asks NN1 to find EN2 and determine what path through the network should be used.

2. NN1 knows that it is the network node server for EN2 and that EN2 has registered its LUs.

3. NN1 determines that the only path available is "EN1 to NN1 to EN2." It passes this information back to EN1.

4. EN1 can now establish an APPC session to EN2 and start exchanging information.

Example 4

Figure 4 shows an APPN network with additional complexity. Notice that EN3 has links to two network nodes. Assume that NN3 is the network node server for EN3.

The following occurs when an APPC application on EN1 wants to start a conversation with an application on EN3:

1. EN1 asks NN1 to find EN3 and determine what path through the network should be used.

2. NN1 is not EN3's network node server, so it needs to get help from the other network nodes.

3. NN1 sends a request to all of its adjacent network nodes looking for EN3. This is known as an APPN broadcast. NN2 is the only network node adjacent to NN1.

4. NN2 passes the same request to all of its adjacent network nodes. NN3 is the only network node adjacent to NN2.

5. Although EN3 has a link to both NN2 and NN3, NN3 is acting as its network node server. This means that even though NN2 knows where EN3 is, it will not reply on its behalf.

6. NN3 asks EN3 what communications links it currently has. This information is important, because NN1 must be able to determine all the possible routes through the network to get from EN1 to EN3. The ability to receive and respond to this request is another function the end nodes perform that LEN nodes cannot.

7. EN3 replies that it has links to both NN2 and NN3.

8. NN3 tells NN2 "Yes, I have EN3" and passes along EN3's link information.

9. NN2 passes the information back to NN1.

10. NN1 now has to compute which of the two routes to get to EN3 is best. The methods for computing routes are described in the next section. NN1 passes the route it selected back to EN1.

11. EN1 can now establish an APPC session to EN3 and start exchanging information.

APPN networks get much larger than three network nodes. The broadcast NN1 did to find EN3 would go to every network node in the network. Doing a broadcast every time a session needs to be started could use a considerable amount of network resources. To minimize the number of broadcasts, NN1 remembers where it found EN3. If another of NN1's end nodes asked to talk to EN3, NN1 would go straight to NN3 to verify that EN3 is still there and get any new link information from EN3. This *caching* of information is useful because the end nodes attached to a single network node will normally be in some form of workgroup. People in a workgroup frequently run the same kinds of applications to the same destinations. Where this is true, network node broadcasts will seldom be needed.

Route Calculations

The previous examples have used simple networks. For each session request there were few possible routes through the network. APPN can handle complex networks and provide intelligent, *class-of-service* routing. Class-of-service routing means that different types of data will each be routed using paths optimized for that specific type. All APPN nodes have several predefined classes-of-service including "batch," "interactive," "batch-secure," and "interactive-secure." Batch is used for sending large volumes of data, such as file transfers. Interactive is normally used for programs that need quick responses. The secure class-of-service definitions give users the option of protecting special data when it is sent through the network.

Which class-of-service to use is determined by the mode name, which is passed on the APPC Allocate call. Given a class-of-service, APPN determines the importance of eight values defined for every link in the network. These include propagation delay, cost per byte, cost for connect time, effective capacity, and security. Each class-of-service assigns a different numerical importance to these values. For instance, when an application asks APPN for a batch session, APPN will try to find a path through the network with high capacity and low cost. For an interactive session, APPN will try to minimize the propagation delay. This means avoiding links such as satellite links because of the time it takes to send a signal from the earth to a satellite and back. This delay could be enough to prevent acceptable performance. For secure sessions, APPN will only pick paths built from secure links. If no secure path is available, the session will not be started.

Also included in the decision is information about each intermediate network node along the route. Each network node maintains a value called its *route addition resistance*. By altering the default value, network administrators can select which nodes they prefer traffic to go through. Network nodes can also detect that they are under a heavy load and go into a congested state. This means that as long as they are congested, they will not be selected for new sessions through the network.

Given a class-of-service, APPN determines the importance of eight values defined for every link in the network.

A network node chooses the lowest class-of-service cost. It does this by first calculating the cost of each of the different possible routes to the destination. This is not a dollar cost, but rather the result of a numerical calculation adding the weight of each node and link along the route being examined. If the lowest cost is offered by more than one route, one of them will be chosen randomly. This distributes the load across equal-cost routes.

Setting Up a Node

Because of the advanced features APPN provides, configuring a node can be as simple as the following.

Each end node provides:

- Its LU name
- Its network node's address (which allows the end node to bring up a link and automatically get a network node server)

Each network node provides:

- Its LU name
- Addresses to connect to any other network nodes

With this minimal amount of definition, nodes throughout the network can communicate with each other. All the necessary information is either dynamically exchanged or is determined from defaults. When considering these savings at each node in the network, APPN can have a tremendous impact.

APPN Products

Today, APPN capabilities are available on the following IBM products:

- OS/400®
- Networking Services/2 (for OS/2 Extended Edition)
- 3174 Establishment Controller
- System/36
- DPPX/370

Each of these products can be configured as an end node or a network node, except the 3174 (since it does not run end-user applications). Additionally, when APPN was announced as part of SNA, four companies announced their intent to create end node products: Apple® Computer, Novell, Siemens-Nixdorf, and Systems Strategies Inc. The end node architecture is available in *Systems Network Architecture: Type 2.1 Node Reference* (SC30-3422-2).

Summary

Advanced Peer-to-Peer Networking is a major step in SNA's evolution to support distributed applications in customer networks. It allows large, complex networks to be built using many types of cooperating computers. APPN also offers dramatic reductions in the amount of network definition needed to configure each node in a network.

For more information on APPN, order *APPN Architecture and Product Implementation* (GG24-3669).

ABOUT THE AUTHORS

Steven T. Joyce is an advisory programmer in the APPC Market Enablement department in the architecture and telecommunications organization at IBM in Research Triangle Park, North Carolina. He managed the planning and testing of the IBM SAA Networking Services/2 product. In 1983, he joined IBM in Raleigh, North Carolina, and was involved with the quality assurance and testing of IBM office systems. Steve received a BS in computer science from North Carolina State University.

John Q. Walker II is the manager of the APPC Market Enablement department in the architecture and telecommunications organization at IBM in Research Triangle Park, North Carolina. John recently managed one of the development teams responsible for the implementation of IBM SAA Networking Services/2. In 1978, he joined IBM in Rochester, Minnesota, where he helped develop and test System/38 operating system software. He was an architect for the IBM Token-Ring Network and served as co-editor of IEEE 802.5 local area network standards. John received a BS, BA, and MS from Southern Illinois University and a PhD in computer science at the University of North Carolina at Chapel Hill.

Using IBM SAA Networking Services/2

Kimberly D. Murray
IBM Corporation
Research Triangle Park,
North Carolina

In March 1991, IBM announced a new product for the OS/2 Extended Edition environment: IBM SAA Networking Services/2. This product provides significantly improved performance, configuration, usability, tools, and sample programs for Advanced Program-to-Program Communications (APPC). It also adds support for Advanced Peer-to-Peer Networking (APPN) and the Common Programming Interface for Communications (CPI-C). This article shows how an application developer can easily use these new features to create and use peer-to-peer communications in the ever-growing cooperative processing environment.

New Configuration File Structure

Networking Services/2 is part of the new OS/2 Extended Services product for use in OS/2 Version 2.0 base environments. Before the installation of Networking Services/2, all Communications Manager configuration information is contained in a .CFG file in the subdirectory CMLIB. Once Networking Services/2 is installed, an APPN subdirectory is created within CMLIB that contains all Networking Services/2 files. Among the files in the APPN subdirectory are three new configuration files.

Although it has its own set of configuration files, Networking Services/2 does not completely remove the need for the original Communications Manager .CFG file. Instead, these four files have the following interdependencies on one another:

- *filename*.CFG: Communications Manager binary configuration file

  Contains DLC, SNA Gateway, X.25, 3270 LU, and LUA information used by Networking Services/2. This file can be modified only through the Communications Manager panels.

- *filename*.NDF: Networking Services/2 node definitions file

  Contains an ASCII representation of the verbs used to configure Networking Services/2 at startup time. This file does not have to exist to start Networking Services/2. It can be modified with a text editor or through the Networking Services/2 configuration panels.

- *filename*.CF2: Networking Services/2 binary configuration file

  Contains a binary representation of the verbs used to configure Networking Services/2 at startup time. This file must exist and cannot be empty to start Networking Services/2. It can be modified through the Networking Services/2 configuration panels.

- *filename*.SEC: Networking Services/2 security file

  This file can only be modified through the Networking Services/2 configuration panels.

The preceding files should have the same *filename*, as specified by the user. They are distinguished by their file extension.

The following commands can be issued from the OS/2 command line and are useful in creating and maintaining various configuration files:

- APPNMIG *filename*.CFG

  Migrates an existing .CFG to a .CF2 file that can be used by Networking Services/2 (At installation time, there is the option to migrate your default configuration file.)

- APPNV *filename*.NDF

  Verifies the information in the .NDF file and creates a corresponding .CF2 file

- APPNRST *filename*.CF2

  Creates an .NDF file from an existing .CF2 file

Configuring Your Workstation as a Simple End Node

Once Networking Services/2 has been installed, the workstation should be configured as either an end node or a network node. This section describes the steps necessary to configure a workstation as a simple APPN end node. Although an individual configuration can be as complex as required, very little configuration is needed to immediately gain access to all resources in an existing APPN network or attached network. In fact, this configuration can be as simple as answering three questions on one screen. The use of defaults by Networking Services/2 enables this straightforward initial configuration.

To invoke this quick and easy configuration utility, called Quick Configuration, type the following at the OS/2 command line:

APPNC *filename* /q

where *filename* is the name of the configuration file that you want to update.

This command causes the Quick Configuration screen to be displayed. On this screen, you will notice that three parameters must be supplied:

- Network ID: Identifies the SNA network name of the network to which you belong
- Local Node Name: Identifies the local name of your workstation

  Also known as the Control Point (CP) name, this node name must be unique within the network.
- Network Node Server Address: Identifies the adapter address of the machine that will be your network node server

  The type of information provided here depends on the type of adapter selected.

For this example, assume that the workstation name is ENDNODE in a network named APPNNET. Also assume that it will be attached via Token-Ring to a network node with the address of 60000C730000. The corresponding Quick Configuration panel should be completed as shown in Figure 1.

After supplying these three parameters and successfully verifying the new configuration file by selecting the Verify pushbutton, use Cancel to leave this panel. This simple procedure updates the Node Definitions File and creates a corresponding binary (.CF2) configuration file. The .NDF should now include the information shown in Figure 2.

Notice that only those parameters shown in italics in Figure 2 were specified manually. All others are standard defaults. Your .NDF file may contain additional information if you chose to migrate any existing configuration files during the installation of Networking Services/2.

Modifying Your Configuration

When configuration changes are required, a user now has three options for making these changes:

- Networking Services/2 Configuration Panels

  These panels provide a user-friendly way to make configuration changes. All panels are Presentation Manager-based and provide online help for all parameters supported by the panels.
- ASCII Node Definitions File

  The .NDF file is an ASCII representation of the binary configuration file. Changes can be made to this file with any text editor.
- Configuration Application Programming Interface (API)

  A configuration API is available for certain configuration changes. This API allows users to make temporary changes from an application program.

The following sections show the definition of a sample Transaction Program (TP), using the three previously mentioned methods. Each method has its advantages. A method should be selected based upon the individual situation. For example, the panels take longer, but they are easier for new users.

Networking Services/2 Configuration Panels: Use the following directions to define a TP from the configuration panels:

1. Select IBM SAA Networking Services/2 from the Desktop Manager.

2. Select Configuration Management.

3. Select Configuration File.

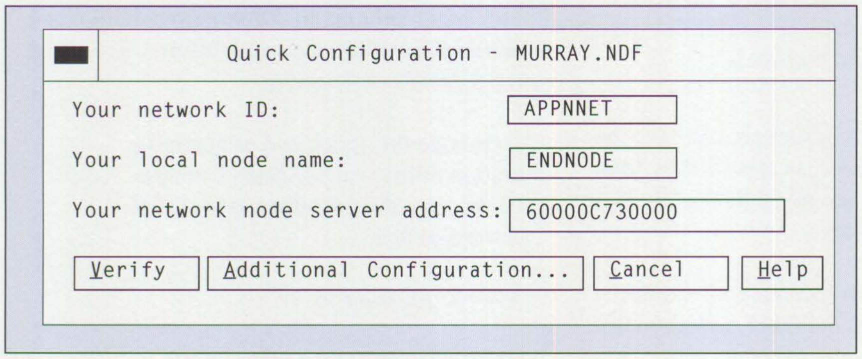
4. Select Configure Advanced SNA.

5. Select Transaction Program Definitions.

6. On the Creating a Transaction Program Definition panel, enter the following information and then select Continue:

- Transaction Program name: FUNAPP
- OS/2 program path and file name: C:\CMLIB\APPN \APPLS\FUNAPP.C

7. On the Creating Additional Transaction Program Parameters panel, specify the following information, and then select OK.


Quick Configuration - MURRAY.NDF

Your network ID: APPNNET

Your local node name: ENDNODE

Your network node server address: 60000C730000

Figure 1. Quick Configuration Screen

```
DEFINE_LOCAL_CP        FQ_CP_NAME(APPNNET.ENDNODE   )
                       CP_ALIAS(ENDNODE   )
                       NAU_ADDRESS(INDEPENDENT_LU)
                       NODE_TYPE(EN)
                       NODE_ID(X'00000')
                       HOST_FP_SUPPORT(NO);


DEFINE_CONNECTION_NETWORK FQ_CN_NAME(APPNNET.RISLAN   )
                          ADAPTER_INFO( DLC_NAME(IBMTRNET)
                                        ADAPTER_NUMBER(0));


DEFINE_LOGICAL_LINK    LINK_NAME(LINK0001)
                       ADJACENT_NODE_TYPE(NN)
                       PREFERRED_NN_SERVER(YES)
                       DLC_NAME(IBMTRNET)
                       ADAPTER_NUMBER(0)
                       DESTINATION_ADDRESS(X'60000C730000')
                       CP_CP_SESSION_SUPPORT(YES)
                       ACTIVATE_AT_STARTUP(YES)
                       LIMITED_RESOURCE(USE_ADAPTER_DEFINITION)
                       LINK_STATION_ROLE(USE_ADAPTER_DEFINITION)
                       SOLICIT_SSCP_SESSION(NO)
                       EFFECTIVE_CAPACITY(USE_ADAPTER_DEFINITION)
                       COST_PER_CONNECT_TIME(USE_ADAPTER_DEFINITION)
                       COST_PER_BYTE(USE_ADAPTER_DEFINITION)
                       SECURITY(USE_ADAPTER_DEFINITION)
                       PROPAGATION_DELAY(USE_ADAPTER_DEFINITION)
                       USER_DEFINED_1(USE_ADAPTER_DEFINITION)
                       USER_DEFINED_2(USE_ADAPTER_DEFINITION)
                       USER_DEFINED_3(USE_ADAPTER_DEFINITION);


DEFINE_DEFAULTS        IMPLICIT_INBOUND_PLU_SUPPORT(YES)
                       DEFAULT_MODE_NAME(BLANK)
                       MAX_MC_LL_SEND_SIZE(32767)
                       DIRECTORY_FOR_INBOUND_ATTACHES(*)
                       DEFAULT_TP_OPERATION(NONQUEUED_AM_STARTED)
                       DEFAULT_TP_PROGRAM_TYPE(BACKGROUND)
                       DEFAULT_TP_CONV_SECURITY_RQD(NO)
                       MAX_HELD_ALERTS(10);


START_ATTACH_MANAGER;
```

Figure 2. Quick Configuration Node Definitions File

- Presentation Type: Background
- Operation Type: Non-queued, Attach-Manager started

8. Close the Advanced SNA panel.

9. Select Verify from the File pull-down menu to verify the new configuration.

Both the binary and ASCII representations of the configuration file should now be updated.

Node Definitions File: The TP can be defined in the .NDF file. With any text editor, edit *filename*.NDF, where *filename* is the name of your configuration file. Enter the following information and then save the file.

```
DEFINE_TP TP_NAME(funapp)
  FILESPEC(c:\cmlib\appn\appls
                    \funapp.c)
  CONVERSATION_TYPE(either)
  CONV_SECURITY_RQD(no)
  SYNC_LEVEL(either)
  TP_OPERATION(nonqueued_am_started)
  PROGRAM_TYPE(background)
  RECEIVE_ALLOCATE_TIMEOUT(infinite);
```

Once this information has been added to the .NDF file, to verify the change and make it effective immediately, the following command must be specified at the OS/2 command line:

APPNV *filename*.NDF /e

Unless the new .NDF file is verified, the change will not be made to the binary .CF2 file, which is used by Networking Services/2 at start-up time.

Configuration API: Use the DEFINE_TP verb to define the sample TP from an application. This verb allows users to add a new definition or replace an existing one. If the TP is already defined, and it is currently active or being started, the verb is rejected. The implementation of the DEFINE_TP verb is described thoroughly in the *Networking Services/2 System Management Programming Reference* (SC52-1111).

Establishing a Connection Network

In Figure 2, one of the default definitions that is added to the Node Definitions File during Quick Configuration is the DEFINE_CONNECTION_NETWORK verb. A connection network allows the direct communication between workstations in a local area network without partner definitions and without the intermediate routing of a network node.

To customize a connection network, it is necessary to specify the fully-qualified connection network name and the adapter information. The

```
DEFINE_CONNECTION_NETWORK FQ_CN_NAME(CNNET.C73)
                   ADAPTER_INFO(DLC_NAME(IBMTRNET)
                            ADAPTER_NUMBER(0));
```

Figure 3. Updated Connection Network Definition

updated connection network definition could appear as in Figure 3.

This verb can only be specified in the Node Definitions File; it cannot be added from the configuration panels or the configuration API. All workstations wishing to communicate directly via the connection network must have the same fully-qualified connection network name specified in their .NDF file.

Improving Performance with Networking Services/2

APPC performance is significantly improved with Networking Services/2. In LAN environments, APPC applications run at least twice as fast as with the previous OS/2 Extended Edition (EE) APPC support. When compared with NetBIOS, APPC performance is not only very competitive, but also faster in cases of large data transfers. And unlike NetBIOS and certain other protocols, APPC can be used between various machine types (both IBM and non-IBM) and across various communications media. With this performance improvement, APPC is now a very viable communications choice in both wide and local area networks.

Figure 4 shows a performance comparison among Networking Services/2, OS/2 Extended Edition, and NetBIOS.

In Figure 4, the following assumptions apply:

- All numbers are for two PS/2 Model 80s across a 16 Mb Token Ring.
- Most NetBIOS application writers will not see this level of performance because numbers represent a device-driver interface.
- Read a Record: Send 100 bytes followed by receive 100 bytes.
- 100 KB File Transfer: Send 100 KB with 16 KB verbs, ending with DEALLOCATE_CONFIRM.
- For the 100 KB file transfer, the frame size is 4 KB.
- Tests with OS/2 EE 1.3 showed no significant difference.

Using PMDSPLAY to Display the Workstation Characteristics

Networking Services/2 contains a program called PMDSPLAY that allows users to display the characteristics of local, as well as remote, workstations. This Presentation Manager program uses the DISPLAY verb and the new DISPLAY_APPN verb.

Through PMDSPLAY, we can view the changes that have made to the configuration file in the last few sections. By selecting Display, APPC, and Global SNA from the pull-down menus, the new node definitions created by Quick Configuration can be shown (Figure 5).

By selecting Display, APPC, and Transaction Program Definitions, we see the definition of our sample TP, FUNAPP (Figure 6). Finally, the connection network definition is shown (Figure 7) by selecting Display, APPN, and Connection Network.

In addition to viewing the local machine characteristics, users can view characteristics of remote machines. By selecting Options and Select Target, users can then select the target machine to display.

Tracing Your APPC Program

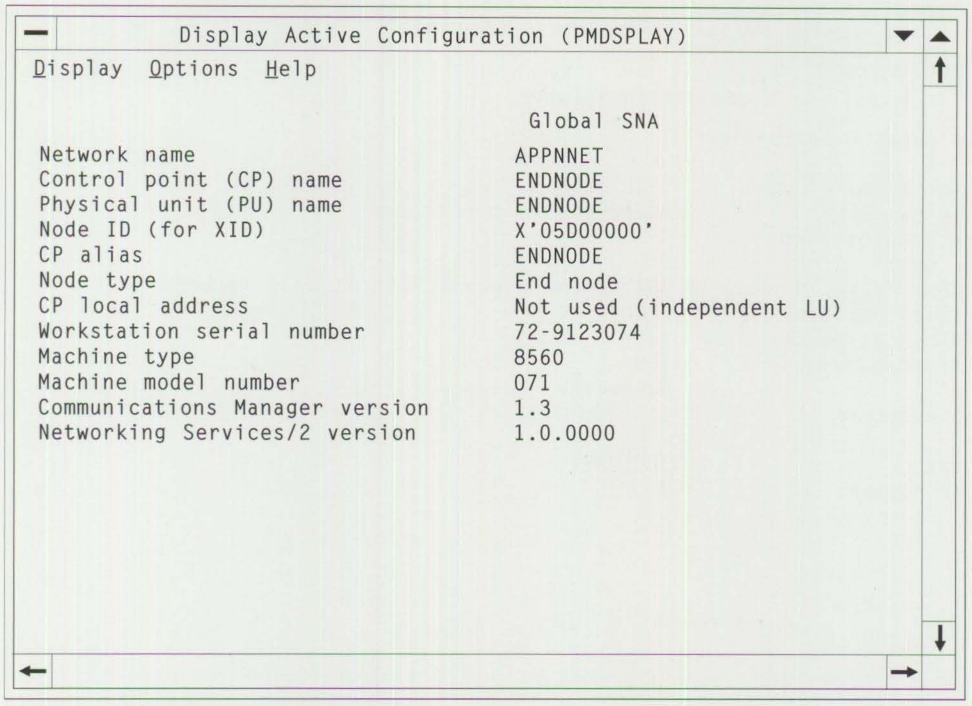
Before Networking Services/2, all tracing was started and stopped through the Communications Manager panels. Networking Services/2 now provides a command, CMTRACE, that allows users to start and stop traces and copy traces to files from the OS/2 command line. CMTRACE can provide the following trace information:

- API program problems, including CPI-C or APPC problems
- Problems establishing connections to or from another node
- System event traces that are used to trace internal events for IBM service personnel

Networking Services/2 also contains another command, FMTTRACE, that

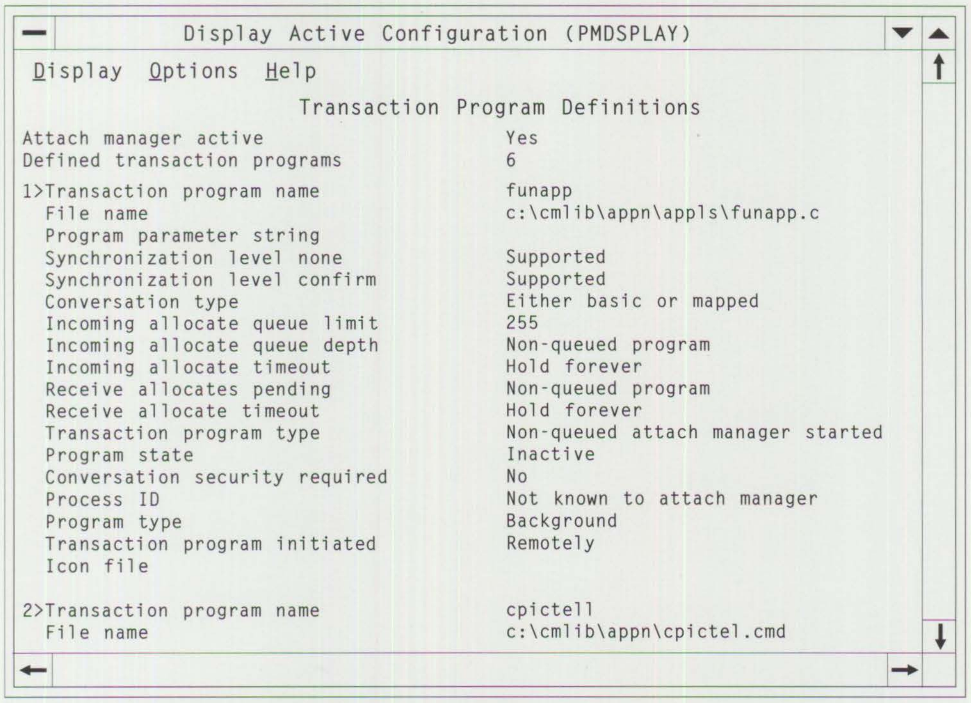
Transaction	NetBIOS	OS/2 EE 1.2	Networking Services/2
Read a Record	5.5 ms	28.0 ms	11.2 ms
100 KB File Transfer	225.0 ms	450.0 ms	159.1 ms

Figure 4. Networking Services/2 Performance Comparison



```
                                        Global SNA
Network name                           APPNNET
Control point (CP) name                ENDNODE
Physical unit (PU) name                ENDNODE
Node ID (for XID)                      X'05D00000'
CP alias                               ENDNODE
Node type                              End node
CP local address                       Not used (independent LU)
Workstation serial number              72-9123074
Machine type                           8560
Machine model number                   071
Communications Manager version         1.3
Networking Services/2 version          1.0.0000
```

Figure 5. PMDSPLAY - Quick Configuration

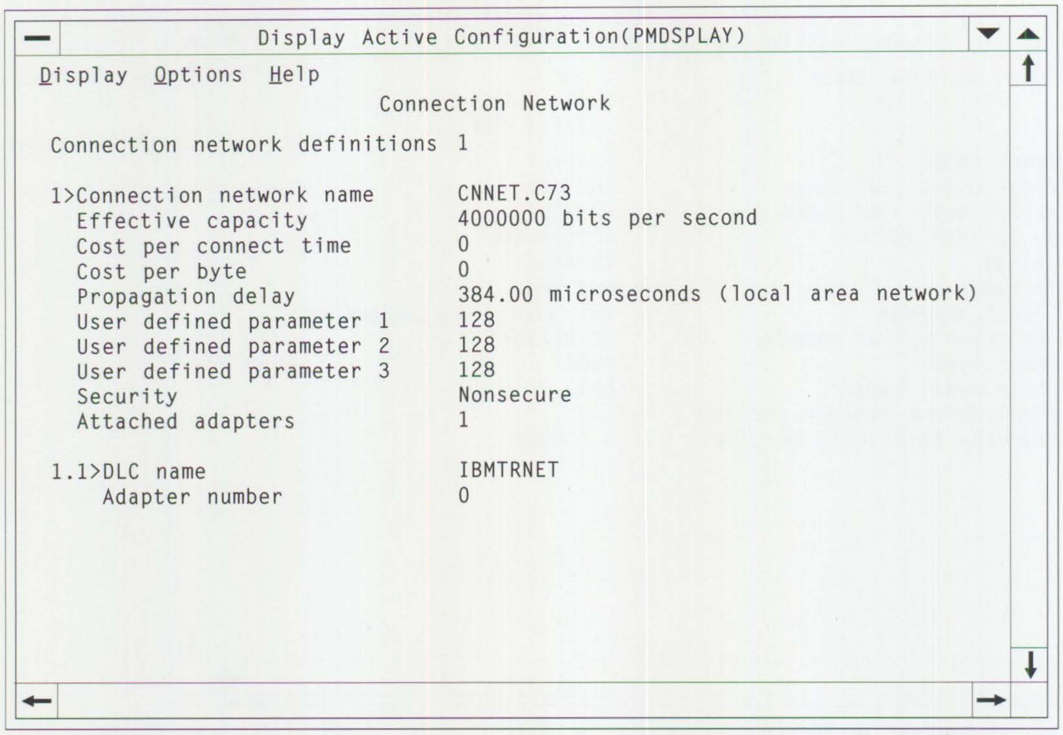


```
                  Transaction Program Definitions
Attach manager active                    Yes
Defined transaction programs             6

1>Transaction program name               funapp
  File name                              c:\cmlib\appn\appls\funapp.c
  Program parameter string
  Synchronization level none             Supported
  Synchronization level confirm          Supported
  Conversation type                      Either basic or mapped
  Incoming allocate queue limit          255
  Incoming allocate queue depth          Non-queued program
  Incoming allocate timeout              Hold forever
  Receive allocates pending              Non-queued program
  Receive allocate timeout               Hold forever
  Transaction program type               Non-queued attach manager started
  Program state                          Inactive
  Conversation security required         No
  Process ID                             Not known to attach manager
  Program type                           Background
  Transaction program initiated          Remotely
  Icon file

2>Transaction program name               cpictell
  File name                              c:\cmlib\appn\cpictel.cmd
```

Figure 6. PMDSPLAY - Transaction Program Definitions



```
Display Active Configuration(PMDSPLAY)
Display  Options  Help
                         Connection Network

 Connection network definitions 1

 1>Connection network name      CNNET.C73
   Effective capacity           4000000 bits per second
   Cost per connect time        0
   Cost per byte                0
   Propagation delay            384.00 microseconds (local area network)
   User defined parameter 1     128
   User defined parameter 2     128
   User defined parameter 3     128
   Security                     Nonsecure
   Attached adapters            1

 1.1>DLC name                   IBMTRNET
     Adapter number             0
```

Figure 7. PMDSPLAY - Connection Network

```
FMTRACE
(C) Copyright IBM Corporation 1990, 1991

-Line-  --TpId--  -Convid-                  ------------------API----------------
-Line-  --DLC---   #  ------DA------  LFSID

     2                                      ------------CMINIT(KIM      )----------->
     4              0800ef4a                <-----------CMINIT(CM_OK)-------------
    21              0800ef4a                ----------------CMALLC--------------->
    38  bef1371d                            McAllocate(USIBMSER.B673C328,#INTER,...>
    52  bef1371d    0800571d                <-------------McAllocate--------------
    66              0800ef4a                <-----------CMALLC(CM_OK)-------------
    68              0800ef4a                -----CMSST(CM_SEND_AND_DEALLOCATE)----->
    70              0800ef4a                <------------CMSST(CM_OK)-------------
    72              0800ef4a                ---------------CMSEND(6)-------------->
    93  bef1371d    0800571d                --------McSendData(6,DealSync)--------->
    97  bef1371d    0800571d                <-----McSendData(0x0001,0x00000006)-----
   101  bef1371d    0800571d                --------McSendData(6,DealSync)--------->
   105                                00102                                          --------FMH5,ApplData---->
   113  bef1371d    0800571d                <---------------McSendData------------>
   117              0800ef4a                <-------------CMSEND(CM_OK)----------->
   120                                00201                                          <------FMH5,ApplData-----
   128  0a085f1d    0800571d                <-------RcvAlloc(KIM,KIM<,INTER)--------
   140              0800df4a                <------------CMACCP(CM_OK)-------------
   157              0800df4a                -----------------CMEPLN-------------->
   160              0800df4a                <----CMEPLN(USIBMSER.B673C328,CM_OK)----
   163              0800df4a                ----------------CMRCV(256)------------>
   181  0a085f1d    0800571d                --------------McRcvWait(256)---------->
   186  0a085f1d    0800571d                <-----McRcvWait(0x0009,0x00000000)------
   194              0800df4a                <CMRCV(CM_OOMPLETE_DATA_RECEIVED,6,CM...-
   197  0a085f1d                            --------------TPEnded(Soft)----------->
   201  0a085f1d                            <-------TPEnded(0x0001,0x00000001)------
```

Figure 8. FMTTRACE Summary Trace File

```
FMTTRACE
(C) Copyright IBM Corporation 1990, 1991
    :
    :

Line:     21 CPI-C request
Time stamp: 15:15:23:68
CMALLC
    Conversation ID = 0x0800EF4A00000200
Conversation characteristics:
    Conversation type = CM_MAPPED_CONVERSATION
    Return control = CM_WHEN_SESSION_ALLOCATED
    Sync level = CM_NONE
    Partner LU name = USIBMSER.B673C328
    Mode name = #INTER
    Partner TP name = cpichear

Line:     38 API request
Time stamp: 15:15:23:68
McAllocate
    TP ID = 0xbef1371d3c04b39a
    Sync level = None
    Return control = When session allocated
    Conversation group ID = 0x00000000
    Partner LU alias =
    Mode name = #INTER
    TP name = cpichear
    Security = Same
    Fully-qualified partner LU name = USIBMSER.B673C328
         :
         :
```

Figure 9. FMTTRACE Detailed Trace File

allows users to format the trace file into an easier-to-use form. This formatted trace file can be a summary of the original trace (one line per trace event) or a detailed file (one line per formatted field).

Figure 8 shows a summary trace file of a simple CPI-C application.

Figure 9 shows a portion of the detailed trace file for the same CPI-C application. Note that this is an expanded trace of the summary trace file shown in Figure 8.

Benefitting from Networking Services/2

Networking Services/2 provides many new helpful tools for developing and using distributed applications with APPC. Increased performance, better tools and samples, and more flexibility in configuration now make APPC an excellent choice when developing applications for a wide range of platforms and communications media, resulting in high-performance solutions for both wide area and local area networks.

ABOUT THE AUTHOR

Kimberly D. Murray is a senior associate programmer in APPC Market Enablement at IBM's laboratory in Research Triangle Park, North Carolina. After joining IBM in 1988, she was a product planner for IBM SAA Networking Services/2. Kim received a BS in computer science from the University of Tennessee at Knoxville.

The AAI Family of Products

Thomas C. Miller
Creative Systems Interface Inc.
Lexington, Massachusetts

The AAI family of products makes programming your cooperative applications easy. Application-to-Application Interface (AAI) supports the client-server and peer-to-peer architecture across many systems. It is a consistent Remote Procedure Call (RPC) function using a call interface on multiple platforms.

AAI simplifies cooperative programming because it eliminates the need for programmers to write communication-specific code. AAI allows you to call and execute a program on any system – local (on the same system) or remote (on a different system) – exactly as you would call a subroutine. The AAI product has a consistent user (programmer) interface on all systems.

Today, the only prerequisite is that the system has Advanced Program-to-Program Communications (APPC) or CPI-C capability. Cooperative processing is supported with IMS through CICS™ and its ISC (LU6.1) communication facilities and directly through the LU6.2 IMS adapter, an IBM offering. Creative Systems Interface will, over time, support all IBM systems and most non-IBM systems.

AAI products now available are:

- AAI/CICS™ (MVS™, VSE, and VM)
- AAI/VTAM (for MVS batch and TSO)
- AAI for MVS (using APPC/MVS)
- AAI/400 (for OS/400®)
- AAI/OS (for OS/2 EE)
- AAI/DOS (for PC DOS)
- AAI/WIN (for Windows 3.0)
- AAI for AIX® (and UNIX)

All AAI products connect to IMS/DC and the IMS LU6.1 to LU6.2 adapters. Your application program can reside on any of these systems, and call and execute an application on any of the others. Any system can be a client (requesting a service), and any system (except PC DOS) can be a server (providing a service).

Cooperative, or peer-to-peer, processing provides an effective and economic answer for your users' requirements. Most cooperative solutions use the client/server architecture as a model. RPC supports the client/server model, but is used mostly within a local area network. Other cooperative models offer limited or focused function.

For example, the distributed data access model allows you to read a remote database as if it were local. This is an important element of IBM's Data Warehouse direction. This model assumes that all application logic resides in a single system; only the database is remote. Other models use LU2 (3270 transmissions) to develop a cooperative application. This model has the host system viewing the remote application program as a terminal. The partner, however, is a slave to the host, meaning this is not a true cooperative model.

There are many solutions available to connect one system to another. Most provide only a physical link. Once connected, only a small part of the problem is solved. Nothing in the physical connection lets an application program access and execute another application program. You still have to write the instructions to make the programs communicate.

There are different communication protocols that can support a client/server conversation (exchange of data). APPC is the richest protocol in function and broadest in support.

Cooperative processing with APPC does not limit you to one partner. Your program can request processing from multiple servers simultaneously, regardless of its location. Therefore, users see a seamless transaction: one screen that executes more than one program at multiple locations.

Because programs can be executed on different systems, you can pick the best system for the job. Many experts believe that processing logic and data should be as close to the user as possible. Other experts believe specific hardware should be used for its best purpose, for example, a personal computer used for graphics.

These two opinions can be put together cooperatively for a sound solution. For a large customer database, a mainframe system has the best hardware to manage it, and probably the best software. Users can have a personal computer with graphical display capability to process cooperatively with a data retrieval program on the host. This not only provides an effective solution, but an economic one. You can have many PCs using the same retrieval program. Because the graphics are on the PC, the amount of data sent across the communication line is minimized. Also, it is advisable to take optimal advantage of your PC's processing capability while the mainframe manages the I/O activity.

This is a powerful tool for cooperative processing. You can design an application to solve a problem and let each part of the solution run where it makes the most sense. But it is not easy to use APPC to develop cooperative applications – it requires an experienced, highly skilled and trained systems-level programmer.

AAI eliminates this need and lets you develop cooperative applications with today's application programmers – all with little impact on the project schedule.

Application Programming

Simple Application Programming: Most customers would like to develop applications using traditional programming techniques. They make programs simple to design and implement. In Figure 1, program A calls program B and sets up two data areas. Area 1 is the information that A wants B to process and area 2 is where B places the result. This design is simple and easy to test, and can be implemented by any application programmer. This approach is responsive to many business needs.

Cooperative Application Programming: If program A and program B execute on different nodes (or systems), the coding of a call is more complex (Figure 2). APPC provides a set of rules where one program can exchange data with another on a different node. APPC uses the LU6.2 protocol of SNA for communication.

Coding the APPC function is difficult because the commands and logic for managing a communication session are complex.

The programmer replaces the standard call in program A with APPC code, and adds APPC code to program B where no code existed previously. This is difficult to write, difficult to test, and requires different code on every system that supports APPC. With CPI-C, the base commands are the same across Systems Application Architecture (SAA) platforms. The other complexities remain the same.

Coding the APPC function is difficult because the commands and logic for managing a communication session are complex. The code in both programs A and B must be identical. If an error occurs, each program needs to know its exact state. That is, each program needs to know where it is in the flow of communication – whether it's sending, receiving, waiting for commit, and so forth. Each program then needs to decide what to do with the error and what the other program will do.

There are many advantages in designing a cooperative application using APPC. Because different hardware and software systems provide unique advantages, you can design a system that provides the best solution to a problem without the restrictions of one particular machine.

An APPC application is complex in many areas. The first is the skill level required by the designer and pro-

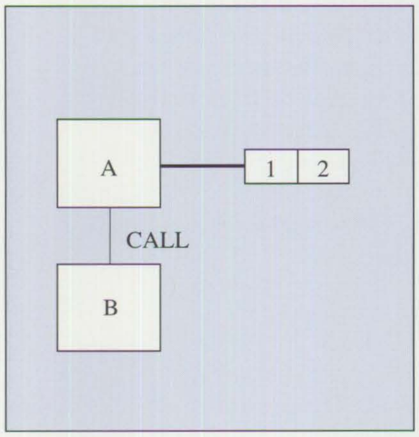


Figure 1. Simple Call

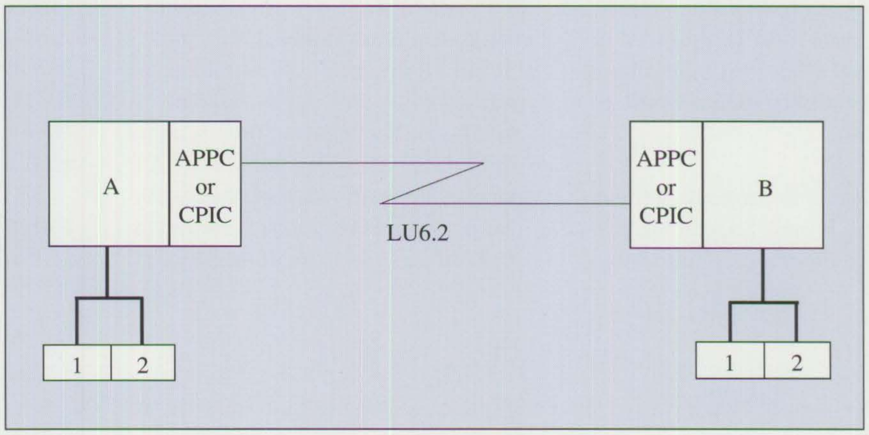


Figure 2. Cooperative Call

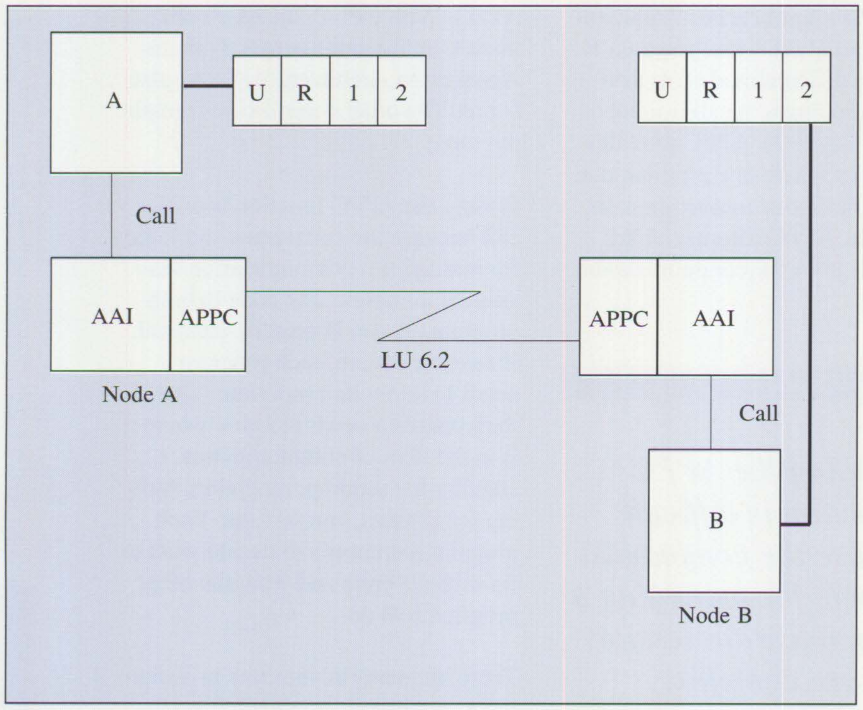


Figure 3. Call with AAI

grammer. Creative Systems Interface Inc. has found that a senior-level programmer requires a minimum of six months education and experience to write quality APPC code. Second, the coding effort requires communications help from the systems programmers. Testing is the next area, and the most complex. Your application program must be prepared to recover from all the error conditions that can occur in a communications environment.

Finally, once your program contains APPC code, it must be maintained. Any change in your communication environment requires a change in the program, and every program may have APPC code.

Programming with AAI: In Figure 3, program A calls program B using AAI. The call is now as simple to code as Figure 1. The difference here is that program A defines two additional data areas (labeled "U" and "R" in Figure 3). These two areas are defined using ten simple move instructions. The learning time is about 15 minutes. The programmer for A codes a standard call for program B, and program B responds the same to A. Both programmers think they are working with local routines, executing with standard calls. AAI manages all the communications, including the movement of data from node A to node B. AAI also translates the data if the two systems use different data types, like the PC and IBM mainframe.

Summary

Your staff can now code and test programs as easily as shown in our first example. Testing is easier because of the call interface. Both programs can run on the same machine during testing and can then be moved to separate systems for production. While the design effort is the same for any cooperative application, AAI eliminates all the communication considerations required with APPC. You will gain all the advantages of a cooperative processing application with none of the APPC hassle. This is the best of both worlds, which is the power of AAI.

ABOUT THE AUTHOR

Thomas C. Miller is vice president and partner of Creative Systems Interface Inc. (CSII), Lexington, Massachusetts. He is responsible for CSII's product marketing and sales. After 14 years in various marketing and marketing management positions with IBM, Tom joined CSII in 1988. He earned a BA and MBA from the University of Wisconsin.

Creative Systems Interface Inc.
(508) 872-0965

Securing the Enterprise Workstation

Arthur L. Goddu
IBM Corporation
Boca Raton, Florida

The protection of enterprise information continues to be essential in maintaining an organization's competitive edge. Programmable workstations compose the fastest-growing element of the enterprise data-processing environment. To respond to the demand for a secured workstation, IBM has announced Secured Workstation Manager/DOS featuring identification and authentication, discretionary access control, and audit capabilities for DOS workstations.

As part of its commitment to be the leader in PC workstation solutions, IBM has announced a next generation product of the TRIUMPH!® Workstation Manager, Version 2. The new product, Secured Workstation Manager/DOS, is available in two versions. One version uses the Data Encryption Standard (DES) and the other uses a non-DES algorithm for masking data. Both versions are referred to here as Secured/DOS.

Secured/DOS requires DOS 3.3, 4.0, or 5.0 and has a windows mode that requires Microsoft Windows 3.0. The memory requirement for Secured/DOS is 8 KB with extended/expanded memory, and 54 KB without (the DES version requires 60 KB without) and runs on most IBM PCs. Secured/DOS has all the functionality of TRIUMPH! along with a simplified installation procedure and anti-virus protection. It's designed to meet the D2 subsystem requirements of the Department of Defense. It supports a graphical user interface that conforms to much of the Common User Access (CUA) architecture in windows mode and has equivalent character-based menus in DOS mode.

Secured/DOS is used to guard the software assets and data on the DOS workstation. It provides a wide range of tools to prevent sensitive data from being exposed or modified. It also simplifies the administration of large numbers of computers, allows supervisors to track users' activities, and automates complex logon procedures. Its tools can support the following activities:

- Control access to the workstation.
- Control access to individual files on the workstation and encryption of these files. Users can control access to their own files; system administrators can also control access to other users' files.
- Provide information about users' activities on the workstation. Users' activities will be recorded,

but only administrators can manipulate audit information.

- Allow a central administrator to control the rights of users on network terminals and remote, unconnected workstations.
- Guide the user through complex logon procedures, particularly in network environments. The administrator can perform this function for the users or allow users to participate in customizing their own environments.
- Allow application programs to access and use the security information stored by Secured/DOS. These tools are typically used by application programmers.

It also provides additional utilities that:

- Help users move protected information from place to place
- Allow users to check the status of files

Workstation Access Control

Secured/DOS provides for the use of passwords or tokens to assure user identification and authentication. As a complementary security service, it has an interface to the IBM Transaction Security System (TSS), which has smart-card support, signature verification support, and Personal Identification Number (PIN) support. This support is required in applications such as banking ATM services, or can be used as a supplementary proof to logon to certain sensitive systems where highly sensitive data is stored. Secured/DOS controls access to your workstation by installing a security kernel that:

- Limits access to a set of users defined by the administrator of the system. If the administrator has specified that tokens are required for logon, additional security is added. Tokens and other physical security devices not only require the user to know the correct password (which may be guessed or stolen) but also require the user to produce a physical device to logon. If the user elects to use tokens, neither the token nor user password by itself will allow access to the system – both must be produced in order to logon.
- Replaces part of the hard disk's boot sector so that users cannot access the system by exiting the booting procedure before the security software is invoked.
- Does not allow users access to the hard disk if the workstation is booted from the floppy disk.

Secured/DOS provides for the use of passwords or tokens to assure user identification and authentication.

File Protection

Secured/DOS has commands that allow the user to encrypt and define rights to files on an individual file-by-file basis or by subdirectory. Users can protect files that are already in existence as well as files that have not yet been created. File creation involves user-defined rules that immediately and automatically protect files. The user and administrator versions of these file protection commands are collectively referred to as Discretionary Access Control (DAC).

DAC becomes active after user setup and logon has occurred. DAC is discretionary because the user can selectively allow rights to other users. DAC capabilities are optional to those who do not choose to protect their workstations on a file-by-file basis. There are two levels of DAC commands: user-level commands that are available to all users, and more powerful commands that are available to the administrator, which when installed cannot be superseded.

File Encryption

Using any DAC command on a file automatically protects the file, adds additional information to the file in the form of a label (512 bytes), and imbeds the contents of the access rule in the label. In addition, both the label and the file itself are encrypted.

Secured/DOS offers both a proprietary encryption scheme and DES encryption. If the DES command is not used to implement DES keys (or the DES version is not available), Secured/DOS will encrypt, using its own encryption algorithm keyed by the domain phrase that is entered during installation. If DES encryption is used, Secured/DOS prompts for additional keys to be used specifically for the DES encryption algorithm. Only the system administrator can enter or change DES keys. Users can receive updated DES keys as part of Central Site Administration (CSA), which is explained later, but they will not normally be involved in the creation of DES keys.

Note: When a DES key is part of the CSA procedure, files encrypted with the old DES key will no longer be accessible. The administrator will normally decrypt files before keys are updated using administrator-level commands.

Monitoring User Activity (Auditing)

In addition to controlling the access of users, Secured/DOS creates a record of user activity. Such a record allows the system administrator to:

- Detect security violations such as unauthorized logon attempts or attempts to access sensitive or restricted information
- Make management decisions regarding the allocation of resources (by project or department), employee performance, and scheduling

The term *auditing* has been used in the security industry to describe the process of evaluating an existing corporate security system. In Secured/DOS, however, it refers to the process of creating, protecting, translating, and evaluating a record of user activity on a protected workstation.

The Secured/DOS auditing function collects and reports details about each event occurring on a protected workstation. Auditing recognizes 14 common events. You can also use the Micronyx® System Integration Support (SIS) product to define up to 19 custom events of your own, if desired. The auditing function records, protects, and stores user activity information in a file of a predefined name. This information can be translated to common file formats for use in spreadsheets, word processors, databases, and so on.

Further analysis can be done with the Micronyx Browser utility. This utility allows the administrator to query by example the audit file and create custom reports of the type needed by your installation. Auditing functions are administrator-level activities.

Central Site Administration

The process by which an administrator controls the user lists, access rights, and encryption schemes on multiple workstations with Secured/DOS is called Central Site Administration. CSA permits the administrator to "bundle" configurations and rules, and changes to such, without compromising security, so that they can be duplicated, delivered, and installed by anyone.

The auditing function records, protects, and stores user activity information in a file of a predefined name.

The administrator creates a rule/configuration package, then stores it as a file on diskette or a file server, where it can be accessed but not altered by users. By entering a command (ATTN), users install the new package, from wherever the administrator has delivered it, on the security kernel of that workstation. This is an auditable function; therefore, the administrator can track the installation if necessary. In a LAN environment the administrator can set up the AUTOEXEC.BAT file so that when the user logs on, the file server will automatically be searched by that workstation to see if there is a "package" to install.

Automating and Controlling User Logon

Secured/DOS has a set of tools for simplifying logon procedures, which allow the user to automate procedures that are complex, unique to each user, or to be controlled by the system administrator. Users often use these tools, known collectively as Common Sign On (CSO), to simplify the process of network access. CSO is equally effective in placing a user in a specific drive and subdirectory and invoking the software package most often used. CSO routines can be protected so that the Control and Break keys will not allow users to exit the logon process. CSO goes beyond the capabilities of even well designed AUTOEXEC.BAT files because:

- Individual logon procedures can be established for each user and can be automatically invoked when the user logs on.
- CSO utilities can access secure information, such as network passwords, for each user and pass them to predefined systems such as LANs and hosts.

Additional Features

Secured/DOS also has the following security features:

- An object reuse feature that reboots the workstation when a user logs off, so there is absolutely no residual data left for the next user to inadvertently see. All memory is cleared.
- A Guest User option, which can be set up so that simply pressing Enter on the keyboard allows administrator-defined, limited usage of the workstation, in keeping with the enterprise security policy rules.
- A Secure Delete function that deletes all traces of a file so that it cannot be retrieved.
- A Suspend feature that can be configured to trigger after a given period (configured by the admin-

istrator) of inactivity of the keyboard and can be armed by the user so that any keyboard input suspends the workstation. When suspended, the only access to the workstation is by the user who was logged on at the time of suspense, and the system will verify the same user by prompting for the user ID and password. The system must be rebooted in order for another user to log on.

- Help panels are available with Secured/DOS to help users through most actions and make it easy to use.
- Mouse support has been added for the new CUA-conformant panels.
- Access Control can be granted to groups as well as individuals, and each user can be configured into any groups by the administrator.
- Directories and files can be defined as protected by the administrator or user.

Secured/DOS gives the security needed to guard software assets and data on the enterprise workstation.

Summary

Secured/DOS gives the security needed to guard software assets and data on the enterprise workstation. It guards the files as individual files and as resident in directories, and prevents access to protected data. This protected data cannot be compromised if the system is booted from a diskette; therefore, misappropriation, corruption, disclosure, and theft can be prevented with Secured/DOS.

ABOUT THE AUTHOR

Arthur Goddu is an advisory programmer at IBM's OS/2 program office in Boca Raton, Florida. Presently, he is the technical interface between IBM and Micronyx Inc. After joining IBM in 1967 in San Jose, California, Art worked in several programming positions, one of which was a state-of-the-art language and maintained numerous IBM finance and capitalization programs. His experience includes quality assurance for the Series/1™ and teaching programming techniques using Pascal.

Little Solutions

We invite you to share your "little solutions" in this column. Send them to us in care of the editor.

Subclassing Made Easy!

Every now and then I hear a comment about how difficult it is to subclass a window. Perhaps an explanation of what subclassing does and how it is invoked will make this subject a little clearer.

OS/2 is a message-based system, and its messages are constantly being routed by the PM message queue. These messages carry all types of information: status, activity, input, output, behavior, and so on. There are messages for the OS/2 PM system, for each application, and for each window. Subclassing gives the programmer a way to examine messages earlier than normal or take a look at those hidden from view.

Each window in an OS/2 PM application has a certain predefined behavior (or class). By subclassing a window, the behavior of that window can be modifed to suit the user. In subclassing a window, messages enroute to the window procedure for that window are rerouted. The new destination is a procedure written to perform a special function. After messages are examined in the new window procedure, most are sent to that window's regular procedure (Figure 1) to let the system do most of the work. A few of the messages passing by will be intercepted by the new procedure so the new behavior can be implemented.

Figure 2 shows how the messages are diverted from their original destination to the New Window Procedure. In most cases, messages are routed back to the Regular Window Procedure so the system can act on them.

The implementation of a subclassed window procedure is deceptively simple. First you must declare a function prototype of the new procedure as shown in the following.

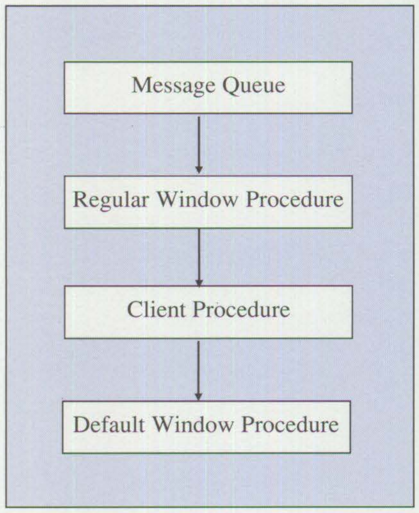


Figure 1. This shows the message path defined before a window is subclassed.

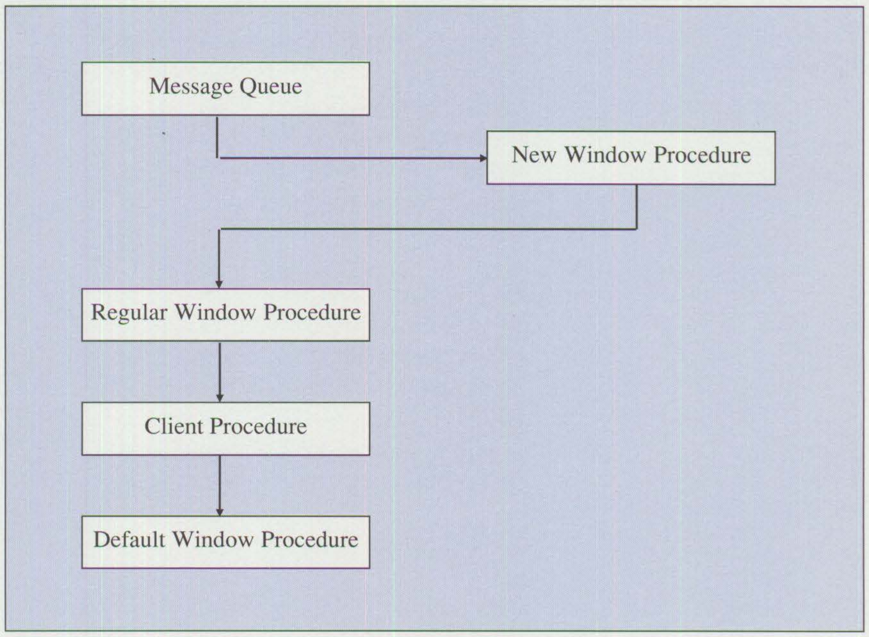


Figure 2. This shows how messages are diverted from their original destinations to the New Window Procedure.

```
  MRESULT EXPENTRY
      NewWindowProcedure(
            HWND,
            USHORT,
            MPARAM,
            MPARM);
```

Next (somewhere in the initialization section of the application), a function pointer variable must be declared to save a pointer to the Regular Window Procedure such as:

```
PFNWP RegularWindowProcedure;
```

The last step is to call WinSubclassWindow. This API substitutes the pointer to New Window Procedure for the pointer to the Regular Window Procedure. The old pointer is returned and saved in the variable RegularWindowProcedure as shown here:

```
  RegularWindowProcedure =
      WinSubclassWindow(
         hwndWindow,
         NewWindowProcedure);
```

The New Window Procedure is written like any window procedure. The only difference is that at the end of the procedure, the return must be to the Regular Window Procedure as shown here:

```
  MRESULT EXPENTRY
    NewWindowProcedure(
                HWND hwnd,
                USHORT msg,
                MPARAM mp1,
                MPARAM mp2)
    {
      switch(msg)
      {
         .
         .
         .
      default:
       return((
        *RegularWindowProcedure)
                (hwnd,
                msg,
                mp1,
                mp2));
         .
         .
         .
      }
    }
```

I'll show a simple example of changing the default behavior of a listbox in the next issue. — *Larry Pollis, IBM, Dallas*

OS/2 Version 1.3 Hints

Having your menus and programs in the same location every time you boot allows you to customize your desktop to your needs. It is helpful and is very easy to do. The procedure is as follows:

1. Size and place your windows on your desktop.

2. Go to the Desktop Manager group.

3. Select Desktop from the action bar.

4. Select Save from the pull-down menu.

5. Select OK.

You also have the option to save your desktop settings at shutdown time. Just make sure you have your desktop arranged like you want it and check the Save checkbox. Then click on the Shutdown button. If you do *not* want your saved desktop to be modified at shutdown time, make sure the Save checkbox is *not* checked.

Here's another hint. Many OS/2 users want to have their most commonly used applications automatically started when they boot up OS/2. This is a great time saver and it is equally easy to do! The procedure is as follows:

1. Highlight the program entry.

2. Select Program from the action bar.

3. Select Properties from the pull-down menu.

4. Check the box for Open when the system is started.

5. Select OK.

6. Select Change.

The program will be started when the system is booted.

As an example, assume you want File Manager to be started at boot time. The procedure is as follows:

1. Highlight the File Manager entry.

2. Select Program from the action bar.

3. Select Properties from the pull-down menu.

4. Select Options.

5. Check Open when the system is started.

6. Select OK.

7. Select Change.

File Manager will be started when OS/2 is booted.

You can undo this by going through the same procedure and removing the check in the "Open when system is started" box.
— *Darin Divinia, Dallas*

New Products

Hardware

IBM PS/2 Model 95 XP 486

IBM enhances the performance of the entry models of the PS/2® Model 95 XP 486 family with the introduction of two new entry models. These new models are based on Micro Channel® architecture and use the new i486SX (486SX) 25 MHz microprocessor. The 486SX consists of a 32-bit 80486 processor without the integrated numeric co-processor unit. Full-function 486/25 MHz capability for numeric-intensive applications can be economically achieved through the addition of the optional 487SX Math Co-Processor. In addition, these entry-level models can be upgraded to the more powerful 486/33 MHz or the 486/50 MHz processors via the PS/2 486/33 and 486/50 Processor Upgrade Options.

The PS/2 Model 95 XP 486 (8595-0H9) is a 486SX/25 MHz system with a 160 MB Small Computer System Interface (SCSI) fixed disk installed as the standard Direct Access Storage Device (DASD). The PS/2 Model 95 XP 486 (8595-0HF) is a 486SX/25 MHz system with a 400 MB SCSI fixed disk installed as the standard DASD. In addition, these new models feature 8 MB of memory standard on the system board and support interleaved and non-interleaved memory. The new PS/2 Model 8595-0HF with the addition of 400 MB fixed disk options can now provide up to 2 GB of internal data storage. This capacity better meets customer requirements for large databases typical in Local Area Network (LAN) server and multi-user host applications.

All other standard features remain unchanged: processor upgradability, XGA graphics, and expandability attributes of the PS/2 Model 95 XP 486 platform announced October 30, 1990.

The recently announced 487SX Math Co-Processor allows numeric processing capabilities to be added to the 486SX versions of the Models 90 and 95. The IBM PS/2 Math Co-Processor 487SX/20–25 MHz contains the new i487SX co-processor with installation instructions. The i487SX provides all the capabilities of the i486SX plus integrated floating-point processing to enhance performance for numeric-intensive applications.

The PS/2 Model 95 XP 486 is supported by OS/2 Standard Edition (SE) 1.30.1, OS/2 Extended Edition (EE) 1.30.1, OS/2 Version 2.0 (when available), IBM DOS Versions 3.3, 4.0, and 5.0, AIX PS/2 Version 1.2.1, and IBM 4680 Operating System Versions 2 and 4.

Highlights:

- New processor complex featuring the 486SX/25 MHz microprocessor
- 8 MB standard parity memory, expandable to 64 MB on the system board
- PS/2 SCSI 32-bit bus master adapter with cache
- Up to 2 GB of internal high-speed data storage
- Eight 32-bit Micro Channel expansion slots (one slot for the SCSI adapter and one for the XGA Display Adapter/A)
- Seven internal storage device bays supporting a combination of 3.5-inch half-height drives and 5.25-inch full-height drives
- XGA Display Adapter/A with 512 KB video memory providing 1024 x 768 resolution with 16 colors
- One DMA serial port and one DMA parallel port
- Selectable boot and easy-to-upgrade licensed System Programs

Letter # 191-194, October 17, 1991

IBM PS/2 Model 90 XP 486

IBM enhances the performance of the entry models of the PS/2 Model 90 XP 486 family with the introduction of two new entry models. These new models are based on Micro Channel architecture and use the new i486SX 25 MHz microprocessor. The 486SX consists of a 32-bit 80486 processor without the integrated numeric co-processor unit. Full-function 486/25 MHz capability for numeric-intensive applications can be economically achieved through the addition of the optional 487SX Math Co-Processor. In addition, these entry-level models can be upgraded to the more powerful 486/33 MHz or the 486/50 MHz processors via the IBM PS/2 486/33 and 486/50 Processor Upgrade Options.

The PS/2 Model 90 XP 486 (8590-0H5) is a 486SX/25 MHz system with an 80 MB SCSI fixed disk installed as the standard DASD. The PS/2 Model 90 XP 486 (8590-0H9) is a 486SX/25 MHz system with a 160 MB SCSI fixed disk installed as the standard DASD. In addition, these new models feature 4 MB of memory standard on the system board and support interleaved and non-interleaved memory. The new PS/2 Model 8590-0H9 with the addition of 400 MB fixed disk options can now provide up to 960 MB of internal data storage. This capacity better meets customer requirements for large databases typical in LAN server and multi-user host applications.

All other standard features remain unchanged: processor upgradability, XGA graphics, and expandability attributes of the PS/2 Model 90 XP 486 platform announced October 30, 1990.

The PS/2 Model 90 XP 486 is supported by OS/2 SE 1.30.1, OS/2 EE 1.30.1, OS/2 Version 2.0 (when available), IBM DOS Versions 3.3, 4.0, and 5.0, AIX PS/2 Version 1.2.1, and IBM 4680 Operating System Versions 2 and 4.

Highlights:

- New processor complex featuring the 486SX/25 MHz microprocessor
- 4 MB standard parity memory, expandable to 64 MB on the system board
- Enhanced performance XGA graphics integrated on the system board providing 1024 x 768 resolution with 16 colors
- Up to 960 MB of internal high-speed data storage
- PS/2 SCSI 32-bit bus master adapter with cache
- Four internal storage device bays supporting three 3.5-inch half-height drives and one 5.25-inch half-height drive
- Four 32-bit Micro Channel expansion slots (one for the SCSI adapter)
- Two DMA serial ports and one DMA parallel port
- Selectable boot and easy-to-upgrade licensed System Programs

Letter # 191-195, October 17, 1991

IBM PS/2 Model 90 XP 486

IBM is extending the PS/2 Model 90 XP 486 family of systems by providing additional PS/2 fixed disk storage models (8590-0K9 and 0KF). The PS/2 8590-0K9 features the 486/33 MHz 32-bit microprocessor with a high-performance 160 MB SCSI fixed disk installed as the standard DASD. The 8590-0KF features the 486/33 MHz 32-bit microprocessor with a high-performance 400 MB SCSI fixed disk installed as the standard DASD. In addition, these new models feature 8 MB of memory standard on the system board, and with the addition of the 400 MB fixed disk options can now provide up to 1.2 GB of internal data storage, allowing these systems to better meet customer requirements.

All other technologically advanced, standard features such as processor upgradability, XGA graphics, and the expandability attributes of the PS/2 Model 90 XP 486 platform remain unchanged.

The PS/2 Model 90 XP 486 is supported by OS/2 SE 1.30.1, OS/2 EE 1.30.1, OS/2 Version 2.0 (when available), IBM DOS Versions 3.3, 4.0, and 5.0, AIX PS/2 Version 1.2.1, and IBM 4680 Operating System Versions 2 and 4.

Highlights:

- Processor complex featuring the 80486 33 MHz microprocessor
- 8 MB standard parity memory, expandable to 64 MB on the system board
- Enhanced performance XGA graphics integrated on the system board providing 1024 x 768 resolution with 16 colors
- 400 MB and 160 MB SCSI fixed disks with 11.5 ms and 16 ms average seek time, respectively
- Up to 1.2 GB of internal high-speed data storage
- PS/2 SCSI 32-bit bus master adapter with cache
- Four internal storage device bays supporting three 3.5-inch half-height drives and one 5.25-inch half-height drive
- Four 32-bit Micro Channel expansion slots (one slot for the SCSI adapter)
- Two DMA serial ports and one DMA parallel port
- Selectable boot and easy-to-upgrade licensed System Programs

Letter # 191-196, October 17, 1991

IBM PS/2 486SX/25, 486/33, and 486/50 Processor Upgrade Options

The PS/2 486SX/25 Processor Upgrade Option enhances the PS/2 Model 90 XP 486 and Model 95 XP 486 by providing additional processor growth capability. The PS/2 486SX/25 Processor Upgrade Option features the 32-bit Intel i486™ processor running at 25 MHz on a processor complex designed to upgrade 486SX/20 MHz or 487SX/20 MHz versions of the PS/2 Models 90 XP 486 and 95 XP 486 systems.

The newly announced PS/2 Model 90 XP 486 (8590-0H5 and 0H9) and Model 95 XP 486 (8595-0H9 and 0HF) can be upgraded by the previously announced IBM PS/2 486/33 and 486/50 Processor Upgrade Options. These processor upgrade options (feature numbers 5269 and 5270) significantly enhance the range of processor performance that customers can now achieve with the expandable processor concept introduced on the PS/2 Model 90 XP 486 and 95 XP 486 systems.

The operating systems and extensions supported by the 486SX/25, 486/33, and 486/50 Processor Upgrade Options are OS/2 SE Version 1.30.1, OS/2 EE Version 1.30.1, OS/2 Version 2.0 (when available), OS/2 LAN Server Version 1.30.1, DOS Versions 3.3, 4.0, and 5.0, AIX PS/2 Version 1.2.1, and the IBM 4680 Operating System Versions 2 and 4.

Highlights:

- 25MHz, 33 MHz, and 50 MHz 80486 32-bit microprocessors
- Internal memory cache controller and 8 KB internal memory cache
- Internal floating-point processor unit (standard in 33 MHz and 50 MHz Processor Upgrade Options; optional with 25 MHz Processor Upgrade Option)
- Allows processor upgrade in PS/2 Model 90 XP 486 and 95 XP 486 systems
- Well suited for compute-intensive and for numeric-intensive (with internal floating-point processor unit) applications, as well as heavy use multi-user, multitasking applications

Letter # 191-198, October 17, 1991

IBM PS/2 Model 35 LS

IBM is extending the PS/2 Model 35 SX and LS family of systems by providing an additional LAN station. The PS/2 Model 35 LS (14X) is a medialess Ethernet LAN station that supports Ethernet Version 2 and IEEE® 802.3 (CSMA/CD) interfaces, and data transfers at 10 Mbps via 10Base-T (twisted-pair) or 10Base5 (thick cable) connection. Data transfers via 10Base2 (thin cable) can be accomplished through the Attachment Unit Interface (AUI) connection and user-provided external transceiver. Remote Initial Program Load (RIPL) support is a standard feature on the Ethernet adapter.

The PS/2 Model 35 LS (14X) features the 80386SX microprocessor operating at 20 MHz with zero to two wait states and has the following integrated functions: parallel port, serial port, pointing-device port, 16-bit VGA port, keyboard port, 1.44MB diskette drive support, Math Co-Processor socket, and three memory SIMM sockets (two available for memory expansion) with 2 MB of memory standard on the system board (expandable to 16 MB).

The PS/2 Model 35 LS (14X) is shipped with all the standard features of the Model 35 SX with the following exception. No DASD devices are installed in this medialess system, but an Ethernet adapter card with RIPL occupies one of the three adapter slots. It is fully upgradable to the Model 35 SX configurations.

The PS/2 Model 35 SX (14X) supports the IBM Enhanced Keyboard (101/102 keys), Space Saving Keyboard (84/85 keys), or the IBM Host Connected Keyboard (122 keys). The keyboard selection can be specified only in new equipment orders and cannot be ordered separately for on-order or installed equipment.

The IBM PS/2 Cable Cover (#5610, 95F5610) is a new option available for PS/2 Models 35 SX and 35 LS.

This option gives the user the ability to secure the rear cables. If this option is installed, rear cables (such as serial, parallel, and communication cables) cannot be removed without the key for the system unit.

Highlights:

- Unshielded twisted pair, thick, and thin cable support via AUI
- RIPL, selectable boot feature
- 20 MHz 80386SX, 16-bit VGA, 2 MB memory (expandable to 16 MB)
- Two full-size, 8/16-bit expansion card slots and two DASD bays (supporting 1.44 MB, 2.88 MB, and 1.2 MB diskette drives options, and 40 MB and 80 MB fixed disk options)
- 3270 affinity, Selectable Keyboard option (84, 101, or 122 keys)

Letter # 191-199, October 17, 1991

IBM PS/2 8518 Color Display

The PS/2 8518 is a high-quality 14-inch VGA color display. It has a larger viewing area than comparable VGA color displays; when combined with outstanding front-of-screen quality and clarity, this makes it the display of choice for the VGA user. The Extremely Low-Frequency Magnetic Field (ELMF) and Very Low-Frequency Magnetic Field (VLMF) have been reduced in response to customer requests and emerging international requirements. The 8518 is available in four models to cover worldwide marketing areas.

Highlights:

- A 14-inch anti-reflective screen, 0.28mm dot pitch, high-contrast glass, with large viewable area
- Full VGA compatibility
- Designed for high reliability
- Meets Swedish MPR 1990:08 guidelines for ELMF and VLMF emissions

Letter # 191-200, October 17, 1991

IBM PS/2 Cached Processor Option for IBM PS/2 Model 8557

The PS/2 Cached Processor Option enhances the announced IBM PS/2 Model 8557 systems. The processor option features the new IBM 386SLC 20 MHz microprocessor and provides the capability of a processor performance upgrade for the IBM PS/2 Model 8557. This new microprocessor includes an internal memory cache controller and 8 KB memory cache. The IBM PS/2 Cached Processor Option significantly increases the processor performance in various applications, especially compute-intensive applications, while providing investment protection.

Highlights:

- 20 MHz IBM 386SLC
- Internal memory cache controller and integrated 8 KB cache memory
- Improves processor throughput up to 88% depending on application
- Enhances system performance for compute-intensive applications
- Allows upgrades of IBM PS/2 Model 8557 systems
- Installs into a math co-processor socket connector
- Provides a socket for 80387SX 20 MHz or equivalent math co-processor

Letter # 191-201, October 17, 1991

IBM 7537 Industrial Computer

The IBM 7537 Industrial Computer is designed to complement and expand the low end of the IBM industrial computer line. It provides product improvements such as increased processor speed, configuration flexibility, and VGA graphics over the current IBM 7531 and 7532 Industrial Computers. The 7537 provides a 20 MHz 80386SX microprocessor, 2 MB of memory, selectable no-fixed-disk or 80 MB fixed disk drive versions, a 1.44 MB media-sense diskette drive, selectable 197 watt or 205 watt power supply with battery backup capability (requires the optional backup battery), selectable bench-top or rack-mount versions, in a mechanical package with five adapter slots and four storage-device bays.

The 7537 is recommended when increased performance or capability is needed, or when future growth is anticipated. For IBM 7531 and 7532 Industrial Computer users (Personal Computer AT® compatible), the 7537 provides significant improved performance.

Highlights:

- Improved performance with the 20 MHz 80386SX microprocessor
- 2 MB of 85 nanosecond (ns) memory standard on the system board (expandable to 16 MB)
- Five full-size 8/16-bit expansion card slots and four DASD bays (two bays available in fixed-disk model and three bays available in diskette-only model)
- Selectable disk drive, power supply, and mounting configuration capabilities
- Faster, industry-standard VGA graphics

Letter # 191-150, September 4, 1991

IBM 7546 Industrial Computer

The IBM 7546 Industrial Computer is a Micro Channel system designed to complement and expand the low end of the IBM industrial computer line. It provides increased processor speed compared to the predecessor 7541 and 7542 Industrial Computers, combined with enhanced configuration flexibility, VGA graphics, and an integrated SCSI I/O interface.

The 7546 Industrial Computer provides a 20 MHz 80386SX microprocessor, 4 MB of standard memory, selectable 80 or 160 MB SCSI fixed-disk drives, a 2.88 MB media-sense diskette drive, available battery-backed power supply, and selectable bench-top or rack-mount versions, in a mechanical package with five adapter slots and four storage-device bays. The 7546 supports up to 16 MB of 70 ns memory on the planar in three SIMM sockets. The system ships with one 4 MB SIMM installed.

The 7546 Industrial Computer is recommended when increased performance or capability is needed, or when future growth is anticipated.

Highlights:

- Improved performance with the 20 MHz 80386SX microprocessor
- Memory expandable to 16 MB on the system board
- Increased number of adapter slots and storage device bays
- Selectable and upgradable disk-drive capacity, power supply, and mounting configuration
- Increased capacity 2.88 MB diskette drive with media sense, providing support for 720 KB, 1.44 MB, and new 2.88 MB diskettes
- Faster, industry-standard VGA graphics

Letter # 191-151, September 4, 1991

Real-time Interface Co-processor Six-Port V.35 Interface Board/A and Related Features

The Real-time Interface Co-Processor Six-Port V.35 Interface Board/A and related features provide another electrical interface option to the currently available Real-time Interface Co-Processor Portmaster Adapter/A. This feature for the Micro Channel machines, Industrial Computer, and PS/2 provides six ports of compatible V.35/V.36 electrical interfaces with data rates up to 256 Kilobits Per Second (Kbps) simultaneously on six ports configured as Data Terminal Equipment (DTE), with external clocking provided, or up to 230 Kbps simultaneously on six ports configured as Data Circuit-Terminating Equipment (DCE), with internal clocking. Up to 2.048 Megabits Per Second (Mbps) can be achieved using only one port configured as a DTE.

(Actual performance may vary depending on the user's applications.)

Letter # 191-205, October 22, 1991

IBM PS/2 Ultimedia Model M57 SLC

The PS/2 Ultimedia™ Model M57 SLC (8557-255) is a Micro Channel system designed to complement and enhance the PS/2 Model 57 family with multimedia capability. It adds the following standard product improvements to the Model 57 SX:

- Multimedia front panel with stereo headphone jack, mono microphone jack, volume control, and enhanced loudspeaker
- 16-bit XGA adapter card with 1 MB VRAM supports 640 x 480 with 65,000 colors or 1024 x 768 with 256 colors
- 16-bit audio adapter card with I/O to the front panel supports FM-quality stereo
- IBM PS/2 mouse
- 80 MB SCSI fixed disk
- A new 600 MB CD-ROM/XA drive (PS/2 CD-ROM II) with connection to the multimedia front panel, supporting existing Compact Disk (CD) formats and enabled to support new CD-ROM/XA formats
- A CD containing three operating environments, a variety of IBM/vendor multimedia application samplers, and an "Introducing Ultimedia" demonstration

PS/2 Ultimedia Model M57 SLC usability features include an installation program (supplied on diskette) that is designed for minimal customer interaction when installing operating systems from the supplied CD.

Also included are IBM OS/2 Version 2.0, IBM DOS 5.0, and Microsoft Windows 3.0 with Microsoft Multimedia Windows Extensions 1.0.

The PS/2 Ultimedia Model M57 SLC is designed for desktop as well as floorstanding operation (a floorstand is included). The mechanical package has five Micro Channel slots and four bays for I/O devices. The Audio Capture and Playback Adapter and the XGA Adapter are installed in two of the slots, leaving three slots for future expansion. A 3.5-inch 2.88 MB media-sense diskette drive, a SCSI fixed disk, and a CD-ROM/XA drive are installed in three of the bays. An additional 5.25-inch or 3.5-inch device, optical diskette drive, fixed disk drive, tape drive, CD-ROM drive, or a similar device can be installed in the remaining bay.

Highlights:

- Faster, high-resolution display via 16-bit XGA graphics
- Quality sound via stereo 16-bit audio subsystem
- Improved convenience with front panel audio I/O jacks, volume control, and enhanced loudspeaker
- Increased data access via CD-ROM/XA drive
- Quicker, easier setup via the installation program and the operating systems provided
- Easier familiarization via the interactive multimedia system demonstration
- Improved performance with the new 20 MHz IBM 386 SLC microprocessor
- Expandable memory, up to 16 MB on the system board
- 2.88 MB diskette drive with media sense, providing support for 720 KB, 1.44 MB, and 2.88 MB diskettes
- Immediate system utilization with samplers and demonstrations supplied by IBM and independent software vendors

Letter # 191-188, October 17, 1991

IBM PS/2 Actionmedia II Display Adapter 2MB IBM PS/2 Actionmedia II Display Adapter/A 2MB IBM PS/2 Actionmedia II Capture Option

These ActionMedia II multimedia adapters, which are part of the Ultimedia product family from IBM, allow the full range of still natural images, motion video, and quality audio information to be incorporated into new and exciting PS/2 multimedia applications. The Action-

Media II Display Adapter 2 MB (feature #9730) and the ActionMedia II Display Adapter/A 2MB (feature #9732) provide for the replay of DVI™ technology applications; and, with the addition of the ActionMedia II Capture Option (feature #9734), live sound and motion video can also be used as input to such applications. These adapters are supplied with device drivers and programming libraries that support a new ActionMedia II Audio-Video Kernel (AVK) programming interface. AVK Version 1.0 will be provided for use with OS/2.

Highlights:

- Improved user productivity results from more natural ways of using personal computers
- Supports all DVI technology compression formats and a baseline JPEG still-image algorithm
- The all-digital DVI multimedia technology fits into existing information technology environments and, with suitable applications, becomes operable over LANs and data communications networks
- Configurations to suit most personal computer platforms with configuration options to grow with application demands

Letter # 191-189, October 17, 1991

IBM PS/2 TouchSelect for 12-Inch Displays

The PS/2 TouchSelect for 12-Inch Displays is a feature that a customer can install to bring touch-screen operation to the IBM PS/2 8513 Color Display. This feature is intended for application areas where touch is used for user input and interaction. The feature attaches easily to the 8513 display, enabling customers with installed displays to upgrade to touch-screen capability. The operation of the 8513 display is not affected by the addition of the IBM TouchSelect.

Touch-screen device drivers are supplied with the feature to support touch operation with IBM DOS, OS/2 SE, OS/2 EE, or Microsoft Windows 3.0 operating environments.

Highlights:

- A customer-installed feature that enables direct touch input on IBM 8513 Color Displays and protects investment in installed displays
- Enables solutions for individuals who are not familiar with computers or keyboards, thereby increasing user productivity and enabling growth into new business areas
- Leadership features include the ability to use any input medium (finger, stylus, and so on) and to measure varying pressures exerted on the screen
- Device drivers for IBM DOS, OS/2, and Microsoft Windows 3.0 provide compatibility with applications created for the IBM PS/2 8516 Touch Display
- System attachment is provided via the PS/2 system unit pointing device port, eliminating the need for a serial port, expansion slot, or external controller

Letter # 191-191, October 17, 1991

Enhancements to the Audio and Video Multimedia Adapters from IBM

The M-Audio Capture and Playback Adapter, M-Audio Capture and Playback Adapter/A, and the Video Capture Adapter/A have been enhanced to provide, at no extra charge, new audio and video Application Programming Interfaces (APIs) and audio device drivers. Support for Microsoft Multimedia Windows Extensions has been added. The new high- and low-level programming interfaces simplify authoring by the software developers producing multimedia applications that take advantage of the capabilities of these adapters. To expand the application development possibilities, these high- and low-level interfaces will operate under the IBM DOS, DOS Windows 3.0, and OS/2 operating environments. The availability of these enhancements with the adapters provides end users with the tools to run newly developed multimedia applications using these interfaces.

These new device drivers and APIs will be shipped with the M-Audio Capture and Playback Adapter (feature #3908; new part number 92F3378 replaces 87F9908), M-Audio Capture and Playback Adapter/A (feature #3909; new part number 92F3379 replaces 87F9909), and the Video Capture Adapter/A (feature #2785; new part number 92F3380 replaces 34F2785). All unshipped orders will be converted to the new part numbers. Customers who purchased and received these adapters prior to the availability of this new interface code will be able to obtain these enhancements at no charge.

Highlights:

- Audio: The new audio interface tools provided with the M-Audio Capture and Playback Adapters:
  - Support Microsoft Multimedia Windows Extensions
  - Provide a common audio interface to multimedia application programs
  - Simplify the addition of audio capabilities to newly developed multimedia applications
  - Enable software developers to write independent of the com-

puter environment or audio interfaces being supported
  - Offer the flexibility to write applications in multiple audio data formats
  - Contain a synthesizer feature allowing creation of applications that can playback Musical Instrument Digital Interface (MIDI)
  - Include a JPEG API that provides a video image compression/decompression function for multimedia applications
- Video: The new video interface tools provided with the Video Capture Adapter/A:
  - Simplify writing applications for the adapter
  - Minimize the amount of memory overhead needed for specific applications as a result of a new modular design

Letter # 191-192, October 17, 1991

IBM PS/2 TV

The PS/2 TV (2460) unit is a low-cost video and audio option that enables the user to display full-motion National Television Systems Committee (NTSC) analog video on a standard PS/2 color display from a variety of sources. These sources include cable TV, an external TV antenna, and traditional baseband video sources such as a VCR or videodisc player. Video can be presented as either a full-screen display or a fixed size Picture-in-Picture (PIP) that places a video window over the user's full-screen application display. TV, audio, and video attributes – including TV channel selection, volume, color, tint, brightness, contrast, and the PIP location – can be adjusted using simple keyboard sequences or using software menu selections.

The PS/2 TV is fully supported in all VGA modes. In XGA mode, all functions of the PS/2 TV are supported except the PIP function.

The PS/2 TV is packaged in an external enclosure that will normally be positioned directly below the display on the desktop. The unit contains a speaker and headphone jack, has its own power source, and does not require a PS/2 system unit expansion slot.

Letter # 191-193, October 17,1991

IBM PS/2 8515 Color Display Model 021

The new PS/2 8515 Color Display is a 14-inch screen multimode, analog, color display featuring high contrast and clarity for alphanumeric, graphics, and image applications that support VGA and XGA modes. When using IBM PS/2 XGA products, 256 colors at a time may be selected. The IBM PS/2 8515 Color Display Model 021 replaces the IBM PS/2 8515 Color Display Model 001.

The Extremely Low-Frequency Magnetic Field (ELMF) and Very Low-Frequency Magnetic Field (VLMF) have been reduced in response to customer requests and emerging international requirements.

Highlights:

- 14-inch display supporting VGA and XGA modes, featuring high contrast, picture clarity, and color definition; XGA compatibility enabling high-quality text fonts and graphics that increase user performance and productivity
- Investments protected through compatibility with IBM PS/2 products, VGA and XGA modes, and applications generated for XGA mode
- Meets Swedish MPR 1990:08 guidelines for ELMF and VLMF emissions

VGA and XGA compatibility, coupled with a large viewing area, good contrast, reduced glare, high picture clarity and color definition provide the PS/2 user with a cost/performance display for using high-quality fonts and graphics.

Letter # 191-155, September 4, 1991

IBM PS/2 8516 Touch Display's Low Emissions and Software Support

The ELMF and VLMF of the PS/2 8516 Touch Display have been reduced in response to customer requests and emerging international requirements.

The PS/2 8516 Touch Display primarily supports IBM DOS, OS/2 SE, and OS/2 EE, but touch device drivers for Microsoft Windows 3.0 are also included with the display.

The PS/2 8516 Touch Display now offers support for a greater variety of software applications.

Letter # 191-174, September 17, 1991

IBM Token-Ring Network 16/4 Adapter/A

The IBM Token-Ring Network 16/4 Adapter/A operates with PS/2 Micro Channel architecture and offers improved performance, increased function, and a new size for use in either a short or full-sized slot in the PS/2. Remote Initial Program Load (RIPL) is standard with the adapter.

Highlights:

- Additional throughput extends the life of the host machine as LAN activity increases. The smaller size allows it to be used in the short slot of a PS/2.

- RIPL, at no extra cost, provides the capability for improved security and simplified IPL content administration for workstations.

Letter # 191-216, November 5, 1991

IBM PS/2 Communications Cartridge I

The PS/2 Communications Cartridge I (3541) is an expansion unit for use on IBM's PS/2 Model L40 SX. The PS/2 Communications Cartridge I is a one-slot (half-size card) unit that supports connectivity to host or LAN networks by using the IBM Token-Ring, 3270, or 5250 adapter cards.

The PS/2 Communications Cartridge I is designed to give mobile professionals connectivity capability for their portable system in the office.

Highlights:

- Improves productivity of customers who need to maintain synchronization of portable files and host/LAN files
- Increases the customer flexibility to use the portable/notebook system in home/office as well as mobile environments
- Permits the use of portables with existing peripherals and networks through simple installation and operation

Letter # 191-204, October 17, 1991

IBM LaserPrinter Option for Appletalk Networks

The IBM LaserPrinter Option for AppleTalk® Networks (feature number 3540; part number 1333540) connects an IBM LaserPrinter with the PostScript Option to a Macintosh computer via the AppleTalk network. The Option plugs directly into the IBM LaserPrinter's interface connector and converts AppleTalk network protocols to serial communication. To communicate with the IBM LaserPrinter, Macintosh computer users can use the IBM LaserPrinter Desk Accessory (DA) program supplied with the product. The IBM LaserPrinter DA allows the Macintosh computer user to access features and functions of the IBM LaserPrinter 4029 or IBM LaserPrinter 4019 not supported by the standard Apple LaserWriter® printer driver.

Highlights:

- Connection to Apple Macintosh computers via the AppleTalk network
- Support for IBM LaserPrinter 4029 features and functions
  - Paper handling (500-sheet second drawer and envelope feeder)
  - Print resolution (PQET and 600 dpi with 4 MB of additional printer memory installed)
- Support for IBM LaserPrinter 4019 features and functions
  - Paper handling (500-sheet second drawer and envelope feeder)

Letter # 191-214, October 29, 1991

Software

IBM Extended Services

IBM announces two new program products that provide database and communications function to OS/2: IBM Extended Services with Database Server for OS/2, and IBM Extended Services for OS/2. Both programs also contain database administration tools, Query Manager, and enhanced communications support for LAN and stand-alone workstations. The Extended Services with Database Server program includes a distributed feature called Database Client Application Enablers. This feature enables DOS, DOS Windows, and OS/2 clients to access any Extended Services database server on a LAN. This same client support can provide read/write access to IBM host servers supporting Distributed Relational Database Architecture (DRDA) via the IBM Distributed Database Connection Services/2 Version 1.0 product. The IBM Extended Services for OS/2 program includes DOS, DOS Windows, and OS/2 client support similar to that in the distributed feature.

Extended Services is designed to:

- Run on IBM OS/2 SE Version 1.3 (Refresh Level 1.30.1) or replace the Communications and Database Manager functions of IBM OS/2 EE Version 1.3 when upgraded to Refresh Level 1.30.1. Extended Services will also run on the OS/2 Version 2.0 32-bit operating system. In addition, Extended Services supports selected versions of OS/2 that have been determined to be compatible with IBM OS/2 SE Version 1.3 Refresh Level 1.30.1 or later (marketed by other vendors).
- Be supported on OS/2-compatible versions of the IBM Personal Computer and PS/2 hardware as well as on selected non-IBM hardware

Extended Services includes many function, performance, and usability enhancements to the database and communications applications previously available in OS/2 EE Version 1.3. The base OS/2, previously part of the Extended Edition's Database and Communications Manager components, is now a separately-packaged program. The LAN Requester is now provided with IBM OS/2 LAN Server Version 2.0.

Limited availability is restricted to customers who have ordered and will install DATABASE 2™ (DB2®) Release 2 Version 3 and/or Operating System/400® (OS/400) Version 2 Release 1 Modification Level 1 before April 24, 1992. Customers who meet these criteria must have ordered the above products, as well as IBM Extended Services and SAA Distributed Database Connection Services/2 Version 1.0 prior to March 27, 1992.

IBM Extended Services has National Language Support for twelve languages: Canadian French, Dutch, Finnish, French, German, Italian, Norwegian, Portuguese, Spanish, Swedish, U.K. English, and U.S. English.

Highlights:

- Extends communications and database functions to selected non-IBM hardware platforms and to selected, non-IBM OS/2-compatible operating systems
- Enhances the LAN transport layer in the communications function for improved performance and enables exploitation of the Network Driver Interface Specification (NDIS) by both IBM and vendors
- Offers new communications and database function and usability improvements over those offered in OS/2 EE
- Offers an Administrator's Kit to facilitate planning, configuration, and administration for a multi-workstation or host-connected environment

Letter # 291-599, October 22, 1991

No-Charge Upgrade Promotion for OS/2 Standard Edition Version 1.X to OS/2 Version 2.0

For a limited time only, eligible customers who acquired any release of OS/2 SE Version 1.X before March 31, 1992, can receive a no-charge upgrade to IBM OS/2 Version 2.0 (84F7586, 3.5-inch; 10G2991, 5.25-inch).

To qualify for this promotion, eligible customers must acquire the upgrade on or after March 31, 1992, but no later than July 31, 1992. Information on how to acquire the upgrade will be provided before March 31, 1992.

Each eligible customer will receive the following for each qualifying location (a qualifying location is defined as a single building designated by a single mailing address):

- One complete IBM OS/2 Version 2.0 package containing diskettes and documentation
- One IBM OS/2 Version 2.0 "Proof of Additional License" certificate for each additional complete package, and one for each "Additional License Copy" of OS/2 Standard Edition Version 1.X being upgraded

Customers can also receive one *OS/2 Version 2.0 User's Guide* at no charge for each "Proof of Additional License" certificate they receive.

All IBM customer types, except IBM internal customers, are eligible for this promotion, which is offered only for licenses granted in the United States and Puerto Rico.

IBM reserves the right to modify or withdraw this promotion at any time.

Letter # 391-178, October 22, 1991

No-Charge Upgrade Promotion for OS/2 Extended Edition Version 1.X to Extended Services with Database Server for OS/2

For a limited time only, eligible customers who acquired any release of OS/2 EE Version 1.X before April 24, 1992, can receive at no charge:

- Upgrades to IBM OS/2 Version 2.0 (84F7586, 3.5-inch; 10G2991, 5.25-inch)
- Upgrades to IBM Extended Services with Database Server for OS/2 (04G1049, 3.5-inch; 04G1050, 5.25-inch)
- "Proof of License for Distributed Feature" certificates for Database Client Application Enablers

To qualify for this promotion, eligible customers must acquire this upgrade on or after March 27, 1992, but no later than August 24, 1992. Information on how to acquire the upgrade will be provided before March 27, 1992.

Each eligible customer will receive the following for each qualifying location (a qualifying location is defined as a single building designated by a single mailing address):

- One complete OS/2 Version 2.0 package containing diskettes and documentation
- One complete Extended Services with Database Server for OS/2 package containing diskettes and documentation
- One OS/2 Version 2.0 "Proof of Additional License" certificate and one Extended Services with Database Server for OS/2 "Proof of Additional License" certificate for each additional complete package, and one for each "Additional

License Copy" of OS/2 Extended Edition Version 1.X being upgraded

- "Proof of License for Distributed Feature" certificates for Database Client Application Enablers; certificates received to equal in number the copies of DOS Database Requester Library (DBDRQLIB), up to a maximum of 128 per OS/2 Extended Edition Version 1.X license being upgraded

Customers can also receive one *OS/2 Version 2.0 User's Guide* at no charge for each OS/2 Version 2.0 "Proof of Additional License" certificate they receive.

All IBM customer types, except IBM internal customers, are eligible for this promotion, which is offered only for licenses granted in the United States and Puerto Rico.

IBM reserves the right to modify or withdraw this promotion at any time.

Letter # 391-179, October 22, 1991

OS/2 Lan Server Version 1.X to Version 2.0-Entry No-Charge Upgrade and Distributed Feature Promotion

For a limited time only, eligible customers who acquired any release of OS/2 LAN Server Version 1.X before April 24, 1992, can receive at no charge:

- Upgrades to IBM OS/2 LAN Server Version 2.0-Entry (04G1051, 3.5-inch; 04G1052, 5.25-inch)
- "Proof of License for Distributed Feature" certificates for LAN Requesters

To qualify for this promotion, eligible customers must acquire this upgrade on or after April 24, 1992, but no later than August 24, 1992. Information on how to acquire the upgrade will be provided before April 24, 1992.

Each eligible customer will receive the following for each qualifying location (a qualifying location is defined as a single building designated by a single mailing address):

- One complete OS/2 LAN Server Version 2.0-Entry package containing diskettes and documentation
- One OS/2 LAN Server Version 2.0-Entry "Proof of Additional License" certificate for each additional complete package, and one for each "Additional License Copy" of OS/2 LAN Server Version 1.X being upgraded
- "Proof of License for Distributed Feature" certificates for LAN Requesters; certificates received to equal in number the IBM DOS LAN Requesters (up to a maximum of 128 per OS/2 LAN Server license being upgraded) and/or the OS/2 Extended Edition Version 1.X licenses currently in use in support of IBM LAN Server products.

All customer types, except IBM internal customers, are eligible for this promotion, which is offered only for licenses granted in the United States and Puerto Rico.

In addition to the above promotion, customers can choose during the promotion period to upgrade any or all of their OS/2 LAN Server Version 1.X licenses to OS/2 LAN Server Version 2.0-Advanced for an upgrade charge. If customers choose to do so, they can still receive the no-charge "Proof of License for Distributed Feature" certificates as noted above.

IBM reserves the right to modify or withdraw this promotion at any time.

Letter # 391-180, October 22, 1991

IBM OS/2 Version 2.0 Tools for Application Development

IBM introduces four new products and two new convenience kits.

The two convenience kits are:

- IBM OS/2 2.0 Developer's Workbench, which consists of IBM OS/2 2.0 Developer's Toolkit and IBM WorkFrame/2 Version 1.0 program packages. The IBM OS/2 2.0 Developer's Workbench convenience kit provides the user with a flexible, language-independent base for application development.
- IBM C Developer's WorkSet/2, which adds the IBM C Set/2 Version 1.0 program package to the above IBM OS/2 2.0 Developer's Workbench. The IBM C Developer's WorkSet/2 convenience kit provides the user with a self-contained, 32-bit OS/2 C language application development environment.

The licensed programs that constitute the IBM OS/2 2.0 Developer's Workbench and IBM C Developer's WorkSet/2 products are available separately. These three program packages and the IBM OS/2 2.0 Technical Library are described as follows:

- IBM OS/2 2.0 Developer's Toolkit is a comprehensive selection of language-independent build tools, productivity tools, sample programs, online reference information, and a kernel debugger. The IBM OS/2 2.0 Developer's Toolkit is designed to help the developer exploit the OS/2 Version 2.0 APIs.
- IBM WorkFrame/2 Version 1.0 is a configurable, project-oriented

application development environment, featuring an SAA/CUA conforming user interface. WorkFrame/2 is built with an open interface to serve as the integration point for the tools in the IBM OS/2 2.0 Developer's Toolkit as well as tools supporting C and other languages. Both 16-bit and 32-bit OS/2 tools can be plugged into the IBM WorkFrame/2.

- IBM C Set/2 Version 1.0 consists of both a 32-bit SAA C Compiler with its runtime libraries, which generates code for IBM OS/2 Version 2.0, and a fully interactive, full-function, source-level PM debugger.
- IBM OS/2 2.0 Technical Library is a comprehensive set of publications designed to complement the IBM OS/2 2.0 Developer's Workbench and help the programmer take full advantage of the programming interfaces provided in IBM OS/2 Version 2.0.

Letter # 291-625, October 22, 1991

NetWare Network Computing Products from IBM

Additional products are offered as part of IBM's product distribution, licensing, and support relationship with Novell, Inc., consistent with IBM's interoperability strategy. Under the terms of this relationship IBM markets, services, and supports NetWare® computing software, now including the new NetWare Lite from IBM for small businesses, and additional NetWare communications products for enterprise-wide computing.

- NetWare Lite from IBM Version 1.0
- NetWare from IBM Version 3.11 (10 users)
- NetWare for SAA from IBM Version 1.1 (16 sessions)
- NetWare for SAA from IBM Version 1.1 (64 sessions)
- NetWare 3270 LAN Workstation for DOS from IBM Version 2.0
- NetWare 3270 LAN Workstation for Macintosh from IBM Version 1.0

Highlights:

- NetWare Lite from IBM: Low cost, easy to use, peer-to-peer network providing file and printer sharing
- NetWare from IBM Version 3.11: Additional lower cost, 10-user product
- NetWare 3270 LAN Workstation from IBM: IBM terminal and print emulation for DOS and Macintosh, consistent with NetWare for SAA from IBM
- NetWare for SAA from IBM: Additional lower cost, 16-session product with new features – two PUs per server, APPC interface, and status display

Letter # 291-631, October 22, 1991

IBM PS/2 Internal Tape Backup Program Version 2.0 (OS/2 PM Compatible)

The PS/2 Internal Tape Backup Program Version 2.0 (OS/2 Presentation Manager Compatible) enhances the PS/2 8540, 8550, 8557, 8560, 8565, 8570, 8580, 8590 and 8595 by allowing the user to transfer up to 120 MB of formatted data from a disk storage device to a removable minitape cartridge for later recall. The IBM PS/2 Internal Tape Backup Program Version 2.0 (OS/2 PM Compatible) is written specifically for the OS/2 Presentation Manager (Version 1.2 and later). It provides an easy-to-use window environment for use with either a mouse or a keyboard. For convenience, the IBM PS/2 Internal Tape Backup Convenience Kit Version 2.0 (OS/2 PM Compatible) has in one package the IBM PS/2 Internal Tape Backup Program Version 2.0 (OS/2 PM Compatible), the IBM Internal Tape Backup Unit (#5279), and a formatted minitape cartridge.

In addition, the customer can upgrade from earlier versions of DOS and OS/2 compatible programs by ordering the IBM PS/2 Internal Tape Backup Program Upgrade Version 2.0 (OS/2 PM Compatible).

Letter # 291-626, October 22, 1991

Sytos Plus Version 1.30 for OS/2 and DOS

Sytos Plus File Backup Manager Version 1.30 for OS/2 and Sytos Plus File Backup Manager Version 1.30 for DOS now include support for the PS/2 2.3 GB External Full-Height SCSI Tape Drive and the PS/2 3.5-inch Rewritable Optical Drive. In addition, Sytos Plus Version 1.30 continues to support the IBM 6157 Tape Drive and the PS/2 2.3 GB Internal Full-Height SCSI Tape Drive. One program supports IBM OS/2 SE Version 1.2 (and later) and IBM OS/2 EE Version 1.2 and 1.3. The other version supports IBM DOS Version 3.30 (and later).

Highlights:

- Systems management features for stand-alone and network server workstations provide a high-performance archive, backup, compare, and restore utility.
- Scheduled/unattended procedures and error correction code options increase efficiency.
- A data compression option can reduce the amount of media used by over 50%.

- User productivity can be enhanced with an intuitive file selection sequence, contextual and indexed online helps, and a CUA-designed menu interface.

Letter # 291-563, September 17, 1991

Multimedia Presentation Generator Version 1.5

Multimedia Presentation Generator (MPG) is a LinkWay application that functions as an authoring system for the DOS platform. The software allows users to create an application using objects and to fully control their multimedia resources through a user-friendly interface. MPG Version 1.5 is the first offering from IBM and is the current offering from the vendor. MPG Version 1.5 fully supports a variety of video sources as well as many IBM multimedia products such as the M-Motion Video Adapter/A.

Highlights:

- Users can create and edit full-motion video presentations
- Supports IBM M-Motion Adapter/A, IBM M-Control Program, IBM 3510 CD-ROM, and IBM LinkWay 2.01
- Provides a user-friendly interface to LinkWay, which brings access of multimedia peripherals (videodisc, CD-ROM, VCR, NTSC camera) to computer novices
- Takes advantage of LinkWay's script language
- Mouse-driven "point and shoot" environment

Letter # 291-637, October 29, 1991

IBM M-Control Program/2 Version 2.0

The IBM M-Control Program/2 Version 2.0 (04G3544) includes programming interfaces for DOS and separate multimedia toolkits for OS/2 Presentation Manager and Microsoft Windows 3.0.

Significant changes have been made to support the Media Control Interface (MCI) for the Microsoft Multimedia Windows Extensions 1.0.

The IBM M-Control Program/2 Version 2.0 allows application developers to use the multimedia features of the IBM M-Motion Video Adapter/A. It replaces the current IBM M-Control Program/2.

Enhancements include:

- Support for the IBM M-Motion Video Adapter/A and various videodisc player device drivers for Microsoft Multimedia Windows Extensions
- Sample MCI application with source code
- Digital-audio support for the IBM M-Motion Video Adapter/A under Windows 3.0 in both MCI and non-MCI environments
- Improved video windowing performance under Windows 3.0
- Additional videodisc player support

Users who acquire the IBM M-Control Program/2 Version 2.0 are entitled to a future update of the program at no additional charge when the OS/2 multimedia extension support becomes available.

Highlights:

- Provides new videodisc player drivers supporting the Microsoft Windows MCI standards
- Provides accessibility to the IBM M-Motion Video Adapter/A's digital audio function through the low-level audio services of the Microsoft Windows MCI. Also, provides emulation of the low-level audio services that enables digital audio in a non-MCI environment. Both mono and stereo sample rates are supported.
- Provides a string-level MCI device driver to access the motion and still video functions of the IBM M-Motion Video Adapter/A. This MCI device driver includes new customized commands that support the capabilities of the IBM M-Motion Video Adapter/A under the Microsoft Multimedia Windows 1.0.
- Enhances video windowing performance under Multimedia Toolkit for Microsoft Windows 3.0
- Provides upward compatibility with IBM M-Control Program/2 supporting the IBM M-Motion Video Adapter/A:
  — Includes an IBM DOS programming interface
  — Includes an OS/2 Presentation Manager and a Windows 3.0 Toolkit containing programmable window class objects for motion video and videodisc players
- Provides investment protection with compatibility to IBM M-Motion Video Adapter/A enhancements and expanded multimedia devices

Letter # 291-592, October 17, 1991

ActionMedia II Developer's Toolkit Version 1.0

The ActionMedia II Developer's Toolkit (92F2731) allows developers of applications and tools to become more productive in the creation of ActionMedia II programs. The Developer's Toolkit provides sample C source code for use with the ActionMedia II Audio-Video Kernel when running under OS/2, together with some Audio-Video Support System (AVSS) file utilities. The Developer's Toolkit also includes the *Actionmedia II Technical Reference Manual* and a programmer's guide.

Highlights:

- Provides source C code and a programmer's guide for use by developers to gain skills in use of the ActionMedia II Audio-Video Kernel programming interface
- Provides file utilities that allow developers to edit and check the AVSS files they create

Letter # 291-594, October 17, 1991

Microsoft Multimedia Windows Extensions 1.0

The Microsoft Multimedia Windows Extensions 1.0 adds multimedia capabilities to Microsoft Windows 3.0. This program enhances the Windows graphical enviromnent in three key areas: user interfaces, multimedia accessories, and Control Panel applets.

Highlights:

- Hardware support for audio, Musical Instrument Digital Interface (MIDI), CD-ROM, displays, and joystick in multimedia environments
- Multimedia Control Panel enables easy user interfaces
- Driver applets to install new and configure existing drivers

Letter # 291-595, October 17, 1991

IBM Multimedia Presentation Manager/2 Statement of Direction

IBM intends to make available IBM Multimedia Presentation Manager/2, which will provide multimedia extensions to the OS/2 32-bit environment. In addition, a multimedia developer's toolkit will be available and contain C sample programs and documentation to assist the multimedia application developer. These products will be generally available in the U.S. and several national language countries in the first half of 1992.

Letter # 291-596, October 17, 1991

Multimedia Tools Series Statement of Direction

IBM intends to provide, through the Multimedia Tools Series, a set of programming packages developed by IBM and industry leaders that will offer, over time, the ability for the multimedia creator to plan, create, and produce multimedia titles and applications. The contents of the tools series will be available individually or in groupings optimized for various industry segments. These packages will be available on IBM PS/2 machines running OS/2 with Presentation Manager, DOS with Windows 3.0 (with their respective multimedia extensions), and DOS. Details of these offerings will be announced at a later date.

Letter # 291-597, October 17, 1991

IBM SpeechViewer Application Software Version 1.0

SpeechViewer™ is an entry-level clinical tool for speech pathologists, teachers, and other professionals trained in treating communication disorders. With the SpeechViewer hardware option, the SpeechViewer application software creates a system that accepts a client's speech input. The system then digitizes, stores, and analyzes the speech to provide critical visual and auditory feedback to the client. Although previously announced, the SpeechViewer application software is now available through certified IBM-Authorized Personal Computer Dealers and IBM-Authorized Industry Remarketers – Personal Computers.

Highlights

- Provides clinicians and clients with critical visual and auditory feedback
- Supports a wide range of clinical activity suitable for clients of all ages, and with a variety of communication disorders
- Contains exercises for pitch, loudness, voicing, vowel pronunciation, and speech timing
- Contains highly motivating exercises that utilize animated graphics and game-like strategies for capturing and maintaining interest
- Provides exercises that are easily integrated into established and accepted clinical practice

Letter # 291-660, November 12, 1991

IBM SpeechViewer II Version 1.0

The IBM SpeechViewer II is a versatile clinical tool for speech pathologists, teachers, and other professionals trained in the treatment of communication disorders. SpeechViewer II, when used with the appropriate IBM M-Audio Capture and Playback Adapter, enhances the efficiency of speech therapy by combining motivating exercises for direct client therapy, powerful speech analysis features, and clinical management features into an easy-to-use, integrated package. SpeechViewer II is an enhanced version of the IBM Independence Series™ SpeechViewer application software.

Highlights

- SpeechViewer II's full set of clinical exercises and selection of animated graphics make the system applicable to clients of all ages and with a wide range of communication disorders.
- SpeechViewer II contains exercises for pitch, loudness, voicing, vowel pronunciation, consonant pronunciation, syllable/word pronunciation, and speech timing.
- SpeechViewer II's clinical management features collect and report performance data for tracking client progress.
- Productivity gains are realized through SpeechViewer II's ease of use and through its customization features.
- An upgrade offering and compatible interfaces protect the software and skills investment of SpeechViewer customers, while SpeechViewer II's use of standard hardware assures investment protection for the future.
- Growth is assured through the flexibility and adaptability of SpeechViewer II in meeting the needs of a wide range of clients. Standard hardware allows the clinician to run a wide variety of other applications.

Letter # 291-661, November 12, 1991

IBM Presentation Manager Office/2 Version 1.1.2 and IBM Presentation Manager Office Support/MVS Version 1.1.2

IBM Presentation Manager Office Version 1 was released in Denmark early in 1991 and is now available in the United States. IBM Presentation Manager Office/2 Version 1.1.2 and IBM Presentation Manager Office Support/MVS Version 1.1.2 provide a state-of-the-art tactical solution integrating OfficeVision™/MVS (OV/MVS) function with OS/2 Presentation Manager using icons, windows, and a mouse. IBM PM Office/2 provides a workstation direct-connect GUI to mail services, file transfer, and file cabinet on OV/MVS. The OV/MVS host system element is provided by IBM PM Office Support/MVS. IBM PM Office/2 can be used with the OfficeVision Usability Aid and Application Desktop.

Highlights:

- PM Office/2 is a direct-connect workstation to OV/MVS.
- It offers cooperative processing with office services on OV/MVS.
- It improves usability through icons, windows, a mouse, and workstation tools.

Letter # 291-603, October 22, 1991

Parameter and Tuning Guide for OS/2 EE and LAN Server, Versions 1.2 and 1.3

The *Parameter and Tuning Guide for OS/2 EE and LAN Server, Versions 1.2 and 1.3* (G33F-9437) contains information to help OS/2 users tune their applications and systems for performance. The guide contains:

- A methodology on how to approach performance tuning, including:
  - — General performance concepts
  - — Steps for isolating a performance problem
  - — Benchmarking suggestions
  - — Ideas for collecting performance data
- Suggestions on how to use IBM System Performance Monitor/2 (SPM/2) for solving performance problems
- Information on performance tuning and system design to better understand which aspects affect performance for the following components:
  - — Standard Edition
  - — Communications Manager
  - — LAN Server and Requester
  - — Database Manager
- DOS Compatibility Mode hints

Contact your IBM representative to order this publication through the IBM System Library Subscription Service (SLSS).

Index to Past Issues of IBM Personal Systems Technical Solutions

Issue 4, 1991 (G325-5013)
Power Factor: Non-Linear Loads and the Power Distribution System
Database Manager: Highlights and Direction
OS/2 Communications Manager
Improving OS/2 Application Performance
Creating PM Windows with Dialog Templates
REXX Program for OS/2 LAN Server
Micro Focus COBOL/2 and the DOS Database Requester
IBM DOS 5.0 Facts and Features
IBM DOS 5.0 Upgrade
DOS 5.0 Performance Improvements
DOS Memory Management Facilities
Disk Caching Under DOS
NetWare Client-Server Interaction
LANACS Protocols

Issue 3, 1991 (G325-5012)
PS/2 Model L40 SX Laptop Portable Computer
OS/2 2.0 Considerations
Comparing PC-DOS, OS/2, and AIX PS/2 – Part 2
Using Dual Displays with OS/2
Local Area Networks: The New Utility
And You Thought LANs Were Just for the Office!
Remote LAN Management Tools
New Horizons for IBM's Shielded Twisted Pair Cabling
Tuning and Self-Tuning Features of OS/2 LAN Server
NetWare Communications and Routing Protocols
Little Solutions for LANs

Issue 2, 1991 (G325-5011)
IBM PS/2 Model 90 XP 486 and Model 95 XP 486
Choosing an I/O Bus Architecture
The Network Is the Message
Invoking Printer Job Properties
Comparing PC-DOS, OS/2, and AIX PS/2
Programming PM Using the COBOL/2 Bindings
Installing and Using the DOS Database Requester
OS/2 LAN Server 1.3 Overview
IBM Windows Connection 2.0
SNA Definitions for 3270 Emulators – Part II
IBM THINKable

Issue 1, 1991 (G325-5010)
XGA – Raising Video Expectations
Choosing betw. Shielded and Unshielded Wiring for Data Transmission
Compatibility of LAN Servers and Requesters
Running DOS LAN Requester and Novell NetWare Concurrently
Breaking the 640 KB DOS Memory Barrier
Understanding an OS/2 CONFIG.SYS File
OS/2 EE 1.2 Database Manager Performance
OS/2 EE 1.2 Competitive Performance
An Intelligent Front-End EASEL Application
Enabling Software for National Language Support
SNA Definitions for 3270 Emulators
Diskette Failures Caused by Contamination

Issue 4, 1990 (G325-5009)
First Look at the New IBM PS/1 Computer
Using the IBM 4019 LaserPrinter Effectively
Micro Channel Interface Chip Sets
Token Ring Bus Master LAN Adapters
Extension of Wiring Rules for 4-Mbit/s Token Ring Using UTP Lobes
SCSI and DISK.386.SYS
Operating System Platforms: A Business Perspective
Minimum OS/2 1.2 DASD Requirements
User Profile Management
Understanding OS/2 1.2 LAN Server Performance
PM: An Object-Oriented Approach
DOS 4.00 SHARE
A "C" Programming Model for DOS Device Drivers
An Electronic Bulletin Board for PC Users

Issue 3, 1990 (G325-5007)
DOS – A Look under the Hood to See How It Spins
Memory Management in a DOS Environment
FASTOPEN – The DOS Performance Enhancer
DOS 4.00 Compatibility Issues
'Out of Environment Space' Errors
A New LAN Requester for DOS Systems
Creating a Dialog Box Dynamically Using WinCreateDlg
An Alternative for the OS/2 START Command
CUA: A Consistent Interface

Issue 2, 1990 (G325-5006)
OS/2 End User Advantages
What's New in OS/2 Standard Edition Version 1.2?
An Application Developer's View of OS/2
Object-Oriented Programming with C and OS/2 PM – Is It Possible?
Design Goals and Implementation of the New HPFS
OS/2 EE 1.2 Database Manager – Remote Data Services
OS/2 EE Database Manager Precompiler API
UNION, INTERSECT, EXCEPT
Writing a Database Manager COBOL/2 Program
Database Manager Programming with Procedures Language 2/REXX
APPC Performance Tips for OS/2 EE
EASEL OS/2 EE PROFS: Host Code Interface
PS/2 RPG II Application Platform and Toolkit
The IBM Independence Series Products

Issue 1, 1990 (G325-5005)
Introduction to Local Area Networks
IEEE 802.3 LAN Considerations
The IBM Token-Ring Network: A New Generation
How to Design and Build a 4-Mbit/s Token-Ring LAN
How to Design and Build a 16-Mbit/s Token-Ring LAN
Making the Cabling Decision
IBM Cabling System Highlights
Communication Strategy for Growth

Issue 4, 1989 (G325-5004)
Requirements for Advanced Bus Architecture
Micro Channel System Configuration Considerations
Features and Benefits
Bus Masters and Applications
Overview of Extended Micro Channel Functions
New Micro Channel Features
SCB – An Architecture for Micro Channel Bus Masters
Design Alternatives with Micro Channel Systems
The IBM PS/2 Micro Channel SCSI Adapters
PS/2 Wizard Adapter
Experience in Bus Master Design
Bus Master Adapters from Independent Option Vendors
Bus Masters and OS/2
Micro Channel Issues in AIX PS/2
Information for Developers
Book Report: "The Winn Rosch Hardware Bible"

"All the new PS/1 models use an 80386 SX microprocessor running at 16 MHz. (page 4)

"The LaserPrinter 4029 Series features a control panel adjustment for Print Darkness, permitting user selection of stroke thickness. (page 7)

"OS/2 2.0 has all the required Windows code elements to support Windows applications without the need for Windows itself. (page 16)

"The DOS Settings feature gives the user more control over the consumption of system resources by a DOS application. (page 23)

"The architecture of OS/2 LAN Server 2.0 permits implementation on a co-processor. (page 31)

"Under 32-bit OS/2, memory is divided into fixed, non-movable 4 KB blocks called pages. (page 42)

"Private semaphores and unnamed pipes require less overhead, so you should use them whenever possible. (page 48)

"Given a class-of-service, APPN determines the importance of eight values defined for every link in the network. (page 71)

"APPC performance is significantly improved with Networking Services/2. (page 76)

"Secured/DOS provides for the use of passwords or tokens to assure user identification and authentication. (page 84)

G325-5014-00