Patricia Seybold Group

Editor-in-Chief
Michael A. Goulde

INSIDE

OPEN INFORMATION SYSTEMS

Guide to Unix and Other Open Systems

Vol. 7, No. 8 • ISSN: 1058-4161 • August 1992

Windows NT 3.1

Microsoft's Bid for Desktop Dominance

By Michael A. Goulde

IN BRIEF: Now that the technologists have nearly completed their work on Windows NT, Microsoft is revving up its marketing machine for a frontal assault on the competition. Its new multithreaded, multitasking, operating system has peer-to-peer networking built in, and it supports symmetric multiprocessing designs. Promising robustness, security, integrity, and compatibility, Windows NT will challenge Unix, NetWare, and OS/2 for dominance on both desktops and servers. In addition to serving as the high-end implementation of Microsoft's palmtop to data center strategy, Windows NT is the focal point for its open systems strategy. However, the definition of open systems that Microsoft prefers is not one that will win it many friends among proponents of POSIX standards. *Report begins on page 3.*

Open Systems and Systems Integration

Building Custom Solutions from Standard Parts

SYSTEM INTEGRATORS have always lived at the bleeding edge, blending immature technologies together to deliver working solutions to customers. For many, open systems are at the bleeding edge, and we see an increasing number of integrators publicizing their capabilities to deliver them.

When considering whether or not to use a system integrator, users often fear that the integrator will have to build customized pieces and "glue" in order to make all of the parts work together. Behind this fear are concerns that the finished system will be difficult or impossible to maintain and that a costly and constraining dependency on the integrator will develop. In the past, integrators didn't necessarily object to that dependency, since it meant that a steady revenue stream could be generated from follow-on enhancements.

The open systems movement is beginning to change this picture. To the extent that standard interfaces exist between various components of a new system, individual pieces can be mixed and matched to build customized solutions. The principle isn't really different from the way Leggo blocks work. These plastic building blocks adhere to an interface definition, and, although they come in specific shapes and sizes, they can be assembled into an infinite variety of toys. In a similar fashion, an infinite variety of applications can be built with open systems products that comply with standard interface definitions.

What does this mean for systems integrators? It could be viewed as a threat, since their specialty is to make irregularly shaped blocks fit together in a workable fashion. If open systems products comply with standard interfaces, then anybody should be able to assemble them. It is not that easy, however, as applications and the tools to build them become ever more complex. The role of the system integrators is now shifting more toward making what should work actually do so and less toward making the impossible possible. Complexity, not incompatibility, is becoming their bread and butter.

First, however, system integrators have to adopt open business practices. This means that they have to be willing to support open systems standards and the products that implement them, to build solutions around open systems interfaces, and to be equally willing to transfer the knowledge and expertise necessary for maintaining and upgrading these systems to their customers. In some sense, open systems integrators have to be willing to leave their customers in the position of having high-quality, maintainable systems and never having to use their services again. This new breed of open systems integrators is emerging, and demand for their services will continue to grow. Users really can't afford any other approach. ◐

OPEN INFORMATION SYSTEMS

Editor-in-Chief
Michael A. Goulde

MCI:
MGoulde
Internet:
mgoulde@mcimail.com

Publisher
PATRICIA B. SEYBOLD

Analysts and Editors
JUDITH R. DAVIS
ROSEMARY B. FOY
DAVID S. MARSHAK
RONNI T. MARSHAK
JOHN R. RYMER
ANDREW D. WOLFE, JR.

Art Director
LAURINDA P. O'CONNOR

Sales Director
PHYLLIS GUILIANO

Circulation Manager
DEBORAH A. HAY

Customer Service Manager
DONALD K. BAILLARGEON

Patricia Seybold Group
148 State Street, 7th Floor,
Boston, Massachusetts 02109

Telephone: (617) 742-5200 or
(800) 826-2424
Fax: (617) 742-1028
MCI: PSOCG
Internet: psocg@mcimail.com
TELEX: 6503122583

Open Information Systems (ISSN 0890-4685) is published monthly for $495 (US), $507 (Canada), and $519 (Foreign) per year by Patricia Seybold Group, 148 State Street, 7th Floor, Boston, MA 02109. Second-class postage permit at Boston, MA and additional mailing offices.

POSTMASTER: Send address changes to *Open Information Systems*, 148 State Street, 7th Floor, Boston, MA 02109.

Windows NT 3.1

Microsoft's Bid for Desktop Dominance

Microsoft's Path to the Future

The Windows NT Professional Developer's Conference held in early July was not merely a meeting about a new product; it was a coming-out party for the technology that will become the focal point of Microsoft's strategy for the next decade. That strategy is to have a single, scalable Windows architecture with different implementations for different platforms. If this strategy sounds familiar, that is because it is patterned closely after the "single VAX architecture—multiple implementations" strategy that brought Digital so much success in the 1980s. The architecture that Microsoft is betting on is the Win32 application programming interface (API) that will be implemented at the low end in Windows for DOS, in the midrange in Windows for Workgroups, and at the high end in Windows NT. Applications written to the Win32s API, a subset of the full API, are meant to be upwardly compatible across all platforms.

The high-end implementation, Windows NT, is a fully preemptive, multitasking, multithreaded, 32-bit operating system. It is contemporary in design, much like OSF/1 1.0 based on Carnegie Mellon's Mach 2.5, but not revolutionary. It has a microkernel, like Carnegie Mellon's newest version of Mach, 3.0, but many of its services run in privileged space rather than in user space. (Privileged space is where high priority operating system code runs, while user space is under control of the kernel.) The advantage of services in user space is that extending the system is easier. That can also be a great disadvantage when consistency and compatibility are an issue. However, when compared to OS/2 2.0 and Unix System V Release 4, Windows NT is highly modular, has implicit support for symmetric multiprocessing, and relies on dynamically linked libraries (DLLs) to implement a great deal of its functionality.

LAN Manager Takes on a New Role

NT could just as well stand for Network Technology. Unlike previous Microsoft operating systems, Windows NT is designed to provide complete networking capabilities without requiring any additional software. Theoretically, there is no limit to how many Windows NT machines can participate in a network. Instead of having to install a LAN Manager package on a server to enjoy file, print, and other network services, Windows NT machines get those services from one another. In fact, the new LAN Manager for the Windows NT package is just what its name implies: management capabilities for LANs over and above what is in Windows NT. LAN Manager offers services to help configure and manage large networks and network resources. It adds robustness to those services and additional security. LAN Manager for Windows NT should probably have a different name, but the name was retained to indicate the degree of compatibility that exists between old LAN Manager server-based networks and Windows NT networks with LAN Manager services. This compatibility exists in spite of the fact that LAN Manager for Windows NT shares no code with LAN Manager for OS/2. On the other hand, Windows for Workgroups, based on Windows 3.1 and interoperable with Windows NT, uses a lot of LAN Manager for OS/2 code.

Implementing the Windows Strategy

Implementing the Windows Strategy

Windows has been central to Bill Gates's vision ever since Steve Jobs showed him a Macintosh prototype. Gates's, however, has always been a software vision. While control over hardware design has allowed Apple to implement features that Windows hasn't been able to, like knowing when a floppy disk is inserted in a drive and ejecting floppies from a drive under software control, the Microsoft software strategy has isolated it from the ups and downs of the PC hardware business. In many ways, Windows NT further isolates Microsoft from underlying hardware, expanding the range of platforms for which it can provide system and application software.

The Goals behind the Design of Windows NT

Development of Windows NT began in earnest in 1988 as a follow-on to the portable OS/2 work that Microsoft had been doing with IBM. David Cutler, an engineer hired from Digital Equipment Corporation after the portable VMS project he was working on was canceled, convinced Microsoft that it needed to develop a much better design than portable OS/2 as it was shaping up at that time. This led to the company's setting key design objectives in several areas, including portability, security, compatibility, extensibility, robustness, support for symmetric multiprocessing, connectivity, and internationalization.

Portability from Intel to RISC. Portability across different processor architectures was seen as critical. The ancestor of Windows NT was Portable OS/2, designed in the days when IBM and Microsoft were close development partners. The original Portable OS/2 was primarily targeted as a port to IBM's Power Architecture. When Microsoft and IBM parted, the emphasis for the new operating system shifted to being more than a port—it had to be highly portable. At the time of Windows NT's conception, RISC processors were on a much steeper performance curve than Intel's CISC processors. Since Unix operating systems have always dominated RISC, Microsoft needed to have an operating system that could run on RISC at least as well as it ran on the Intel architecture. Early development for Windows NT was actually done on Intel's i860 RISC processor instead of Intel's x86 in order to prevent developers from falling prey to x86 instruction set contamination. Serious work on the i860 ended when it became clear that it would not emerge as a popular general purpose processor. Focus shifted to the MIPS R3000/R4000, in part because, if the R4000 had been available on schedule, it would have offered significant performance advantages over the next generation x86 processor and also because MIPS didn't compete with Microsoft neither in system software as Sun or IBM did, nor in object-oriented technology as did Hewlett-Packard. Current Windows NT development work includes ports to the Intel 386/486, MIPS R3000/R4000, and Digital's Alpha processor architectures.

Security at the C2 Level. NT is designed from the kernel up to qualify for Department of Defense C2 certification in its initial release. C2 specifies discretionary access controls on all potentially sharable objects through the use of access control lists (ACLs). This requirement is met in part through a new file system, NT File System (NTFS), which allows permissions to be assigned to files. This capability does not exist either in the DOS FAT file system or OS/2's HPFS. The kernel is designed to provide the pieces necessary for security at the B2 level, which specifies mandatory access controls. However, actual implementation of B2 is much further down the road since B2 would cause problems with functions like DDE and windowing. It is more likely that a third party would license and use the Windows NT kernel to build a B2 secure system than it is for Microsoft to do so—at least for the next few years.

Compatibility with Existing Applications. The ability for the new operating system to run DOS and existing Windows 3.1 applications unmodified was a key requirement. DOS on

Intel was simply a matter of using the virtual machine capability of the i80386 and above. On RISC architectures, the easiest way to support DOS was to license an emulator from Insignia Solutions, Ltd. (Wycombe, UK). Windows NT was also designed to support character mode 16-bit OS/2 applications, but only on Intel platforms, not on RISC. Microsoft claims that OS/2 is too dependent on features of the Intel architecture to make trying to support those applications on MIPS or Alpha worthwhile. Limiting OS/2 to Intel also happens to be a way for Microsoft to limit the appeal and spread of OS/2. It may also help channel some OS/2 development to Windows NT. Presentation Manager support for OS/2 applications is under development for 1993, but it is not planned for the first release. LAN Manager compatibility is ensured, and both FAT and HPFS file systems are brought forward with Windows NT.

Extensibility without Compromise. An important consideration was to allow extensions to the operating system without requiring the addition of privileged code. This characteristic allows Windows NT to support subsystems, like POSIX, OS/2, and DOS, without their conflicting with one another. This is in contrast to NetWare, in which all NetWare Loadable Modules (NLMs) run in privileged mode. Windows NT does support certain privileged, kernel-level extensions, however, for components like device drivers, installable file systems, and installable network redirectors. This allows the operating system to be enhanced without having to change the basic system. The mechanisms that support these extensions prevent compromising the integrity of the system.

A Robust Environment. DOS and Windows have been notorious for their lack of robustness, often due to applications overwriting the address spaces of one another. To help correct this, every process that runs under Windows NT has its own separate address space. This helps prevent poorly behaved applications from crashing the system. In addition, per user quotas on system resources are used to help protect those resources from being monopolized. In addition, all APIs return an error status, and exception-handling has been improved by employing a structured approach.

Symmetric Multiprocessing for More Power. While uniprocessor designs have been sufficient throughout the PC era, symmetric multiprocessing (SMP) designs will become increasingly prevalent for high-end workstations as well as for servers. In recognition of this, SMP support is built into the basic architecture of Windows NT. The same release will support from 1 to 16 processors of like type in the same system. Each vendor of a multiprocessor machine will supply its own Hardware Abstraction Layer (HAL) for its system, which presents a consistent view of various hardware implementations to the operating system kernel. Microsoft will provide a number of HALs on the Windows NT distribution media, including the Compaq SystemPro (which is a master/slave multiprocessor design), NCR 3450 and 3550, the Wyse 7000i, and multiprocessor machines from ALR and ACER. The Windows NT model is fully symmetrical, with uniform memory access. Performance scales in a linear fashion with additional processors, and the kernel executes on any CPU. Contention for CPU resources is handled by a fine-grained locking mechanism.

Connectivity without NetWare. Windows NT represents Microsoft's big chance to neutralize the dominance of Novell's NetWare in the PC networking market. The need for network operating systems (NOS) came about because DOS is network oblivious. NetWare succeeded in gaining market dominance primarily because Microsoft did not directly market its networking technology, MS-NET, but relied on licensed OEMs like 3Com and Digital to promulgate it. In the meantime, Novell built a dealer channel that overwhelmed its competition and established NetWare as a de facto standard NOS. Now, Windows NT virtually eliminates the requirement for using a NOS by integrating network functionality directly in the operating system while providing additional functional and management enhancements with LAN Manager for Windows NT. In fact, it is in the area of networking interoperability that Windows NT is most closely aligned with open systems standards. In

addition to running its NetBIOS protocol over IBM's NetBEUI transport, Windows NT also supports TCP/IP, includes TCP/IP utilities and an OSF DCE-compatible remote procedure call (RPC), and supports SNMP for network management. For interoperability with proprietary systems, IBM 3270 capabilities will be supplied by third parties. NetWare client support, announced by Novell, will still allow Windows NT machines to access NetWare servers.

Internationalization. Worldwide markets are important to Microsoft, third-party developers, and their customers. Internationalization support makes Windows NT easy to localize. All strings in the Executive are in Unicode, and the Win32 API supports Unicode strings as well as ANSI code pages. All system resources are Unicode strings, and there is C run-time support for Unicode strings as well as for ANSI code pages.

Windows NT Architecture

In the architecture of Windows NT, the NT Executive sits above the kernel layer but still within privileged space. In that regard, it differs significantly from Carnegie Mellon's Mach 3.0 kernel architecture, which places most of the functionality found in the NT Executive in the Mach user space. The more code that resides in user space, the easier it is to extend the system and add value to it. In Windows NT, there are formal interfaces between every component of the Executive, facilitating enhancement and extension of the operating system—but on Microsoft's terms. This kind of modularity is lacking in most of Windows NT's competitors.

The components of the Executive include object management, memory management, the I/O subsystem, interprocess communication, process structure, and security.

Everything is an Object

The object management component of the Executive exports the functions necessary to support the APIs that are used to build user (application) visible objects, including processes, threads, events, and files. Object handles are assigned on a per process basis, and access validation for objects is provided by access control lists associated with each object. Before any action takes place on an object, an access check is made.

The Unicode Standard

Early in 1988, a group with extensive experience in multilingual computing, agreeing that there was no encoding methodology that possessed the elegance and simplicity of ASCII, established the Unicode character encoding as a fixed-width, 16-bit encoding. This system was intended to provide a sufficient number of unique codes for all the world's scripts and commonly used technical symbols while promoting efficient and flexible system design.

In January 1991, the Unicode Consortium was incorporated as Unicode, Incorporated, a nonprofit organization chartered with maintaining and promoting the Unicode standard worldwide. The consortium, working in conjunction with the International Standards Organization (ISO), went on to merge Version 1.0 of the Unicode Standard with the 32-bit, character-encoding standard that ISO was working on, ISO DIS 10646. A proposal for the merger of the two standards was approved by ISO in 1991, and the final standard was accepted in July 1992.

The framework for the actual encoding in ISO 10646 provides for the Unicode standard as a 2-byte subset of a canonical 4-byte international standard character encoding. Version 1.0 of the Unicode standard is being revised in Version 1.1 to reflect the merger with ISO 10646.

Windows NT Architecture

Memory Management

Windows NT provides a 32-bit memory environment, in which the 4GB that are addressable are divided equally between 2GB available for user processes and 2GB available for kernel processes. Memory is demand-paged, and multiple paging files are provided, which allows faster paging, especially when the striped disk option is used. It is the memory management subsystem that exports memory-mapped files that applications can read for additional speed.

I/O Subsystem Supports Large Files and Disks

The file address space in Windows NT is 64 bits long, which means that it can manage over 18,000 terabytes of disk storage. It provides services for drivers and the file system through a layered model with clearly defined and specified interfaces. Those interfaces related to device driver development are published in the Microsoft Device Driver kit. Others may be made available in the future. The operating system uses an integrated single global cache, which is integrated with memory management. This eliminates the need to set cache size, since the operating systems takes care of that.

NT Kernel Architecture

The kernel is the lowest layer in the Windows NT Executive. It provides processor architecture-specific support, but not platform-specific support. It exports the APIs that the rest of the NT Executive and everything else in the operating system are built on. (See Illustration 1.) However, those APIs are not exposed to developers, restricting access to kernel services on the one hand while protecting system consistency on the other. The kernel implements scheduling and context-switching, multiprocessor synchronization, exception- and interrupt-handling, and low-level hardware functions. About 80 percent of the kernel code is machine-independent and 20 percent, machine-dependent. Of the portion that is machine-dependent, about 10 percent is written in assembler, primarily portions that deal with processor privilege operations, which are different for each processor supported.

Abstractions Ensure Compatibility

The combination of the kernel, device drivers, and HAL constitute NT's interface to the underlying hardware functionality. HAL is platform-specific code that isolates the NT Executive from platform-specific implementations of functions like I/O devices, DMA control, bus mapping, clocks/timers, cache control, interrupt dispatches, and access to the privileged architecture. This means that hardware designers will have much greater freedom to innovate without having to worry about hardware compatibility. While this may create a new market for Windows NT machines, it might make DOS and possibly even Unix support of these innovative designs more difficult. As long as the hardware is properly abstracted to Windows NT, compatibility is not an issue.

Preemption, Scheduling and Synchronization

The kernel architecture is non-pageable (can't take page faults), non-preemptable (not context switchable), but interruptible, e.g., when servicing interrupts. The Executive itself is multithreaded. It exports abstractions in the form of dispatcher objects and control objects. Dispatcher objects control scheduling and synchronization. They have "signal" state and are waitable. ("Signal state" means that they wait until satisfied. "Not signaled" means that they wait until signaled.) The types of dispatch objects include threads, mutual exclusion, events, semaphores, times, and event pairs. Kernel objects, like naming and security, are encapsulated in executive objects before they are exported to user space.

Control objects provide Executive and device driver control. They are passive and are used by the Executive and by device drivers. They have no "signal" state and are not waitable. They include processes, interrupts, device queues, profiles, asynchronous procedure calls, and deferred procedure calls.

Multithreading Operations

Windows NT is multithreaded, and threads are a central part of the system. Threads are execution agents which register a context. They are always associated with a process and run in that process's' address space. They may have processor affinity, which is a subset of

processors in a multiprocessing environment on which the thread can run. Affinity is system control. It is not under programmatic control because the API controlling it is not exposed.

Windows NT System Structure

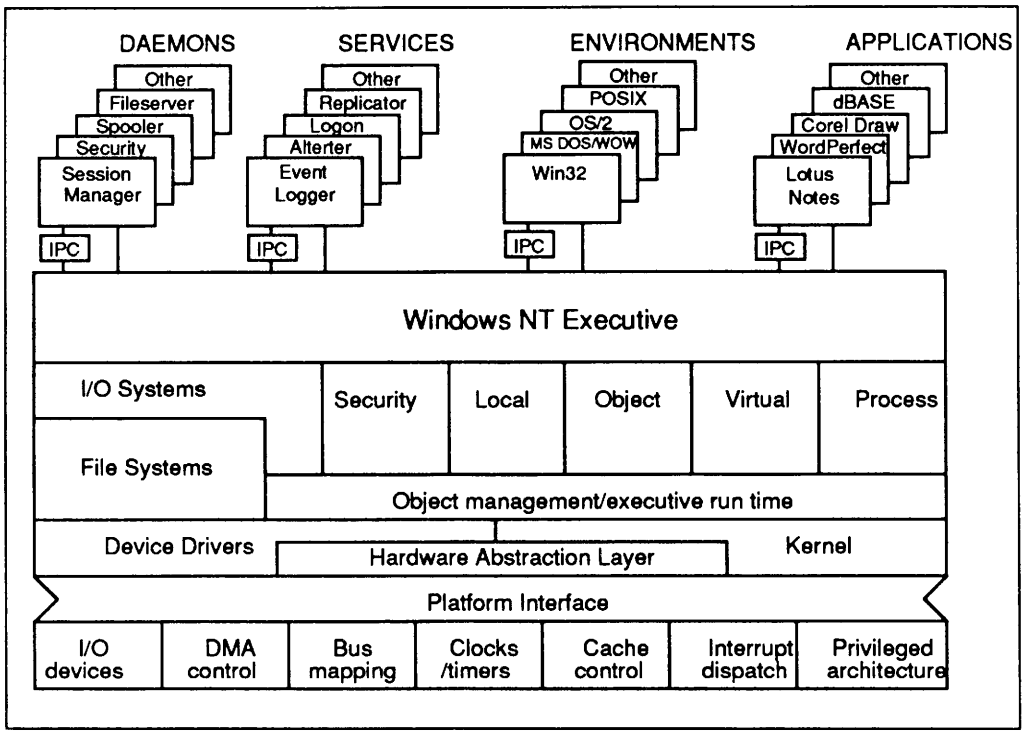

Illustration 1. The Windows NT system has a privileged layer, the NT Executive, services built on the interfaces provided by the Executive, and protected environments, such as Win32, and POSIX. At the lowest level of the Executive, the Device Driver Interface, HAL, and the Kernel interface to the actual hardware. Privileged mode services, such as I/O, Security, Object, and IPC services use services from underlying layers and provide interfaces that user-level services can access.

HAL Layer: The ACE Legacy

HAL isolates Windows NT from specific hardware features, allowing Microsoft to supply a shrinkwrapped operating system on a wide range of systems. It provides a uniform model for device drivers, allowing the same driver to work on systems with different I/O subsystems. Most importantly, it permits vendors to make system-specific optimizations without concern for compatibility issues. Hardware dependencies are isolated in specific components of the operating system, including the HAL, the kernel, and the management of virtual memory.

HALs will be supplied by Microsoft for all PC-compatible computers and some SMP machines. Otherwise, computer manufacturers will write their own HALs. Writing the complete HAL for the Wyse 7000i, a three-processor machine, took just a few weeks, according to the developer. HALs are small independent routines. Simple, fast linkage is supplied as each is loaded and bound by the operating system loader.

Among the features that are abstracted in the HAL are the system bus, or buses. This means that vendors no longer have to be limited to ISA, EISA, or MicroChannel, at least from a software perspective. The direct memory access controller (DMA) is abstracted, eliminating

the need for the PC standard 8237 chip. Similarly, the interrupt controller is abstracted, eliminating the 8259 chip that currently provides that function in PCs. System timers (8254), system-specific cache coherency, cache-flushing (even systems without DMA coherence are supported), and system-specific SMP support are all handled in the HAL.

SMP: The Economical Route to High Performance

SMP Support Included—No Extra Charge

Symmetric multiprocessing (SMP) has become an accepted design center for yielding optimum price/performance (See *Unix in the Office*, Vol. 7, No. 1). Microsoft designed Windows NT from the ground up to support symmetric multiprocessing designs. The first release will support up to 16 processors. Windows NT runs a single copy of all its code—it is just a question of which processor runs any particular process at any given moment in time. There is also just one copy of the operating system data structures residing in memory. Not just processes are location-independent; any thread, including device drivers, can also run on any processor. Mechanisms in the kernel, such as spin locks, provide for simple, automatic load-balancing, thereby speeding up both the operating system and applications.

There are two kernels provided with the NT distribution, a uniprocessor kernel and a multiprocessor kernel. The uniprocessor kernel doesn't contain the functions that support multiple processors, such as the scheduler. Otherwise, the rest of the Executive is the same. If the operating system is being installed on a multiple processor system, the proper kernel is picked up during the installation. Applications and device drivers are not affected by the type of kernel installed.

Device drivers, if written according to Microsoft's guidelines, are MP ready. The key here is that the driver may be running on any processor in the system at any given time and can't be assured of running on a particular processor.

The Kernel Is Not Fault-Tolerant

Unlike some Unix implementations, Windows NT is not tolerant of processor failures. If a processor dies, so does NT. A machine like the Wyse 7000i running Wyse's Unix can keep running if one of its three processors fails. However, that same machine running Windows NT would fail completely. While processors don't often fail, this is a consideration in some applications.

Optimum Designs Yield Optimum Performance

Several design parameters characterize multiprocessor designs that benefit most from additional processors under Windows NT. Each processor must have identical views of physical memory, each processor must have identical access to devices; hardware must keep caches coherent, and hardware must provide atomic references. In addition, each processor must be able to interrupt any other processor, must support a periodic timer interrupt, and must support a profile interrupt. One restriction in designs is that processor architectures cannot be mixed. This means that 386 and 486 processors can't be mixed either, since there are several 486-specific instructions.

Going beyond the minimum requirements, better multiprocessor designs will have write-back secondary caches, fast internal buses, a fast path to memory, and will implement Scatter/Gather DMA in hardware. The latter function is carried out in the HAL layer in PC designs, which introduces additional overhead and impacts secondary cache designs.

There are several multiprocessor designs on the market today that do not meet these criteria and are not truly symmetrical in their designs. However, there are many others that do. Windows NT will drive more vendors to provide SMP machines once its acceptance is determined.

Win32 API: The Next de Facto Standard?

Although DOS, OS/2, and POSIX applications are supported in Windows NT, Win32 is the strategic API. It is a 32-bit programming interface that provides all of the functionality of the Windows 3.1 16-bit API, but it has been extended to 32 bits while adding additional functionality, such as multithreading. Win32 can only run on 80386 and above machines or comparable RISC architectures that support 32-bit addressing and paged virtual memory. Windows programmers will find the Win32 API easier to program than the 16-bit Windows API since they no longer have to deal with 64KB chunks of segmented memory. Programmers will, however, have to discard their custom virtual memory schemes, because virtual memory is handled by the operating system.

Win32 supports a higher degree of system integrity because it places different memory objects in different pages of memory and allows an application to control access permissions to memory objects. It also allows an application to map files into its address space (memory mapping). This allows data within the file to be accessed directly from its address space instead of requiring more complicated and less efficient I/O functions.

Win32s Provides a Path to Windows NT

The migration path for applications from Windows 3.1 to Windows NT will be through the Win32s API, which is a fully compatible subset of Win32. Developers can hedge against Windows NT adoption by revising their 16-bit Windows applications to Win32s. Revised applications will then run on both Windows 3.1 and Windows NT. They will not, however, run on the current version of OS/2 2.0, and, unless IBM explicitly supports Win32s, they never will.

Win32s provides 32-bit equivalents for 16-bit Windows 3.1 APIs. It is a subset because there are APIs in the full Win32 API that have no equivalent in Windows 3.1. Programs written for Win32s will run on Windows 3.1 but will require support from additional libraries and from a DLL. (See Illustration 2.) However, they will run as native applications on Windows NT without changes or additional support. Win32s supports all Windows 3.1 windowing and Graphic Device Interface (GDI) functions. OLE 1.0, DDE and DDEML, TrueType, and common dialogues are also included in Win32s.

Conceptual Architecture of Win32s

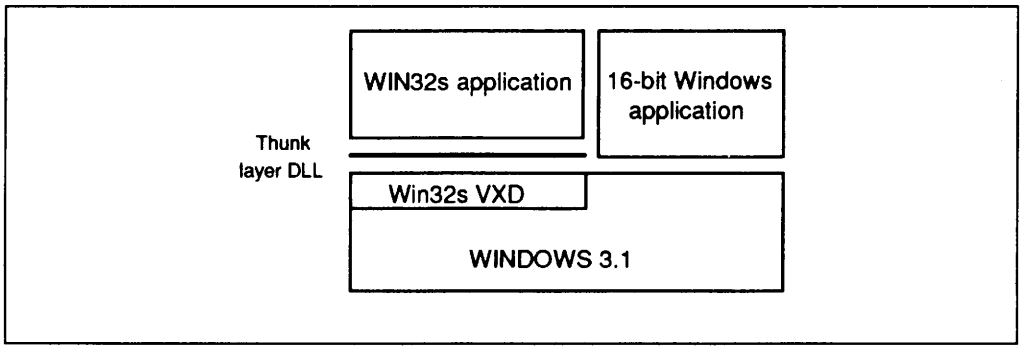

Illustration 2. A Win32s application has its 32-bit calls transferred to 16-bit calls by the thunk layer DLL. Those calls then use the VXD libraries, which map to Windows 3.1. This all happens concurrently and invisibly to regular 16-bit Windows applications.

Flat Memory Model. Win32s provides flat 32-bit addressing for Windows applications on Windows 3.1. It requires a 386SX with 4MB of memory running in enhanced mode. Applications are binary compatible with Windows NT and share the same EXE format, the NT portable executable format. Win32s applications are source-compatible with Windows NT on x86, MIPS R4000, and DEC Alpha. It bears repeating that programs written to the

Win32s API are real Win32 applications but don't use certain advanced features like multithreading. The difficulty involved in converting a 16-bit Windows application to Win32s will vary depending on the design and coding of the application. In addition to the different memory model, the semantics of some of the APIs are different. In spite of porting issues that may arise, we expect that native Windows NT applications will be in ample supply in a short period of time, although many of these will be written to Win32s.

Win32s Slated for Fall Availability. The first Beta of Win32s will be released at the end of summer 1992 and will be shipped with the September update of the Win32 SDK. The September release will not have network support, NetBIOS, named pipes, console, Unicode, Pen, Multimedia, or RPC support. This support will be included in the final release, which will coincide with the retail release of Windows NT at the end of 1992. As MAPI and ODBC evolve and are finalized, support for those interfaces will be included. However, a subset of MAPI will probably be in the initial release. There are some features that are not planned for the first release that many developers would be very interested in, including memory-mapped files and support for mixing 16-bit and 32-bit DLLs.

Win32s will also be provided by various Windows 3.1 DOS toolkits, including those from Microsoft and Borland. No special tools are required to build Win32-compatible applications for Windows 3.1. There are neither a special build process nor unique link libraries required for building Win32s applications. Both MS-DOS and Windows NT will host Win32s development. The difference is that the Win32 SDK will provide Windows NT platform support, while Microsoft Languages and those from others will provide the MS-DOS/Windows 3.1 development platforms.

Getting Win32s on Windows 3.1. In order to run Win32s applications, users will have to install approximately 500KB of code contained in the Win32s VXD library and the Win32s DLL to extend Windows 3.1 to support 32-bit operations. The actual working set of code is about 200KB, in addition to the memory already required for Windows 3.1. The library and DLL will be shipped by ISVs with their 32-bit applications. A setup program will determine if they are already installed and will install them if necessary. Future versions of Windows will contain these libraries and DLL.

An Important Win32s Caveat. Win32s applications cannot load 16-bit DLLs. Developers will have to either convert all DLLs to 32 bits or use a client/server structure within the application. Under this model, the 32-bit program would use a 16-bit server application to link with a 16-bit DLL. We believe this limitation will spur developers to move to 32-bit code as fast as they can.

Extensibility without Changing the System

It is possible for NT to be extended with or without privileged code. The operating system subsystems, like OS/2 and POSIX, are in non-privileged space. Extensions in privileged space are under Microsoft's control and include functions like device drivers, installable file systems, and installable network redirectors. The latter are thought of as kernel-level extensions that enhance the system without changing it. Microsoft will release APIs for privileged operations gradually, as it learns to understand where it can do so without jeopardizing control over system integrity and compatibility.

High-Level Abstractions

System services are exposed to applications as high-level abstractions in the form of APIs. There are many services in the privileged space that are not exposed by a published API which third parties could conceivably use to add value on top of Windows NT. For instance,

a third party could port another file system or another operating system personality if the necessary interfaces were available. In this way, Microsoft controls the services available to applications, ensuring consistency, but, as a result, limits the opportunity for a third party system software industry to grow up around NT the way it has around Unix.

NTFS Provides Advanced Features

Windows NT is able to support multiple active registered file systems. They are loadable just like any other driver. Support is provided for automatic volume mounting and verification. In addition to FAT, HPFS, and NTFS, Windows NT also supports CD-ROM file systems, named pipe file systems, mailslot file systems, and the LAN Manager redirector. All file systems share a single global cache manager, and cache can expand to include up to all available physical memory.

NTFS Is Key to Many NT Features

NTFS supports large disks and files, independent of hardware sector size. It uses 64 bits for file sizes and offsets as opposed to the 32 bits of HPFS. File names are Unicode-based, including volume and all other names. Unicode file names are stored on the disk. It supports long file names while still autogenerating standard DOS FAT 8.3 names in instances when MS-DOS cannot uniquely express an NTFS name. NTFS itself has a 255-character limit on file names. The MS-DOS name is a fully functional alias that is stored in the same directory structure index with the NTFS name.

File System Structure

Each file in the NTFS has a set of attributes, including its type, its data, and an optional name. Each file is represented as a record in the master file table. The file record contains standard information, such as the time stamp, link count, file name, volume version, volume name, and volume information. Additional attribute types include multiple data attributes, directory information, index allocation, extended attribute information, and security descriptor information. Small files may contain their data right in their file record, while large files contain pointers to the location of the data which are recorded as virtual cluster numbers. Attributes may be resident in the file record or nonresident, in which case NTFS allocates separate extents in the volume to contain them.

NTFS Addresses Reliability Requirements

To address concerns about data integrity, NTFS has been designed to be fully recoverable, to remove fatal single-sector failures, to hot-fix bad clusters, and to coordinate with a fault-tolerant driver that will be supplied.

Additional Functionality

NTFS has been designed to support multiple file servers, including NFS, AppleShare, and others. It is designed to implement a typical corporate security model in the form of access control lists. It also supports multiple concurrent data STREAMS, has an extensible design, and will support storage quotas and the collection of accounting information.

Recoverability Features

NTFS is a recoverable file system which provides some of the advantages of both a Careful Write and a Lazy Write policy. Volume consistency is guaranteed across crashes via transaction-logging and recovery techniques. The advantage is that performance benefits of Lazy Writing are maintained with very rapid crash recovery. The disadvantage is the amount of overhead incurred for volume update operations. However, Microsoft estimates that this overhead will be less than 10 percent.

NTFS logs each modifying Windows NT I/O request, treating it as an atomic transaction. The transaction is committed on success and aborted on error. Redo/undo information is logged for all NFS metadata updates, and the undo is used for error recovery. It uses a Lazy Commit policy and treats all requests as either complete or not started at all. There are periodic log file checkpoints that monitor the progress of cache manager lazy writes and free up log file space. This is important to avoid Log File Full conditions, even though those only result in retries with the message "Abort, Reque, Flush."

NTFS Provides Advanced Features

On restart after a crash, three passes are made, the first to determine what needs to be updated, the second to update the status of cache to the state where it existed at the time of the crash, and the final pass to abort incomplete transactions.

Data Protection

NT provides support for tape backup, disk management, fault tolerance, and uninterrupted power supplies. Tape backup APIs and applications support all three NT files systems, FAT, HPFS, and NTFS. A new Microsoft Logical Tape Format (MLTF) is used, which is being made available to the industry as another Microsoft standard in order to promote interchange among vendors. MLTF provides on-tape catalogues of files, making restoration faster and easier. The tape APIs will make it easier for developers to write additional backup applications.

Span Sets = Logical Volumes

Disk management features include span sets, usually known as volume sets; stripe sets; stripe sets with parity; mirror sets; and "sticky" drive letters. Stripe sets with parity and mirror sets are only included in LAN Manager for NT and not in the standard NT configuration. Sticky drive letters are drive designations that stay with the volume when other volumes are added to the system.

Span sets allow a partition to span several physical drives. A volume set can contain up to eight unequally sized partitions across drives. Each span set is exposed to users and applications as a single drive letter. They have only a marginal impact on performance and are not bootable.

Stripe Sets for Performance

Stripe sets can be configured with or without parity. Without parity, between two and eight equally sized partitions can be configured both across disks and/or across controllers. Striping without parity can lead to improvements in read performance proportional to the number of drive spindles in the set with no degradation on writes. Stripe sets are not bootable.

LAN Manager for NT adds RAID 5 (stripe sets with parity) support using between three and eight partitions with parity data spread across all of the partitions. RAID 5 can be configured both across drives and across controllers. This means that a stripe set can consist of a combination of SCSI, ESDI, and IDE drives. Read performance is the same as striping without parity, but there is a significant penalty on writes. RAID 5 has a cost advantage over mirror sets, however, while providing a high level of data integrity.

Mirror Sets Provide Safety

Using the enhancements offered by LAN Manager for Windows NT, any two partitions can be mirrored with mirror sets. A mirror set has no primary or secondary member—both are equal. Mirror sets can provide up to a two times performance improvement when reading under heavy I/O load, particularly when small bits of data are being transferred. Microsoft estimates less than a three percent overhead for writes.

Mirror sets are bootable, and mirroring can be configured both across drives as well as across controllers for duplexing.

Fault-Tolerance Driver

The fault-tolerance driver, also provided with LAN Manager for Windows NT, only breaks its relationships with a drive on complete device failures. It loads at boot time and attaches all hard-drive partitions. Fault-tolerance information is maintained in the system registry as it intercepts all disk I/O and writes it to the log.

Networking Inherent in the Design

Microsoft has virtually eliminated the need for a network operating system (NOS) with Windows NT. Although this doesn't mean that Novell customers will abandon NetWare, it does mean that a separate NOS, such as NetWare, will become less important as Windows for Workgroups (which has basic network file and print services built in), Windows NT adoption ramps, and these peer-networked Windows machines become more prevalent on corporate networks.

Networking capabilities are an integral part of the NT design. The base I/O system includes features needed by remote file systems and servers. It supports installable file systems and provides a library of common file system functions and cache manager interfaces. It provides security through access control lists. The architecture of Windows NT is open to additional client-side providers, including third-party redirectors or requesters and transport providers. This is accomplished through the transport device interface (TDI) and STREAMS, and transparent support for network adapter drivers provided by NDIS.

Integrated LAN Manager for Peers and Servers

A compatible superset of LAN Manager 2.x functionality is provided for in Windows NT, even though the code implementing that functionality is all new. Every Windows NT workstation has server and client capabilities. The list of integrated features that support or facilitate networking includes:

- SMP support
- Disk fault-tolerance support
- TCP/IP client utilities, including ftp, telnet, rsh, rexec, rcp
- An extendable SNMP agent
- A fast, local area NetBEUI and TCP/IP transports
- NDIS 3.0 network interface card drivers
- LAN Manager services, including file replication, alerter, etc.

Several new administrative tools are provided, many in basic Windows NT, and others in LAN Manager for Windows NT. These include a server manager, user account manager, network control panel, security editor, event/audit log viewer, and a user profile editor. All of these have a graphical interface, addressing one of the most common complaints about LAN Manager and, for that matter, NetWare.

Transport Architecture

The TDI is usable from both the kernel mode and from the user mode. It provides a common transport interface for sockets and NetBIOS libraries to offer transparent access to NetBEUI, XNS, TCP/IP, and DECnet transports. Windows NT provides a System V STREAMS-compatible environment for protocol stacks through a kernel DLL. STREAMS-based stacks are implemented as loadable drivers; therefore, other transports may be furnished by third parties. Applications may be written to the WinSock API or to the NetBIOS API.

Remote File Systems

Remote file systems appear to Windows NT applications and users as local file systems. Because of the installable file system architecture, multiple local and remote file systems can coexist concurrently. The Windows NT object-naming architecture allows the remote file systems to be accessed directly without the physical path being specified. The multiple provider router (MPR) provides a WinNet API for file manager services and a Provider Interface for access by foreign file systems such as those provided by NetWare, VINES, or LAN Manager. The provider is a DLL that implements WinNet functions. The necessary DLLs for accessing remote file systems are recorded in the registry database and are accessible to any application.

Transport Architecture

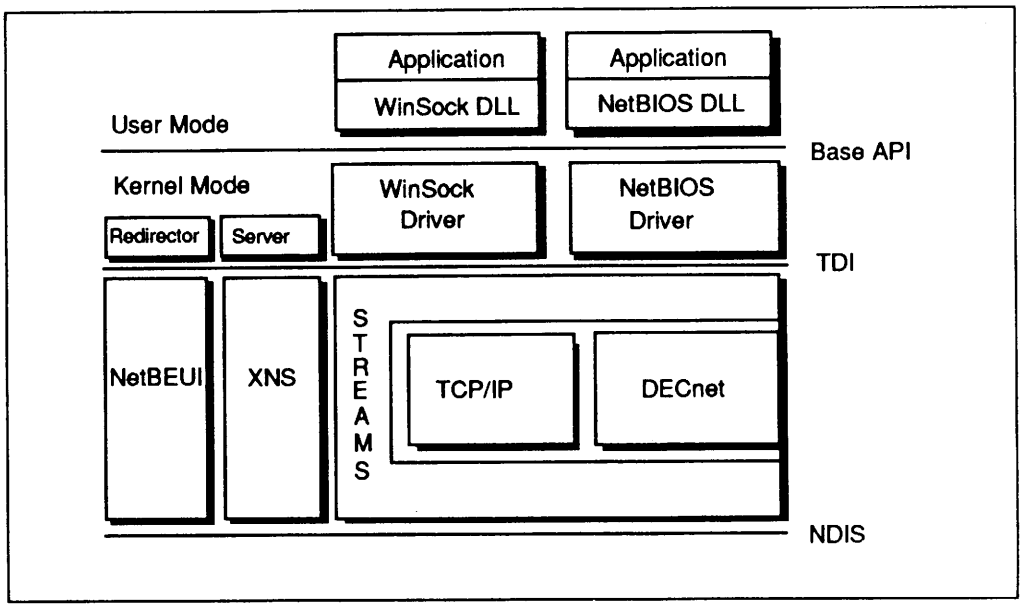

Illustration 3. The Transport Driver Interface (TDI) provides a common transport interface for Sockets and NetBIOS libraries to provide transparent access to NetBEUI, XNS, TCP/IP, and DECnet transports. Other transports may be provided by third parties. The STREAMS interface is compatible with that defined in SVR4 and is provided by a kernel DLL.

Windows NT continues support for the Universal Naming Convention (UNC). UNC is a standard for ensuring consistent and unique naming of resources across a network. A Multiple UNC Provider (MUP) driver, which controls the various providers, is loaded during system boot. Each provider is implemented as a remote file system driver, registering itself to the system by using control functions issued to the MUP. Provider preference information is stored in the registry database in the same location as the MPR, again making it available to applications.

UNC opens are directed to the MUP. Win32 functions translate the UNC name to the MUP name. MUP then offers the name to all providers simultaneously by issuing "fsctl" I/O requests to each. The provider then determines whether it owns the name and completes the I/O request with success or failure. MUP tracks the higher-preference provider that claims the name. Future requests to open a file with the same prefix are then routed directly to that provider. Name ownership eventually times out if it is not accessed for a configurable period of time. MUP is involved only in path-based operations, not handle-based operations.

Developing Distributed Applications with RPCs

An important benefit of Windows NT networking capabilities will be the development of distributed applications. Microsoft is providing a remote procedure call (RPC) mechanism to support this development. The Microsoft RPC is a network-independent interprocess call for heterogeneous distributed applications. It is interoperable with the OSF DCE RPC, using a portable interface definition language (IDL). An IDL compiler is provided, assuring interoperability. Its API is equivalent to that of the OSF DCE RPC, although it is not the same OSF code. It uses the familiar procedure call model and simplifies writing distributed applications. The RPC provides transport, naming, and security for Windows Open Services Architecture (WOSA) loadable service providers, using either named pipes, NetBIOS, or TCP/IP transports.

The RPC will support DOS and Windows machines as clients but not as servers. This means that they can initiate but cannot service a call. The required run-times for DOS and

Networking Inherent in the Design

Windows will eventually be packaged with those two products. In the interim, they are included in the Software Development Kit and are freely distributable by ISVs. Windows NT will provide run-time support in the form of a DLL for both client and server applications which will be included in the final package.

Network Management Base on SNMP and NetView

Supports MIB1 and LAN Manager

One of Microsoft's design goals was to make Windows NT manageable in enterprise management systems. To accomplish this, the company focused on de facto standards, SNMP and IBM's NetView. Windows NT has an extendable agent Management Information Base (MIB) interface with a DLL API optimized for the Windows NT environment. This API hides the complexity of ASN.1 and SNMP from the programmer. A Messenger API is provided that serves as a high-level, easy-to-use SNMP API. Windows NT supports both MIB1 and the LAN Manager MIB. Microsoft has committed to support other MIBs as they become standard, and the company also intends to track work being done in SNMP security. Third parties can extend Microsoft's core SNMP product through the documented agent extension DLL API that will be published and available for review.

NetView Requires SNA Services

Microsoft will look to third parties (Digital Communications Associates of Alpharetta, Georgia, is an obvious candidate) to provide SNA services for Windows NT. Support for NetView services will depend on those underlying SNA services. Support will be provided for user-defined generic NetView alerts for Windows NT events. This can be accomplished without programming by the customer.

Support will also be provided for the NetView Run command. Windows NT service interprets the content of the Run command packet, routing service requests to the appropriate DLL.

Hermes Will Provide System Management

Hermes is the code name for a set of system management tools for Microsoft's system products that will be released in 1993. Hermes provides improved manageability for Microsoft computing environments. Its open architecture will provide a number of significant areas for third-party enhancement.

Functionality Targets for Hermes

Hermes is being designed to manage hardware and software inventory, software distribution and installation, networked applications, Hermes jobs, and the system itself. All Microsoft systems platforms are being targeted, including Windows NT, LAN Manager, MS-DOS, and Windows 3.1. A key component of Hermes will be ease of use—all functionality will be available to the user or administrator from an integrated, user-friendly GUI.

Hermes will support the distribution and installation of new software to any Windows system anywhere on the network, track the hardware and software in use on the network, manage networked applications, and meter applications running from Windows NT servers. The administrator will be able to configure for central or remote operation.

Designed to Be Scalable

The design of Hermes will make it possible to add network nodes without measurably adding to administrative overhead. It is being developed in response to requirements fed to Microsoft from its largest corporate customers. Among those requirements are for LAN administrators to be able to view and manage the entire network, no matter how large, from a single point.

Another Microsoft Open Standard

Microsoft will publish APIs that will allow Hermes to interoperate with other management systems (including IBM's SystemView, OSF's Distributed Management Environment—

Hermes Will Provide System Management

DME—and possibly Tivoli's WiZdom). It will support standard network transports and will provide senders for standard LAN transports as well as for Remote Access Server (RAS) and LU6.2. Each node in a Hermes-managed network will run code to determine the hardware and software inventory. A binary result file is written to a collection server. Changes for all sites are stored in a central SQL database which an administrator can query and use to generate reports.

Windows NT LAN Manager technology will be the basis for Hermes, and the NTFS will be used to enforce security. Software distribution management capabilities from the server will also be Windows NT-based.

Interoperability Is Again the Focus

Hermes provides another illustration of Microsoft's strategy of interoperating with other de facto standard systems rather than natively adopting their technology. In order to succeed with this approach, Microsoft will have to go the extra mile by making a significant investment in testing both standards compliance and interoperability with other products to ensure that its implementations truly interoperate.

POSIX Support May Not Appeal to Unix Developers

The POSIX support provided on Windows NT is limited exclusively to the 1003.1 standard. An X server from eXcursions will be made available for Windows NT from Digital and others. X applications do not have access to OLE or DDE, and Win32 applications cannot make POSIX calls. Features such as case-sensitive naming, additional time stamps, and hard links are supported for POSIX compatibility, and symbolic links and sparse files will also be supported in the future. Symbolic link support will be added when the POSIX specification is completed. While access to advanced Windows NT features from the POSIX subsystem is limited, it does include security and control of process threads. It is not possible for a POSIX application to use the graphical API, and screen control or access is limited.

Access to Win32 functionality is possible through the IPC mechanisms and named pipes, allowing POSIX applications to exchange data with Win32 applications. However, POSIX doesn't access the RPC mechanism or sockets, which makes it difficult to develop POSIX applications that can interoperate with other POSIX systems. Sockets has to be accessed through the Windows NT console interface. No support for SLIP or PPD is provided in NT's TCP/IP.

Why Bother with POSIX?

It is not Microsoft's intention to provide a full XPG/3-compliant system with Windows NT nor to encourage any serious development using the POSIX interface. The company makes it clear that POSIX 1003.1 is there because it has to be there in order to qualify for federal government bids. Microsoft is encouraging ISVs with character cell Unix applications to port them first to Windows NT's console interface. This eliminates dealing with the complexity of GUI development, but does give access to the Win32 API. Character applications running in the console can call graphical features, such as file and print management.

Windows NT Follows Microsoft's Definition of Open Systems

There is much debate about Microsoft's definition of open systems and its open process. How are they alike, and how are they different? Like the OSF, Microsoft has an open process. Both organizations circulate draft specifications and invite comment from the industry before finalizing them. Both publish their API specifications. Why, then, is the OSF process generally considered to be open while Microsoft's is not? Two factors are operating. First, Microsoft views many open systems standards as being closely aligned with Unix and, therefore, inconsistent with its technical direction and market strategy. The OSF,

while seeking to become unencumbered by USL licensing, still embraces open systems standards based on Unix in its products when those standards are available or under development. Second, the OSF distributes source code in addition to API specifications. Microsoft does not generally license or distribute any source code. In rare instances, source code has been licensed so that a third party could build a complementary product that needed hooks into the Microsoft operating system, and the product was not of interest to Microsoft.

Microsoft is much more interested in standards that support interoperability than in becoming Unix compatible. Whether it is TCP/IP, SNMP, the DCE RPC, sockets, or database access standards, if a standard supports bringing information from foreign systems to Windows users, Microsoft will be inclined to support it.

Will Mach 3.0 Challenge Windows NT?

Many believe that the state-of-the-art in microkernel design is represented by Mach 3.0, developed by Carnegie Mellon University. The Mach design has been optimized for system software support through:

- Integrated virtual memory management and interprocess communication
- Multiple threads of control
- Support for transparent system trap callout
- An object programming facility integrated with the Mach IPC mechanisms

Current ports of Mach 3.0 range from laptop computers to massively parallel systems such as the Intel Hypercube. The Mach 3.0 kernel source code does not require any prerequisite licenses from AT&T or Berkeley and can even be downloaded from CMU. Mach 3.0 isolates the Mach foundation from Unix interfaces, providing a more reliable and more easily maintained system. With Mach 3.0, a task doesn't have to be a Unix task; it is just an entity managed directly by the kernel. Because of this isolation of the kernel from system services, multiple operating system personalities can be supported. A Mach device interface has been introduced and made accessible to applications. The IPC has evolved, and its performance has increased; a new type of port has been created which allows transparent distribution of messages across nodes.

Mach Supports Multiple Personalities

In addition to the monolithic OSF/1 1.0 server, a number of other servers are under development to run on top of Mach 3.0. These include a BSD 4.3 server for Intel 386 platforms, a Mac emulator, and servers for SVR4, BSD 4.4, and SPRITE. The Mach 3.0 kernel has the ability to support several of these operating system personalities at the same time. However, like Windows NT, the different operating system servers do not interoperate on the same system. The environment can be extended to address this with the coupling between the environments, ranging from loose coupling, such as support of different types of utilities to communicate data between environments, to a more tightly coupled scheme that includes full sharing of resources and management of their concurrent access.

Who Will Bring Mach 3.0 to Market?

The biggest question surrounding Mach 3.0 is: What organization will provide commercialized implementations that can be sold to customers? The OSF does not have the resources necessary to do any more than coordinate the efforts of various researchers working on Mach 3.0 extensions and refinements. It is possible that some OSF member may assume a contractor role, building a commercial-quality Mach-based product and licensing it either back to OSF or to other vendors. Who might be interested? Any Unix-oriented company that is not particularly interested in doing business or continuing to do business with USL is a safe bet.

Summary and Conclusions: Will Windows NT Succeed?

It is likely that, for the first year after its release, Windows NT will be closely evaluated by customers and ISVs alike. Some may adopt it in its first release. Others may reject it out of hand. History has shown that when Microsoft commits to a product direction, it sticks with it for the long haul. Windows 1.0 shipped in 1985 but wasn't a major success until 1989. It is unlikely that the jury will be out on Windows NT for four years, but it is clear that Microsoft will put whatever resources are necessary into its refinement to make it acceptable to a significant number of customers.

Will Windows NT Succeed?

Unlike the situation that existed with DOS and Windows, vendors with large distribution organizations, like IBM and Hewlett-Packard, are not automatically falling in line to support Windows NT. Vendors who realized that supporting DOS was important even if it wasn't central to their strategy, such as Apple and Sun, aren't making any moves either. This represents a key challenge to Microsoft because it probably won't be able to sell Windows NT through the same channels that it sold DOS and Windows. Windows NT is a complex system and will require a level of support that dealers are unlikely to be able to provide. Microsoft needs to develop a distribution channel for Windows NT with a higher level of competency in supporting complex systems. This is not unlike the challenge Compaq faced with the SystemPro.

The agreement between Digital Equipment and Microsoft is certainly aimed at developing that channel (see *Unix in the Office*, Vol. 7, No. 5), but Digital cannot do the job alone. Novell is not going to be interested in distributing Windows NT. It is unlikely that IBM will throw its weight behind the product and equally unlikely that major Unix adherents, like HP and Sun, will, either. Microsoft faces a real challenge.

Applications Will Help

Windows NT will siphon off a lot of resources from developers who are willing to gamble on selling software to enough of the 10 to 15 million Windows users who will perceive Windows NT as a natural upgrade to create a sizable applications market. Many of these will be Unix developers who will see a market that is two orders of magnitude larger than they currently sell into. Until now, many of these developers have not been able to port their functionality over to Windows because of its limited capabilities. Although they are not likely to abandon their Unix development, it is possible that their OS/2 development could dry up completely.

Next month's *Open Information Systems* will address
Oracle Version 7
For reprint information on articles appearing in this issue,
please contact Donald Baillargeon at (617) 742-5200, extension 117.

Open Systems: Analysis, Issues, & Opinions

FOCUS: DEVELOPMENT TOOLS

Crossroad and SuperNOVA: Sharpening the Focus on Rapid Prototyping and Deployment

There is a gap between the vision of distributed object management and the reality of the kinds of distributed applications that can be built today. Attempting to bridge that gap are Crossroad from Crossroad Systems and SuperNOVA from Four Seasons Software. These products offer practical help to the application developer chartered to develop distributed applications within a standards-based framework.

Crossroad Offers Hope to the "Front-Line" Developer

Crossroad Systems (Boston, Massachusetts) recently introduced Release 1.5 of Crossroad, its framework for integrating Unix applications in networked environments. This release follows the initial introduction of the product by less than a year. Application developers, particularly in the technology-aggressive financial services sector, have found that the Crossroad framework fits well architecturally in cases where the development task is to extend, integrate, or distribute existing applications. Crossroad explicitly supports rapid prototyping of peer-enabled interaction among applications, databases, and networks. Crossroad applications can integrate diverse applications and data sources, and can be used to manage and reconfigure the interapplication process and data flows on the fly.

General Purpose Integration Focus. Compared with products such as PowerBuilder from PowerSoft (Burlington, Massachusetts), which focus on integrating applications and data sources, Crossroad has more of a general purpose orientation toward integration. Within its graphical application development environment, Crossroad focuses on support for building "integrating applications"—applications that integrate other applications. Developers have used Crossroad for integrating databases with analytical, presentation, and publishing applications, and for integrating system and network applications that do not use a database.

The Crossroad Architecture. The Crossroad development environment consists of three subsystems: CrossFrame Graphical Interface Builder, CrossScript Application Builder, and CrossLink Network Services Builder (see Illustration 1).

CrossFrame Offers Standard GUI-Builder Functionality. The CrossFrame Interface Builder offers developers standard GUI-builder capability, supporting the full set of Motif widgets. The CrossFrame GUI programming environment is tightly integrated with the CrossScript language and interpreter, supporting the development of both a fully functional, Motif-compliant user interface and the underlying application with just one programming endeavor. Because Crossroad applications run interpreted rather than compiled, the developer can build and test the user interface and the application incrementally, switching between Run and Edit modes. In-house developers and consultants working on an iterative basis with users do not have the opportunity up front to devise an elegant design for integrating applications. The incremental Crossroad approach can be a real boon for them.

CrossScript Builds on Familiar Programming Concepts and Features. The CrossScript Application Builder is actually a language shell that layers on top of the C libraries that are native to the target platform. Developers use CrossScript to program the behavior of CrossFrame user interface objects and screens, and to build the underlying application by determining the way CrossLink interapplication communication agents are managed and connected to the user interfaces. CrossScript also supports lower-level programming when required to handle complex interactions of specialized or "ill-behaved" applications. Though proprietary, CrossScript combines features of C, C++, and the C-shell scripting language, building on the open systems development environments with which programmers are already familiar.

The CrossLink Messaging API Supports Peer Interoperability. The CrossLink Network Services Builder is used for programming the interapplication communication mechanisms, the agents, that interact with the applications and each other to create an integrated environment. The CrossLink Agent concept is

based loosely on the dictionary meaning of "one who acts for or in the place of another under the other's authority." The CrossLink Network Services Builder includes a comprehensive library of C functions for defining and exchanging messages between CrossLink agents. Within the CrossLink programming environment, developers create agents that interface to existing applications through the application's native lexicon, and, at the same time, use the standard Crossroad C-based messaging API. The CrossLink libraries offer developers a rich environment for writing Crossroad API-conformant interfaces to existing applications without changing the application source code. The Crossroad messaging system uses a TCP/IP-based interprocess communication (IPC) design that ensures that the developer need not be aware of where on the network the called agent is running. The Crossroad messaging API is fully documented, allowing developers to incorporate external tools or to extend or enhance the API to meet specialized needs.

Crossroad Offers Turnkey Agents. Crossroad offers optional pre-built agents for Open Server from Sybase, Lotus 1-2-3, and Unix E-mail. More pre-built agents are being added based on the needs of the customer base, which is currently concentrated in the financial services industry.

Interpreter-Based for Rapid Prototyping. Crossroad's goal is to maximize the flexibility and configurability of its integrating applications. In order to support this goal, Crossroad applications are developed and run in the interpreted mode. This approach is consistent with the rapid prototyping/rapid deployment/configure-on-the-fly philosophy that has attracted "in-the-trenches" developers to Crossroad. However, the fact that Crossroad does not currently offer the option of generating compiled C code is an obstacle to developers who require control over their applications and insist on the greater portability of applications that run as compiled C code. A C-code generator is on the Crossroad enhancement schedule for 1992.

Building Applications with Crossroad. Crossroad developers can work with high-level tools throughout the development process. In the CrossFrame environment, developers work with the Motif look-and-feel to build "frames" of Crossroad-enhanced and standard Motif widgets by selecting objects such as buttons and list boxes from menus and dragging and dropping the ob-

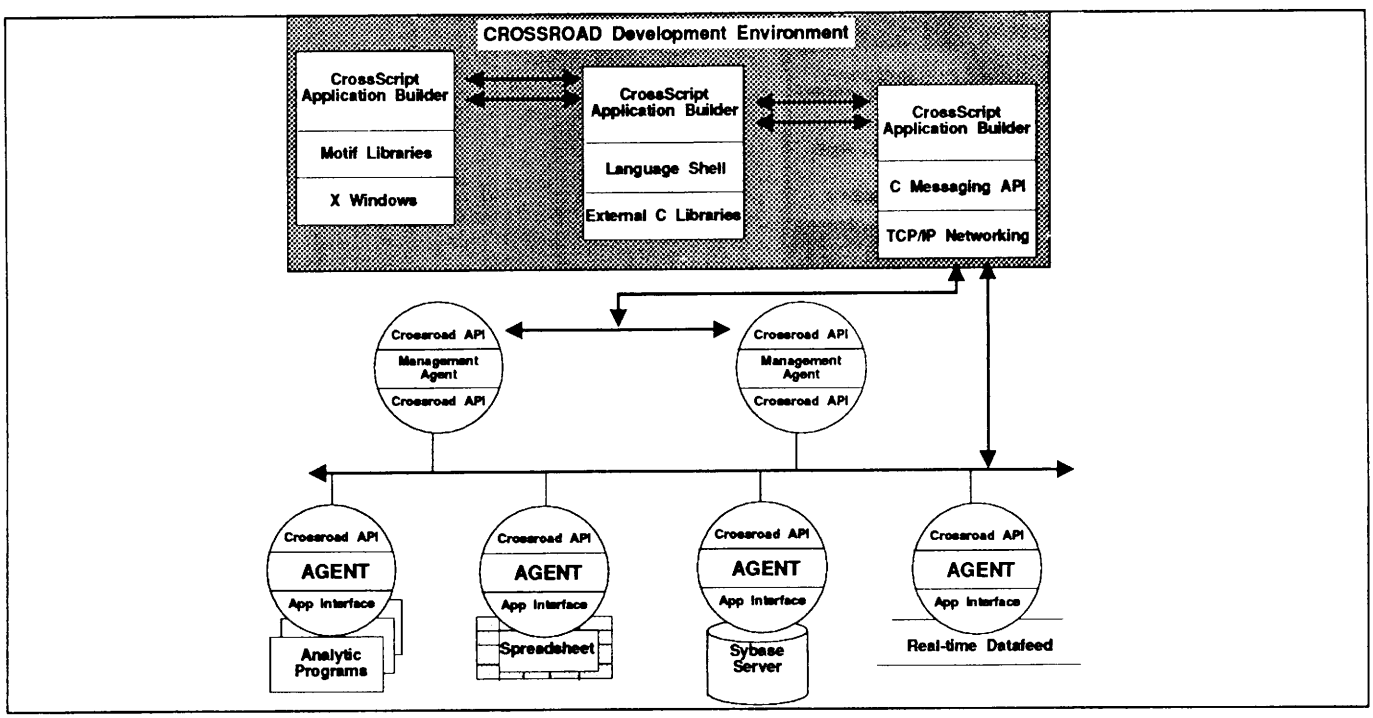

Illustration 1. The Crossroad environment supports configurable, dynamic relationships between integrated applications. The Crossroad messaging system is generalized to use TCP/IP interprocess communication mechanisms to access applications that are local and remote. In some cases, a special type of Agent, the Management Agent, can be developed in CrossLink and managed with scripts built with CrossScript to coordinate the interactions of application Agents, especially for event-driven complex interactions between applications that use different data types or file formats.

jects onto the workspace. The static layout becomes a live application incrementally as the developer writes CrossScript 4GL-style scripts that determine the behavior and the resources of the user interface objects. The scripting language is used to connect user interface objects with agents to configure or reconfigure the process flows or data flows in the integrated system. In building agents with CrossLink, developers can work with the C-based function libraries, or program at a lower level for specialized applications.

CrossScript programs facilitate the interactions of diverse agents by handling format or structure changes for data to pass smoothly between applications such as databases, spreadsheets, and graphics programs. Because CrossScript programs are interpreted, they can be modified on the fly to dynamically reconfigure the interactions among agents as new applications come on line or have to be changed.

Is Crossroad Object Oriented? Crossroad operates on some principles of object orientation, for example, the way agents appear to encapsulate applications and export the application functionality. However, Crossroad does not support core concepts of object orientation such as object classes, attribute inheritance, or object persistence, and it does not extend the configurability of applications to the user level, as distributed object management systems do.

Peer-Level vs. Client/Server Architecture. Even as client/server enabling technologies and distributed applications begin to arrive in the marketplace, concern is mounting about the overhead of server-based resource management. Complex interactions of applications in the network beg this question if all interactions have to be managed through a centralized server-based program. The Crossroad architecture addresses this issue through support for peer-level interactions between CrossLink agents, interactions that do not require the intervention of a supervisory software function. Peer-level agent interactions are event-triggered and can be completely automated. For example, a triggering event could be the arrival of an electronic mail message from a remote location into a file that is checked regularly for changes by a phantom process. When the change is detected in the monitored file, the phantom process sends a "wake-up call" to an E-mail agent, which then sends a message to a particular database agent, which, in turn, converts the data to a format that can be loaded into a database record, passes arguments to the database to open it for input, and actually loads the new data into the database. Upon completion of the database input, the database agent sends a message to the E-mail agent to confirm to the sending system that the data was received and the central database has been updated. Other variations on this theme might involve on-the-fly graphing of data with a graphics program; loading data into Frame for publishing; or reporting of system management events that trigger the automated re-balancing of system or disk loads through special purpose, real-time agents.

Pricing and Availability. Crossroad is priced at $7,500 per seat for the full development environment. It is available on SPARC machines, IBM RS/6000, and HP PA workstations. The run-time license is $500 per user for all platforms. Turnkey agents range from $500 to $5,000, depending on the agent.

SuperNOVA from Four Seasons Software

SuperNOVA from Four Seasons Software (Edison, New Jersey) enables developers to build new applications within a graphical programming environment that can be deployed independent of back-end databases, GUIs, and networks. SuperNOVA's functionality is primarily oriented toward integrating new or existing data sources into new applications that are developed with the SuperNOVA development environment and 4GL.

SuperNOVA Architecture. The SuperNOVA development environment consists of a relational model data dictionary, a 4GL editor for creating the application logic, and a window editor for creating the user interface. The SuperNOVA model builds applications as objects, and functions are performed on those objects. Applications do not run as compiled executables, but instead are run as meta-code that is interpreted at runtime by the SuperNOVA engine.

SuperNOVA Distribution Mechanism. SuperNOVA supports distributed processing transparently to the application code through the distribution of the run-time SuperNOVA engine. When any of the objects in the application has been configured to run distributed, the SuperNOVA engine must be configured with network interface libraries and dispatch tables to direct the requests for data to appropriate remote engines, and to return the data to the requesting application. When an application calls a function or database that is not running locally, the SuperNOVA engine refers to dispatch tables configured by the developer that contain location information for remote resources or data. The local SuperNOVA engine packages requests for data or services through its network interface, which has been configured with the appropriate libraries to support the network protocols needed to communicate to the SuperNOVA engine where the resource is running. The remote SuperNOVA engine then returns the data or results to the local application. The remote resource could be an existing application that has been modified to in-

terface with the SuperNOVA engine, or one which has been front-ended with a hand-coded C or a high-level scripted program to which the SuperNOVA engine talks.

GUI Independence. The SuperNOVA GUI selection decision is made at run-time. Developers work with a particular windowing system and look-and-feel during development, but the user can choose to work with the application in an alternate mode when starting an application. Since the application code is not compiled, the display of different GUIs is a parameter selection. SuperNOVA currently supports Windows 3.0, CRT terminals, X Window, and SunView.

Portability Strategy: Standard Run-Time Engine API with Platform-Specific Libraries and Drivers. Developers working with SuperNOVA may develop applications in any of the supported operating environments, including DOS, several versions of Unix, or VAX VMS, and deploy them on or across those platforms. SuperNOVA applications can run in a heterogeneous distributed environment without any changes to the source code. An application runs as SuperNOVA meta-code, independent of the platform or the location of other resources on which it depends. The meta-code is built during the development process and is stored in an internal database, and the SuperNOVA engine interprets the meta-code at run-time. SuperNOVA engines are configured and customized as needed to run on specific platforms and to operate locally or in a distributed manner.

The SuperNOVA run-time engine interfaces to local host systems through a set of platform-specific device drivers and libraries (see Illustration 2) that cover interactions with operating systems, graphical user interfaces and display devices, databases, and network protocols. Supported run-time operating systems include DOS, Unix (XENIX, AIX, Ultrix), and VAX VMS. SuperNOVA user interface libraries include drivers for the display hardware, the windowing environment, and graphical presentation for character terminals, workstations, and Windows PCs. The SuperNOVA database interface interacts directly with the data storage mechanism for the supported databases. Supported file systems and databases include flat files, C-ISAM, Informix, Ingres, Oracle, Sybase, HP Allbase, and Teradata. Network protocols currently supported are TCP/IP, StarLAN, and X.25. The latter two protocols reflect the predominance of telecommunications customers in the SuperNOVA base.

The Engine Is Extensible through a Toolkit. Four Seasons offers a toolkit of function libraries along with the SuperNOVA product. Developers can extend the SuperNOVA functions through these C-based libraries, or they can write new C code to create new SuperNOVA engine functions. Developers use the library toolkit to enable existing C-based applications to call SuperNOVA functions and to embed SuperNOVA calls in existing applications. This extends the capability of existing C-based applications with SuperNOVA functions such as database access.

Prototyping with SuperNOVA. SuperNOVA supports rapid prototyping of applications that have complex data requirements. The prototype developer works with

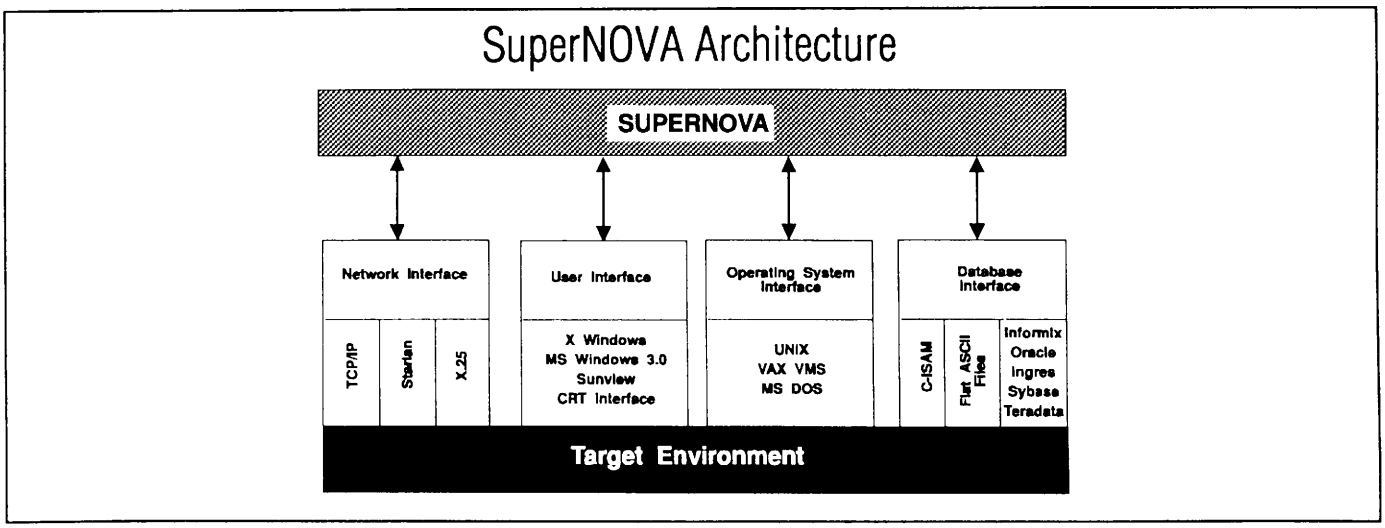

Illustration 2. Portability of SuperNOVA applications is achieved through insulation of the application logic from the specifics of the host systems. The logic remains unchanged whether the application runs as a standalone on a Unix system or in a distributed manner on a network of PCs running Windows.

form objects in the SuperNOVA development environment, employing flat files as the data source. When the application has been debugged and worked through the user-input cycle, the flat files can be replaced with live links to multiple RDBMSs or other data sources simply by changing the "datatype" parameter and by reconfiguring the SuperNOVA engines to operate over the network as required to request and receive data from multiple data sources.

Transparent Multi-Source Data Access. SuperNOVA supports simultaneous access of data from multiple data sources and can be located in any combination of local or remote in respect to the client system. For example, a developer may be building a marketing application to generate a request for a report projecting sales by product by quarter, based on input from a market researcher. The developer must integrate data from an acquired subsidiary in one city which is in an Informix database running on an IBM RS/6000 AIX system with data at the parent company which is in flat files for pre-1985 sales information with data in an Oracle RDBMS which is running on a VAX system for post-1985 sales data. The developer creates an internal data dictionary that includes location and access method information required to fulfill a request for data from within the application. The SuperNOVA engine local to the application handles the requests made to all three sources of data simultaneously and returns data to the application, which then builds the report. Neither the marketing application nor the user would have any explicit awareness that the report was built from a combination of remote sources of data.

Pricing and Availability. SuperNOVA was first introduced in 1988. Its current release is 3.0, priced at $4,000 for a complete development license for a 386-based workstation and $6,700 for a 486-based workstation. These prices include a database interface. Runtime licenses are $900 for a 386 workstation and $1,500 for a 486-based workstation. These are sample prices only, since SuperNOVA has a very complex pricing scheme that covers 45 vendors and almost a hundred platforms, including high-end VAXs. For example, on a VAX 9000 Model 440, a distributed development license is over $150,000.

Comparison of Crossroad and SuperNOVA

Crossroad and SuperNOVA both offer effective tools to address the thorny problems of integrating existing applications and databases in the distributed model. SuperNOVA tools orient strongly to integrating new applications with diverse sources of data. Crossroad is oriented more toward integrating existing applications and data sources with each other, and dynamically managing the process and data flows between a set of applications and data sources. SuperNOVA offers a set of data-integration-specific features, such as an internal data dictionary and data model, a multilevel referential integrity enforcement option, and data security options in the form of fine-grained access control and encryption. Through the robust messaging model of CrossLink agents, Crossroad specializes in making Unix-based applications work together, which is applicable to data integration but doesn't have a specific focus on integrating databases.

Crossroad tools can produce integration environments that offer configurable relationships between the integrated entities. Application integration with SuperNOVA is between fixed static integration and configurable integration. (See *Open Information Systems*, Vol. 7, No. 7, for a discussion on levels of application integration.)

Summary and Conclusions

Demands placed on developers for fast turnaround of development projects that directly serve the needs of users are becoming increasingly intense. Developers must look for ways to leverage existing application functionality in meeting evolving user requirements. With Crossroad and SuperNOVA, as with other tools in this segment, developers can quickly prototype new graphical applications that serve the practical business needs of users. As users begin to deal with the slow introduction of DCE-based applications and tools and the even slower introduction of distributed object management development environments and applications, they should give serious immediate consideration to development frameworks such as Crossroad and SuperNOVA. —*S. Dolberg*

FOCUS: GRAPHICAL USER INTERFACES

OSF Releases Motif 1.2: Filling out and Speeding Up

Look-and-Feel the Enabling Technology for X Window

When the X Window System was conceived in the mid-1980s, its design goal was principally to be a portable and networkable windowing system for bit-mapped displays. Its developers positioned it as an underlying graphical user interface (GUI) service rather than as a user-level environment. Because of this, X provided no end-user style nor functionality. Alternatively, it could provide any kind of style or functionality. It is the look-

and-feel layer that adapts the technical capabilities of X Window for the user at the display. This layer provides a consistent set of display components—such as window titles, text regions, and check boxes—that users can learn and internalize as their tools for accomplishing work through the GUI. Motif, now released in Version 1.2 by the Open Software Foundation (Cambridge, Massachusetts), has the broadest industry support as a GUI for X Window.

Motif Dominates X Window

When Apple pioneered the windowed GUI with its Macintosh personal computer, it provided software "toolboxes" for developers. With these, one could write applications with standardized Macintosh components and behaviors. Apple pressured developers to use the toolbox instead of writing directly to the hardware, as IBM PC developers often did. Because of this, Mac software from different developers has a high degree of consistency, making it easy for users of one Mac program to learn another.

The X Window environment, however, had no such direction when it started. Its developers at MIT implemented basic look-and-feel libraries or "widget sets," but computer vendors could alter them or implement their own. This played havoc with application developers. How could they develop X software that would run on different vendors' systems with their different display characteristics without rewriting the application for each platform? End-user organizations were equally dubious. How or why should they commit to X Window as a GUI technology that was supposedly open, portable, and interoperable, but that looked and felt radically different on different vendors' platforms?

The Open Software Foundation (OSF) stepped into the breach with its first and most successful Request for Technology (RFT) to date. It solicited specifications for toolkits and GUI look-and-feel from the industry as a whole, and settled on a submission bearing a strong resemblance to Microsoft Windows and to the OS/2 Presentation Manager. Motif, as the product was called, was an instant success. Adopted immediately by major system vendors, Motif unified large segments of the Unix industry and gave application vendors a single GUI to which they could write their applications. Sun Microsystems was the apparent loser, after championing the Unix System Laboratories OpenLook specification. Sun continues to support and ship OpenLook, but many customers have switched their Sun workstations to Motif implementations provided by third parties. In the rest of the Unix market, Motif is taken for granted.

Motif 1.2 Adds Helpful Interface Behaviors

DRAG-AND-DROP. From its inception, Motif provided the basic presentation and behavior necessary for the end-user environment, but that is a far cry from saying it implemented or supported comprehensive functionality. One of the most widely discussed needs has been partially answered in Motif 1.2's drag-and-drop capability. The idea is simple enough in the context of a single application: You select text, a picture, or any object, and you drag it over to a new placement in the document. Equally important is the fact that the drag target can act on the object appropriately. But the user also needs this capability *between* applications, which raises numerous technical issues. How does the receiving application know that a foreign object is being dropped into it? What is the communication mechanism? How do the applications decide what type of data to exchange or what actions to take?

Motif's solution to these problems was actually leveraged from new work by the X Consortium. The X approach dictates that the display server act as a clearinghouse mechanism between applications. Hence, the X clipboard, which had supported cut-and-paste between applications, was implemented through X server software. Guidance for all such operations is contained in a document called the Inter-Client Communication Conventions Manual (ICCCM).

ICCCM revisions are just being completed to define interapplication drag-and-drop, and Motif 1.2 has been written to make use of the revisions. Now, it will be necessary for applications to be revised as well. But the new Motif drag-and-drop will not only support data transfer, but it will also cause functions to be activated. Dropping a file icon onto a mail application, for example, could send that file as a letter. A spreadsheet range could be dropped on a printer icon to print out that range.

TEAR-OFF MENUS. While all of an application's functions are normally accessible through its pull-down menus, functions or settings that the user employs very frequently require some quicker, more convenient access. "Accelerators," or special key combinations, can be provided, but only so many can be registered before the end user loses track of them. An alternative is to allow a pull-down menu to stay on the screen after use, so that its functions remain available to the user. Motif 1.2 implements this capability in "tear-off menus." The user can click on a menu to pull it down, then click on a perforation line to "tear it off" and place it elsewhere on the screen. It stays there, allowing the user to invoke its various functions, until it is explicitly closed.

MORE CONVERGENCE WITH WINDOWS/PM STYLE. One of the original thrusts of the Motif definition was compatibility with PC windowing systems. This makes GUI skills transferable between Motif and Microsoft Windows and the OS/2 Presentation Manager. Text input, selection, and editing are now almost exactly as they were under Windows/PM. Also, the handling of dialog boxes has been augmented with support for cursor keys as under Windows/PM. With these changes, the OSF has also made Motif more "keyboard friendly."

"MULTI-HEADED" DISPLAYS. The more GUI work one performs, the more screen space one wants. Macintosh users ran into this first, and Apple responded by implementing a capability to add monitors to a Mac display. X Window also supports such "multi-headed" displays, but, until recently, it was difficult to use displays of different characteristics. Without custom C programming to manipulate the X Window server, display characteristics would be reduced to the least functional monitor that was attached, e.g., a true-color monitor running with grayscale would become grayscale. Motif 1.2 allows monochrome, grayscale, and color screens to be used in the same logical display without coercing them to the least common denominator of functionality.

WINDOW MANAGER UPGRADES. OSF has made a number of improvements to the Motif Window Manager (mwm). First of all, it now handles non-rectangular windows, such as circular "wall clocks." The mwm 1.2 version also displays a full window image during moves, instead of just outline bars, and it packs iconified windows more tightly to conserve display space.

Incremental Technical Improvements

SUPPORTS C++ AND FULL ANSI C. The Motif 1.2 widget libraries have been revised from the least-common-denominator style of the C programming language to employ full ANSI-standard C. This revision makes the Motif toolkit more reliable and easier to use correctly. In addition, OSF provides a set of header files that can be used on most platforms to invoke the Motif widgets from the C++ programming language.

TOOLKIT ENHANCEMENT WITH BACKWARD COMPATIBILITY. Toolkit enhancements in Motif 1.2 basically amount to the implementation of an API to support drag-and-drop functionality. This API allows applications to register in the drag-and-drop system, either statically when they are started up or dynamically as the user works with the application. Dynamic drag/drop functionality allows Motif 1.2 applications to customize drag/drop behaviors on the fly, but at a perceptible cost in performance.

Other toolkit enhancements include the development of APIs for some of the critical Motif widgets. Motif 1.2 has extended programmatic use of the Text, List, and Label widgets by implementing APIs for them. Ordinarily, widgets are invoked through the X Window event-loop/callback mechanism, rather than through an explicit API. And while Motif developers will still ordinarily use this latter mechanism, the new APIs will allow some additional flexibility in GUI development.

Software built with Motif 1.1 will recompile directly against Motif 1.2—no source code modification is necessary. In many cases, relinking against the Motif 1.2 libraries is all that is needed.

X11R5 SUPPORT. Release 5 of the X Window System, Version 11, was released last year. This was called the "performance release" of X11. Motif 1.2 retains compatibility with X11R4 display servers but includes rewrites to exploit X11R5's speedier execution and improved internationalization capabilities.

PERFORMANCE IMPROVEMENTS. Motif 1.2 exploits X11R5 performance improvements in its widget set and also in the Motif window manager. A key approach is "event compression," by which mwm optimizes its interactions with the X display server. Instead of dealing with each GUI event sequentially and responding to it, Motif can detect a stream of events and "add them up" to one or more compressed "net events." Ordinarily, Motif would respond twice as the user pulls the cursor over an unused window—once as the cursor enters, again as the cursor leaves, regardless of the user's intentions. Event compression code would recognize that entering and leaving a window is, effectively, a null event, and simply ignore both of them. This greatly speeds refreshes and window manipulations.

INTERNATIONALIZATION EXTENDED. Motif 1.1 included basic internationalization, but it awaited X11R5 to really complete the facility. X11R5 not only allows output in multiple languages and character sets, but its keyboard "input method" facility also allows on-the-fly entry and editing in any language, even those using multi-byte character encoding like Japanese Kanji. "Locale" functionality now fully supports the ANSI C definition, allowing input and output of certain information according to local custom. For example, in Europe, dates are customarily abbreviated day.month.year, rather than month/day/year as they are in the United States. Also, application messages and labels can now be loaded from language- and locale-specific text files. The result is that the developer can construct an application in which the same binary will support fully-localized interaction in English, French, Japanese, or any other language.

OSF Also Focuses on Management Issues

COMPLIANCE WITH STANDARDS. One perceives in Motif 1.2 OSF's commitment to deliver technologies based on standards. It adheres not only to X11R5 and to ANSI C, but it also boasts POSIX compliance and XPG3 branding.

VALIDATION AND QUALITY ASSURANCE PROGRAMS. Taking a cue from the U.S. government's Ada programming language, the Unix community some years ago recognized that some form of validation is necessary for every standard. Hence, the rise of validation test suites for Unix System V, POSIX, and the X/Open Portability Guide. Since Motif is a standard as well (albeit de facto), OSF has implemented an automated test suite to verify conformance of each hosting of Motif. Complementing this is a quality assurance suite that verifies the robustness of each implementation.

STRONGER INTERNAL QA, TOO. The OSF undertook many internal quality initiatives in Motif 1.2 as well. It had an outside organization perform quality assurance (QA) testing to prepare for release, and it seeded numerous "snapshot" licensees to get external comments on the product. We feel these actions are laudable.

But What Is This Licensing Fee?

Many have commented on the redirection and re-evaluation of the Open Software Foundation necessitated by its rapprochement with its former arch rival, the Unix System Laboratories (Summit, New Jersey). One obvious goal in OSF's redirection is building toward financial self-sufficiency. Motif, as the OSF's only current commercial success, would naturally take a key role. This has necessitated significant changes to OSF's business model, with the attendant result of some market confusion. The OSF plans or hopes to gain considerable additional outside revenue, instead of depending exclusively on its members and on its major sponsors—but without burning its bridges. It has raised unlimited-redistribution source-code licensing fees for Motif from a mere $2,000 to $15,000. However, most current Motif customers will find that a $2,000 source license, restricting licensee distribution of the product, adequately meets their needs. QA suites, which had a hefty price tag under Motif 1.1, are included gratis on the Motif 1.2 distribution. While OSF obviously tried to fine-tune its licensing approach, this alteration fundamentally changes the way that outside parties, especially those outside the OSF, will use Motif. The added functionality, plus the total commitment of most vendors to Motif, will most likely keep this change from backfiring. However, as Unix System Labs enhances its combined Motif and OpenLook Toolkit (MoOLIT), third parties may find it a more attractive development system and run-time to use than OSF's.

Conclusions: Motif Consolidating Its Hold on X Window

PUTTING THE POLISH ON. OSF/Motif Release 1.2 is not new construction but good finish work. It plugs some holes in the look-and-feel specification, and in mwm, and it improves overall performance and responsiveness. Although we are concerned about OSF's attempt to pull more money out of Motif with this new version, we are nonetheless pleased with the improvements in Motif 1.2.

WHERE ARE THE HUMAN FACTORS? OSF retains its overriding concern that Motif have maximal compatibility with Windows and Presentation Manager. While Motif 1.2 certainly improved this compatibility, we are concerned that the user has been left out of the user-interface development process. We would like to see some degree of substantial, independent scientific research into GUI ergonomics that OSF and others could use to turn usability claims into usability proofs.

THE BIG FISH IN THE POND. Motif has the pole position in the Unix GUI market. Apart from Sun, no major vendor supports OpenLook, the only alternative look-and-feel specification for X Window. And while some figures indicate that Sun desktops are at a par with those of the Motif vendors, a lot of Motif implementations are cropping up on Suns. But the problem isn't Unix infighting; the real problem is that Microsoft Windows is shipping in huge quantities, completely out-muscling Unix in the desktop market. Industry wags might explain Microsoft's next rev, Windows NT, as meaning "not there." But they could have said the same about Windows two years ago, and now it's the largest selling GUI. So, while Motif might be the big fish in the Unix pond, Microsoft Windows dominates the water supply. Motif will not be the product to change that reality.

—A. Wolfe

Letter to the Editor

In the feature article "The X Window System" (*Unix in the Office*, Vol. 7, No. 4.), Andrew D. Wolfe touches on the debate over the merits of PCs and X terminals. While he makes a number of thoughtful assessments, he fails to mention specific cost and performance issues which make the PC-based X server decidedly right for some organizations and equally wrong for others.

For compute-intensive environments that require dedicated X devices, the hidden cost of a PC solution may make it a less viable option than an X terminal network, not to mention that the PC alternative might fall short on performance.

To illustrate, users need to consider three criteria when analyzing the relative points of PCs vs. X terminals:

- The performance quality of the PC-installed base
- The existence and quality of the PC network
- The type of display devices in use

Since a PC must be at least a 386/20MHz machine to function reasonably with an X server, companies with less powerful systems must upgrade. Likewise, to work as an X server, PCs must be networked with a TCP/IP LAN, like PC NFS or PC-TCP.

However, one also must remember that TCP/IP network software for PCs is less robust than X terminal network software and much harder to install. PCs running X Window are extremely dependent on the network software and may not be able to simultaneously access more than a few windows. As new variables are introduced on the PC network, such as new memory cards, there is also a risk that incompatibilities with existing hardware and software will be introduced.

Companies lacking adequate networking capabilities for PCs but wanting to turn PCs into dedicated X devices, may have to invest up to $1,250 per seat, considering the cost of the network card, the TCP/IP software, and the X server software.

Mr. Wolfe also did not mention that most installed PCs lack the graphics capability to display multiple X windows. The vast majority of installed PCs rarely exceed EGA or VGA with 16 colors and 640 x 480 resolution. Even with a high resolution, 256-color VGA card installed in the PC, the performance may not be adequate for X Window operations. Depending on the card and the monitor, an accelerated graphics package upgrade can cost up to $1,500.

In short, to use PCs as X terminals may cost more than $6,000 per seat, for a total bundle consisting of the 386-based PC, an accelerated graphics card with 256 colors, a high-resolution 14-inch monitor, a network card, network software, and an X server for the PC. Even after users address their hardware needs, PC X-terminal emulation software is not guaranteed to run all X programs. In fact, users may even have to change their existing software to solve a problem.

Dan Fullerton
Business Development Manager
Tektronix Network Displays Division

Editor's reply: At the Association for Image and Information Management (AIIM) Conference in June, I ran into two developers who were debating the question of whether they should write their applications for X or for MS Windows. One of the developers had come to the conference convinced that X was the right approach, but, after seeing so many exhibits on the show floor with MS Windows front ends, he decided that was the way to go. The other developer had come to the conference convinced that MS Windows was the right approach, but, after seeing the cost of the hardware required to get adequate performance and too many "General Protection Fault" messages, he decided that X Window terminals were the better choice. When these two met at a reception, each was astounded that the other had reached the opposite conclusion from his own! A more serious issue for open systems than X terminals vs. PC X servers may be the X Window System vs. Win32 battle that will be heating up over the next two years.