Patricia Seybold's Office Computing Group

Editor-in-Chief
Michael A. Goulde

UNIX IN THE OFFICE

Guide to Open Systems

Vol. 7, No. 2 • ISSN: 1058-4161 • February 1992

System V.4 and OSF/1

Matching up in the Marketplace

By Judith S. Hurwitz and Michael A. Goulde

IN BRIEF: System V Release 4 (SVR4) from Unix System Laboratories (USL) represents a major step forward in Unix technology. By merging the three major variants of Unix that were creating confusion in the marketplace-System V, Berkeley, and Xenix—it holds the promise of resolving one the greatest barriers to the wide-scale adoption of Unix by new customers. However, USL paid a severe price in bringing SVR4 into existence: the creation of the Open Software Foundation (OSF) and its competing technology, OSF/1. While the two technologies appear to stack up very evenly, the competition between them will continue to be fierce.

Report begins on page 3.

When Change Is Constant

The only constant is change

THIS ISSUE REPRESENTS the beginning of the seventh year for *Unix in the Office* and the inauguration of a new Editor-in-Chief, Michael Goulde. Michael has had broad experience in the industry, having followed Unix, distributed computing, personal computers, image management, and open systems over the past eight years. His involvement with users and vendors has led him to be concerned about the impact that social, economic, and political changes have on business, technology, and the interdependency between the two. He believes that technology and standards need to be viewed as enablers. Their role is to support the efforts of organizations as they search for new ways to add value to information as it is captured, managed, and disseminated. The important issues from his point of view are those that address how businesses can get the biggest payback from their investments in both time and money without having to struggle with barriers placed in their path by closed products, vendors acting in their own self interests, or Quixotic pursuits of utopian visions.

We have a challenge here at *Unix in the Office*. Like Unix, we must keep evolving and improving if we are to meet our commitment to our customers. As we have said before, a publication and its readers are parties to a contract. Our end of this bargain is providing you with the right information about the issues that concern you and your organizations in a timely, analytical fashion. Your end of the bargain is to keep us informed about what those issues are so that our newsletters can evolve and continue to meet your needs as they change over time. Your feedback is vital in allowing us to assess how well we are meeting your needs.

Michael is committed to being responsive to the interests and issues his readers feel are important. He will encourage you to give him feedback and suggestions. Over the coming months, with your help and assistance, he will be assessing the future direction for *Unix in the Office*. Next month's newsletter will include a survey of readers' interests and concerns. Start thinking now about how this publication can best address your personal and organizational requirements for information and analysis. Think about the topics and the type of coverage you would like to see in the newsletter. Most importantly, think about the role Unix will play in an ever-changing information technology industry.

Just as Unix is catching on in the mainstream, it faces a severe challenge from the very open systems movement it helped spawn. With a few exceptions, it is now generally acknowledged that having an open system doesn't necessarily require Unix. As you will read in this month's News, Analysis, and Opinions section, higher-level interface standards, such as POSIX, can be supported by non-Unix operating systems that can be purchased today. What does this mean for Unix—and what does this mean for *Unix in the Office?* With your help, we will explore answers to both these questions over the coming months. ◐

UNIX IN THE OFFICE

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.
News Editor
DAVID S. MARSHAK
Art Director
LAURINDA P. O'CONNOR
Sales Director
RICHARD ALLSBROOK JR.
Circulation Manager
DEBORAH A. HAY
Customer Service Manager
DONALD K. BAILLARGEON

Patricia Seybold's
Office Computing 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

Unix in the Office (ISSN 0890-4685) is published monthly for $495 (US), $507 (Canada), and $519 (Foreign) per year by Patricia Seybold's Office Computing 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 *Unix in the Office*, 148 State Street, 7th Floor, Boston, MA 02109.

FEATURED REPORT: BY JUDITH S. HURWITZ and MICHAEL A. GOULDE

System V.4 and OSF/1

Matching up in the Marketplace

Now that OSF/1 is a real product and no longer just a work-in-progress, the showdown between the two major commercially distributed Unix technology alternatives, Unix System Laboratories' (USL) System V Release (SVR4) and the Open Software Foundation's (OSF) OSF/1, has begun. Each of these two technologies has its proponents, its strengths, and its weaknesses. Customers and OEMs will have choose one to support and use, although it will be customers who will ultimately have the loudest voice in determining which technology will dominate.

In the case of SVR4, we are looking at technology that has been evolving for two decades. USL is committed to supporting the Unix installed base for whom SVR4 is the logical migration path. OSF/1, on the other hand, is new technology, with no installed base to consider. This has allowed OSF to learn many lessons from the evolution of SVR4 in the process of designing OSF/1. Although OSF has to consider compatibility with the installed Unix base, it needs to be concerned with a different set of issues, not the least of which is convincing potential customers that OSF/1 is a superior substitute for SVR4.

History of SVR4—the Merged Operating System

Twenty years ago, when Unix was being rewritten in C by Dennis Ritchie of Bell Laboratories, AT&T was forbidden by law from competing in the computer business. Therefore, the company had no commercial interest in Unix and licensed it liberally, first to educational institutions and then to commercial computer vendors. As it turns out, AT&T's liberal licensing of Unix source code has been both a pro and a con. Over the years, programmers (and hackers) made substantial revisions to the code received from AT&T for the following reasons:

- To improve on the code supplied by AT&T
- To add functionality that was lacking
- To adapt the operating system to new applications
- To adapt the operating system to new hardware
- To add proprietary features for differentiation

By the early 1980s, a multitude of different Unix versions had proliferated, all of which were incompatible with one another to varying degrees. As long as AT&T was constrained from competing in the computer industry, it had little to gain or lose from the situation and took no action. However, once AT&T was freed from its previous legal bonds by Judge Green's Modified Final Judgment (MFJ), the company quickly recognized two things about Unix. First, there would be a significant business opportunity in making Unix an industry standard, and second, the fragmentation that existed would have to end if Unix was to emerge as a standard. The vision of one standard Unix led to a plan for merging the major Unix variants into a single release, SVR4. It was to be a hybrid consisting of three major components:

- A superset of its most popular version, System V Release 3.2
- The best technology from Sun Microsystems' SunOS (including selected facilities from the Berkeley version of Unix)
- Integration with Microsoft's Xenix, a 16-bit version of Unix oriented towards personal computers.

The potential for SVR4 was obvious: A single version of Unix would incorporate the best technology from the most popular versions of Unix and offer heretofore unknown levels of compatibility, portability, and interoperability. It was just what the market ordered.

The Vision Becomes Clouded

Things came apart for this unifying vision in 1988, when AT&T recruited Sun Microsystems as its development partner. Sun, whose Unix operating system was based on a Berkeley Software Distribution (BSD) version, was to work with AT&T in merging Berkeley and System V, using Sun's SPARC as the reference platform. AT&T, in turn, made an equity investment in Sun. Unfortunately, there were a number of other vendors, including IBM, Digital Equipment, Hewlett-Packard, Siemens, Bull, and Hitachi, that didn't like the idea of their biggest Unix competitor, Sun, playing such a central role in the development of SVR4. Another gripe was the requirement that Unix licensees would, as a condition of their license, have to assure that their products would pass the System V Verification Suite (SVVS), which meant that licensees would have to follow AT&T's lead in Unix development. This is when the Open Software Foundation was formed and the vision of a single unified version of Unix was lost.

SVR4's Design Philosophy

Because SVR4 was designed to support the old while migrating to the new, it had to be restructured to be more modular and it had to include a series of layered interfaces. Modularity allowed new components to be added while older facilities remained available, though enhanced, thereby maintaining compatibility for applications written for earlier versions of System V, SunOS, and Xenix. Improved modularity would also make it easier for other vendors to create device drivers to match the hardware components of their systems. In addition, the layered approach would allow licensees and third parties to add value on top of the operating system in a variety of ways, ranging from including different file systems to using different user interfaces.

SVR4: A Better Unix

In spite of what some of its detractors say—"Just because it has a higher release number (than V.3.2) doesn't mean that it is better!"—SVR4 is a much better operating system than its predecessors. Functional and design issues have been addressed both in the base operating system and in additional SVR4 versions being made available that support multiprocessing (MP) and enhanced security (ES). These issues include the modularity of the kernel, the functionality and organization of the file system, and capabilities in other versions of Unix that System V lacked. In order to move quickly in what has rapidly become a competitive environment, USL has brought in a slew of partners to accelerate the implementation of new functionality. For example, USL licensed software from Veritas to add disk-mirroring and fault tolerance to SVR4. Sequent, which has extensive experience implementing multiprocessing Unix system kernels, has been working closely with USL developers in that area. Security extensions have been implemented in conjunction with Amdahl and Motorola. By using all of these developments in various combinations, USL has released a set of richly featured operating system options and has accomplished the Xenix-BSD-SunOS-System V merge as originally envisioned.

Multiprocessing for SVR4

The benefits of multiprocessing were discussed in last month's *Unix in the Office*. For the past two years, a working group within Unix International (UI) has been hard at work providing input to USL about multiprocessing requirements for System V. Because of this pressure, USL signed up several of its partners to provide multiprocessing capabilities for SVR4. Implementing true multiprocessing with SVR4 required a kernel that supported multiple threads. The threading that was implemented in SVR4 MP (the multiprocessing release of SVR4) extends to the file system, device drivers, Streams modules, and schedulers. This new kernel is intended to scale from 1 to 16 processors, and is targeted to run on machines that have symmetric, homogeneous, shared memory architectures and systems that either have fully symmetric I/O or have all I/O handled by one processor.

The structure of the MP implementation is designed so that hardware dependencies are isolated from the platform-independent code. To take advantage of the MP release, the underlying hardware must at least support inter-processor interrupts. This means that each processor in the system must be able to send an interrupt to any other processor. While the MP release will work on hardware that supports only a single interrupt priority, it will work better if the platform supports the ability to interrupt at several different levels.

Process Scheduling in SVR4 MP

MP scheduling is more complicated than single processor scheduling; therefore, additional MP extensions have been necessary. While traditional scheduling has been priority based, the MP release offers an option to bind a specific process to a specific processor so that a critical process is assured fast response. In addition, a pool of processes is maintained so that the process with the highest priority runs first.

Security Support

SVR4.1 ES Appeals to Government Buyers

Operating systems by themselves are not evaluated for security. It is the implementation of an operating system on an architecture that gets certified by the National Computer Security Center (NCSC). According to the NCSC's specifications, certification may occur at four levels: D, C, B, A (where D is the least secure and A is the most secure). Within each level are additional security levels. For example, a C2 level of security is more secure than a C1 level.

USL has identified system security as a key feature for selling to government agencies and to certain highly security-conscious commercial industries like banking. Its first product that addressed the high-security market was Unix System V/Multi-Level Secure (MLS), which was based on SVR3.2. The current Enhanced Security (ES) release is designed to conform to the criteria for the B2 level and has been submitted by USL for NCSC certification. SVR4.1 ES offers users B2 level security with some B3 features (remember: B3 is more secure than B2). Level B2 is intended to allow the most secure data to coexist with data that do not have restrictions. SVR4.1 ES can be configured by the licensee for level C2 or B1 security if the higher level of security is not required.

USL had to make several significant changes to the operating system to achieve this increased security, including a major restructuring of the Unix kernel. The monolithic kernel of previous releases has been partitioned into subsystem-level modules. Each component of the kernel, including source files, was packaged as a module containing the associated public header files, headers for use within the kernel, private headers used only within the module, and the code implementing the subsystem. By modularizing the kernel, USL made the security enhancement effort more straightforward and reliable, while addressing government criteria for the design of a secure system.

USL's implementation of security in the ES release is extremely efficient, resulting in significantly less performance overhead than there is in many security implementations. This stems more from the reorganization work that was done than from having written a lot of new code. In fact, less than 10 percent of the code in SVR4.1 ES is new or changed code.

Enhanced System Administration

System administration has also been enhanced for the V.4.1 ES release. The administrative user interface, backup and restore services, and package installation services have all been improved. Included are a messaging daemon and a message-handling interface that supports writing to the console. The administrative interface also allows for management of the new security features introduced in this release, including remote backup and restore.

Enhancements for Secure Networking

Several new networking facilities have been added that apply to secure configurations. Connection Server (CS) allows a single server to perform connection establishment, identification, and authentication. The CS library interface allows the network to obtain the address of the specified server on a given machine and return a connection to the application. The primary use of CS is for mutual authentication of server machines in a network. It is designed in such a way that it could support Kerberos authentication, although this support is not included from USL.

Identification mapping is a facility that translates a user id from a remote machine into a known identity on a local machine. This is helpful in the authentication of a remote user. Remote execution facilities, including *rexec,* have been implemented using the CS. This provides a more secure implementation of the Berkeley "r" commands and allows for control over services executing on remote hosts, giving the administrator more direct control over a remote machine. This is a critical component in helping to control security in a distributed networked environment.

Two networking facilities have been enhanced for security purposes, SAF (Service Access Facility) and Unix-to-Unix Copy Protocol (UUCP). Now SAF can use the services of the Identification and Authentication Facility (IAF) when starting up the requested services. The administrator can specify the identification, authentication, ID mapping, and environment setup schemes to be used. UUCP has been improved to increase the security of batch-oriented file transfer and remote execution. Likewise, the remote file system Remote File Sharing (RFS) and Network File System (NFS) have been modified to support B2 security.

Additional Security Options

Optional security features for SVR4.1 ES include C2 Auditing and B2 Enhanced Security, including mandatory access control, discretionary access control, a least privilege mechanism that replaces the superuser, and trusted facility management. The latter is the separation of administration responsibilities into distinct roles, allowing for the two classes of users-administrator and operator. Trusted path supports a secure communication path between a user and the system for initial login and authentication.

Do Customers Want Multiple Releases?

The principle drawback to both the ES release and the MP release is that they are separate releases. In other words, you may choose one capability or the other, but you can't have both. This will be fixed in an SVR4.2 ES/MP release that was placed into early access by Unix International (UI) in January 1992, but which will not be generally available to licensees until mid 1993. This means that it probably won't be available to users until mid 1994. Therefore, having multiple versions of SVR4 from USL is a potential barrier to the widespread adoption of either of the enhanced releases.

Common Components Shared by SVR4 Releases

While some components of SVR4 are different for the MP and ES releases, all have common underpinnings. These include kernel components, such as the handling of virtual memory, file systems, basic communication and networking, shells and commands, and administration.

VIRTUAL MEMORY. SVR4 has borrowed the virtual memory (VM) management system from SunOS. It includes support for memory-mapped files, allowing users to access file system data much more easily than was possible with earlier releases of System V. With VM, users can map part of a file into the address space rather than bringing data into buffer memory. This allows for a more efficient use of system resources and provides a named memory facility that allows processes to map files or devices into their memory spaces. Once mapped, the contents of files can be accessed as memory locations.

USE OF SINGLE-LEVEL STORE. SVR4's memory mapping also allows physical memory to be constructed as a single-level store. A single-level store is a unified mechanism for accessing file system data, using physical memory more economically by treating all user memory as cache. This eliminates the use of buffer cache for file I/O. In contrast, programs executing in other versions of Unix, such as Berkeley, are mapped to address space and then automatically brought from system into disk page by page, requiring repeated, memory-intensive I/O calls. Traditionally, it has been difficult to fine-tune the use of memory because of the split between memory used for I/O and memory used for programs. Single-level store reduces overhead and makes operations more efficient.

SHARED MEMORY. While shared memory has existed in System V for some time, it was enhanced by the use of Sun's implementation of shared memory. Another aspect of shared memory is dynamic linking, which has been added to SVR4 from SunOS. This capability allows subroutines to be dynamically linked, providing flexibility at run-time and in the program development cycles—compile, debug, and recompile. Dynamic linking is especially important for low-end systems because it allows the user to automatically shift pages into and out of memory. It also plays a critical role in implementing object-oriented concepts, where objects need to have a dynamic linking mechanism at run-time.

File Systems

The file system has always been one of the key weaknesses of the Unix operating system. The native System V file system was not modular and treated all types of files equally. A longstanding irritation for users has always been the difficulty involved in changing file system size. To change the size of a local volume, the user had to dump the files to tape and rebuild them. These problems have all been addressed in SVR4.

Because USL has to contend with the past as well as look to the future, a Virtual File System (VFS) is offered, allowing users to switch between different file system options. As more users decide to move to more advanced file systems, such as the AFS (Andrew File System), now part of OSF's Distributed Computing Environment (DCE), they face an easier migration by using the VFS. The VFS is a merger of the traditional System V File System Switch (FSS) and the SunOS VFS mechanism.

Achieving file system independence required the use of a vnode mechanism. The vnode acts as a master switcher, allowing users to select among file systems. USL has defined the VFS kernel interfaces so that third parties can implement new files systems under SVR4. The first example of this is the work USL is doing with Veritas to link its file and disk management technology into SVR4.

Reliability Greatly Improved

Some of the important improvements over earlier versions of System V have been made in the area of file system reliability. The manner in which file I/O is performed is a critical component for implementing commercial-grade transaction processing systems and database managers since new information must never be written to disk in an incomplete form, in case the system should crash before the write is complete. Once the system has been recovered after a crash, the information has to be consistent. SVR4 meets these requirements by handling bad blocks dynamically and by including a deferred-write scheme for modified files. However, full commercial quality will have to wait for roll-back and recovery features as well. In the initial design, SVR4 included a write-through option but not disk-mirroring. The Veritas technology adds a disk-mirroring option.

FILE-LOCKING. Another significant improvement is the way SVR4 implements file-locking. The key changes include the addition of advisory and mandatory file- and record-locking, synchronous write mode, and Xenix file- and record-locking compatibility. However, as SVR4 moves ahead into areas such as multiprocessing and, eventually, parallel processing, even more granular locking will be required.

POSIX Forces Some File Changes

The IEEE POSIX specification is having an impact on some subtle but important aspects of the way files are handled in operating systems. For example, in order to be POSIX 1003.1 compliant, SVR4 has added file renaming, file truncation (previous versions of System V had no direct way to truncate a file), file synchronization, ENAMETOOLONG error (an error message meaning that a user has input a name that is too long), and NONBLOCK mode.

Other File System Changes

OPEN FILES. The number of open files per process has been increased from a limit of 20 file descriptors per process to 64. However, this default can be overridden through a tunable soft limit of up to 2,048 open file descriptors. The SVR4 standard I/O package allows a program to access as many as 256 open files, a limit required for binary compatibility. This type of flexibility is critical for creating an operating system that needs to scale from low-end 386s to mainframes.

SUPPORT FOR SYMBOLIC LINKING. SVR4 has added support for symbolic linking, supported in BSD but not in earlier versions of System V, which only supported hard links. Symbolic links are important in the network computing environment because they increase the efficiency of disk utilization by cutting down on the number of copies of files stored across the network. Symbolic linking allows different nodes on the network to share files. Symbolic links serve as pointers, allowing a file to be stored on a server anywhere on a network. They are also important on non-networked systems for easier file access and better system management. However, symbolic links in all implementations still need to be improved to be more secure and reliable so that they are less likely to be destroyed and so that files are less likely to be accidentally deleted.

PROCESS FILE SYSTEM. A new facility in SVR4, the /proc file system, generalizes the file system concept beyond physical files. This facility allows Unix processes to look like files and, therefore, to have names. It makes the debugging process easier because programmers can access processes by name as if they were simple files, rather than having to use special facilities as they did under earlier versions of Unix.

Scheduling and Real-Time Processing

While the MP version of SVR4 adds different scheduling and real-time support, even base-level SVR4 has improved the way scheduling and real-time processing are handled. A key issue for real-time is the latency time for processing switching. Preemption points have been added to the longest system calls to improve real-time processing in SVR4. The process scheduler architecture supports three classes of policies: traditional time-sharing, system, and fixed-priority processes. Fixed-priority scheduling allows users to set fixed priorities on a per-process basis. The highest-priority fixed-priority user process always gets the CPU as soon as it is runnable, even if system processes are runnable. Fixed-priority processes are never swapped, although they may be paged. To prevent paging, text and data may be locked into primary memory using the "memctl" call. With the fixed-priority policy, the system call "priocntl" can modify the scheduling class, priority, and minimum time-slice; priocntl is restricted to an effective user ID of zero.

Communications and Networking

Streams Implementation

Since its introduction as part of System V.3, Streams has been viewed as a major improvement over traditional Unix I/O mechanisms. Prior to the development of Streams, new kernel software had to be written any time a new device driver was developed in order to interface to that device. In effect, adding a new device required the Unix kernel to be rebuilt! Therefore, it is easy to understand why Streams was heralded so widely by USL as an answer to ISVs' prayers. Streams is a framework for character I/O that allows a set of standard interfaces to be placed within it, making an interface between each device driver and the kernel unnecessary. The added benefit (and perhaps the most important for distributed computing) is that Streams also hides the network protocol and media, and enables a program to link to resources across the network transparently. Whether an application will be using the X.25 protocol or TCP/IP is hidden from the developer since either can be run over any Stream.

A key enhancement to Streams is the addition of Streamed-based pipes and Named Streams facilities. Also, a TTY subsystem in the kernel has been rewritten to use the Streams mechanisms. This helps to unify the interfaces that a program uses to communicate with character devices and other processes.

Another important facility in SVR4 is Transport Level Interface (TLI), which provides for protocol and media independence. Any network conforming to the Transport Provider Interface specification can be accessed by a program using TLI.

INTERPROCESS COMMUNICATION. IPC mechanisms supported in System V.3.2 are all supported in SVR4. Several additions come from Xenix, including Xenix Semaphores and shared data. BSD sockets (features of 4.2 and 4.3) have also have been added. TCP/IP is now officially a part of SVR4.

NFS SUPPORT. Sun's de facto standard Network File System (NFS) has been added, as has Sun's External Data Representation (XDR). USL's RFS and NFS now share integrated administration.

DCE SUPPORT. Even more important than the intertwining of RFS and NFS will be USL's forthcoming support for DCE. The logjam between USL and OSF in distributed computing was broken when Unix International announced Atlas, its proposed distributed computing infrastructure. One of the key components of Atlas is OSF's DCE. USL has publicly stated that it will support DCE within SVR4. Supporting this emerging de facto standard will help keep SVR4 competitive in the eyes of users who are beginning to plan for implementation of DCE within the next three to five years. Ironically, facilities such as DCE make the differences among core operating systems less important.

NEW NETWORKING FEATURES. New facilities introduced in SVR4 include major revisions that uniformly support the many different underlying mail architectures found on Unix systems. The enhancements will maintain compatibility with ATTMAIL and DARPA mail systems, and will handle binary messages as well as the conventional text messages. Changes to mail will now allow name-to-address translation for handling the network addresses of servers in a transparent manner.

The first release of SVR4 doesn't include a lot of support for OSI communications or for mainframe and PC connectivity. USL does have a suite of OSI product that can be added as an option. We expect that USL's newly inaugurated joint venture with Novell, Univel, will handle the PC and mainframe connectivity issues well from the workstation perspective, but we wonder what Novell will bring to USL in the realm of network management for PCs. We can expect OSI connectivity to be gradually rolled into future releases when user demand increases.

Developers' Tools

One of the important changes in SVR4 for developers is the addition of the Extensible Link Format (ELF) to the Common Object File Format (COFF). These are both object formats that define how the binary and object files are formatted. This change is particularly relevant to programmers writing compilers and loading different binaries together. It is also important for developers who write linkers and debuggers. ELF, the object format from SunOS, is more suitable for high-level languages such as Ada and Cobol, which COFF doesn't support. However, because COFF has been widely accepted and used for the C programming language, the large number of utilities that have been developed using COFF are fully supported in SVR4. Gradually, programmers will want to rewrite and recompile their code to support ELF. But, in the meantime, USL promises to continue to support COFF and will provide migration tools to ELF.

Internationalization

Internationalization has been a top priority at USL in order to satisfy overseas markets and worldwide customer requirements. SVR4 implements common Extended Unix Codes (EUC) support for ideographic languages, such as Japanese and Korean. Improvements have included support of full 8-bit code sets, as well as commands that can handle code sets in which all 8 bits are used. The Multi-National Language Supplement (MNLS) includes a separation between tools and libraries. In effect, MNLS is an API between system services and each native language. This means that the operating system makes no assumptions about the national language, code set, or local conventions. This allows you to set up libraries for textual interaction with the user in the appropriate language, which can be selected at installation.

Desktop and Networked SVR4 Will Be Welcome Additions

Two of the most interesting forthcoming announcements should be of great interest to users of Unix. A version of Unix for the desktop that will be stripped of components needed by larger systems and a LAN networked version from Novell are both signs of a new and increasingly competitive Unix Systems Laboratories.

Summary

System V has come a long way over the past three years. Although it is evolving into a modular and modern operating system, this cannot happen overnight. The bulk of Unix applications are based on older versions of System V, BSD, SunOS, and Xenix. Therefore, the evolution of System V might be slower than some in the industry would like. But, now that USL is chartered with becoming a profit-oriented software company, aggressive change is inevitable and welcome. If USL can achieve its new mission, users will be well served. In the long run, SVR4 will mature beyond the 4.2 release coordinated with a desktop version and the tight integration with OSF's Distributed Computing Environment. In the future, we expect that USL will add OSF's Distributed Management Environment (DME) to its environment as well.

OSF/1: Open Software Foundation's Answer to SVR4

Although a new vision of Unix reunification has begun to emerge, it is based on a common set of application programming interfaces (APIs) for higher level services, including distributed application services, distributed management, and the like. This version of the vision also includes OSF migrating away from a USL Unix license toward compliance with standards for Unix-like operating system behavior, including Issue 3 of the System V Interface Definition (SVID 3), the complete suite of POSIX standards, and X/Open's XPG/4. In effect, OSF will be providing a vendor-neutral, Unix-like operating system called OSF/1, and USL will provide trademarked 0.11. Both products, as well as other, non-Unix operating systems, will comply with a common set of standards.

This evolution will keep the competitive pressure on both USL and OSF to incorporate the best technology the industry has to offer into their operating systems. Users should be grateful for this competitive spirit. It will force USL to keep the Unix operating system moving forward at a rapid and exciting pace by providing alternatives to customers.

Let's look at the alternative. The fact that OSF/1-based products are now available from Digital Equipment and Kendall Square Research, combined with announcements of availability from many others, signals that OSF's Unix alternative has finally reached the market. Because it represents a departure from many of the traditional Unix variants, there is a tendency to misunderstand what OSF/1 really is and where it is headed.

What Is OSF/1?

OSF/1 consists of a Mach kernel integrated with Unix functionality that is based on Berkeley 4.4. The kernel is Mach 2.5 from Carnegie Mellon University, a simple, extensible kernel that was designed for both parallel and distributed environments. The Mach kernel technology has been under development for the last six years at Carnegie Mellon under an agreement with the U.S. Defense Advanced Research Project Agency (DARPA) with contributions from other industry sources. There already are commercial systems based on the Mach kernel from a number of vendors including NeXT, Encore, and Sequent. The base code was modified by Encore Computer, parallelizing most user and kernel processing for implementation on multiprocessing architectures. Required System V functionality, such as an SVR3-compatible Streams package, has been added, allowing transparent parallelization of Streams modules.

Unix Functionality on Mach

The Unix functionality that is implemented with Mach is integrated with the kernel. This is not the modular, microkernel architecture that is the focus of OSF's microkernel development (see "Organization of the OSF/1 MK" below). Even though it is integrated, there is a cleanly defined interface between the Unix and the Mach portions in the kernel that will come in handy when the Unix functionality is separated in a future release (see "Looking Ahead" below). When OSF/1 was first announced in 1988, it was to have been based on AIX code from IBM. OSF soon recognized that it would be too difficult to get multiprocessing support into AIX and made the decision to use Mach for its kernel technology instead, in spite of the delay this would mean in getting OSF/1 out the door. Even so, AIX commands and libraries remain the source of much of the Unix functionality in OSF/1, while Mach handles scheduling, interprocess communications, and virtual memory, replacing the virtual memory support of Unix.

Organization of the OSF/1 Mach Kernel

Organization of the OSF/1 Mach Kernel

Introduction to Mach

Unix functionality coexists with Mach in the kernel of OSF/1 and provides a pageable, interruptable, multithreaded, extensible Unix. Mach's value to OSF/1 lies in its notions of tasks and threads. Mach is responsible for scheduling its multithreaded processes and supplies different types of interprocessor locks to support multiprocessor synchronization. In Mach, each task may have multiple threads. In other Unix implementations, each task has a single thread.

Mach's I/O structure is based on ports, which are both communication channels and object references. A port is best thought of as an entire communication channel, differing from Berkeley sockets, which represent only the end points of a communication channel. Ports as memory objects are entities that can be mapped into a task's address space. They might be files, temporary storage, or other objects defined by user-provided servers.

Threads and Parallelism

Mach provides for concurrency in the kernel, meaning that a single process may have multiple threads at one time, although, on a single processor system, they are not running simultaneously. The kernel handles that job for the server application by using multiple threads—one for each client—instead of having to start a separate instance of an application to handle each client. Mach also provides for parallelism, with multiple threads in process simultaneously, although this, of course, requires multiple processors. A shared memory architecture is assumed in this case, with each processor having equal access to memory. Different threads can communicate by exchanging messages, although two threads in the same task communicate more efficiently using shared memory.

OSF/1 Has Been Parallelized

In order to take advantage of Mach's multiprocessor and multithreading support, many operating system functions have been parallelized to take full advantage of a shared-memory multiprocessor. In order to accomplish this, work was required in:

- Scheduling
- Virtual Memory
- Virtual File System and Buffer Cache
- UFS and NFS
- Logical Volume Manager
- Streams
- Sockets
- Unix Domain and Internet Domain Protocols

The modifications that were made allowed these areas to take advantage of multithreading and shared memory, although they could have been implemented without doing so.

Synchronization on Multiprocessors

The Mach kernel uses a number of different synchronization techniques to eliminate conflicts in access to shared data structures. Spin locks, for instance, keep other processes from accessing a processor until the process that has placed the lock has finished. Interrupt masking is employed to ensure that certain interrupts don't interfere with a thread that is about to access or modify a data structure that an interrupt could access or modify. Spin locks are used to protect against potentially destructive effects of two threads interacting from different processors. However, spin locks must be used carefully because all threads or interrupts attempting to take the lock monopolize their processors while they are waiting for the lock to be freed.

A better solution, particularly if the lock is going to last for a long time (for example, during an I/O operation), is provided by blocking locks. In this type of lock, the waiting threads put themselves to sleep and are awakened by the process that held the blocking lock when it has completed. This frees up those processors for other work while the lock is on. When the blocking lock is removed, the process that held the lock notifies the sleeping threads of that occurrence, and they reactivate themselves. This is accomplished through an event-notification mechanism provided by the kernel that notifies all processes waiting for a particular event that it has occurred.

Synchronizing Threads and Processes

Although a thread can be put to sleep and later awakened, interrupt processing must run to completion, except for brief interruptions by higher priority interrupts. Therefore, only spin locks may be taken in the interrupt context, while both blocking and spin locks may be taken in a thread context.

Multithreaded Processes and Single-Threaded Heritage

Allowing an application to use concurrency within an application is accomplished through multithreaded processes. The standard Unix process has only a single thread of control within it. In Mach, processes may contain any number of threads of control. Since the original definition of Unix system calls assumed a single-threaded process, this can present some problems. The "fork" and "exec" system calls are examples of where single-threaded assumptions have created problems in a multithreaded environment. Use of fork and exec system calls are not really appropriate in a multithreaded environment, although applications using them are supported. Certain global variables, like "errno," could be accessed by multiple threads simultaneously in a multithreaded environment. OSF/1 deals with this by associating a separate error location with each thread, but only when the POSIX-threads (Pthreads) library is linked with the application.

Another challenge addressed in OSF/1 is Unix signals, which come in two flavors, synchronous and asynchronous. Synchronous signals come in response to a trap within a thread, clearly belong to a specific thread, and are sent to that thread. Asynchronous events, like key presses, are external to a process. They need to be sent to a process, but, if it is a multithreaded process, the question arises of which thread within that process should receive the signal. OSF/1 handles this by having the oldest thread within the process deal with the signal.

Heritage Creates Headaches

Even standard C libraries create headaches for multithreaded processes since they too were designed to be single-threaded. Many routines return pointers to statically allocated data structures. If two threads call such a routine simultaneously, both will expect to find the result in the same location, creating a conflict. In this case, OSF/1's solution is to change the interface to such routines, allowing the caller to indicate where the result is to be placed. Again, this facility is only available if the application links the Pthreads library.

POSIX-Threads Support for the Future

Obviously, the POSIX-threads package built on top of the OSF/1 operating system threads support plays a crucial role in providing support for multithreaded processes. The Pthreads library is OSF's implementation of draft 4 of the IEEE POSIX Standard P1003.4a and provides a standard API for multithreaded applications. Although the OSF/1 kernel interface for threads may or may not become a standard, it is used to support the Pthreads interface, which is, or will be, a standard. Therefore, a programmer is isolated from changes OSF may make to the interface between the kernel and the Pthreads library in the future and may use the POSIX-threads interface with confidence that it will not go away. The intent is that programmers manage threads not by using the system call interface, but by using the Pthreads interface.

POSIX-Threads Support for the Future

Scheduling in Multiprocessor Designs

There are two aspects of scheduling that involve the kernel: processor allocation and processor-sharing. Processor allocation entails user-controlled partitioning of the processors to satisfy application requirements. On the other hand, processor-sharing has to deal with the equitable sharing of processors among the threads that are running, and it has to take into account threads priorities.

The OSF/1 kernel partitions processors into containers for threads called processor sets, each of which contains zero or more processors. Any one processor can only be in one container. A thread may run on only one processor in a processor set. Also, two types of queues are maintained, global run queues and local queues. Each processor set has one global run queue and each processor has a local run queue. Threads that are just becoming active are placed in the global run queue to wait until a processor is available. Once running on a given processor, the thread will go into a local queue if it is interrupted and will resume running on that processor when it becomes available again. The exception is the case of those few threads that are specifically written to run on a specific processor, i.e., the threads have processor affinity. In that case, they go directly to that processor's local queue. A processor always looks to its local queue first for a thread to run before taking a thread from the global queue.

The system maintains a list of idle processors which have no thread running. If this list isn't empty when a thread is made runnable, then the agent making the thread runnable selects the first processor in the idle list and dispatches the newly runnable thread to that processor.

Scheduling Policies

OSF/1 supports two scheduling policies, time-shared and fixed-priority policies. Determining which of the policies is allowed is a property both of the thread and of the processor set. Time-shared policy means that sharing of the processors among the various threads is equitable. Fixed-priority policy provides preferential treatment to particular threads.

Virtual Memory

The concept of lazy evaluation is used throughout the OSF/1 virtual memory system. It is based on the principle: "Postpone everything until the last possible moment; if you put something off long enough, maybe you won't have to do it." [Editor's note: I thought I invented this in high school.] It is an effective optimization, since many operations often turn out not to be necessary. For example, you don't want to load all of an application's code into memory if only some of that code is actually going to run. Applications sometimes allocate memory but do not use it, and it is left unavailable to other applications. Lazy evaluation helps save memory that ends up unused.

OSF/1 Supports Multiple File Systems

OSF/1's Virtual File Systems

OSF/1 supports multiple file system types, e.g., UFS, NFS, and AFS. The virtual file system (VFS) is a generalization of the standard file system data structures used to represent the different types. The scheme is based on Sun's virtual file system technology with code adapted from BSD 4.4. The VFS abstraction provides a common interface to the many different file systems that are supported.

File System Options

OSF/1 currently supports the local Unix file systems, System V and UFS, and a reimplementation of Sun's NFS. Within the virtual file system, *vnodes* are the abstractions of individual files, containing generic information about files, and they refer to the file-system-specific information on files, e.g. *inodes* for Unix files and *nfsnodes* for NFS files. This is essentially the same as in SVR4.

Support for the System V file system is primarily for compatibility purposes, since it is almost always slower than the UFS file system. OSF/1's UFS file system is a parallelized version of the Berkeley file system. It lays out files on disk so that they can be accessed as quickly as possible and so that only a minimal amount of disk space is wasted. It allows longer component names than the System V file system. OSF/1's NFS is a parallelized implementation of the NFS that is a part of BSD 4.4.

Logical Volume Manager (LVM) Offers Robust Functionality

Role of the LVM

The Logical Volume Manager (LVM) is based on technology provided by IBM and exists as a layer between the physical disk volumes and the files systems. It presents a device-driver interface to the file system and provides for the notion of logical volumes so that:

- Logical volumes may span multiple physical volumes.
- Logical volumes may be mirrored on multiple physical volumes.
- Logical volumes may grow and shrink under the control of the administrator.
- Logical volumes support software bad-sector remapping.

Unix files have always been limited by their inability to span multiple volumes, but, since logical volumes can span multiple physical volumes, this restriction is removed in OSF/1, just as it has been in SVR4.

Managing Logical Volumes

Logical volumes are organized within volume groups that contain a collection of both logical volumes and physical volumes. Logical volumes are divided into logical extents, the size of which may be any power of 2 between 1MB and 256MB. Each logical extent is mapped to one, two, or three physical extents on physical volumes. The size of physical extents is equal to the size of logical extents, which is the same throughout a volume group. Logical volumes appear to be real devices to most of the system, so they have a name as a special file within the /dev directory.

Each logical volume contains a single file system. The size of the logical volume may be easily changed by adding or removing logical extents and associating them with physical extents. In OSF/1 Release 1, neither the System V nor the UFS file systems support the notion of growth or shrinkage in a file system's underlying volume. This will be addressed in Release 1.1.

Mirroring: A Key to Performance and Availability

The LVM supports mirroring and augments the bad-sector remapping provided by the hardware. For mirrored volumes, the LVM can fix newly detected bad sectors by relocating the sector, reading the mirror, and writing the data into the relocated sector. Disk-mirroring provides better speed and crash recovery to OSF/1. Read accesses to a logical volume are accelerated by translating a read of a physical extent to the least busy physical volume. The performance gain is most noticeable on volumes that are read-mostly, including logical volumes that contain binaries. This functionality is similar to that provided by Veritas in SVR4.

OSF's Streams and Sockets Implementations

Streams Implementation Included in OSF/1...

OSF/1 Streams are a reimplementation of SVR3 Streams that have been parallelized. The parallelization work was done in such a way that implementing an SVR3 Streams module is transparent, i.e., does not require any change to the module's code. The OSF/1 kernel allows fully parallel execution with essentially no changes to the code, apart from some synchronization that has to be implemented within the standard routines that are called.

Streams are the kernel analogy to pipelines as used in the shell and are bidirectional channels for messages being processed in one or more modules. The end points of a kernel stream can be in two user processes, or, more commonly, one end point may be in a user process and the other in a device driver. Because of the high number of Streams modules that are likely to be active at once, it was not desirable to implement Streams using kernel threads because of the overhead. Instead, each stream module is a collection of procedures that can be called in a variety of contexts. There are data structures associated with each module that contain its state, so the module's execution can be started in the context of one caller and continued in the context of another.

... Along with Berkeley Sockets...

OSF/1 uses Berkeley's socket model to implement its communication protocols. The actual code that is used is a parallelization of BSD 4.4 sockets. To interface to this layer, OSF recommends using the X/Open Transport Interface (XTI), an enhancement to AT&T's Transport Layer Interface (TLI). SVR4 does not support XTI yet because the standard was published after SVR4 was released. Using XTI results in code that is more portable than if one communicated directly with the socket layer.

...And Made to Work Together

Unfortunately, sockets and Streams constitute two competing network interfaces, one from BSD and one from System V. While there are probably more applications currently built on sockets than on Streams, that may be changing. Even so, both still have to be supported. SVR4 supports sockets with an emulation library in user mode. OSF/1 supports both in the kernel without an emulation library. Instead, it provides a means for accessing a protocol implemented in the socket framework through the Streams interface. This mechanism is the X/Open Transport Interface to Sockets (XTISO). The XTISO Stream consists of a standard stream-head module, a fairly simple timod module, and the XTISO driver module, which is the interface to the socket level. The XTISO driver takes transport interface (TPI) messages and converts them into operations on sockets.

Dynamic Configuration Makes System Management Easier

Many system components can be added to a running OSF/1 system dynamically because of the dynamic configuration feature, a feature that SVR4 lacks. Typically, in BSD-style autoconfiguration, device drivers are statically linked into the kernel. At boot time, autoconfiguration code determines which devices are present and "activates" the appropriate drivers. OSF/1 supports this, but, in addition, it can dynamically add or unload device drivers, file systems, Streams modules and drivers, and network protocols. This is accomplished using the run-time loader discussed below (see "OSF/1's Loader Adds Flexibilty"). The run-time loader links the driver to the rest of the operating system, but the loaded driver is responsible for linking the rest of the operating system to itself.

OSF/1 provides an approach for adding and removing interrupt handlers dynamically, although this involves highly machine-dependent facilities and must be tailored for each machine architecture. In the BSD kernel, the notion of interrupt vectoring is "wired into" the kernel and there is no convenient way to add interrupt handlers dynamically.

OSF/1's Loader Adds Flexibility

Role of the Loader

While OSF/1 supports older style program invocation where a program has been fully bound and relocated, it also allows much of the binding and relocation to be postponed until runtime. OSF/1's run-time loader can cope with load formats unrecognized by the kernel, linking the image to shared libraries, loading additional modules as required, and relocating the entire image as necessary.

The loader can be used to load or unload modules from the kernel as well. It is used in conjunction with dynamic configuration to support loadable/unloadable device drivers, Streams modules and drivers, file systems, and protocols. Kernel loading is managed by a privileged user-mode task, the kernel-loader server.

Shared Libraries Save Resources

An important feature of the OSF/1 loader is shared libraries. Programmers often don't realize what happens to system resources, especially memory, as repeated instances of the same functions and libraries are called by different programs. Whether stdio or Xlib, as libraries are linked, the image grows and grows. In OSF/1, the loader prelocates libraries, combining them into a single image that is located to fit at a fixed location in the address space. The libraries are then available for linking to programs without any further relocation. The standard libraries are already provided in this format and are linked in this form with the standard Unix commands.

Modules may be explicitly loaded or unloaded from a running program. To do this, the runtime loader remains in the address space even after the program starts up. The user program can call the loader by its load and unload entry points. Since shared libraries may be dynamically linked, they represent another foundation block for object-oriented environments.

Security

It is important to keep in mind that security certification requires a formal evaluation process, and that not just the operating system but the implementation of the operation system on a particular hardware architecture with a specific option set is evaluated. In addition, the U.S. Department of Defense "Orange Book" criteria for secure systems is limited to standalone systems—a networked system cannot be secure under the criteria. There are separate criteria that must be addressed for networks found in the "Red Book," or Trusted Network Interpretation. OSF/1 does not address "Red Book" requirements at this time. DCE addresses some issues, as will future releases of OSF/1.

Regarding "Orange Book" requirements, OSF/1 can be compiled to be either C2 or B1 compliant, or neither. Security has been implemented using technology from SecureWare. It provides for auditing, discretionary access control (DAC), mandatory access control (MAC), and management of authorizations and privileges. Security is set at compile time, not at boot time. Since secure systems may be slower in performance than nonsecure systems, not all installations may want security.

OSF/1 has augmented traditional Unix discretionary security policies with the use of access control lists (ACLs). In addition, it breaks down the traditional privilege classes of superuser/mere-mortal into:

- Root replacements—a set of rights that can be individually granted.
- Unix mode—privileges that ordinary users have in Unix may be restricted in OSF/1, e.g., one must have the execsuid privilege to execute SETUID programs.
- Trusted mode—privileges allowing a process to operate in modes that gain it special treatment with respect to trusted system features.
- Trusted function—privileges allowing a process to define new trusted functions, e.g., the writeadit privilege allows a process to append records to the audit trail.

Internationalization

Locale databases are supplied in OSF/1 for Belgium, Canada, Denmark, Finland, France, Greece, Italy, Japan, the Netherlands, Norway, Portugal, Spain, Sweden, Switzerland, the United Kingdom, and the United States. Multi-National Language Support (MNLS) support extends to the syntax for regular expressions. While separate message catalogues are supplied for each locale, some programs don't use the message catalogues, including the operating system kernel (panic messages are in English), ftpd (the protocol sends English ASCII text back to indicate serious configuration problems, and gcc (the GNU C compiler), which is not internationalized. Asian-language support is achieved through support of the Shift-JIS encoding scheme.

Looking Ahead

Release 1.1

OSF will be rolling out OSF/1 Release 1.1 later in the year. The primary enhancements in Release 1.1 are in the areas of internationalization, SVID Issue 3 compatiblity, which defines System V Release 4 interfaces, and scalability. Becuase it is not a major upgade, it should be available to customers by mid 1993.

Internationalization in Release 1.1

In addition to the Shift-JIS and 8-bit clean support provided in Release 1.0, Release 1.1 will include Extended Unix Codes (EUC) support for ideographic languages and conformance to X/Open's XPG4 draft specification for wide-character interfaces. Also, a Streams-based terminal (tty*)* and pseudo-terminal subsystem will be provided to handle supported encodings that will give greater flexibility to vendors. Release 1.1 will also include internationalization of system administration commands for security and the LVM.

SVID Issue 3 Compliance

Release 1.1 is compatible with the System V Interface Definition for System V Release 4 for base and kernel extensions, including Streams. Obviously, if a future POSIX standard comes to be in conflict with the SVID, OSF would follow the POSIX standard. Equally obviously, the SVID would also change to follow the POSIX standard.

Scalability

A feature of Release 1.1 will improve operating system scalability, with the initial focus on smaller systems. The base configuration for Release 1.1 will be a system with as little as 4MB of memory.

OSF/1 MK

Also ahead is the development of an OSF/1 operating system based on Mach 3 microkernel technology. Whether or not Release 1.1 and OSF/1 MK eventually merge will depend on the requirements that emerge from the market. The microkernel in the MK release is a minimalistic kernel, with as much function as possible migrated into user space, reducing the number of transitions to supervisor mode required to get work done. There are some applications where a microkernel is actually less desirable, so it is not necessarily a foregone conclusion that the future direction of OSF/1 is to the microkernel. Further enhancements to the Release 1.0 code base are not likely to occur for some time.

Conclusions

OSF/1 is real, and it is available. Does anyone care yet? OSF members care. Users are watching intently, but they are forced to wait until applications have been ported from their favorite Unix flavor to OSF/1 before they can really begin evaluation. The pace at which this happens will be driven by the vendors and will depend on how fast they roll out their OSF/1 products and how much support they give ISVs in the porting effort. The more systems that are sold with or are converted to OSF/1, the better the opportunity ISVs will see for an OSF/1 market. Over 100 ISVs have already announced their intention to develop for OSF/1 in conjunction with Digital's OSF/1 product announcement, and more can be expected to follow.

Penetrating the market with a new operating system is a painful process that is, ultimately, driven by application software. With proper development, once an application is ported to one vendor's OSF/1, it will be source code compatible with any other's, just as any application written to POSIX standards or to X/Open's XPG will be portable. Digital's inclusion of support of Ultrix binaries is a slick migration strategy that allows existing applications to run under OSF/1 while they are being ported to native OSF/1. Other vendors can be expected to take a similar path.

Conclusions

The biggest test that faces OSF/1 is to prove that it is a commercial grade product ready for prime time. Release 1.0 of any product can be scary stuff. If OSF has brought out a relatively bug-free operating system that licensees can port easily to their platforms, its credibility in the market will go up dramatically. On the other hand, if the porting effort is any where near as difficult as SVR4 licensees found, OSF will have to find other ways to establish itself. OSF/1 is a modern operating system with a bright future. But without customer experience yet, all we can say is, "Let the best OS win!"

Next month's *Unix in the Office* will address
Europe's Harness Project

For reprint information on articles appearing in this issue,
please contact Donald Baillargeon at (617) 742-5200, extension 117.

$200 OFF

The Fourth Annual

Executive UniForum Symposium ▲▲▲

Migrating to Integrated Open Systems: Tools, Tactics, and Tradeoffs

May 5 - 7, 1992, Santa Barbara, California

Sponsored by: UniForum, X/Open, and Patricia Seybold's Office Computing Group

Registration Fee: $895, if paid by April 5, 1992
$1095, if paid after April 5, 1992

The 1992 Executive UniForum will go beyond basic concepts and abstract issues—it will zero in on the issues and challenges of actually building and delivering business applications that leverage the best that open systems have to offer.

Executive UniForum is designed for decision makers and managers responsible for developing and implementing information technology strategies for their organizations. This conference will be of particular interest to individuals whose organizations are planning to migrate from proprietary systems to an approach based on open systems interfaces and technologies.

To register, or for more information:

Call: (800) 826-2424 or (617) 742-5200 **Fax:** (617) 742-1028

Mail: Patricia Seybold's Office Computing Group
148 State Street, 7th Floor, Boston, Massachusetts 02109

Open Systems: Analysis, Issues, & Opinions

FOCUS: STANDARDS SUPPORT

POSIX-Compliant Proprietary Operating Systems Demonstrated

The key to a vendor-neutral definition of open systems is the specification of system interfaces that are independent of the underlying technology. In the case of operating systems, the work of the IEEE POSIX standards committees is far from complete. However, efforts are far enough along to let us begin to think about true standards-based portability.

Much of the work of the 1003.1 committee on System Services and 1003.2 committee on Shell and Utilities was based on the System V Interface Definition (SVID). For that reason, there had been a tendency to assume that POSIX only applied to Unix operating systems. In theory, there was no reason why most non-Unix operating systems could not support the POSIX standard, but announcements by many vendors of proprietary, non-Unix operating systems of their intent to support POSIX were met with skepticism by the Unix community.

UniForum Announcements

At UniForum in San Francisco, however, theory became reality as Digital Equipment, Hewlett-Packard, and Unisys demonstrated POSIX-compliant versions of their proprietary operating systems, VMS, MPE, and CTOS, respectively. Some areas of POSIX support have had to be based on draft standards. Nevertheless, these products raise many interesting possibilities for open systems based on non-Unix operating systems.

The joint announcement centered on the three vendors who have licensed 1003.2 capabilities from a small Canadian software company, Mortice Kern Systems (MKS), and demonstrated code portability across the three platforms for a real application (IXI's X Desktop) written to POSIX interfaces.

A brief review of the relevant POSIX standards will help place the event in context.

POSIX 1003.1—SYSTEM SERVICE INTERFACES AND C LANGUAGE BINDNGS. The 1003.1 standard defines basic operating system services and describes how POSIX applications access system-level services. Examples of the types of services that are covered by 1003.1 are process creation and execution, file system access, and I/O device management. Applications written to the 1003.1 interface are shielded from the specific system on which the application is compiled and run. This standard has been voted and approved.

POSIX 1003.2—SHELL AND UTILITIES SERVICES. The 1003.2 standard is close to formal approval. It specifies a shell command language based on the Unix Bourne shell with some features from the Korn shell. It provides both an interactive interface to shell and utility services as well as a callable interface. It also provides commands and utilities with many of the same features and functions from the Korn shell.

POSIX 1003.2a. POSIX 1003.2a, an extension to 1003.2, is a set of enhancements and additional commands designed to provide a consistent user interface across systems for character-oriented terminals only.

POSIX 1003.4—REAL-TIME APPLICATION SERVICES. The POSIX definition of real-time is that the system must have the ability to provide a required level of service with a predictable, bounded response time. There are three parts to this effort: the real-time file system, the multithreaded architecture, and other services, including shared memory semaphores and signals. Real-time is a little further away from adoption than 1003.2. The 1003.4 extensions provide support for such functions as enhanced interprocess communication, scheduling and memory management control, and asynchronous I/O operations.

The MKS Connection

MKS is a Canadian-based software company that develops and markets developer tools and a standards-conformance offering called InterOpen. InterOpen is actually a family of products and services that are designed to provide POSIX conformance for their customers' operating systems. InterOpen customers are primarily system vendors, and they include Digital, HP, and Unisys. MKS is working with at least three other vendors, whom we speculate to be Microsoft (NT), IBM (MVS or OS/2), and Tandem (Guardian). MKS's POSIX products and services include:

- InterOpen/POSIX.1 Conformance Evaluation Services—on-site conformance testing
- InterOpen/POSIX.2 and POSIX.2a source code products—300,000 lines of portable, source code unencumbered by a USL license, which includes documentation
- InterOpen Porting and Consulting Services—a range of services designed to help a vendor through the entire compliance process, from initial specifications through porting and testing

Strategies behind POSIX Conformance

The three vendors who were demonstrating POSIX conformance at UniForum all have somewhat different motivations and approaches to POSIX.

Digital Goes for the Gold. In many ways, Digital has more to gain by introducing POSIX compliance for VMS. Unlike HP, which pushes Unix much more than it does MPE, Digital has a blatant two-operating-system strategy. VMS will continue to be a strategic operating system for Digital on its new generation of Alpha systems through the end of the decade. While Digital believes that existing VMS customers will migrate from the current VAX architecture to the Alpha architecture without leaving VMS, the company also believes those same customers will be looking at open systems when they are evaluating the move. Digital realizes that, if VMS is going to compete with Unix-based systems, the issue about support for open systems' standards has to be neutralized. The best way to do that is to fully support open systems standards on VMS.

In fact, Digital is being much more aggressive in support for draft POSIX standards than other vendors, particularly in its support of draft 9 of the 1003.4 standard. POSIX pricing is also very aggressive when compared to other vendors' offerings: $165 above the VMS 5.5 license for media and documentation, or $340 with full IEEE documentation. VMS 5.5 includes the right to use POSIX. The offering has been available to customers since the end of February.

A Smaller Opportunity for HP MPE/iX. Hewlett-Packard achieved a major breakthrough in the mid 1980s on the HP 3000 and HP 9000 series with the first port of both Unix and a proprietary operating system to the same RISC architecture. With the announcement of a POSIX-compliant release of MPE, MPE/iX, HP now bridges the gap between those two environments. Although support of MPE customers and continued enhancement of MPE is a commitment of HP, MPE does not play the same role in HP's future strategy as VMS plays in Digital's.

MPE/iX is not a separate offering; it is the latest release of MPE and has support for 1003.1 integrated in the kernel as well as support for other services that are not POSIX standards but are necessary to run Unix-derived applications. These include support for OSF Motif; Berkeley sockets, AT&T SVID Interprocess Communication (IPC) and curses.

At UniForum, HP demonstrated its POSIX.1 implementation as well as POSIX.2, using the MKS technology. POSIX.2 is expected to be available mid 1992. HP has not yet announced plans for 1003.4 support.

HP supplies a POSIX Development Kit for HP 3000 computers that contains:

- A library to support the 1003.1 standard interfaces
- Support for the 1003.2 standard shells and utilities
- An ANSI-compliant C compiler
- The Network File System (NFS) for MPE
- A library to support AT&T's SVID IPC
- A library to support curses
- Documentation in the form of programmers' manuals
- Support services, including up to 40 hours of engineering support

Pricing for the development kit is processor based, and ranges from $10,000 for the HP 3000 Model 917LX to $50,000 for the HP 3000 Model 980. The release of MPE/iX that supports 1003.1 is due by mid 1992. Since this is the standard release of MPE, there is no incremental pricing.

The development kit is for developing new POSIX-compliant applications on the HP 3000 or porting POSIX applications to it. In some instances, a developer may wish to program for the HP 3000 from within a Unix development environment. In that case, the developer simply uses the normal workstation environment, sticks within the POSIX interfaces, and then uses NFS to move the source to the HP 3000 and compiles and links it there.

HP's strategy for pricing the full POSIX development environment stands in marked contrast to Digital's. While MPE/iX by itself provides a base level of support for POSIX source code, additional services are necessary for development of POSIX applications on the HP 3000. The pricing strategy may stem from an expectation that this kit will be a low-demand item and from HP's desire to at least recoup its costs. Digital, on the other hand, foresees a large volume opportunity for VMS POSIX.

UNISYS CTOS

CTOS may be one of the industry's best kept secrets. CTOS used to stand for Convergent Technologies Operating System before Convergent Technologies, which was an OEM supplier to Burroughs, was acquired by Unisys. A CTOS machine is hard to categorize, since it is both a server and a workstation, but whatever it is, almost a million of them are installed worldwide.

One of the largest customers for CTOS products has been the federal government. It is no wonder, then, that Unisys was very quick to get POSIX support onto CTOS as quickly as possible. CTOS was the first non-Unix operating system to be certified POSIX 1003.1 compliant, receiving certification in September 1991. (In fact, it was during CTOS's tests that a flaw in the POSIX certification test suite was found. The test suite is supposed to be architecture neutral. But, when CTOS was run, it was discovered that the test suite expected to find a 32-bit operating system. CTOS is 16-bit. The test broke, and the suite had to be rewritten. Unisys is planning a migration of CTOS to being 32-bit, but no availabiity date has been set.)

POSIX.1 is supplied as a separate product with single-user pricing of $150; for a cluster (supporting an average of 24 users), the price is $750; and for the LAN configuration (supporting a maximum of 128 users), the price is $1,125. The Microsoft C compiler is required to compile applications for the POSIX.1 interface.

Unisys will be providing POSIX 1003.2 support in CTOS in a product by the end of second quarter 1992. Pricing for this support has not yet been announced, but it will be higher than the POSIX 1003.1 support.

Benefits to Customers

What do customers gain from these products? Three primary benefits. First, applications that are written to POSIX interfaces can have their source code compiled and can run with a minimum of porting effort. Second, a developer can be working in his or her native environment and produce POSIX-compliant source code that can then run on any other POSIX-compliant system. Third, users trained in a Unix environment can move around to other environments without having to be retrained. In each case, the benefits are reduced development effort, greater opportunity to leverage applications, and a greater assurance of compatibility.

As the open systems movement progresses, the competition to provide the best environments for supporting standards like POSIX between Unix and Unix-like operating systems and proprietary operating systems will continue to heat up. The competition is healthy. It should help the standards process, and customers will benefit from better technology and more competitive business practices. —*M. Goulde*

FOCUS: DIGITAL EQUIPMENT CORPORATION

Digital Takes the Wraps off Alpha

In briefings and scientific papers, Digital Equipment has begun peeling back the covers on the Alpha Project. As you know, Alpha is the code name for Digital's next-generation RISC microprocessor and the systems that will be built around it. The Alpha RISC microprocessor was described in February in a presentation to the 1992 International Solid State Circuits Conference (ISSCC92) meeting in San Francisco. However, Digital has been holding roundtable discussions for industry analysts for the past few months in an effort to lay the groundwork for unleashing Alpha on the world.

Alpha—Today's Hot Processor

Digital claims that Alpha will make clear that company's world leadership position in microprocessor technology. Other microprocessor developers may disagree, but no one will dispute the prediction that Alpha will put Digital's hardware back in the top tier of performance.

The first generation of Alpha processors is called EV4. Actually, EV4 is the second generation, since there was an EV3 design that was used to build 50 or so "Application Development Units" (ADUs) for software development. ADUs are configured with between one and four processors. EV3 is pin compatible with EV4 but lacks the chip's on-board floating point unit. Internal software developers are using the ADUs to port

OSF/1 and VMS to the Alpha processor. We believe some of Digital's most important ISV partners, like Oracle and Microsoft, also have had access to ADUs. All ADUs have been upgraded to the EV4 chip.

EV4 will have 1.68 million transistors (not a remarkable number, but time-to-market is a consideration here), will run at between 150 and 200 MHz (a remarkable speed), at 3.3 volts (a remarkably low voltage—standard in the industry is 5 volts), and will be able to run two instructions per cycle. The architecture includes four units, integer, floating point, address, and branch, all integrated on a single chip. Alpha has a full 64-bit architecture. While the chip can process 400 MIPs at peak instruction issue rate, in actual practice, we estimate it will run at between 150 and 200 MIPS, about twice as fast as anything else out there at the moment. This even includes HP's next-generation PA-RISC 7100, which will operate at speeds up to 100 MHz and is claimed to be capable of up to 120 SPECmarks. Alpha EV4 will easily exceed this performance, and EV5, due in 1993, is likely to be a year ahead of HP's next generation.

Future Timetable

Digital's plans for succeeding generations are:

	1993	1996	1999
Generation	EV5	EV6	EV7
No. Transistors	4M	10M	30M
Clock Speed	225-275 MHz	325-375 MHz	450-500 MHz
MIPS*	300 MIPs	425 MIPs	575 MIPs

* Seybold estimate

Digital believes that new Alpha systems, deliverable in the first half of 1993, will vault it to the forefront of the industry in performance. The first generation of systems will be high-end servers and workstations, although they will be priced to provide leadership price/performance. The MIPS products will fill out the midrange and low-end product line. The second generation of systems, appearing late in 1993 and 1994, will probably begin to replace the high end of the MIPS products, with the low-end MIPS the last to go in 1995-1997. Although current MIPS DECstation customers might be concerned about the migration to a new architecture, this shouldn't be a major issue, since they will have all migrated to OSF/1 by then. Porting applications written for OSF/1 from MIPS to Alpha when the need arises should be relatively straight-forward.

Operating System Support

The Alpha systems will run a port of VMS and OSF/1, and it is fairly safe to assume that Windows/NT will follow. Migrating software over to Alpha can occur in one of two ways. If the software is written strictly to software interfaces and in one of the languages whose compiler has been ported, then a recompilation of the source is all that will be required. If not, then a "binary image conversion" can be performed to transform the VAX/VMS binary to an Alpha/VMS binary. Some of VMS is being ported using binary images since some was written in languages that will be slower to be ported (e.g., PL/1). Porting OSF/1 applications, including Ultrix applications, should be very straightforward. Digital's approach to marketing Alpha reflects its Open Advantage business strategy. Other companies may either license the Alpha design or buy chips directly from Digital or another chip-maker. Digital correctly recognizes that, if it is to be something other than an also-ran in the industry, it must have a dominant chip architecture, but that architecture must be open to others.

Alpha: Are You Going to DECWorld?

The big coming out party for Alpha systems is likely to occur before DECWorld, which opens in Boston on April 27. Since DECWorld consumes a large chunk of Digital's marketing budget, it is inconceivable that Alpha-based systems won't at least be shown as a technology demonstration. It is more likely that Digital will have program-announced the first systems by then, with delivery by the end of calendar 1992. We also expect that Digital's software partners and best customers will receive early support on Alpha to assist them in migrating their mission-critical applications. We plan extended coverage of both the hardware and software aspects of the announcement afterward. The importance of the hardware is that its success will provide the basis for a healthy Digital moving on into the 1990s. *—M. Goulde*

LETTERS TO THE EDITOR

Downsizing with Open Systems

The article in the January issue of *Unix in the Office* tackles one of the major issues facing organizations today: how to effectively take advantage of more flexible, less expensive, mainframe-equivalent computers without compromising system functionality.

I thought your readers would be interested in the ways our company's product line, the Data General AViiON, has been supporting downsizing.

- We have been shipping symmetric, multiprocessing, RISC-based servers since 1989. Today, AViiON's SMP servers are available in single, dual, and four-way configurations, with higher performance yet to come. The initial TPC-A benchmarks on our AV 5225 dual processor system deliver $11,498/tpsA-Local.
- In January 1992, we concluded an agreement with Dun & Bradstreet Software to port key D&B software to Data General's AViiON Unix-based servers. The agreement is significant because it enables D&B's 12,000 mainframe customer sites to significantly reduce the costs of running their mainframe applications without compromising mainframe performance.
- DG/UX 5.4 also delivers significant data integrity and availability through features such as software-logical disk-mirroring, fast-recovery file systems, and rapid failover in a dual-ported system environment using fault-tolerant RAID 5 disk subsystems.
- Serviceability is ensured through capabilities such as machine-initiated service calls to country service centers, which automatically register problems for review by expert personnel.
- In the area of systems management, Data General functionality includes system capacity planning and performance monitoring, system network management, backup and restore facilities, and others.

These capabilities have helped many Data General customers to replace their mainframes and achieve significant operating and organizational savings.

As always, thank you for the on going, valuable analyses. I look forward to your future research on downsizing issues.

Stephen Paul Baxter
Vice President, Corporate Marketing
Data General Corporation

Thanks for the additonal information. It will serve as a good supplement for our readers. DG has done noteworthy pioneering work to make Unix "ready for prime time" in commercial accounts. —*Editor*