VMS POSIX'S CHALLENGE TO UNIX

Although they started out together, POSIX and UNIX are on the road to rivalry.

BRADFORD T. HARRISON

IN THE MID-1980s, while Digital was selling an unbelievable number of VAXs using the "one architecture, one operating system" sales pitch, UNIX was quietly making its way out of the universities and research centers and into corporate America.

By the time myopic Digital finally realized what was happening, it was too late. The company's handful of ULTRIX and related products was no competition for the likes of Sun Microsystems and Silicon Graphics on the West Coast and Apollo Computer on the East Coast. VAXstations and MicroVAXs took a serious pummeling, and DEC stock took a bath. Digital's DECstations and DECsystems arrived too late to stop the tailspin. The party was over.

But now it might be starting again. The VAX has continued to sell briskly (though at a decelerated pace), and VMS continues to challenge UNIX as a popular general-purpose multitasking operating system. There are some 500,000 VAX/VMS systems in the field, and there are more than 10 million VMS users. Digital continues to invest heavily in the operating system.

Meanwhile, UNIX is faltering as AT&T — via Sun Microsystems, UNIX Int'l and UNIX Systems Labs — attempts to gain control over the operating system it accidentally lost long ago. Products such as OSF/1, developed in direct opposition to the intentions of AT&T, are beginning to look decidedly un-UNIX-like. OSF/1 is based more on the capabilities of the Mach kernel than on generic UNIX. Apart from POSIX, UNIX, the so-called open platform, is proving to be the source of more incompatible applications programs and contentiousness among computer companies than any other product in the history of the industry.

Furthermore, the groundswell of support for POSIX as a standard independent of UNIX indicates that it is not UNIX that is so popular, but the concepts of portability, scalability and system software layering. All of these are long-time hallmarks of VMS.

UNIX, it now appears, could go down in history as a useful "reference platform" that spawned products such as the Open Software Foundation's DCE, X/Open's XPG and IEEE's POSIX. These have little to do with the generic UNIX kernel, so they are developing a life of

their own. They may very well find their most popular platform in VMS rather than UNIX, since VMS has the required installed base, such features as clustering, and the unflagging support of a computer company that puts technical research and development at the top of its list of priorities. Ironically, because of UNIX, VMS may be reborn.

Alpha/POSIX

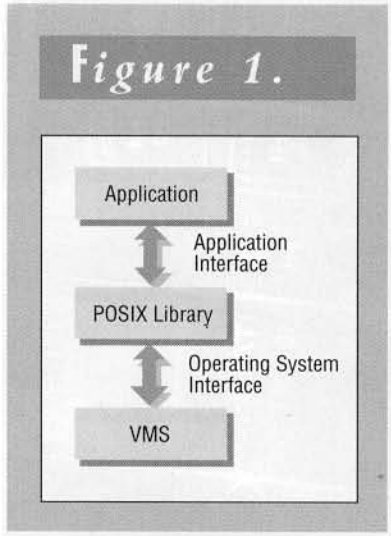
The renewal appears to be under way. DCE on VMS is in the works, XPG3 is in field test, and POSIX.1, .2 and .4 compliance are already here in VMS V5.5. In addition, Alpha systems will soon arrive to provide the raw horsepower required to drive all the VMS software needed to support these standards along with the traditional VMS capabilities.

With Alpha POSIX, Digital can directly address the concerns of the market that allowed RISC/UNIX to steal VAX/VMS market share. RISC/UNIX was a technology concept rather than a specific product implementation. Hewlett-Packard, with its powerful Precision Architecture and proven HP/UX UNIX-based operating system, was largely responsible for propagating the idea that UNIX running on a RISC architecture represented the state of the art, while proprietary VMS running on the clunky old CISC VAX architecture was a historical concept. Sun, for its part, also seemed to have a good time bashing VMS, and continues to do so.

But now Alpha is every bit as powerful as its RISC counterparts across the industry, and POSIX compliance in VMS provides the same portability and sophisticated programming environment that HP and Sun claim are the hallmarks of UNIX. There are some architectural features of Alpha that provide a close fit with VMS, similar to the original VAX architecture, and Digital is initially supporting the current 32-bit version of VMS for Alpha, while Alpha for OSF/1 will support 64 bits in the first implementation. DEC is supporting 32-bit VMS to ease the transition to Alpha for VMS users. In addition, Digital will continue to improve the performance of the CISC VAX architecture with newer, faster chip technologies.

Furthermore, Digital is now licensing both VMS and Alpha to whoever wants them. Although the company is expected to maintain tight control over use of the technologies, the practice will ensure that Alpha VMS is kept competitive with the many RISC/UNIX implementations now available and will ensure a solid supply of Alpha chips from several sources.

Even more important, VMS is the first proprietary operating system to truly take the plunge into the world of standards-based computing. According to X/Open's Xtra World Open Systems Requirements Survey (see box, "X/Open: CAE, XPG and OSD"), X/Open's Common Applications Environment (CAE) and POSIX will prove to be the most important operating system interfaces in the coming years. VMS, according to X/Open, has been the subject of more work toward XPG3 and CAE compliance than any other non-UNIX operating system. Two-thirds of the sample in this X/Open survey held the top-level information technology management job at their company or reported directly to the person who did.



The distinction between the operating system kernel and libraries is allowing non-UNIX operating systems to move into the open systems fold.

The trend toward so-called "standard proprietary" operating systems is evident. Some companies, such as Mortice Kern Systems (MKS), which provided Digital with the source code that became VMS POSIX.2, have made markets out of providing the proprietary world with software that will make these systems standards-compliant.

According to Richard Kittle, VMS marketing manager, Digital is addressing three markets with "standard proprietary" VMS: the installed base of VMS users, competitive non-UNIX operating systems, and users who already have UNIX installations.

"VMS users want to keep what they have," says Kittle. "Foremost, we have to consider the needs of that base. POSIX opens a whole new realm of applications to them without having to change operating systems." POSIX-conformant applications can coexist, as well as interoperate, with traditional VMS applications on the same system.

The huge installed base of competitive non-UNIX operating systems is an almost equally important market. "These people are increasingly interested in a combination of things," says Kittle. "They want openness plus the features currently available only in proprietary systems. They want price and performance a la UNIX, as well as the high integrity of VMS."

Finally, Kittle says that VMS POSIX will also have a strong appeal to users who already have UNIX installations. POSIX originally came from the UNIX world, and POSIX.2 gives these people a familiar programming environment, with all the VMS capabilities also available. "But the POSIX in VMS effort is complementary to the ULTRIX/OSF effort," he says. "Openness means giving users a choice."

UNIX Versus POSIX

The openness question is at the heart of the UNIX versus POSIX debate. Since UNIX has been ported to many platforms, it has become synonymous with the term "open systems." UNIX is relatively easy to port, since it is a small operating system written in a fairly high-level language, C. OSF/1, for its part, is regarded as highly portable, since there

is a strong separation in the operating system between the machine-dependent and machine-independent code.

The UNIX kernel is accessed using system calls. System calls allow the programmer to access kernel facilities such as the filesystem, the multitasking mechanisms and the interprocess communication facilities. System libraries are made up of binaries that use system calls to provide an interface to the operating system for application programs, although not all library routines use system calls.

As the concept of open systems matures, the distinction between the operating system kernel and libraries is allowing older, non-UNIX operating systems to move into the open systems fold. The POSIX standard, as shown in Figure 1, is the primary vehicle by which this is being accomplished.

POSIX is based on interfaces in UNIX System V and Berkeley UNIX and defines the interface between an application and the operating system. This interface is implemented as a library of functions available to an application but communicating with the operating system.

POSIX is actually a series of standards, as shown in Figure 2, that will continue to grow. POSIX.1 is based on C calling semantics, incorporating 110 functions from the ANSI C standard library, while POSIX.5 is a set of Ada bindings to the POSIX.1 services and POSIX.9 is a similar set of FORTRAN language bindings. POSIX.16, assigned to the POSIX.1 working group, is developing an ISO 9989 C language binding to a language-independent version of POSIX, while POSIX.19 is doing the same for FORTRAN 90. Real-time Ada is being addressed by POSIX.20. POSIX.1 is the only project that has progressed out of the draft standard stage to official standard.

POSIX was first specified by the IEEE (IEEE 1003.1) in 1988 but was also specified by ISO/IEC (9945-1) in 1990. The U.S. government has also specified POSIX as Federal Information Processing Standard 151.

POSIX.1, the core POSIX standard, is supported by every major computer company, including Digital, HP, IBM,

Figure 2.

Committee	Name	Purpose
POSIX.1	System Interface	To define the interface between portable application programs and the operating system based on UNIX system models.
POSIX.2	Shell and Tools	To define the shell command language and tools that might be used by portable applications.
POSIX.3	Testing and Verification	To develop a standard methodology for testing an implementation's conformance to POSIX standards.
POSIX.4	Real-Time	To define the minimum extensions to POSIX.1 that allow portability of real-time applications.
POSIX.5	Ada Language Bindings	To define how the POSIX.1 features will be used from an Ada environment and to deal with special features of Ada.
POSIX.6	Security Extensions	To develop interfaces that would let a POSIX.1 implementation become a trusted system as defined in the Orange Book.
POSIX.7	System Administration	To define a common set of utilities required to manage the system.
POSIX.8	Networking	To look at transparent file access, a protocol-independent network interface, remote procedure calls and OSI protocol-independent application interfaces.
POSIX.9	FORTRAN Language Bindings	To define how POSIX.1 features will be used from FORTRAN.
POSIX.10	Supercomputing	To produce an Application Environment Profile (AEP) oriented to supercomputer users.
POSIX.11	Transaction Processing	To produce an AEP oriented to transaction processing users.
POSIX.12	File Access	To determine whether transparent access to files will support all of the semantics of POSIX.1 file access.
POSIX.13	Real-Time Application Support	To define an AEP for real-time applications using the POSIX interfaces.
POSIX.14	Multiprocessor Application Support	To define an AEP for multiprocessor application environments.
POSIX.15	Batch Processing	To provide a network queuing and batch system in a POSIX environment.
POSIX.16	C Language Bindings	To provide an ISO 9989 C language binding to the language-independent API that corresponds exactly with ISO 9945-1:1990.
POSIX.17	Directory Services	To provide a standard interface supporting the development of applications that use directory services, including X.500.
POSIX.18	Platform Environment Profile	To establish a Platform Environment Profile based on the ISO 9945 work-related standards.
POSIX.19	FORTRAN Language Bindings	To provide a FORTRAN 90 language binding to the Language-Independent Specification of the POSIX system API.
POSIX.20	Ada Language Bindings for Real-Time	To provide an Ada language binding to the real-time POSIX standards.

The POSIX series of standards, now quite large, will continue to grow.

Sun and Unisys. POSIX is the one common denominator that unites vendor standards consortiums as varied as UNIX Int'l, OSF and X/Open. It cuts across operating systems, including nearly every variant of UNIX, as well as VMS, CTOS and MPE.

In addition, the National Institute for Standards and Technology (NIST) hosts the Accredited NIST POSIX Testing Laboratories to validate products. POSIX.3 provides the detailed testing and verification requirements used by the accredited laboratories to validate products. Digital products have been validated by Mindcraft and DataFocus. You can obtain more information from NIST by contacting James Hall at the NIST POSIX Group, Computer Systems Laboratory, B266 Technology Bldg., Gaithersburg, MD 20899; hall@swe.ncsl.nist.gov.

According to *The POSIX.1 Standard: A Programmer's Guide* by Fred Zlotnick (The Benjamin/Cummings Publishing Co. Inc., 1991), POSIX describes an interface, not a UNIX implementation. "The description," writes Zlotnick, an employee of Mindcraft, "consists largely of a series of specifications of functions callable from C programs, of headers, and of data structures." Zlotnick points out that although the current POSIX.1 standard is written in terms of C, work is under way that will result in a language-independent version that will be accompanied by language-specific interfaces for several languages.

No distinction is made between system calls and library functions in the standard. "In fact," says Zlotnick, "a system call in one implementation may quite possibly be a library function in another." Zlotnick regards POSIX as a generalization of UNIX that generalizes such UNIX concepts as filesystems. "A UNIX system file is uniquely specified by its file system and i-number within the file system," writes Zlotnick. "In POSIX.1 a file system is simply a name-space in which file serial numbers (the equivalent of i-numbers) are unique."

Zlotnick also points out that one of the most powerful features of POSIX.1 is its ability to determine a system's limits and attributes at run time and take appropriate action. This allows POSIX-compliant applications to run on many different systems by polling the system about itself rather than simply by addressing a common denominator of system calls. For a description of the technical foundation of the VMS POSIX implementation, see "POSIX For VMS" by Philip A. Naecker (May 1991).

POSIX Compliance

POSIX compliance is defined in terms of implementations such as VMS POSIX and specific applications that will run on those implementations. A conforming implementation may also support calling conventions that are not part of the POSIX standard but that provide backward compatibility with an existing base of applications (such as older C programs)

or additional capabilities (such as VMS RMS calls, support for Motif and support for SQL). In some implementations, several function calls may have the identical effect but different syntax simply for backward compatibility with a variety of existing applications.

There are four levels of application conformance. At the first level, a "strictly conforming" application uses only the facilities described in ISO/IEC 9945. So far, only C language interfaces for POSIX.1 have been standardized.

The second level, the "ISO/IEC Conforming POSIX.1 Application," uses only the facilities described in ISO/IEC 9945 plus conformance language bindings for any ISO or IEC standard. This level of conformance allows applications to make calls compliant with other recognized standards, such as the GKS or the PHiGS graphics standards, and to add to the minimum requirements specified in the POSIX standard. An example of the latter is an extension of filenames from the POSIX.1 minimum of 14 characters to longer names.

A third level, the "National Body Conforming POSIX.1 Application" is just like an ISO/IEC Conforming POSIX.1 Application, except that it can also use standards adopted by a single member body of the ISO/IEC, such as ANSI.

Finally, the "Conforming POSIX.1 Application Using Extensions" level of conformance uses nonstandard facilities consistent with the POSIX.1 standard. This is a very liberal type of conformance, and according to Zlotnick, almost every C program could satisfy it with suitable documentation.

POSIX.1 And C

According to Donald Lewine, author of *POSIX Programmer's Guide: Writing Portable UNIX Programs* (O'Reilly & Associates Inc., 1991), it is possible to write a book about POSIX without using examples written in a specific language, though this is not really feasible yet. "The POSIX standard is written in terms of the C programming language," Lewine writes. "POSIX supports two programming environments. One is based on the

X/Open: CAE, XPG and OSD

X/Open is not a standards organization per se. It is a consortium of the largest and most influential providers and users of information technology systems. X/Open has been very influential in Europe and is now making an equally strong impact in the U.S. Digital has been very active as a member of X/Open.

X/Open's goal is to combine existing and emerging standards to define a comprehensive, integrated applications environment. The organization has focused primarily on operating system interfaces for software portability but is now including distributed computing environments in its specifications, as well. Much of X/Open's work has focused on bringing standards developed in the UNIX world to proprietary systems, which it views as the most important market during the next decade. In fact, X/Open has stated that many large organizations — both vendors and customers — are resisting the move to purely open systems, preferring a mixture of open and proprietary.

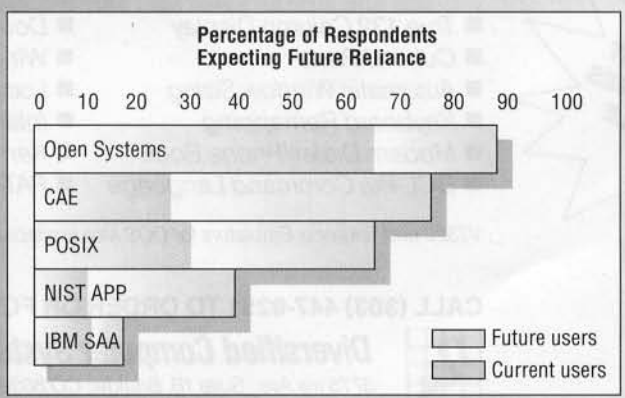
X/Open publishes the specifications for its Common Applications Environment (CAE) in its seven-volume book set, *The X/Open Portability Guide* (XPG). The latest version is XPG3, which is aligned with the POSIX standard. XPG4 will be here this fall. X/Open also hosts the XPG branding program, which certifies operating systems and applications as XPG-compliant.

More than 300 products carry X/Open branding, and the number is expected to increase to well over 1,000 by 1995. VMS is expected to be branded soon. X/Open states that VMS has had more work done on it toward XPG compliance than any other non-UNIX operating system. X/Open now also publishes the *X/Open Registered Applications Directory* so that users who have settled on a particular branded platform can easily locate compatible products. As soon as VMS is branded, this will be a very useful directory for Digital customers.

X/Open also sponsors Xtra, a process that is embodied in a major international research program attracting and involving the "movers and shakers" of the open systems movement from around the world. The Xtra conference is held annually and is user-driven, designed to be the ultimate link between user and vendor in open systems. It provides a forum in which users can identify and prioritize their requirements for the adoption, procurement and implementation of open systems technology. As an annual follow-up to Xtra, X/Open publishes the Open Systems Directive (OSD), a definitive report on the program.

The Figure shows data taken from the third edition of the OSD. This 1992 edition differs from past editions because it presents a detailed compilation of the World Open Systems Requirements Survey, which was conducted simultaneously in 17 nations last summer.

For more information about X/Open and what it can offer Digital customers, call or write Jeff Hansen, X/Open, 1010 El Camino Real, Ste. 380, Menlo Park, CA, 94025; (415) 323-7992. —*B.T.H.*



Respondents to X/Open's Xtra survey believe that CAE and POSIX will be the most important operating system interfaces in coming years.

traditional C language. The other is based on the Standard C language defined by [ANSI X3.159-1989]. Standard C defines the C language in a more precise way and allows for more portability than traditional C."

Since POSIX.1 became a standard in 1988 and ANSI C did not become a standard until a year later, the POSIX committee did not require use of Standard C. But according to Lewine, there is no need to use traditional C if Standard C is available. Lewine says it is unlikely you will find a system that conforms to the POSIX standard and does not also supply a Standard C compiler and libraries.

Mindcraft supplies a POSIX Portability Verifier that you can use to check your C programs to ensure that they conform to the POSIX.1 standard. Books such as Zlotnick's and Lewine's will help you modify existing C programs to ensure that they are strictly conforming and will therefore run on any conforming operating system, including VMS V5.5.

POSIX.2

Digital now supports POSIX.2 and POSIX.4 across its platforms. DEC licensed its POSIX.2 implementation from MKS, which has also licensed its InterOpen/POSIX Shell and Utilities source code to HP and Unisys for implementation on their MPE and CTOS operating systems respectively. The POSIX.2 shell resembles the Korn shell and provides functions similar to DCL.

The package enables developers to build tools and applications that can be easily ported among POSIX-conforming environments. According to Digital's Kittle, DEC implemented all of the standard and optional elements of the toolkit in the VMS POSIX.2 implementation.

POSIX.2 is still an IEEE and ISO draft standard but is expected to become a full standard soon. POSIX.2 defines both an application and user portability interface through the shell programming language and the utilities. The goal of POSIX.2 is application portability, through which application programs may access a standard set of system services at a high level. POSIX.2 complements POSIX.1 by making the underlying POSIX.1 features accessible to software developers, testers and users.

POSIX.2a, the User Portability Extension, is an addendum to the POSIX.2 standard and will eventually be merged with it. POSIX.2a is supported in VMS POSIX and deals with a standardized application interface and a common environment for time-sharing users on a character-oriented display terminal. This extension includes a mail utility, text editors and an interactive standard command language interpreter. The intent of POSIX.2a is to provide POSIX users and developers with a common interactive working environment, reducing retraining time across systems. The POSIX.2 and POSIX.2a utilities appear in Figure 3.

POSIX.4

The POSIX.4 standard, though also still in draft form, is providing the foundation for the standardization of several important operating system services that have traditionally been implemented in very vendor-specific ways. These facilities include semaphores and shared memory as well as threads. Capabilities such as timers, real-time signal extensions, asynchronous I/O, synchronized I/O and shared memory are addressed by POSIX.4.

POSIX.4a is being closely watched by Digital and others, since it will provide the standard threads implementation used in most distributed computing environments. POSIX.4a specifies both kernel-level (in the kernel) and library extension ("above" the kernel) threads. The DCE threads package is based on the library extension specification, while Digital is using DECthreads and the POSIX.4a library extension version in VMS V5.5. DEC is expected to incorporate the kernel-level specification into VMS once the POSIX standard is finalized.

In a paper presented at Uniforum in January, Bill Gallmeister and Chris Lanier of Lynx Real-Time Systems concluded that, though these standards stem from UNIX, "the work required to support

Figure 3.

POSIX.2 Utilities

!	cd	echo	getconf	ls	readonly	touch
(	chgrp	ed	getopts	mailx	return	tr
{	chmod	egrep	grep	make	rm	trap
[	chowm	env	head	mkdir	rmdir	true
:	cksum	eval	id	mkfifo	sed	tty
.	cmp	exec	if	mv	set	umask
ar	comm	exit	join	nohup	sh	uname
asa	command	export	kill	od	shift	uniq
awk	continue	expr	lex	paste	sleep	unset
basename	cp	false	ln	pathchk	sort	until
bc	cut	fgrep	locale	pax	strip	wait
break	date	find	localedef	pr	stty	wc
c89	dd	fold	logger	printf	tail	while
case	diff	for	logname	pwd	tee	xargs
cat	dirname	fort77	lp	read	test	yacc

POSIX.2a Utilities

alias	csplit	fc	mailx	nm	strings	uudecode
at	ctags	fg	man	passwd	tabs	uuencode
batch	df	file	mesg	patch	talk	vi
bg	du	history	more	ps	tput	wait
compress	ex	jobs	newgrp	renice	unalias	
crontab	expand	kill	nice	split	unexpand	

POSIX.2 defines an application and user portability interface through the shell programming language and the utilities. POSIX.2a, the User Portability Extension, is an addendum.

POSIX.4 and POSIX.4a atop an existing UNIX system is significant. ... Even so, the advantages to be gained by widespread support and adoption of a standard programming interface for real-time systems can have a major impact in terms of software quality and productivity."

It is ironic that the work required to build standards into the operating system that served as the reference platform for the development of the standards should be "significant." UNIX is destined to become just another kernel like the VMS kernel. The only thing truly special and open about it will be that it served to spawn the standards that eventually united all operating systems across computer manufacturers.

Would you like to see more articles on this topic?
Circle on reader card: Yes 467 No 468

Companies Mentioned

DataFocus Inc.
12450 Fair Lakes Cir., Ste. 400
Fairfax, VA 22033
(703) 631-6770
CIRCLE 469 ON READER CARD

Lynx Real-Time Systems Inc.
16780 Lark Ave.
Los Gatos, CA 95030
(408) 354-7770
CIRCLE 470 ON READER CARD

Mindcraft Inc.
410 Cambridge Ave.
Palo Alto, CA 94306
(415) 323-9000
CIRCLE 471 ON READER CARD

Mips Computer Systems Inc.
950 deGuigne Dr.
Sunnyvale, CA 94086
(408) 720-1700
CIRCLE 453 ON READER CARD

Mortice Kern Systems Inc.
35 King St. N.
Waterloo, ON N2J 2W9
(519) 884-2251
CIRCLE 472 ON READER CARD

Silicon Graphics Inc.
2011 N. Shoreline Blvd.
Mountain View, CA 94043
(415) 960-1980
CIRCLE 476 ON READER CARD

UNIX System Laboratories
190 River Rd.
Summit, NJ 07901
(908) 522-6000
CIRCLE 478 ON READER CARD