# IEEE's Posix: making progress

*Seven more standards are near completion in this cornerstone of the international open-system software effort*

As the circuit design deadline drew near, the pace became more frantic by the hour, elevating the project leader's anxiety level and his blood pressure. And for good reason—all the users of DEC workstations, where the computer-aided design software resided, were out at a seminar. To be sure, IBM, Sun, and other workstations were available that would have allowed other engineers to complete the job on time. But the software? Only a DEC version was obtainable in house; in no way could it be ported to the IBM or Sun on time to beat the deadline—even though the project leader had received assurances from the software vendor that such versions "were forthcoming."

This scenario is not as fictitious as it might seem. It reflects a growing concern among engineers as well as other users of software with the need for portability and interoperability of software. Portability refers to the ease with which a software system or component can be transferred from one hardware or software environment to another. Interoperability is the ability of two or more computer systems and their software to exchange information and use the information that has been exchanged.

Though the work needed to bring about industrywide portability and interoperability is extensive, there is hope. Efforts have been under way for some years in the form of the so-called open system standards. An important part of that effort is the interface standards within the IEEE portable operating system interface (Posix) environment. (The X in Posix denotes the Unix operating system origin of this effort.) At least one of these standards—No. 1003.1, which covers basic operating system services—was adopted last December as Standard ISO/IEC 9945-1 by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC)—both in Geneva, Switzerland.

D. Richard Kuhn
National Institute of Standards and Technology

Today there are 19 Posix working groups involved in 26 projects. Likely to be completed within the next 12–18 months are standards for such tasks as handling command language and utilities (Standard 1003.2), real-time programming (1003.4), and controlling access to computer files (1003.6).

The Posix efforts have been supplemented by the IEEE Technical Committee on Operating Systems with other projects to develop standards for application interfaces to services [see table, p. 39].

Additional open system standards have been developed through the American National Standards Institute (ANSI), ISO, and other organizations. These focus mainly on such specialized aspects as programming languages, databases, and communication protocols. In contrast, Posix standards cover operating system services in general.

Many of these other specifications have been combined with the developing IEEE standards to define an open systems environment using the Posix interface standards as the basis. Already products abiding by the Posix standards have emerged from vendors, including such industry leaders as IBM, Digital Equipment, Sun Microsystems, Apple Computers, and AT&T's NCR.

**In the offing: a major new industry geared to standard software components**

Not only are the open system standards within the Posix activity expected to resolve the portability and interoperability problems, but they are also expected to open the door to a major new industry of standard software components, or modules. From these components, users will be able to build and modify larger systems to suit their evolving needs. Such components will eliminate the need to produce several versions of an application program to accommodate operating systems with different file system structures and network interfaces, making them more attractive to software developers.

Standard components for software have been advocated for at least two decades, but until recently only limited versions of such components—mathematical subroutine libraries, for example—have been available.

Barriers to more complex components have been both technical and economic. Though such modern programming languages as C++, Objective C (an object-oriented version of C), and Ada are helping to solve technical problems associated with component development, a software component created today in a language such as C++ must still use system services that vary. They must depend on the operating system, database, communication interface, and other vendor-specific functions. If a component makes extensive use of VMS system services, for example, then a different version of the component must be created for Unix System V. (VMS is a popular operating system for Digital Equipment Corp.'s computers.)

**INTERFACE SPEC.** Evolving industrywide through a consensus process, open system standards will eventually change this situation. Generally, an open system standard is an interface specification to which any vendor can build hardware and software products. Posix and related standards, however, refer only to software interfaces. If a vendor of a proprietary operating system software abides by an open system standard, it will provide software with the standard interface. This can be used to build portable software.

There are two kinds of portability—binary and source-code. Specifications for binary portability are designed for object code—a fully compiled or assembled program that is ready to be loaded into the computer. With binary portability, an executable copy of a program can be moved from one machine to another. In contrast, with source-code portability, a program must be recompiled first.

An example of a *de facto* standard for binary portability is the IBM PC machine-language instruction set. Executable copies of software can run on PC clones from many different manufacturers.

Of the two portabilities, binary portability is the more difficult to achieve because it puts constraints on the machine architecture and instruction set. Standards efforts,


0018-9235/91/$1.00©1991 IEEE

therefore, have concentrated on developing interfaces for source code.

Open system standards for source code portability define interfaces available to application programs for essential services like process control, file and directory access, interprocess communication, and graphics.

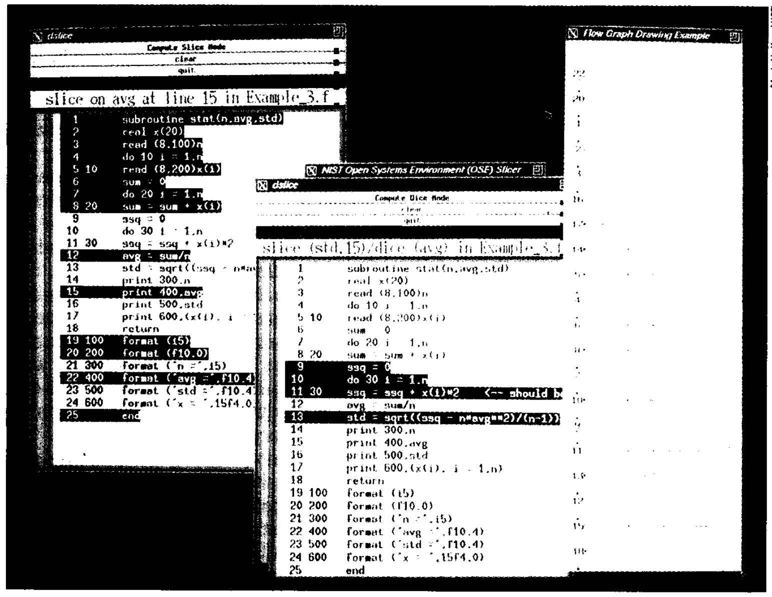
Interoperability standards, though necessary, are not sufficient for a complete open systems environment. An example of this is provided by the X Window System protocol, another *de facto* standard that specifies how graphics primitives can be communicated between an application program and graphics software running on a workstation. The protocol allows, say, an X Window application running on an IBM workstation to interact with a user sitting at a Sun workstation. The interoperability, however, does not mean that the source codes on these two systems are compatible. Each one may use different library functions to generate the X Window protocols.

**OPEN SYSTEMS ENVIRONMENT.** No single standard provides all the functionality needed in a modern computing environment. To provide portability and interoperability requires a comprehensive set of standards.

The Posix open systems environment (OSE) being put together by Working Group 1003.0 of the IEEE Technical Committee on Operating Systems (TCOS) offers a standard set of interfaces to information systems' building blocks, covering both portability and interoperability standards.

Not all the specifications in the Posix OSE are IEEE Posix (1003.x) standards. Posix functions serve as a basis, supplemented by other applicable open system standards—like those under development by the ISO and ANSI.

Two types of standard interfaces are specified in the Posix OSE: the application program interface (API) and the external environment interface (EEI) [Fig. 1]. The APIs generally are the procedure calls made to the application platform—the computer in which




*IEEE portable operating system interface (Posix) standards are being used by a programmer at the National Institute of Standards and Technology, Gaithersburg, Md., to manipulate slices, lines of code that affect the final value of a variable. Highlighted in the leftmost window are all the lines of the program used in generating the output values of "avg" (average). A slice in the center window shows the variable "std" (standard deviation), with lines from the first slice masked. The result shows lines used to compute "std" but not "avg," lines believed to contain an error. The rightmost window graphically illustrates data flow within the slice. Windows and graphics are generated by using the X Window system with a Posix operating system interface; data flow and slicing code was written in ANSI C.*

the application program is running and its operating system—for a particular programming language. Through these calls, APIs provide source-code portability.

The external environment refers to external entities with which the application platform exchanges information, including the human end-user, hard copy documents, and physical devices such as video displays, disks, printers, and networks. Generally in the form of communication protocols, record and document formats, display formats, and distributed systems services, EEIs, in contrast to APIs, provide mainly for interoperability.

**FIVE ROLES.** Examining details of the Posix OSE application program interfaces is helpful in exploring how standards can be used in constructing portable software. Based on services they provide, four general categories and a special-purpose category are available. The general categories cater to system, communications, information, and human-computer interaction services. A typical computing environment will require some, but not all, of the standards contained in each of these four categories. A fifth category—domain services—is provided for such special-purpose environments as transaction processing.

System services include both language and operating system services. Language services are the functions typically provided by programming languages such as C, Fortran, Pascal, and others. Operating system services are those used to control the resources of a computer system—hard-disk storage, printer, and so on.

In the language service area standard interfaces specify instructions in different programming languages—Ada, Basic, C, C++, and Pascal (for example, the ISO/IEC 9899 standard for the C language). To make other services in the OSE accessible from application programs, language bindings (subroutine calls in specific languages) are needed for one or more of these languages.

## Defining terms

**Compiler:** a computer program that translates computer code in a high-order language (such as Fortran) into its machine-language equivalent.

**Environment services:** services related to external objects, such as conditions and processes that influence the behavior of a system.

**Interface:** a shared boundary across which information is passed.

**Kernel:** a software module that encapsulates an elementary function or functions of a system.

**Language binding:** definition of the parameters passed and functions to be performed by a subroutine call for a specific language (C, maybe).

**Language services:** functions provided by programming languages (basic mathematical functions, for example).

**Logical naming:** services that allow the use of system resources by name rather than by hardware addresses.

**Open system standard:** a specification developed in a consensus process, to which any vendor can build products.

**Posix standards:** a family of open system standards developed by the IEEE Technical Committee on Operating Systems.

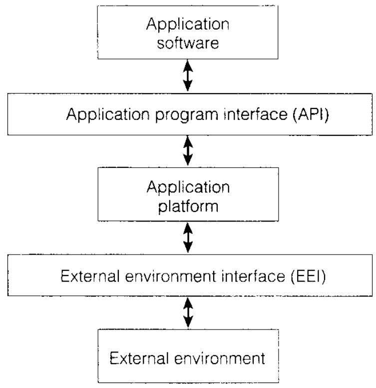
**Real-time programming:** programming for computation to be performed during an external process, so that the computation may be used to respond to the process in a timely fashion.

**Software component:** a piece of software whose interfaces are precisely defined so that programmers can use it without knowing its structure.

**Source code:** computer instructions and data definitions expressed in a form suitable for input to an assembler, compiler (see above), or other translator.

**Transaction:** a data or control element, signal, event, or change of state that causes, triggers, or initiates an action or a sequence of actions.

**Utility:** a software tool designed to perform a frequently used support function.




*[1] The standards being developed under the IEEE portable operating system interface (Posix) open systems environment effort employ this reference model. It includes two important interfaces: an application program interface between the application software and platform, and an external environment interface between the platform and such peripheral systems as printers and displays.*

The Posix kernel standard (1003.1), originally defined using C, will soon have Fortran (1003.9) and Ada (1003.5) language bindings. The most common language for Posix interfaces is C, although language-independent bindings (generic subroutine calls not tied to a specific programming language) are now being developed by IEEE Posix working groups.

Among the major categories of operating system services in the Posix OSE are process management, task management (suspension or resumption of a process, for example), and environment services (like obtaining a terminal identification or user profile). Other services include: process communication and synchronization; input/output; file management; event, error, and exception management (enabling and disabling interrupts, for example); time services; and memory management.

Standards in the OSE providing these services include Posix shell and utilities (1003.2), which provides a command language (similar to DOS commands used in IBM PC batch files); software tools for such common operations as sorting; and real-time extensions (1003.4), which handles real-time programming features.

Communications services, including ISO Open Systems Interconnection, make communication possible for application programs running on networked computers. They include services for file transfer, namespace and directory services, network file access, remote procedure calls, protocol-independent network access, and data representation. Both API and EEI functions are included in this area.

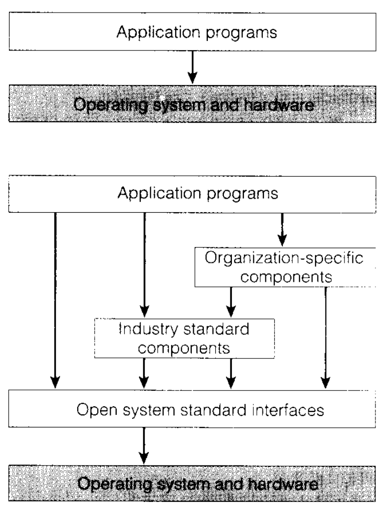
The interface to the interoperability functions is through the standard APIs, such as the protocol-independent interface (1003.12) and the remote procedure call interface being developed by ANSI X3T5.5 working group.

Information services include database services, which provide the capability to store and retrieve data from long-term storage, and data interchange services to exchange data between systems.

Database services are the functions associated with database management systems. These include: data definition and manipulation (the ability to create, update, and delete records, fields, or tables); data access (the ability to retrieve data based on complex search conditions); and data integrity (the locking of data items, transaction control, and synchronous writes—that is, writing of data on an external, backup hard-disk system synchronously with the writing in the main memory).

Application programs use database services extensively, and the APIs in the Posix OSE information services area include such non-Posix standards as Structured Query Language (ISO 9075:1982) and Network Data Language (ISO 8907:1987).

Included in data interchange services are data description protocols, character sets, and data format protocols. Data description protocols provide a standard means of associating a name with individual data elements. Data format protocols add attributes that describe the physical characteristics of the data. Among the standards addressing data interchange services is the Standard Generalized Markup Language (ISO 8879: 1986)—again a non-Posix example—useful for defining the layout and structure of a document.




*[2] In today's architecture, the application programs operate directly with the designated operating system and hardware (top). Aiming at total portability and interoperability of software, the IEEE Posix open system architecture allows three application program interfaces, each with its own possibility for software components. The first is specific to an organization that uses or designs the software, the second relates to industrywide standard software components, and the third—open system standard interfaces—is designed to work with each of the previous two or directly with the application program (bottom).*

The Posix OSE includes national and international electronic data interchange standards being developed for data format protocols—like ISO 9735. Other standards embraced by Posix in the information services category include Computer Graphics Metafile (ANSI X3.122-1986), which provides a standard means for storage and exchange of computer graphics.

**HUMAN-COMPUTER INTERFACE.** Using the window and mouse style of interaction popularized by the Apple Macintosh, the human-computer interaction services in the Posix standards provide functions for communication between user and computer.

Applicable external environment interface standards will include the X Window protocol, which specifies the format and meaning of messages between an application program and a display terminal, and human factors standards.

In development in this category is the IEEE 1201.2 Drivability Recommended Practice. It will recommend a set of window system behaviors designed to make working with different systems of this kind as easy as driving different makes of automobiles.

API standards in this area are still being defined. Among them is IEEE Standard 1201.1, a standard intended to be a set of window system function calls that can be used with any system that provides the services to create and manipulate menus, buttons, scroll bars, graphics, and other common features of window-based interfaces.

It is doubtful that any information-processing system will implement all the standards included in the Posix open systems environment. A subset of them, referred to as a "profile," is typically sufficient to meet an organization's requirements. Profiles for different types of applications, such as transaction processing, real-time programming, and supercomputing, are being developed within the Posix working groups.

Such profiles are incorporated in the domain services area—the fifth component of the Posix open systems environment. Organizations may also have their own profiles, based on their unique needs. For example, the National Institute of Standards and Technology (NIST) has established an applications portability profile, which some Federal agencies have adopted to promote software portability within the Government.

The widespread interest in open systems has encouraged strong support from computer and software vendors. Most vendors now provide a system compatible with the Posix 1003.1—a basic operating system, or kernel standard, as well as the other, non-Posix completed standards (such as those for programming languages). As other Posix standards are completed in the near future, conforming systems from leading vendors should follow.

When Posix standards for the open systems environment become available, what will be the most effective way of using them to achieve applications portability? One approach is to build components that provide services specific to an industry or an individual organization, resulting in a hierarchy of services. These would include generic system services provided by standards such as the Posix kernel, industry-specific services provided by components built on the system services, and organization-specific services provided by components built on the industry-specific and the generic system services.

Application programs for end-users can then be built for the application program interfaces provided by the hierarchy of components. For example, an organization-specific interface might be a specification for a function that displays a company logo, department name, and time of day on a graphics terminal. A software component to provide the specified service would use operating system functions to obtain the time of day, bitmap for the logo, and department information from a database. Many of the organization's application programs use the same service, and the application programs may run on many different computers.

The application programs call these APIs, rather than calling operating system services directly. This approach is sometimes used today to deal with system dependencies, and it will still be necessary when open system standards are used.

It is also possible to specify an API for a particular industry. For example, NIST and the Interactive Multimedia Association (formerly the Interactive Video Industry Association), Washington, D.C., are developing an API for multimedia services to be used in computer-aided training systems. Components providing the services specified in the API can be built on standard interfaces. Because they are built using open system standards, the components can be ported to diverse hardware at low cost.

In application architecture today, the application software interfaces with the operating system and hardware. Open system standards introduce three layers that will make the portability and interoperability possible—organization-specific APIs, industry standard APIs, and an open system standard interface [Fig. 2].

Open system standards are likely to have a significant impact on both cost and competition in the computer industry. Software products can now be made more efficiently because developers can produce a single version for the standard programming interface rather than a different version for each hardware vendor.

Also, vendors will be able to compete for business that previously was denied to them because of users' dependence on another vendor. When all the Posix standards are completed, users will be able to buy software from different vendors without requiring major conversions of their internally developed software. Since software products will be developed more efficiently for a wider range of hardware, software components will become more economically practical.

## Representative IEEE application program interface standards

| Standard | Subject | Scope | Status |
|---|---|---|---|
| 1003.1 | System application program interface (kernel) | Basic operating system services such as file I/O and process control | Complete; became ISO Standard 9945.1 in December 1990 |
| 1003.2 | Shell and utilities | Command language and utilities that can be used in shell scripts[a] or command procedures | Nearing completion |
| 1003.2a | User portability extension | Utilities for time-sharing systems | Nearing completion |
| 1003.4 | Real-time extensions | Real-time programming features such as process locking and synchronization | Nearing completion |
| 1003.4a | Threads extension | Real-time features useful for supporting transaction processing | Nearing completion |
| 1003.5 | Ada language binding | 1003.1 function calls for the Ada language | Nearing completion |
| 1003.6 | Security extensions | Security features such as access control lists and multilevel security | Nearing completion |
| 1003.7 | System administration | System management features for such tasks as adding users and checking device status | In progress |
| 1003.8 | Transparent (network) file access | Functions for making files on several machines appear to reside on a single machine | In progress |
| 1003.9 | Fortran interface | 1003.1 function calls for the Fortran language | Nearing completion |
| 1003.12 | Protocol-independent network interface | Communication services independent of protocol | In progress |
| 1003.15 | Batch scheduling | Functions for batch (noninteractive) processing | In progress |
| 1003.17 | Name space and directory service | Distributed systems directory functions | In progress |
| 1201.1 | Window-based user interface | Window system, graphical user interface functions | In progress |
| 1224 | X.400 message-handling interface | Open systems interconnection (OSI) electronic mail services | In progress |
| 1238.0 | Support functions | Common OSI support functions for lower-level interface | In progress |
| 1238.1 | File transfer access method | OSI file transfer functions | In progress |

[a]Shell scripts: commands similar to DOS commands used in IBM PC batch files.

**TO PROBE FURTHER.** UniForum, the international association of Unix systems users, publishes a series of booklets entitled "Posix Explored." Contact UniForum, 2901 Tasman Dr., Suite 201, Santa Clara, Calif. 95054; 408-986-8840.

The Posix 1003.1-1990 (kernel) standard is available from IEEE Publications, which can be reached at 800-272-6657. IEEE Standard 1003.1-1990 is also referenced as ISO/IEC 9945-1:1990.

The National Institute of Standards and Technology Special Publication, "Application Portability Profile—APP—the U.S. Government's Open System Environment Profile OSE/1 Version 1.0" (Order No. SN: PB91-201004), explains Posix and other open system standards. It is available from the National Technical Information Service, 5285 Port Royal Rd., Springfield, Va. 22161; 703-487-4650.

"The Guide to Posix Open Systems Environment IEEE 1003.0," currently in draft form, should be available from the IEEE Service Center in 1992.

The seven-layer, open-systems interconnection model is discussed in detail in "Helping computers communicate," *IEEE Spectrum*, March 1986, pp. 61–70.

**ABOUT THE AUTHOR.** D. Richard Kuhn is a computer scientist at the National Institute of Standards and Technology, Gaithersburg, Md., where his responsibilities include operating system interface standards, formal methods, and computer security. He received an M.S. in computer science from the University of Maryland in College Park. ◆

The use of specific products and companies as examples in this article does not indicate their endorsement either by the National Institute of Standards and Technology or by the U.S. government. Nor does such use imply that the products named are the best available for the stated purpose.