-- --

IEEE P1003.0 Draft 13 – September 1991

4.2 System Services

Responsibility: Patricia Oberndorf

4.2.1 Overview and Rationale

This clause describes the system services component of the application platform. It presents a reference model for this component and describes the services provided to application software. Those services are those usually considered as part of an operating system or executive and also those services that may be provided by system level entities such as spoolers and device drivers. Standards, current and emerging, that specify the interface to those system services are also described.

System services are a key component of the application platform and represent the focus of the IEEE effort to produce POSIX base standards. A common set of system services provides support for the portability and the interoperability of application software. While other common services can aid application reuse, system services are those that are common to the largest number of applications.

4.2.2 Scope

System services cover those features that users have come to expect from operating systems or executives. They cover the areas of process management, file management, input/output, memory management, and print spoolers. Because there is a wide variety of platform users, ranging from large general purpose time-shared systems to small time-critical, special-purpose systems, services such as timers and clocks, event management, logical device drivers, and system initialization/reinitialization are included. Services related to distributed systems are also discussed, since application software sees these capabilities through the platform.

4.2.3 POSIX OSE System Services Reference Model

4.2.3.1 Reference Model

This subclause identifies the entities and interfaces specific to the system services of the POSIX OSE. The reference model presented here is consistent with and expands upon the reference model of Section 3. It provides the context for the discussion of System Services in this clause. The basis System Services model is shown in Figure 4-2.

This clause describes the system services portion of the application platform as viewed by a software developer (not necessarily the viewpoint of the end user). This view corresponds to the program design level of abstraction.

The system services API provides the interface between the application software and the system services from the source code point of view. The API defines the

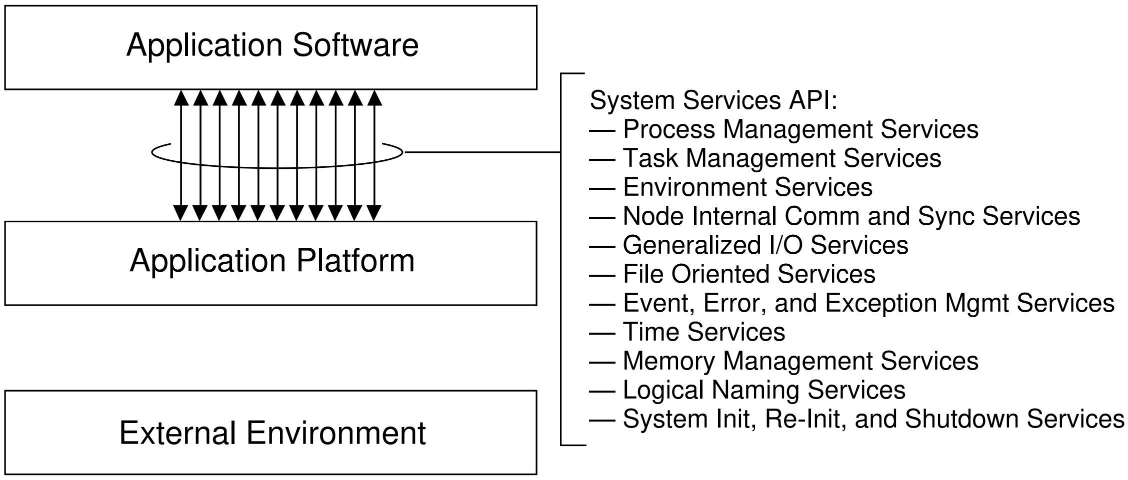

Figure 4-2 – System Services Reference Model

program designer's means of accessing the functions, objects, and services of the system.

In order for the platform to protect system integrity and ensure system database consistency, application software competing for system resources must access all system resources via system service requests. The formal definition of these requests (or system calls) defines the system services portion of the API.

All of the system services may be available locally or remotely. Some of the system services may be performed remotely if the system is a distributed system with multiple processor nodes. Such distribution is not reflected in Figure 4-2 because it is transparent to users of the System Services.

The platform's device drivers and other software entities are seen as being available to an application program via invocation of the system services. Local devices include sensors, effectors, and connections to independent computing systems. The local devices themselves are a part of the external entities element of the system services reference model. The interfaces used by the application software are the logical device interfaces and are part of the system services. It should be noted that, even though the device drivers are represented within the system services portion of the application platform and the devices themselves are represented within the external entities, there is no unique system service interface illustrated at the EEI in Figure 3-3. This is not an oversight; it is due to the belief that such interfaces are not within the scope of this guide.

4.2.3.2 Implementation Aspects

Although this reference model for system services appears to make no distinction, there are some very important real-world considerations for system services brought on by the essential differences between realtime and nonrealtime systems. One of the best ways to understand these differences is to examine the performance requirements for system services.

Several metrics are associated with the performance of system services.

Service Request Latency is defined as the interval between (a) context switch from an application context to the operating system context and (b) the reverse context switch from the operating system context to the application context for a given service request. Note that for asynchronous services, the service request may not have been satisfied within the latency interval. In error situations, return of an error status or dispatch of the error handler shall be the terminating event for the measurement.

Service Delivery Latency is the interval between (a) context switch from an application context to the operating system context and (b) satisfaction of the service request.

In some contexts, the most important metric is the wall clock time from the point that an external, high-priority event occurs to the time the application actually responds to the event.

However, response time exclusively need not always provide an appropriate metric. It is also necessary to take into account what must be done within that time. For example, one system with a one-second response time may be high-powered enough to handle and coordinate many tasks within that time. Another system with a faster response time might not be handling the same amount of processing.

Realtime Metrics

Since a realtime system is defined as one that performs its functions within a bounded response time, the evaluation of a realtime system has two distinct requirements. First, it must have the right functions. Second, each function must be able to guarantee a certain level of performance.

Two major metrics that are concerned with priority interrupts and scheduling are interrupt latency and context switch time. These are the metrics most commonly associated with realtime systems.

Interrupt latency is the longest time interval possible for an interrupt to be received and acknowledged and a process scheduled to service it (see Figure 4-3).

The context switching time is the time required to switch from the task that is scheduled to service the interrupt. It is determined by the time the operating system takes to store the state (context) of the current process so it can continue processing its previous task after servicing the interrupt.

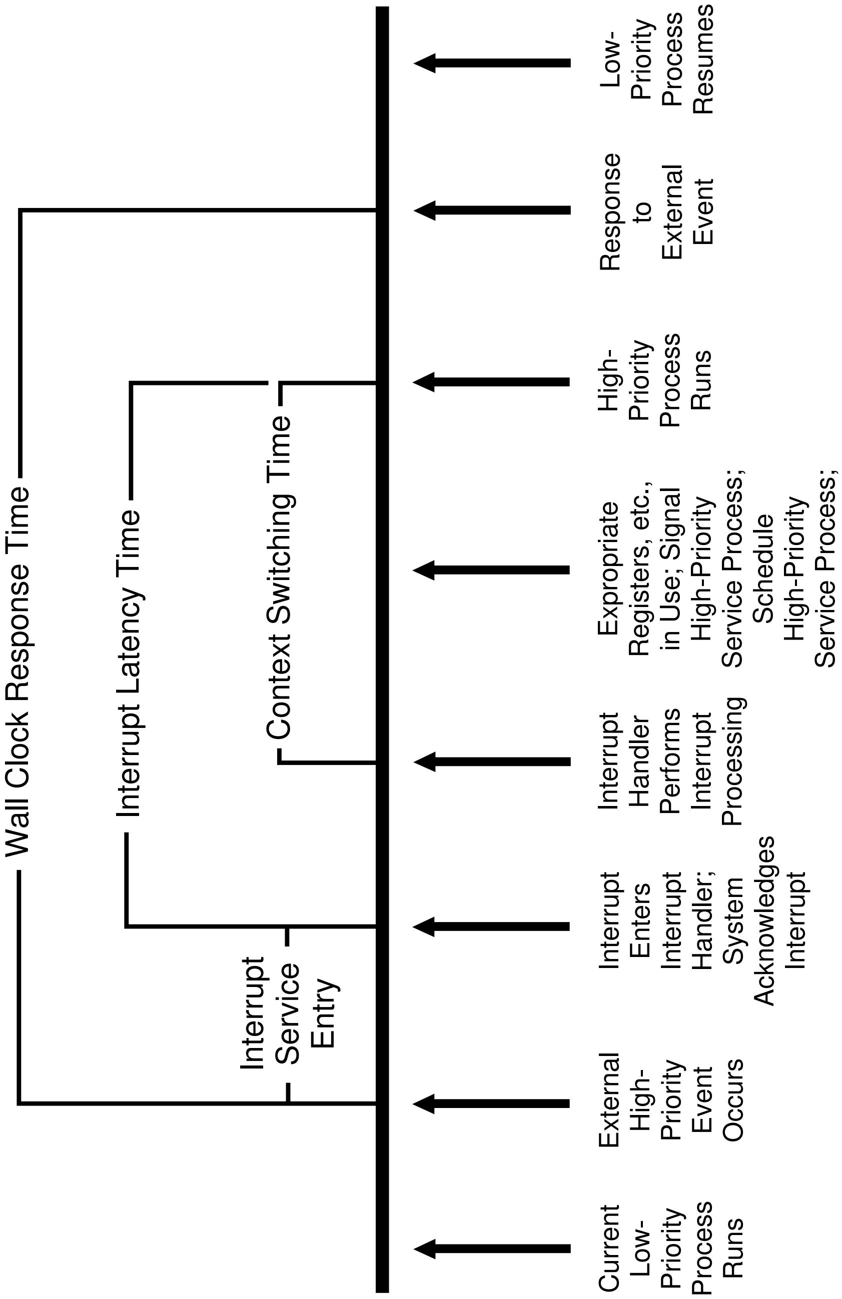

Figure 4-3 – Realtime Metrics

4.2.4 Service Requirements

This subclause identifies those processor-oriented system services required to support application portability and system interoperability. Subclause 4.2.4.1 describes those system services directly available to an application program via the System Services API. Other processor-oriented services are described in 4.2.4.4. Subclause 4.2.5 identifies the applicable standards.

This subclause describes the major groups of system services that an application may require of a platform. Not all of these services require a programming interface; therefore, services are described as either explicit or implicit services. Explicit services are those that can be accessed from an application program (via the API) and generally are only provided when requested. Implicit services, on the other hand, are services that the platform provides without a direct request. An example of an implicit service is the prevention of one program from writing over the memory of another. An example of an explicit service is a call to a system service routine to output the contents of a block of memory to some device.

4.2.4.1 Application Program Interface Services

This subclause describes the major categories of system services available at the System Services API. These services include:

- Process Management Services
- Task Management Services
- Environment Services
- Node Internal Communication and Synchronization Services
- Generalized Input/Output Services
- File Oriented Services
- Event, Error, and Exception Management Services
- Time Services
- Memory Management Services
- Logical Naming Services
- System Initialization, Reinitialization, and Shutdown Services

4.2.4.1.1 Process Management Services

These services relate to the creation, management, and deletion of processes executing within the scope of an operating system. These processes are distinguished from ‘‘tasks’’ via the following characteristics:

- They have a single thread of execution per address space.
- There is substantial overhead for context switches.

— Specific attributes are associated only with processes.

In this context, ‘‘management’’ consists of those services that affect the execution of a process:

— Stop and restart execution of a process (e.g., suspend, resume)

— Modify processor allocation to a process (e.g., priority, timeslice)

— Modify scheduling of the process based on timer (or other) events

— Protect the process from interruption during critical periods

— Create a process and make it ready for execution

— Destroy a process and recover its resources

— Evaluate a reference to a process

— Evaluate a connection to a process, where a connection is a logical communication path between any two processes

These services schedule or arbitrate the usage of various resources of the OS, particularly the central processing unit (CPU). The scheduling services must be able D
to queue up requests to use a particular resource. This situation is made more complicated by the common need to schedule processes to run cyclically at a fixed period. When a resource becomes idle, the scheduler must select one of the ‘‘requesters’’ of the resource to grant use of the resource. These services are listed separately rather than under the services that use scheduling to emphasize that there should be uniformity and consistency of scheduling across the range of resources.

Typically, there are at least two types of scheduling occurring in an operating system: short-term and long-term. Long-term schedulers determine which possible requesters at a given time may actually request a resource. The short-term scheduler selects from among the active ‘‘requesters’’ that currently have need of the resource and allocates the resource to the selected ‘‘requester.’’ For example, if the requesters are processes and the resource is the CPU, the long-term scheduler manages the movement of processes from inactive (waiting in batch queues or in hibernation) to active (in wait or execute). The short-term scheduler, on the other hand, would determine which process should execute next on the CPU. Hybrid services between the two may also be available in the operating system.

When a request for a resource is submitted to the operating system (at some local operating system node), it is not always serviced at that local node. The most advantageous way to service the request may result in part or all of the work being performed at a different processor node. Several reasons may cause this to occur, including load balancing, resource availability, computation speedup, hardware preference, and software preference. These services may hide from the application the fact that the functionality was being performed at a different node. This has the advantage that the code needs to know little about the system on which it is running. Alternately, the services may allow the user to specify directly on which logical resource the function should be executed.

The priority scheduling of resources allows the requester to have associated with it its importance to use the service. More complex schemes also have a criticalness of the request that is used for graceful degradation purposes. The scheduler(s) will use the priority information to arbitrate resource requests and to queue requests in the specific order. A priority scheduler may need to support multilevel queues to support proper execution.

Preemptive schedulers will deallocate a resource from a requester when certain events occur. Usually this is when a requester of a higher priority or importance requests the resource or a specified time limit for the resource has expired.

4.2.4.1.2 Task Management Services

These services relate to the creation, management, and deletion of tasks executing within the scope of an operating system. These tasks are distinguished from ‘‘processes’’ via the following characteristics:

— There may be multiple threads of execution per address space.

— There is low overhead for context switches between threads located in the same address space.

In this context, ‘‘management’’ consists of those services that affect the execution of a task:

— Stop and restart execution of a task (e.g., suspend, resume).

— Modify processor allocation to a task (e.g., priority, timeslice).

— Modify scheduling of the task based on timer (or other) events.

— Protect the task from interruption during critical periods.

— Create a task and make it ready for execution.

— Destroy a task.

— Evaluate a reference to a task.

— Evaluate a connection to a task, where a connection is a logical communication path between any two tasks.

4.2.4.1.3 Environment Services

These services provide an application access to a variety of information relating to the operating system environment in which the application is executing. The specific characteristics are:

— Process-specific attributes (process identification, priority, stack size, scheduling attributes, status, memory allocation).

— Task-specific attributes (task identification, priority, scheduling attributes, status, memory allocation).

— Processor-specific attributes (node identification, electronic nameplate information).

— User-specific attributes (user identification and terminal ID, user interaction profile).

— Environment variables (command-line arguments, menu selections).

— Current time and date

4.2.4.1.4 Node Internal Communication and Synchronization Services

One or more applications and application subcomponents may run on a processor within an application platform simultaneously. The applications run as independent software entities and communicate among themselves via a variety of mechanisms provided or managed by the system services (see Figure 3-2). An important class of system services relates to the coordination and synchronization of these software entities. In traditional systems, entities execute on a single hardware processor. However, it is becoming common to have multiple processors and networked processors that place more requirements on the system services to provide coordination and synchronization among the many truly concurrent software entities.

When a platform has several software entities executing concurrently, the applications need system services so that the entities can be coordinated and synchronized with each other. With respect to applications written using concurrency, there are two levels of concurrency that are usually seen by the application developer. The first level of concurrency, task level concurrency, is seen when the application is split into multiple subcomponents (tasks) that share access to the data and subprograms of the application. Concurrency services at this level concern the relative priorities and scheduling of tasks within a single application program and their communication with each other. At the second level of concurrency, application level concurrency, a unit is a single application including all its subcomponents. Concurrency services at this level concern the relative importance of the individual applications competing for and sharing system resources.

These services are used to communicate among processes, among tasks, and among processes and tasks residing on the same node. The methods outlined do not include the network specific services described in 4.3, but are limited to methods open to entities executing within the scope of a single operating system. Both synchronous and asynchronous services are defined. The specific services are:

— Create, delete, open, close, read, and write shared memory.

— Create, delete, read, and write event flags.

— Create, delete, set, and wait on semaphores.

— Create/send and receive signals.

— Create, delete, open, close, send to, get from, and control message queues.

— Create, delete, send, and receive streams.

4.2.4.1.5 Generalized Input/Output Services

These services are used by an application to perform generalized device I/O operations. These operations include synchronous and asynchronous operations for device and class specific functions. Specifically, these form the services needed to implement or include logical device drivers in a system. These services are device initialization, device attachment, asynchronous operation, and error notification. In addition, they include those services that are used to directly access specific device capabilities, particularly those services often referred to as ‘‘raw I/O.’’

4.2.4.1.6 File Oriented Services

Mass storage in the form of hierarchy of directories, subdirectories and files will be available to an application executing within the application platform. The following paragraphs describe the services available for creating, accessing, managing, and deleting these entities with mass storage. Both synchronous and asynchronous services are defined.

Naming and Directory Services

These services allow the access of files and directories through logical names rather than the actual hardware device naming conventions. The services allow sharing of files at various levels. For example, the services may not allow any shared naming of files and directories between systems, or they may allow shared files by explicit naming, or they may allow shared files by implicit naming. The directory services present a view or views of the directory structure to the application or target system operator.

File Modification Primitives

Primitive services for files and directories are:

— Read a portion of the file.

— Write to a portion of the file.

— Open access to a file.

— Create a new file.

— Close access to a file.

— Delete a file.

— Copy a file.

— Merge two or more files.

— Append one file to another.

— Split one file into two or more files.

— Support read and write locks at both the record and file levels.

These services may be very complex. For example, the access to read or write may be direct (by record number), sequential (one record at a time), or indexed (by a key). The services must also support a variety of file structures, including linked, segmented, contiguous, serial, and directory.

File Support Services

Additional services support the physical devices on which the files and directory reside. These services include the dismounting/mounting of medium, the formatting of medium, and the partitioning of media.

Realtime Files

Realtime systems often need special files to ensure fast, bounded, and consistent performance in time critical situations. The need for a bounded response time for a given I/O function drives the design of these files and services. One service preallocates the complete disk space needed for a file at creation time, while another guarantees that records within files are aligned in an optimal way (such as along word boundaries). Services support the access of records within the file in ways that make response time constant or bounded, including by direct access.

4.2.4.1.7 Event, Error, and Exception Management Services

These services provide a common facility for the generation and communication of asynchronous events among the system and application programs. A major use of the event services is to report error conditions, but they are also used by device drivers and the platform to provide an indication of some condition to the application programs. These services are:

— Event and error receipt.

— Event and error distribution.

— Event and error management, including user-selectable error processing alternatives (filtering, retry, ignore, accumulate occurrences).

— Event logging.

— Enable/disable and mask/unmask interrupts.

4.2.4.1.8 Time Services

Timers may be a static or dynamic resource on the system, necessitating a variety of allocation and management strategies. These services are used by applications to perform a variety of services based on absolute and relative time. These services are:

— Create a timer.

— Delete a timer.

— Initiate the measurement of an arbitrary specified time duration.

— Receive an indication when the specified duration has elapsed.

— Read the current value of a timer.

— Initialize a timer with a value and count direction (i.e., increment or decrement).

— Trigger a timer to begin incrementing or decrementing.

— Associate with a timer some action to be taken when the specified duration has elapsed.

4.2.4.1.9 Memory Management Services

These services are used by application processes and tasks to request additional memory and return it to the processor for reuse. They cover the services required to fulfill the needs of both virtual and fixed memory. Specifically, there is a service for locking pages in real memory to support the needs of virtual memory systems.

4.2.4.1.10 Logical Naming Services

These services allow the usage of system resources through logical names rather than the actual hardware device naming conventions. Furthermore, they allow the resources of other processor nodes to be accessed via a logical name so that no knowledge of the resource's location is needed and the resource's location may change over time. Logical names are also used by security services to hide resources from unauthorized processes by only letting authorized processes know the logical name that is needed to use the physical resource.

The logical name to physical name relationship can be one to many, many to one, or many to many. Many times, one physical resource may have multiple logical names as well as one logical name representing a ‘‘bank’’ of available physical resources. These services must provide the proper resolution of names, logical and physical, in all of these cases.

4.2.4.1.11 System Initialization, Reinitialization, and Shutdown Services

System initialization consists of services for a complete restarting of the software, starting up the attached hardware subsystems devices, doing subsystem and system self tests, and completely initializing the database.

System reinitialization consists of services for restarting the software while using the existing database information. The software may have to be reloaded and the database may have been reestablished by a system recovery. Attached hardware subsystems may also need to be reinitialized.

Reinitialization also includes a function to restart applications redistributed to other processors after a processor module failure. Within a processor, there is a service to initialize applications in a system with the existing software, but with the database reinitialized. Also within a processor, there is a service to restart the applications in a system with the existing software and database retained.

Shutdown services are those required to perform planned orderly shutdown at the local and remote levels for each and all processor(s) throughout a system. These services support both crisis and non-crisis situations that call for system shutdown. They make sure that the persistent store is in a consistent state, see to the clean termination of all processes, programs, devices, etc., and take care of user notification. They also provide for the running of system diagnostics.

4.2.4.2 External Environment Interface Services

Data Interchange External Environment Interface Services are required by the System Services. Of particular interest are the formats, locations, and procedures for using system administration files, such as password files, system startup files, and configuration files.

4.2.4.3 Interapplication Software Entity Services

This could include support for generalized network/multisession services, such as message handling between system components, global object definition specification, and intermediate language definition.

4.2.4.4 Resource Management Services

These services provide general management functions across the entire platform. They consist primarily of system administration-oriented functions (i.e., management of system interfaces within the scope of the administrator, such as setting up defaults and limits.)

4.2.4.4.1 System Operator Services

The system operator needs to access and control the system services in order to allow the platform to perform properly. If a system has an operator, the major functions that need to be supported are system control, reconfiguration, and status reporting. Currently, these services are usually made available to an operator through a command language interpreter, which is an application program that accesses these system services.

Note that the Windowing Services provide the building blocks (menu utilities, command parsers, etc.) for building the user interface while the System Operator Services make available operating system status and control functions to appropriate application programs with the proper security level.

These services support general conventions and specifications for interaction between system components.

4.2.4.4.2 System Administration

These services and procedures are those required to assure management and allocation of system services to system users, both local and remote. They consist primarily of those services required to establish authorized users of the system, with associated allocation of processor resources, including memory, processor time, priority, and mass storage space. These services are both static (as in the establishment of a new user identification) and dynamic (as in login/logout).

4.2.5 Standards, Specifications, and Gaps

Table 4-2 – System Services Standards Activities

Service	Specification	Subclause	
Process Management	ISO/IEC 9945-1	4.2.5.1.1.1	D
Task Management	ISO/IEC 9945-1	4.2.5.1.1.1	D
Environment Services	ISO/IEC 9945-1	4.2.5.1.1.1	D
Node Internal Comm/Synch	ISO/IEC 9945-1	4.2.5.1.1.1	D
Generalized I/O	ISO/IEC 9945-1	4.2.5.1.1.1	D
	OSF AES – OSC	4.2.5.3.1	D
	SVID	4.2.5.3.1	D
File Oriented Services	ISO/IEC 9945-1	4.2.5.1.1.1	D
Event, Error, and Exception	ISO/IEC 9945-1	4.2.5.1.1.1	D
	OSF AES – OSC	4.2.5.3.1	D
	SVID	4.2.5.3.1	D
Time Services	ISO/IEC 9945-1	4.2.5.1.1.1	D
Memory Management	ISO/IEC 9945-1	4.2.5.1.1.1	D
Logical Naming	ISO/IEC 9945-1	4.2.5.1.1.1	D
System Init/Reinit/Shutdown	ISO/IEC 9945-1	4.2.5.1.1.1	D
	OSF AES – OSC	4.2.5.3.1	D
	SVID	4.2.5.3.1	D

4.2.5.1 Current Standards in the POSIX OSE

4.2.5.1.1 International Standards

4.2.5.1.1.1 Portable Operating System Interface (POSIX) Part 1

ISO/IEC 9945-1 (IEEE Std 1003.1) is the first in a set of planned international POSIX standards. It defines services and characteristics that need to be in the platform for portable applications, as do some of the other planned standards. Another type of POSIX-related standard is bindings for those services to specific languages. The third type deals with concepts that cross between various groupings of services, such as security and distributed processing.

Objectives

The purpose of the ISO/IEC 9945-1 standard is to define a standard operating system interface based on the UNIX Operating System documentation to support application portability at the source level. The document is intended for systems implementors and applications software developers.

In addition to ISO/IEC 9945-1, ISO is planning to publish ISO/IEC 9945-3 Conformance Test Procedures (which will be identical to IEEE 1003.3). This standard will cover conformance testing procedures for 9945-1.

Functionality

Table 4-3 outlines the contents of the current document. This document is identical in its ISO/IEC form (ISO/IEC 9945-1) and the US national standard form (IEEE Std 1003.1). Revisions are currently in progress to deal with:

- A language-independent services specification
- A unified data interchange format
- Service interfaces for control of character cell terminals
- Miscellaneous functions identified in comments on the current standard.

Table 4-3 – Functionality of POSIX .1 Standard

File system organization, and file naming conventions
System configuration and file system configuration characteristics
Error messages and reporting mechanism (*errno*)
Application environment information (*environ*)
Process creation, management, and termination: *exec*(), *fork*(), *wait*()
Process environment: user ID, process ID, Group ID
Exception conditions and handling (signals)
Timer operations
File and Directory operations: FIFO files, pipes, status, open/close, read/write
File protection mechanisms
Record and file locking mechanism
Device specific functions: Terminal controls: Processing modes: echo, baud rate, modem termination
C language specific routines: *setlocale*(), nonlocal jumps
User and Group database information (excluding password information)
Data interchange formats (USTAR and CPIO)
Also included is a rationale appendix that provides insight on the selection of various functions and features, including some guidance to developers to understand what types of variations may exist and how that can impact portability.

Options and Variations

The ISO/IEC 9945-1 standard draws heavily upon major implementations of the UNIX Operating System, including System V and the Berkeley versions. Where a specific behavior was clearly needed (e.g., signals), only a single behavior was permitted. However, there are points where functions were considered optional and others where two different behaviors were considered acceptable. However, in many cases, a solid technical argument favoring one approach over the other was not established. In this case, two behaviors (usually System V and BSD) are defined as being permitted. This is of benefit in writing portable applications, since those that can tolerate both behaviors will run on a wider range of systems. It is also a slight disadvantage in writing such applications, since it can mean handling a wider range of implementations.

NOTE: In a related U.S. national standard, the United States Government Department of Commerce National Institute of Standards and Technology has adopted the ISO/IEC 9945-1 standard as a Federal Information Processing Standard (FIPS 151-1) for use in computer systems procurement. FIPS 151-1 requires options and characteristics that have not been required by ISO/IEC 9945-1, requiring implementations to function in a specific way in many of these cases. While buyers may want to make such requirements, the impact should be understood. Applications written for FIPS systems may only run on FIPS systems, whereas applications written for full ISO/IEC 9945-1 implementations will run on both FIPS systems and POSIX conforming systems that are not FIPS conforming. FIPS systems will run applications written either way.

FIPS 151-1 requires all three POSIX.1 options. In addition, it eliminates several additional possible behaviors, thus narrowing the variation between conforming implementations.

4.2.5.1.2 Regional Standards

None.

4.2.5.1.3 National Standards

None.

4.2.5.2 Emerging Standards in the POSIX OSE

4.2.5.2.1 International Standards

None.

4.2.5.2.2 Regional Standards

None.

4.2.5.2.3 National Standards

The IEEE P1003.4 Group is defining realtime extensions to ISO/IEC 9945-1. Draft 9 of the realtime POSIX extensions proposes standardized interfaces to the following functions:

- Response to asynchronous events
- Priority interrupts and scheduling
- Preemptive scheduling
- Memory locking
- High-performance file system (contiguous or other)
- Realtime timers (with nanosecond resolution times)
- Shared memory
- Semaphores
- Interprocess communications (message passing)
- Asynchronous event notification
- Synchronous input and output.

The P1003.4 group is also specifying an interface to threads (P1003.4a).

4.2.5.3 Gaps in Available Standards

While ISO/IEC 9945-1 and P1003.4 both represent very important work, they do not yet address all of the services indicated in 4.2.4. Areas of particular shortfall include Event, Error, and Exception Management Services, some Generalized I/O Services (particularly concerning services for device drivers), and System Initialization, Reinitialization, and Shutdown Services. In addition, Security (see 5.2) and Reliability, Adaptability, and Maintainability (Fault Tolerance; see 5.4) services are not reflected in these two base standards, and some capabilities are explicitly considered to be implementation defined. For some of the services discussed here, adequate consideration is not given to the implications of multiprocessor and distributed implementations of the services and interface provided. Finally, since these are intended to be base standards (or, in the case of P1003.4, an extension to a base standard), profiles are needed in order to select appropriate features and provide appropriate combinations with other related capabilities.

4.2.5.3.1 Public Specifications

The following are public specifications that define interfaces to services for which no formal standards are currently available.

OSF/1

The Open Software Foundation (OSF) ‘‘Application Environment Specification (AES)—Operating System Component’’ (OSC).

Service Gaps Addressed:

- Generalized I/O
- Event, Error, and Exception

— System Init/Reinit/Shutdown D

SVID D

The AT&T System V Interface Definition (SVID), Issue 3. D

Service Gaps Addressed: D

— Generalized I/O D

— Event, Error, and Exception D

— System Init/Reinit/Shutdown D

4.2.5.3.2 Unsatisfied Service Requirements

There are two significant areas of the services described above for which no standards currently exist. One is the considerations implied by the use of multiprocessors to implement some or all of the services described herein. The other area is that of interfaces to logical device drivers.

4.2.5.3.3 Standards/Specifications Outside the POSIX OSE

4.2.6 OSE Cross-Category Services

4.2.6.1 Capability and Security Services

These services support the ability of the system to control usage such that system integrity is protected from inadvertent or malicious misuse. These protection services provide a mechanism for the enforcement of the policies governing resource usage. Note that many of the security services are implicit services; i.e., they are provided without an explicit request to the operating system. There are two distinct classes of system access with which operating system services must be concerned: physical access and logical access.

Security services at the physical level are used to protect against security compromise, given unauthorized personnel may have physical access to system hardware. Typically, the physical access is to a terminal and/or terminal/display cables; however, physical access may also include network cables, central processing units, disk drives, or tape drives. Prevention of physical access by unauthorized personnel may require different operating system services under different circumstances.

Logical access is the ability to interact with the operating system via a terminal/display. Security services at the logical level can be implemented through passwords and watchdog timers.

Capability services attach operation lists that limit a process's ability to act on resource objects. This is to ensure the resources are not misused. Access to resources can be protected by services using capability lists as well as access lists, lock/key mechanisms, global tables, or through dynamic protection structure services.

Prevention of Unauthorized Access

The system may need to be guarded from attempted access by unauthorized personnel. The point of access to the operating system that is typically of concern is through the API. Given the mode of operation (system high, multilevel, open) at which the system is operating, these services differ and have differing implications on other system services (such as reliability and naming) and system performance.

Prevention of Data Compromise

These services prevent access of data by users not authorized to the data. These services may be implemented using access lists on files (and directories) and/or encryption of data or in other ways.

Prevention of Service Denial

These services ensure that a service request will be met by the operating system in a reasonable time if the requester is authorized to use the service. These services ensure that a bandit user or process cannot cause system malfunction by monopolizing system services or resources.

Security Administration

This category involves services to allow the management of the security system, including the administration of permissions to personnel, data, and services as well as capability lists. In addition, it permits the administration access mechanisms (most often passwords and capability lists) and services that allow the system to switch modes of operation. The services will likely be accessed by the target system operator with security responsibilities through the target system operator services.

4.2.6.2 Fault Management

Many of the fault management capabilities described in 5.4 are normally the responsibility of the operating system and fall under the sorts of services described in this clause.

4.2.7 Related Standards

The following emerging standards are related to the services covered in this D
clause, in as much as they address at some level services either explicitly listed in or implied by the services found in 4.2.4:

P1003.6 Security Interface for POSIX.

P1003.12 Protocol Independent Interfaces (for networks).

P1238 OSI Application Program Interfaces (initial effort is to provide at least sufficient facilities for the support of FTAM API specifications).

4.2.8 Open Issues

[PO: PCTE and CAIS -A have been moved here largely because it is not clear what to do with them. They are not adequately accommodated by this model. They are both hybrids of operating system and database management system capabilities that seem to belong either everywhere or nowhere. They could both well be used in conjunction with a P1003.1 implementation, but they could also be implemented on other base operating systems, or implementations could even expand their capabilities to provide full operating systems. P1003.0 must decide what to do with them.]

PCTE

An effort by the European Computer Manufacturers Association (ECMA) has resulted in the definition by Technical Committee 33 of the Basis for the Portable Common Tools Environment (PCTE). This is now an ECMA standard and is referred to as Standard ECMA -149.

CAIS -A

MIL -STD -1838A (CAIS -A) was developed by the US Department of Defense to provide a common foundation for Ada Programming Support Environments. Similar in nature to PCTE (see above), it too covers many of the system services covered by 4.2.4. In addition, it provides data management services such as those discussed in 4.4 and data interchange services (specifically, a Common External Form) similar to those discussed in 4.5.

D

X/Open

[PO: The following text on XPG was given to me at the last meeting. However, I cannot tell from the text whether XPG is more like this guide or more like P1003.1. If it is the former, it does not belong here. If it is the latter, then this text should be moved into 4.2.5.3.1.2.]

X/Open is a consortium formed to create interface definitions and guidelines that will provide portability and interoperability to its members' products. Its primary output is the X/Open Portability Guide. The functional content of this guide is heavily based on the AT&T SVID. The current version is XPG 3.

[PO: Documentation of open issue: what are we going to do with FIPS ? For this section, the decision was made in Chicago to stick FIPS 151-1 in the section on 9945-1, but this proved quite awkward: it couldn't be placed in the regular text because it didn't pertain directly to 9945-1 (i.e., it was inappropriate to discuss it in the text that was to be devoted to ''international standards''), so it ended up in an informative note. I don't find this a satisfactory answer. I think that it properly belongs under Government/Legal Standards; the ''gap'' it fulfills is one of

profiling.

Documentation of new open issue: We are beginning to ‘‘use our heads,’’ as indicated by the discussions we had on this section in Chicago. It is essential that we capture the rationale for our decisions, particularly for our inclusions and exclusions. I have tried to do that here to some extent, but what is our overall plan for making sure that our rationales are not lost?]

4.3 Network Services

Responsibility: Charles Severance

HLJ : This is a complete clause replacement in Draft 13. It is not further diff-marked.

4.3.1 Overview and Rationale

This clause describes the network services component of the application platform. It also describes the services provided to application programs and users, and it describes current and emerging standards that are standardizing these services.

Applications gain direct access to network services via the POSIX API. The network is just another system resource (albeit an important one) allocated among the competing processes.

4.3.2 Scope

Network services include those capabilities required to allow application programs to execute on platforms that may be connected via a network. They cover the areas of file transfer, namespace and directory services, services in support of distributed environments such as remote procedure call, distributed time management, transparent file access, and data representation services. The application programs using these services should be able to access them via a high-level, context-insensitive or low-level, context-dependent interface.

In the open systems and distributed systems environments, interoperability is of equal or greater importance than portability. The interfaces provided by the network services must be network protocol independent and provide for this level of interoperability. The network protocols defined for both Open Systems Interconnect (OSI) and Internet Protocol Suite (IPS) for TCP/IP should provide the basis for the open networking interfaces; however, these interfaces should not preclude the use of some subsequent networking protocol in the future.

Rationale: It is important for an open system to interoperate with more systems than just other open systems. Many open systems users will have requirements to interoperate with non-ISO networks for the near future.

4.3.3 POSIX OSE Network Services Reference Model

This subclause identifies the entities and interfaces specific to the construction of an POSIX Network Environment. This environment is consistent with and extends the environment of Section 3.

As illustrated in Figure 4-4, the components of a network architecture that require standardization are divided into two groups called external environment interfaces (EEI) and application program interfaces (API).