

Information technology — Portable Operating System Interface (POSIX®) —

Part 2: System Interfaces

ICS 35.060

National foreword

This British Standard reproduces verbatim ISO/IEC 9945-2:2002 and implements it as the UK national standard. BS ISO/IEC 9945-2:2002 along with BS ISO/IEC 9945-1:2002, BS ISO/IEC 9945-3:2002 and BS ISO/IEC 9945-4:2002 supersedes BS ISO/IEC 9945-1:1990(adopted as BS EN 29945-1) and BS EN ISO/IEC 9945-2:1995 which are both withdrawn.

The UK participation in its preparation was entrusted to Technical Committee IST/5, Programming languages, their environments and system software interfaces, which has the responsibility to:

- aid enquirers to understand the text;
- present to the responsible international/European committee any enquiries on the interpretation, or proposals for change, and keep the UK interests informed;
- monitor related international and European developments and promulgate them in the UK.

A list of organizations represented on this committee can be obtained on request to its secretary.

Cross-references

The British Standards which implement international publications referred to in this document may be found in the *BSI Catalogue* under the section entitled “International Standards Correspondence Index”, or by using the “Search” facility of the *BSI Electronic Catalogue* or of British Standards Online.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 April 2003

Summary of pages

This document comprises a front cover, an inside front cover, the ISO/IEC title page, the ISO/IEC inside front cover, pages i to xxx, pages 1 to 1690, an inside back cover and a back cover.

The BSI copyright date displayed in this document indicates when the document was last issued.

Amendments issued since publication

Amd. No.	Date	Comments

© BSI 30 April 2003

ISBN 0 580 41307 1

INTERNATIONAL STANDARD

BS ISO/IEC 9945-2:2002

**ISO/IEC
9945-2**

**System Interfaces,
IEEE Std 1003.1**

Second edition
2002-12-15

Information technology — Portable Operating System Interface (POSIX®) — Part 2: System Interfaces

*Technologies de l'information — Interface pour la portabilité des systèmes
(POSIX®) —*

Partie 2: Interfaces systèmes



Reference number
ISO 9945-2:2002
System Interfaces, IEEE Std 1003.1-2001
Documents
Document Sharing Hub

ISO/IEC 9945-2:2002(E)

IEEE Std 1003.1™-2001
(Revision of IEEE Std 1003.1-1996
and IEEE Std 1003.2-1992)

The Open Group Technical Standard
Base Specifications, Issue 6

Information technology—Portable Operating System Interface (POSIX®)

Part 2: System Interfaces

Sponsor

Portable Applications Standards Committee
of the
IEEE Computer Society

and

The Open Group



Adopted as an International Standard by the
International Organization for Standardization
and by the
International Electrotechnical Commission



THE *Open* GROUP



Abstract

This standard defines a standard operating system interface and environment, including a command interpreter (or “shell”), and common utility programs to support applications portability at the source code level. It is the single common revision to IEEE Std 1003.1-1996, IEEE Std 1003.2-1992, and the Base Specifications of The Open Group Single UNIX[®]† Specification, Version 2. This standard is intended to be used by both applications developers and system implementors and comprises four major components (each in an associated volume):

- General terms, concepts, and interfaces common to all volumes of this standard, including utility conventions and C-language header definitions, are included in the Base Definitions volume.
- Definitions for system service functions and subroutines, language-specific system services for the C programming language, function issues, including portability, error handling, and error recovery, are included in the System Interfaces volume.
- Definitions for a standard source code-level interface to command interpretation services (a “shell”) and common utility programs for application programs are included in the Shell and Utilities volume.
- Extended rationale that did not fit well into the rest of the document structure, containing historical information concerning the contents of this standard and why features were included or discarded by the standard developers, is included in the Rationale (Informative) volume.

The following areas are outside the scope of this standard:

- Graphics interfaces
- Database management system interfaces
- Record I/O considerations
- Object or binary code portability
- System configuration and resource availability

This standard describes the external characteristics and facilities that are of importance to applications developers, rather than the internal construction techniques employed to achieve these capabilities. Special emphasis is placed on those functions and facilities that are needed in a wide variety of commercial applications.

Keywords

application program interface (API), argument, asynchronous, basic regular expression (BRE), batch job, batch system, built-in utility, byte, child, command language interpreter, CPU, extended regular expression (ERE), FIFO, file access control mechanism, input/output (I/O), job control, network, portable operating system interface (POSIX[®]†), parent, shell, stream, string, synchronous, system, thread, X/Open System Interface (XSI)

† See **Trademarks** (on page xxii).

International Standard ISO/IEC 9945-2:2002(E)

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

The main task of the joint technical Committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 9945 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 9945-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

This second edition of ISO/IEC 9945-2, together with ISO/IEC 9945-1, ISO/IEC 9945-3 and ISO/IEC 9945-4, cancels and replaces ISO/IEC 9945-1:1996 and ISO/IEC 9945-2:1993, which have been technically revised.

ISO/IEC 9945 consists of the following parts, under the general title *Information technology — Portable Operating System Interface (POSIX®)*:

- *Part 1: Base Definitions*
- *Part 2: System Interfaces*
- *Part 3: Shell and Utilities*
- *Part 4: Rationale*

IEEE

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property, or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied "AS IS".

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of the IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with the IEEE.¹ Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, U.S.A.

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE Standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

A patent holder has filed a statement of assurance that it will grant licenses under these rights without compensation or under reasonable rates and non-discriminatory, reasonable terms and conditions to all applicants desiring to obtain such licenses. The IEEE makes no representation as to the reasonableness of rates and/or terms and conditions of the license agreements offered by patent holders. Further information may be obtained from the IEEE Standards Department.

The IEEE and its designees are the sole entities that may authorize the use of IEEE-owned certification marks and/or trademarks to indicate compliance with the materials set forth herein. Authorization to photocopy portions of any individual standard for internal or personal use is granted in the U.S. by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to the Copyright Clearance Center.² Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center. To arrange for payment of the licensing fee, please contact:

Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923, U.S.A., Tel.: +1 978 750 8400

Amendments, corrigenda, and interpretations for this standard, or information about the IEEE standards development process, may be found at <http://standards.ieee.org>.

Full catalog and ordering information on all IEEE publications is available from the IEEE Online Catalog & Store at <http://shop.ieee.org/store>.

1. For this standard, please send comments via the Austin Group as requested on page iii.

2. Please refer to the special provisions for this standard on page iii concerning permissions from both copyright holders and arrangements to cover photocopying and reproduction across the world, as well as by commercial organizations wishing to license the material for use in product documentation.

The Open Group

The Open Group, a vendor and technology-neutral consortium, is committed to delivering greater business efficiency by bringing together buyers and suppliers of information technology to lower the time, cost, and risks associated with integrating new technology across the enterprise.

The Open Group's mission is to offer all organizations concerned with open information infrastructures a forum to share knowledge, integrate open initiatives, and certify approved products and processes in a manner in which they continue to trust our impartiality.

In the global eCommerce world of today, no single economic entity can achieve independence while still ensuring interoperability. The assurance that products will interoperate with each other across differing systems and platforms is essential to the success of eCommerce and business workflow. The Open Group, with its proven testing and certification program, is the international guarantor of interoperability in the new century.

The Open Group provides opportunities to exchange information and shape the future of IT. The Open Group's members include some of the largest and most influential organizations in the world. The flexible structure of The Open Groups membership allows for almost any organization, no matter what their size, to join and have a voice in shaping the future of the IT world.

More information is available on The Open Group web site at <http://www.opengroup.org>.

The Open Group has over 15 years' experience in developing and operating certification programs and has extensive experience developing and facilitating industry adoption of test suites used to validate conformance to an open standard or specification. The Open Group portfolio of test suites includes the *Westwood* family of tests for this standard and the associated certification program for Version 3 of the Single UNIX Specification, as well tests for CDE, CORBA, Motif, Linux, LDAP, POSIX.1, POSIX.2, POSIX Realtime, Sockets, UNIX, XPG4, XNFS, XTI, and X11. The Open Group test tools are essential for proper development and maintenance of standards-based products, ensuring conformance of products to industry-standard APIs, applications portability, and interoperability. In-depth testing identifies defects at the earliest possible point in the development cycle, saving costs in development and quality assurance.

More information is available at <http://www.opengroup.org/testing>.

The Open Group publishes a wide range of technical documentation, the main part of which is focused on development of Technical and Product Standards and Guides, but which also includes white papers, technical studies, branding and testing documentation, and business titles. Full details and a catalog are available at <http://www.opengroup.org/pubs>.

As with all *live* documents, Technical Standards and Specifications require revision to align with new developments and associated international standards. To distinguish between revised specifications which are fully backwards compatible and those which are not:

- A new *Version* indicates there is no change to the definitive information contained in the previous publication of that title, but additions/extensions are included. As such, it *replaces* the previous publication.
- A new *Issue* indicates there is substantive change to the definitive information contained in the previous publication of that title, and there may also be additions/extensions. As such, both previous and new documents are maintained as current publications.

Readers should note that Corrigenda may apply to any publication. Corrigenda information is published at <http://www.opengroup.org/corrigenda>.

Full catalog and ordering information on all Open Group publications is available at <http://www.opengroup.org/pubs>.

Contents

Chapter 1	Introduction.....	1
1.1	Scope.....	1
1.2	Conformance	1
1.3	Normative References	1
1.4	Change History	1
1.5	Terminology	1
1.6	Definitions	3
1.7	Relationship to Other Formal Standards	3
1.8	Portability	3
1.8.1	Codes	3
1.9	Format of Entries.....	11
Chapter 2	General Information	13
2.1	Use and Implementation of Functions	13
2.2	The Compilation Environment	13
2.2.1	POSIX.1 Symbols	13
2.2.1.1	The _POSIX_C_SOURCE Feature Test Macro	14
2.2.1.2	The _XOPEN_SOURCE Feature Test Macro	14
2.2.2	The Name Space.....	14
2.3	Error Numbers.....	21
2.3.1	Additional Error Numbers.....	28
2.4	Signal Concepts	28
2.4.1	Signal Generation and Delivery.....	28
2.4.2	Realtime Signal Generation and Delivery	29
2.4.3	Signal Actions	31
2.4.4	Signal Effects on Other Functions	34
2.5	Standard I/O Streams.....	34
2.5.1	Interaction of File Descriptors and Standard I/O Streams	35
2.5.2	Stream Orientation and Encoding Rules	37
2.6	STREAMS	38
2.6.1	Accessing STREAMS.....	39
2.7	XSI Interprocess Communication	39
2.7.1	IPC General Description.....	40
2.8	Realtime	41
2.8.1	Realtime Signals.....	41
2.8.2	Asynchronous I/O	41
2.8.3	Memory Management	43
2.8.3.1	Memory Locking.....	43
2.8.3.2	Memory Mapped Files.....	43
2.8.3.3	Memory Protection.....	44
2.8.3.4	Typed Memory Objects	44
2.8.4	Process Scheduling.....	44

2.8.5	Clocks and Timers	48
2.9	Threads.....	50
2.9.1	Thread-Safety.....	50
2.9.2	Thread IDs.....	51
2.9.3	Thread Mutexes.....	51
2.9.4	Thread Scheduling.....	52
2.9.5	Thread Cancellation	54
2.9.5.1	Cancelability States	54
2.9.5.2	Cancellation Points	55
2.9.5.3	Thread Cancellation Cleanup Handlers.....	57
2.9.5.4	Async-Cancel Safety.....	57
2.9.6	Thread Read-Write Locks.....	58
2.9.7	Thread Interactions with Regular File Operations	58
2.10	Sockets.....	58
2.10.1	Address Families.....	58
2.10.2	Addressing	59
2.10.3	Protocols	59
2.10.4	Routing.....	59
2.10.5	Interfaces.....	59
2.10.6	Socket Types.....	59
2.10.7	Socket I/O Mode.....	60
2.10.8	Socket Owner.....	60
2.10.9	Socket Queue Limits	60
2.10.10	Pending Error.....	60
2.10.11	Socket Receive Queue.....	61
2.10.12	Socket Out-of-Band Data State	61
2.10.13	Connection Indication Queue	62
2.10.14	Signals	62
2.10.15	Asynchronous Errors	62
2.10.16	Use of Options.....	63
2.10.17	Use of Sockets for Local UNIX Connections.....	66
2.10.17.1	Headers	66
2.10.18	Use of Sockets over Internet Protocols.....	66
2.10.19	Use of Sockets over Internet Protocols Based on IPv4.....	67
2.10.19.1	Headers	67
2.10.20	Use of Sockets over Internet Protocols Based on IPv6.....	67
2.10.20.1	Addressing	67
2.10.20.2	Compatibility with IPv4.....	68
2.10.20.3	Interface Identification.....	68
2.10.20.4	Options.....	69
2.10.20.5	Headers	70
2.11	Tracing.....	70
2.11.1	Tracing Data Definitions.....	71
2.11.1.1	Structures.....	71
2.11.1.2	Trace Stream Attributes	75
2.11.2	Trace Event Type Definitions	76
2.11.2.1	System Trace Event Type Definitions	76
2.11.2.2	User Trace Event Type Definitions.....	79

Contents

	2.11.3	Trace Functions.....	79
	2.12	Data Types.....	80
Chapter	3	System Interfaces	83
		Index.....	1663

List of Tables

2-1	Value of Level for Socket Options.....	63
2-2	Socket-Level Options.....	64
2-3	Trace Option: System Trace Events.....	77
2-4	Trace and Trace Event Filter Options: System Trace Events.....	78
2-5	Trace and Trace Log Options: System Trace Events.....	78
2-6	Trace, Trace Log, and Trace Event Filter Options: System Trace Events.....	79
2-7	Trace Option: User Trace Event	79

Foreword

Structure of the Standard

This document was originally developed by the Austin Group, a joint working group of members of the IEEE, members of The Open Group, and members of ISO/IEC Joint Technical Committee 1, as one of the four volumes of IEEE Std 1003.1-2001. The standard was approved by ISO and IEC and published in four parts, correlating to the original volumes.

A mapping of the parts to the volumes is shown below:

ISO/IEC 9945 Part	IEEE Std 1003.1 Volume	Description
9945-1	Base Definitions	General terms, concepts, and interfaces common to all parts of ISO/IEC 9945, including utility conventions and C-language header definitions, are included.
9945-2	System Interfaces	Definitions for system service functions and subroutines, language-specific system services for the C programming language, function issues, including portability, error handling, and error recovery, are included.
9945-3	Shell and Utilities	Definitions for a standard source code-level interface to command interpretation services (a “shell”) and common utility programs for application programs are included.
9945-4	Rationale	Extended rationale that did not fit well into the rest of the document structure, containing historical information concerning the contents of ISO/IEC 9945 and why features were included or discarded by the standard developers, is included.

All four parts comprise the entire standard, and are intended to be used together to accomodate significant internal referencing among them. POSIX conforming systems are required to support all four parts.

Introduction

Note: This introduction is not part of IEEE Std 1003.1-2001, Standard for Information Technology — Portable Operating System Interface (POSIX).

This standard has been jointly developed by the IEEE and The Open Group. It is both an IEEE Standard and an Open Group Technical Standard.

The Austin Group

This standard was developed, and is maintained, by a joint working group of members of the IEEE Portable Applications Standards Committee, members of The Open Group, and members of ISO/IEC Joint Technical Committee 1. This joint working group is known as the Austin Group.³ The Austin Group arose out of discussions amongst the parties which started in early 1998, leading to an initial meeting and formation of the group in September 1998. The purpose of the Austin Group has been to revise, combine, and update the following standards: ISO/IEC 9945-1, ISO/IEC 9945-2, IEEE Std 1003.1, IEEE Std 1003.2, and the Base Specifications of The Open Group Single UNIX Specification.

After two initial meetings, an agreement was signed in July 1999 between The Open Group and the Institute of Electrical and Electronics Engineers (IEEE), Inc., to formalize the project with the first draft of the revised specifications being made available at the same time. Under this agreement, The Open Group and IEEE agreed to share joint copyright of the resulting work. The Open Group has provided the chair and secretariat for the Austin Group.

The base document for the revision was The Open Group's Base volumes of its Single UNIX Specification, Version 2. These were selected since they were a superset of the existing POSIX.1 and POSIX.2 specifications and had some organizational aspects that would benefit the audience for the new revision.

The approach to specification development has been one of “write once, adopt everywhere”, with the deliverables being a set of specifications that carry the IEEE POSIX designation and The Open Group's Technical Standard designation, and, if approved, an ISO/IEC designation. This set of specifications forms the core of the Single UNIX Specification, Version 3.

This unique development has combined both the industry-led efforts and the formal standardization activities into a single initiative, and included a wide spectrum of participants. The Austin Group continues as the maintenance body for this document.

Anyone wishing to participate in the Austin Group should contact the chair with their request. There are no fees for participation or membership. You may participate as an observer or as a contributor. You do not have to attend face-to-face meetings to participate; electronic participation is most welcome. For more information on the Austin Group and how to participate, see <http://www.opengroup.org/austin>.

3. The Austin Group is named after the location of the inaugural meeting held at the IBM facility in Austin, Texas in September 1998.

Background

The developers of this standard represent a cross section of hardware manufacturers, vendors of operating systems and other software development tools, software designers, consultants, academics, authors, applications programmers, and others.

Conceptually, this standard describes a set of fundamental services needed for the efficient construction of application programs. Access to these services has been provided by defining an interface, using the C programming language, a command interpreter, and common utility programs that establish standard semantics and syntax. Since this interface enables application writers to write portable applications—it was developed with that goal in mind—it has been designated POSIX,⁴ an acronym for Portable Operating System Interface.

Although originated to refer to the original IEEE Std 1003.1-1988, the name POSIX more correctly refers to a *family* of related standards: IEEE Std 1003.*n* and the parts of ISO/IEC 9945. In earlier editions of the IEEE standard, the term POSIX was used as a synonym for IEEE Std 1003.1-1988. A preferred term, POSIX.1, emerged. This maintained the advantages of readability of the symbol “POSIX” without being ambiguous with the POSIX family of standards.

Audience

The intended audience for this standard is all persons concerned with an industry-wide standard operating system based on the UNIX system. This includes at least four groups of people:

1. Persons buying hardware and software systems
2. Persons managing companies that are deciding on future corporate computing directions
3. Persons implementing operating systems, and especially
4. Persons developing applications where portability is an objective

Purpose

Several principles guided the development of this standard:

- Application-Oriented

The basic goal was to promote portability of application programs across UNIX system environments by developing a clear, consistent, and unambiguous standard for the interface specification of a portable operating system based on the UNIX system documentation. This standard codifies the common, existing definition of the UNIX system.

- Interface, Not Implementation

This standard defines an interface, not an implementation. No distinction is made between library functions and system calls; both are referred to as functions. No details of the implementation of any function are given (although historical practice is sometimes indicated in the RATIONALE section). Symbolic names are given for constants (such as signals and error numbers) rather than numbers.

4. The name POSIX was suggested by Richard Stallman. It is expected to be pronounced *pahz-icks*, as in *positive*, not *poh-six*, or other variations. The pronunciation has been published in an attempt to promulgate a standardized way of referring to a standard operating system interface.

Introduction

- Source, Not Object, Portability

This standard has been written so that a program written and translated for execution on one conforming implementation may also be translated for execution on another conforming implementation. This standard does not guarantee that executable (object or binary) code will execute under a different conforming implementation than that for which it was translated, even if the underlying hardware is identical.

- The C Language

The system interfaces and header definitions are written in terms of the standard C language as specified in the ISO C standard.

- No Superuser, No System Administration

There was no intention to specify all aspects of an operating system. System administration facilities and functions are excluded from this standard, and functions usable only by the superuser have not been included. Still, an implementation of the standard interface may also implement features not in this standard. This standard is also not concerned with hardware constraints or system maintenance.

- Minimal Interface, Minimally Defined

In keeping with the historical design principles of the UNIX system, the mandatory core facilities of this standard have been kept as minimal as possible. Additional capabilities have been added as optional extensions.

- Broadly Implementable

The developers of this standard endeavored to make all specified functions implementable across a wide range of existing and potential systems, including:

1. All of the current major systems that are ultimately derived from the original UNIX system code (Version 7 or later)
2. Compatible systems that are not derived from the original UNIX system code
3. Emulations hosted on entirely different operating systems
4. Networked systems
5. Distributed systems
6. Systems running on a broad range of hardware

No direct references to this goal appear in this standard, but some results of it are mentioned in the Rationale (Informative) volume.

- Minimal Changes to Historical Implementations

When the original version of IEEE Std 1003.1 was published, there were no known historical implementations that did not have to change. However, there was a broad consensus on a set of functions, types, definitions, and concepts that formed an interface that was common to most historical implementations.

The adoption of the 1988 and 1990 IEEE system interface standards, the 1992 IEEE shell and utilities standard, the various Open Group (formerly X/Open) specifications, and the subsequent revisions and addenda to all of them have consolidated this consensus, and this revision reflects the significantly increased level of consensus arrived at since the original versions. The earlier standards and their modifications specified a number of areas where consensus had not been reached before, and these are now reflected in this revision. The authors of the original versions tried, as much as possible, to follow the principles below

when creating new specifications:

1. By standardizing an interface like one in an historical implementation; for example, directories
2. By specifying an interface that is readily implementable in terms of, and backwards-compatible with, historical implementations, such as the extended *tar* format defined in the *pax* utility
3. By specifying an interface that, when added to an historical implementation, will not conflict with it; for example, the *sigaction()* function

This revision tries to minimize the number of changes required to implementations which conform to the earlier versions of the approved standards to bring them into conformance with the current standard. Specifically, the scope of this work excluded doing any “new” work, but rather collecting into a single document what had been spread across a number of documents, and presenting it in what had been proven in practice to be a more effective way. Some changes to prior conforming implementations were unavoidable, primarily as a consequence of resolving conflicts found in prior revisions, or which became apparent when bringing the various pieces together.

However, since it references the 1999 version of the ISO C standard, and no longer supports “Common Usage C”, there are a number of unavoidable changes. Applications portability is similarly affected.

This standard is specifically not a codification of a particular vendor’s product.

It should be noted that implementations will have different kinds of extensions. Some will reflect “historical usage” and will be preserved for execution of pre-existing applications. These functions should be considered “obsolescent” and the standard functions used for new applications. Some extensions will represent functions beyond the scope of this standard. These need to be used with careful management to be able to adapt to future extensions of this standard and/or port to implementations that provide these services in a different manner.

- Minimal Changes to Existing Application Code

A goal of this standard was to minimize additional work for the developers of applications. However, because every known historical implementation will have to change at least slightly to conform, some applications will have to change.

This Standard

This standard defines the Portable Operating System Interface (POSIX) requirements and consists of the following volumes:

- Base Definitions
- Shell and Utilities
- System Interfaces (this volume)
- Rationale (Informative)

Introduction

This Volume

The System Interfaces volume describes the interfaces offered to application programs by POSIX-conformant systems. Readers are expected to be experienced C language programmers, and to be familiar with the Base Definitions volume.

This volume is structured as follows:

- Chapter 1 explains the status of this volume and its relationship to other formal standards.
- Chapter 2 contains important concepts, terms, and caveats relating to the rest of this volume.
- Chapter 3 defines the functional interfaces to the POSIX-conformant system.

Comprehensive references are available in the index.

Typographical Conventions

The following typographical conventions are used throughout this standard. In the text, this standard is referred to as IEEE Std 1003.1-2001, which is technically identical to The Open Group Base Specifications, Issue 6.

The typographical conventions listed here are for ease of reading only. Editorial inconsistencies in the use of typography are unintentional and have no normative meaning in this standard.

Reference	Example	Notes
C-Language Data Structure	aiocb	
C-Language Data Structure Member	<i>aio_lio_opcode</i>	
C-Language Data Type	long	
C-Language External Variable	<i>errno</i>	
C-Language Function	<i>system()</i>	
C-Language Function Argument	<i>arg1</i>	
C-Language Function Family	<i>exec</i>	
C-Language Header	<sys/stat.h>	
C-Language Keyword	return	
C-Language Macro with Argument	<i>assert()</i>	
C-Language Macro with No Argument	INET_ADDRSTRLEN	
C-Language Preprocessing Directive	#define	
Commands within a Utility	a, c	
Conversion Specification, Specifier / Modifier Character	<i>%A, g, E</i>	1
Environment Variable	<i>PATH</i>	
Error Number	[EINTR]	
Example Output	Hello, World	
Filename	/tmp	
Literal Character	<i>'c', '\r', '\'</i>	2
Literal String	<i>"abcde"</i>	2
Optional Items in Utility Syntax	[]	
Parameter	<i><directory pathname></i>	
Special Character	<i><newline></i>	3
Symbolic Constant	_POSIX_VDISABLE	
Symbolic Limit, Configuration Value	{LINE_MAX}	4
Syntax	<i>#include <sys/stat.h></i>	

Reference	Example	Notes
User Input and Example Code	echo Hello, World	5
Utility Name	awk	
Utility Operand	file_name	
Utility Option	-c	
Utility Option with Option-Argument	-w width	

Notes:

1. Conversion specifications, specifier characters, and modifier characters are used primarily in date-related functions and utilities and the *fprintf* and *fscanf* formatting functions.
2. Unless otherwise noted, the quotes shall not be used as input or output. When used in a list item, the quotes are omitted. For literal characters, ' \' (or any of the other sequences such as ' ' ') is the same as the C constant ' \\' (or ' \\').
3. The style selected for some of the special characters, such as <newline>, matches the form of the input given to the *localedef* utility. Generally, the characters selected for this special treatment are those that are not visually distinct, such as the control characters <tab> or <newline>.
4. Names surrounded by braces represent symbolic limits or configuration values which may be declared in appropriate headers by means of the C **#define** construct.
5. Brackets shown in this font, "[]", are part of the syntax and do *not* indicate optional items. In syntax the ' | ' symbol is used to separate alternatives, and ellipses (" . . . ") are used to show that additional arguments are optional.

Shading is used to identify extensions and options; see Section 1.8.1 (on page 3).

Footnotes and notes within the body of the normative text are for information only (informative).

Informative sections (such as Rationale, Change History, Application Usage, and so on) are denoted by continuous shading bars in the margins.

Ranges of values are indicated with parentheses or brackets as follows:

- (a,b) means the range of all values from *a* to *b*, including neither *a* nor *b*.
- [a,b] means the range of all values from *a* to *b*, including *a* and *b*.
- [a,b) means the range of all values from *a* to *b*, including *a*, but not *b*.
- (a,b] means the range of all values from *a* to *b*, including *b*, but not *a*.

Notes:

1. Symbolic limits are used in this volume instead of fixed values for portability. The values of most of these constants are defined in the Base Definitions volume, <limits.h> or <unistd.h>.
2. The values of errors are defined in the Base Definitions volume, <errno.h>.

Participants

The Austin Group

This standard was prepared by the Austin Group, sponsored by the Portable Applications Standards Committee of the IEEE Computer Society, The Open Group, and ISO/SC22 WG15. At the time of approval, the membership of the Austin Group was as follows.

Andrew Josey, Chair

Donald W. Cragun, Organizational Representative, IEEE PASC

Nicholas Stoughton, Organizational Representative, ISO/SC22 WG15

Mark Brown, Organizational Representative, The Open Group

Cathy Hughes, Technical Editor

Austin Group Technical Reviewers

Peter Anvin
Bouazza Bachar
Theodore P. Baker
Walter Briscoe
Mark Brown
Dave Butenhof
Geoff Clare
Donald W. Cragun
Lee Damico
Ulrich Drepper
Paul Eggert
Joanna Farley
Clive D.W. Feather
Andrew Gollan

Michael Gonzalez
Joseph M. Gwinn
Jon Hitchcock
Yvette Ho Sang
Cathy Hughes
Lowell G. Johnson
Andrew Josey
Michael Kavanaugh
David Korn
Marc Aurele La France
Jim Meyering
Gary Miller
Finnbarr P. Murphy
Joseph S. Myers

Sandra O'Donnell
Frank Prindle
Curtis Royster Jr.
Glen Seeds
Keld Jorn Simonsen
Raja Srinivasan
Nicholas Stoughton
Donn S. Terry
Fred Tydeman
Peter Van Der Veen
James Youngman
Jim Zepeda
Jason Zions

Austin Group Working Group Members

Harold C. Adams
Peter Anvin
Pierre-Jean Arcos
Jay Ashford
Bouazza Bachar
Theodore P. Baker
Robert Barned
Joel Berman
David J. Blackwood
Shirley Bockstahler-Brandt
James Bottomley
Walter Briscoe
Andries Brouwer
Mark Brown
Eric W. Burger
Alan Burns
Andries Brouwer
Dave Butenhof
Keith Chow
Geoff Clare
Donald W. Cragun
Lee Damico
Juan Antonio De La Puente
Ming De Zhou
Steven J. Dovich
Richard P. Draves
Ulrich Drepper
Paul Eggert
Philip H. Enslow
Joanna Farley
Clive D.W. Feather
Pete Forman
Mark Funkenhauser
Lois Goldthwaite
Andrew Gollan

Michael Gonzalez
Karen D. Gordon
Joseph M. Gwinn
Steven A. Haaser
Charles E. Hammons
Chris J. Harding
Barry Hedquist
Vincent E. Henley
Karl Heubaum
Jon Hitchcock
Yvette Ho Sang
Niklas Holsti
Thomas Hosmer
Cathy Hughes
Jim D. Isaak
Lowell G. Johnson
Michael B. Jones
Andrew Josey
Michael J. Karels
Michael Kavanaugh
David Korn
Steven Kramer
Thomas M. Kurihara
Marc Aurele La France
C. Douglass Locke
Nick Maclaren
Roger J. Martin
Craig H. Meyer
Jim Meyering
Gary Miller
Finnbarr P. Murphy
Joseph S. Myers
John Napier
Peter E. Obermayer
James T. Oblinger

Sandra O'Donnell
Frank Prindle
Francois Riche
John D. Riley
Andrew K. Roach
Helmut Roth
Jaideep Roy
Curtis Royster Jr.
Stephen C. Schwarm
Glen Seeds
Richard Seibel
David L. Shroads Jr.
W. Olin Sibert
Keld Jorn Simonsen
Curtis Smith
Raja Srinivasan
Nicholas Stoughton
Marc J. Teller
Donn S. Terry
Fred Tydeman
Mark-Rene Uchida
Scott A. Valcourt
Peter Van Der Veen
Michael W. Vannier
Eric Vought
Frederick N. Webb
Paul A.T. Wolfgang
Garrett Wollman
James Youngman
Oren Yuen
Janusz Zalewski
Jim Zepeda
Jason Zions

*Participants***The Open Group**

When The Open Group approved the Base Specifications, Issue 6 on 12 September 2001, the membership of The Open Group Base Working Group was as follows.

Andrew Josey, Chair

Finnbarr P. Murphy, Vice-Chair

Mark Brown, Austin Group Liaison

Cathy Hughes, Technical Editor

Base Working Group Members

Bouazza Bachar

Mark Brown

Dave Butenhof

Donald W. Cragun

Larry Dwyer

Joanna Farley

Andrew Gollan

Karen D. Gordon

Gary Miller

Finnbarr P. Murphy

Frank Prindle

Andrew K. Roach

Curtis Royster Jr.

Nicholas Stoughton

Kenjiro Tsuji

IEEE

When the IEEE Standards Board approved IEEE Std 1003.1-2001 on 6 December 2001, the membership of the committees was as follows.

Portable Applications Standards Committee (PASC)

Lowell G. Johnson, Chair
Joseph M. Gwinn, Vice-Chair
Jay Ashford, Functional Chair
Andrew Josey, Functional Chair
Curtis Royster Jr., Functional Chair
Nicholas Stoughton, Secretary

Balloting Committee

The following members of the balloting committee voted on IEEE Std 1003.1-2001. Balloters may have voted for approval, disapproval, or abstention.

Harold C. Adams	Steven A. Haaser	Frank Prindle
Pierre-Jean Arcos	Charles E. Hammons	Francois Riche
Jay Ashford	Chris J. Harding	John D. Riley
Theodore P. Baker	Barry Hedquist	Andrew K. Roach
Robert Barned	Vincent E. Henley	Helmut Roth
David J. Blackwood	Karl Heubaum	Jaideep Roy
Shirley Bockstahler-Brandt	Niklas Holsti	Curtis Royster Jr.
James Bottomley	Thomas Hosmer	Stephen C. Schwarm
Mark Brown	Jim D. Isaak	Richard Seibel
Eric W. Burger	Lowell G. Johnson	David L. Shroads Jr.
Alan Burns	Michael B. Jones	W. Olin Sibert
Dave Butenhof	Andrew Josey	Keld Jorn Simonsen
Keith Chow	Michael J. Karels	Nicholas Stoughton
Donald W. Cragun	Steven Kramer	Donn S. Terry
Juan Antonio De La Puente	Thomas M. Kurihara	Mark-Rene Uchida
Ming De Zhou	C. Douglass Locke	Scott A. Valcourt
Steven J. Dovich	Roger J. Martin	Michael W. Vannier
Richard P. Draves	Craig H. Meyer	Frederick N. Webb
Philip H. Enslow	Finnbarr P. Murphy	Paul A.T. Wolfgang
Michael Gonzalez	John Napier	Oren Yuen
Karen D. Gordon	Peter E. Obermayer	Janusz Zalewski
Joseph M. Gwinn	James T. Oblinger	

The following organizational representative voted on this standard:

Andrew Josey, X/Open Company Ltd.

*Participants***IEEE-SA Standards Board**

When the IEEE-SA Standards Board approved IEEE Std 1003.1-2001 on 6 December 2001, it had the following membership:

Donald N. Heirman, Chair

James T. Carlo, Vice-Chair

Judith Gorman, Secretary

Satish K. Aggarwal

Mark D. Bowman

Gary R. Engmann

Harold E. Epstein

H. Landis Floyd

Jay Forster*

Howard M. Frazier

Ruben D. Garzon

James H. Gurney

Richard J. Holleman

Lowell G. Johnson

Robert J. Kennelly

Joseph L. Koepfinger*

Peter H. Lips

L. Bruce McClung

Daleep C. Mohla

James W. Moore

Robert F. Munzner

Ronald C. Petersen

Gerald H. Peterson

John B. Posey

Gary S. Robinson

Akio Tojo

Donald W. Zipse

Also included are the following non-voting IEEE-SA Standards Board liaisons:

Alan Cookson, NIST Representative

Donald R. Volzka, TAB Representative

Yvette Ho Sang, Don Messina, Savoula Amanatidis, IEEE Project Editors

* Member Emeritus

Trademarks

The following information is given for the convenience of users of this standard and does not constitute endorsement of these products by The Open Group or the IEEE. There may be other products mentioned in the text that might be covered by trademark protection and readers are advised to verify them independently.

1003.1TM is a trademark of the Institute of Electrical and Electronic Engineers, Inc.

AIX[®] is a registered trademark of IBM Corporation.

AT&T[®] is a registered trademark of AT&T in the U.S.A. and other countries.

BSDTM is a trademark of the University of California, Berkeley, U.S.A.

Hewlett-Packard[®], HP[®], and HP-UX[®] are registered trademarks of Hewlett-Packard Company.

IBM[®] is a registered trademark of International Business Machines Corporation.

Motif[®], OSF/1[®], UNIX[®], and the “X Device” are registered trademarks and IT DialToneTM and The Open GroupTM are trademarks of The Open Group in the U.S. and other countries.

POSIX[®] is a registered trademark of the Institute of Electrical and Electronic Engineers, Inc.

Sun[®] and Sun Microsystems[®] are registered trademarks of Sun Microsystems, Inc.

/usr/group[®] is a registered trademark of UniForum, the International Network of UNIX System Users.

Acknowledgements

The contributions of the following organizations to the development of IEEE Std 1003.1-2001 are gratefully acknowledged:

- AT&T for permission to reproduce portions of its copyrighted System V Interface Definition (SVID) and material from the UNIX System V Release 2.0 documentation.
- The SC22 WG14 Committees.

This standard was prepared by the Austin Group, a joint working group of the IEEE, The Open Group, and ISO SC22 WG15.

Referenced Documents

Normative References

Normative references for this standard are defined in the Base Definitions volume.

Informative References

The following documents are referenced in this standard:

1984 /usr/group Standard

/usr/group Standards Committee, Santa Clara, CA, UniForum 1984.

Almasi and Gottlieb

George S. Almasi and Allan Gottlieb, *Highly Parallel Computing*, The Benjamin/Cummings Publishing Company, Inc., 1989, ISBN: 0-8053-0177-1.

ANSI C

American National Standard for Information Systems: Standard X3.159-1989, Programming Language C.

ANSI X3.226-1994

American National Standard for Information Systems: Standard X3.226-1994, Programming Language Common LISP.

Brawer

Steven Brawer, *Introduction to Parallel Programming*, Academic Press, 1989, ISBN: 0-12-128470-0.

DeRemer and Pennello Article

DeRemer, Frank and Pennello, Thomas J., *Efficient Computation of LALR(1) Look-Ahead Sets*, SigPlan Notices, Volume 15, No. 8, August 1979.

Draft ANSI X3J11.1

IEEE Floating Point draft report of ANSI X3J11.1 (NCEG).

FIPS 151-1

Federal Information Procurement Standard (FIPS) 151-1. Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API) [C Language].

FIPS 151-2

Federal Information Procurement Standards (FIPS) 151-2, Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API) [C Language].

HP-UX Manual

Hewlett-Packard HP-UX Release 9.0 Reference Manual, Third Edition, August 1992.

IEC 60559: 1989

IEC 60559: 1989, Binary Floating-Point Arithmetic for Microprocessor Systems (previously designated IEC 559: 1989).

IEEE Std 754-1985

IEEE Std 754-1985, IEEE Standard for Binary Floating-Point Arithmetic.

IEEE Std 854-1987

IEEE Std 854-1987, IEEE Standard for Radix-Independent Floating-Point Arithmetic.

Referenced Documents

- IEEE Std 1003.9-1992
IEEE Std 1003.9-1992, IEEE Standard for Information Technology — POSIX FORTRAN 77 Language Interfaces — Part 1: Binding for System Application Program Interface API.
- IETF RFC 791
Internet Protocol, Version 4 (IPv4), September 1981.
- IETF RFC 819
The Domain Naming Convention for Internet User Applications, Z. Su, J. Postel, August 1982.
- IETF RFC 822
Standard for the Format of ARPA Internet Text Messages, D.H. Crocker, August 1982.
- IETF RFC 919
Broadcasting Internet Datagrams, J. Mogul, October 1984.
- IETF RFC 920
Domain Requirements, J. Postel, J. Reynolds, October 1984.
- IETF RFC 921
Domain Name System Implementation Schedule, J. Postel, October 1984.
- IETF RFC 922
Broadcasting Internet Datagrams in the Presence of Subnets, J. Mogul, October 1984.
- IETF RFC 1034
Domain Names — Concepts and Facilities, P. Mockapetris, November 1987.
- IETF RFC 1035
Domain Names — Implementation and Specification, P. Mockapetris, November 1987.
- IETF RFC 1123
Requirements for Internet Hosts — Application and Support, R. Braden, October 1989.
- IETF RFC 1886
DNS Extensions to Support Internet Protocol, Version 6 (IPv6), C. Huitema, S. Thomson, December 1995.
- IETF RFC 2045
Multipurpose Internet Mail Extensions (MIME), Part 1: Format of Internet Message Bodies, N. Freed, N. Borenstein, November 1996.
- IETF RFC 2373
Internet Protocol, Version 6 (IPv6) Addressing Architecture, S. Deering, R. Hinden, July 1998.
- IETF RFC 2460
Internet Protocol, Version 6 (IPv6), S. Deering, R. Hinden, December 1998.
- Internationalisation Guide
Guide, July 1993, Internationalisation Guide, Version 2 (ISBN: 1-859120-02-4, G304), published by The Open Group.
- ISO C (1990)
ISO/IEC 9899:1990, Programming Languages — C, including Amendment 1:1995 (E), C Integrity (Multibyte Support Extensions (MSE) for ISO C).
- ISO 2375:1985
ISO 2375:1985, Data Processing — Procedure for Registration of Escape Sequences.

- ISO 8652: 1987
ISO 8652: 1987, Programming Languages — Ada (technically identical to ANSI standard 1815A-1983).
- ISO / IEC 1539: 1990
ISO / IEC 1539: 1990, Information Technology — Programming Languages — Fortran (technically identical to the ANSI X3.9-1978 standard [FORTRAN 77]).
- ISO / IEC 4873: 1991
ISO / IEC 4873: 1991, Information Technology — ISO 8-bit Code for Information Interchange — Structure and Rules for Implementation.
- ISO / IEC 6429: 1992
ISO / IEC 6429: 1992, Information Technology — Control Functions for Coded Character Sets.
- ISO / IEC 6937: 1994
ISO / IEC 6937: 1994, Information Technology — Coded Character Set for Text Communication — Latin Alphabet.
- ISO / IEC 8802-3: 1996
ISO / IEC 8802-3: 1996, Information Technology — Telecommunications and Information Exchange Between Systems — Local and Metropolitan Area Networks — Specific Requirements — Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA / CD) Access Method and Physical Layer Specifications.
- ISO / IEC 8859
ISO / IEC 8859, Information Technology — 8-Bit Single-Byte Coded Graphic Character Sets:
- Part 1: Latin Alphabet No. 1
 - Part 2: Latin Alphabet No. 2
 - Part 3: Latin Alphabet No. 3
 - Part 4: Latin Alphabet No. 4
 - Part 5: Latin / Cyrillic Alphabet
 - Part 6: Latin / Arabic Alphabet
 - Part 7: Latin / Greek Alphabet
 - Part 8: Latin / Hebrew Alphabet
 - Part 9: Latin Alphabet No. 5
 - Part 10: Latin Alphabet No. 6
 - Part 13: Latin Alphabet No. 7
 - Part 14: Latin Alphabet No. 8
 - Part 15: Latin Alphabet No. 9
- ISO POSIX-1: 1996
ISO / IEC 9945-1: 1996, Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language] (identical to ANSI / IEEE Std 1003.1-1996). Incorporating ANSI / IEEE Stds 1003.1-1990, 1003.1b-1993, 1003.1c-1995, and 1003.1i-1995.
- ISO POSIX-2: 1993
ISO / IEC 9945-2: 1993, Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities (identical to ANSI / IEEE Std 1003.2-1992, as amended by ANSI / IEEE Std 1003.2a-1992).
- Issue 1
X / Open Portability Guide, July 1985 (ISBN: 0-444-87839-4).

Referenced Documents

Issue 2

X/Open Portability Guide, January 1987:

- Volume 1: XVS Commands and Utilities (ISBN: 0-444-70174-5)
- Volume 2: XVS System Calls and Libraries (ISBN: 0-444-70175-3)

Issue 3

X/Open Specification, 1988, 1989, February 1992:

- Commands and Utilities, Issue 3 (ISBN: 1-872630-36-7, C211); this specification was formerly X/Open Portability Guide, Issue 3, Volume 1, January 1989, XSI Commands and Utilities (ISBN: 0-13-685835-X, XO/XPG/89/002)
- System Interfaces and Headers, Issue 3 (ISBN: 1-872630-37-5, C212); this specification was formerly X/Open Portability Guide, Issue 3, Volume 2, January 1989, XSI System Interface and Headers (ISBN: 0-13-685843-0, XO/XPG/89/003)
- Curses Interface, Issue 3, contained in Supplementary Definitions, Issue 3 (ISBN: 1-872630-38-3, C213), Chapters 9 to 14 inclusive; this specification was formerly X/Open Portability Guide, Issue 3, Volume 3, January 1989, XSI Supplementary Definitions (ISBN: 0-13-685850-3, XO/XPG/89/004)
- Headers Interface, Issue 3, contained in Supplementary Definitions, Issue 3 (ISBN: 1-872630-38-3, C213), Chapter 19, Cpio and Tar Headers; this specification was formerly X/Open Portability Guide Issue 3, Volume 3, January 1989, XSI Supplementary Definitions (ISBN: 0-13-685850-3, XO/XPG/89/004)

Issue 4

CAE Specification, July 1992, published by The Open Group:

- System Interface Definitions (XBD), Issue 4 (ISBN: 1-872630-46-4, C204)
- Commands and Utilities (XCU), Issue 4 (ISBN: 1-872630-48-0, C203)
- System Interfaces and Headers (XSH), Issue 4 (ISBN: 1-872630-47-2, C202)

Issue 4, Version 2

CAE Specification, August 1994, published by The Open Group:

- System Interface Definitions (XBD), Issue 4, Version 2 (ISBN: 1-85912-036-9, C434)
- Commands and Utilities (XCU), Issue 4, Version 2 (ISBN: 1-85912-034-2, C436)
- System Interfaces and Headers (XSH), Issue 4, Version 2 (ISBN: 1-85912-037-7, C435)

Issue 5

Technical Standard, February 1997, published by The Open Group:

- System Interface Definitions (XBD), Issue 5 (ISBN: 1-85912-186-1, C605)
- Commands and Utilities (XCU), Issue 5 (ISBN: 1-85912-191-8, C604)
- System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0, C606)

Knuth Article

Knuth, Donald E., *On the Translation of Languages from Left to Right*, Information and Control, Volume 8, No. 6, October 1965.

KornShell

Bolsky, Morris I. and Korn, David G., *The New KornShell Command and Programming Language*, March 1995, Prentice Hall.

MSE Working Draft

Working draft of ISO/IEC 9899:1990/Add3: Draft, Addendum 3 — Multibyte Support Extensions (MSE) as documented in the ISO Working Paper SC22/WG14/N205 dated 31 March 1992.

POSIX.0: 1995

IEEE Std 1003.0-1995, IEEE Guide to the POSIX Open System Environment (OSE) (identical to ISO/IEC TR 14252).

POSIX.1: 1988

IEEE Std 1003.1-1988, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language].

POSIX.1: 1990

IEEE Std 1003.1-1990, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language].

POSIX.1a

P1003.1a, Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — (C Language) Amendment

POSIX.1d: 1999

IEEE Std 1003.1d-1999, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — Amendment 4: Additional Realtime Extensions [C Language].

POSIX.1g: 2000

IEEE Std 1003.1g-2000, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — Amendment 6: Protocol-Independent Interfaces (PII).

POSIX.1j: 2000

IEEE Std 1003.1j-2000, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — Amendment 5: Advanced Realtime Extensions [C Language].

POSIX.1q: 2000

IEEE Std 1003.1q-2000, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — Amendment 7: Tracing [C Language].

POSIX.2b

P1003.2b, Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities — Amendment

POSIX.2d:-1994

IEEE Std 1003.2d: 1994, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities — Amendment 1: Batch Environment.

POSIX.13:-1998

IEEE Std 1003.13: 1998, IEEE Standard for Information Technology — Standardized Application Environment Profile (AEP) — POSIX Realtime Application Support.

Referenced Documents

Sarwate Article

Sarwate, Dilip V., *Computation of Cyclic Redundancy Checks via Table Lookup*, Communications of the ACM, Volume 30, No. 8, August 1988.

Sprunt, Sha, and Lehoczky

Sprunt, B., Sha, L., and Lehoczky, J.P., *Aperiodic Task Scheduling for Hard Real-Time Systems*, The Journal of Real-Time Systems, Volume 1, 1989, Pages 27-60.

SVID, Issue 1

American Telephone and Telegraph Company, System V Interface Definition (SVID), Issue 1; Morristown, NJ, UNIX Press, 1985.

SVID, Issue 2

American Telephone and Telegraph Company, System V Interface Definition (SVID), Issue 2; Morristown, NJ, UNIX Press, 1986.

SVID, Issue 3

American Telephone and Telegraph Company, System V Interface Definition (SVID), Issue 3; Morristown, NJ, UNIX Press, 1989.

The AWK Programming Language

Aho, Alfred V., Kernighan, Brian W., and Weinberger, Peter J., *The AWK Programming Language*, Reading, MA, Addison-Wesley 1988.

UNIX Programmer's Manual

American Telephone and Telegraph Company, *UNIX Time-Sharing System: UNIX Programmer's Manual*, 7th Edition, Murray Hill, NJ, Bell Telephone Laboratories, January 1979.

XNS, Issue 4

CAE Specification, August 1994, Networking Services, Issue 4 (ISBN: 1-85912-049-0, C438), published by The Open Group.

XNS, Issue 5

CAE Specification, February 1997, Networking Services, Issue 5 (ISBN: 1-85912-165-9, C523), published by The Open Group.

XNS, Issue 5.2

Technical Standard, January 2000, Networking Services (XNS), Issue 5.2 (ISBN: 1-85912-241-8, C808), published by The Open Group.

X/Open Curses, Issue 4, Version 2

CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2 (ISBN: 1-85912-171-3, C610), published by The Open Group.

Yacc

Yacc: Yet Another Compiler Compiler, Stephen C. Johnson, 1978.

Source Documents

Parts of the following documents were used to create the base documents for this standard:

AIX 3.2 Manual

AIX Version 3.2 For RISC System/6000, Technical Reference: Base Operating System and Extensions, 1990, 1992 (Part No. SC23-2382-00).

OSF/1

OSF/1 Programmer's Reference, Release 1.2 (ISBN: 0-13-020579-6).

OSF AES

Application Environment Specification (AES) Operating System Programming Interfaces
Volume, Revision A (ISBN: 0-13-043522-8).

System V Release 2.0

- UNIX System V Release 2.0 Programmer's Reference Manual (April 1984 - Issue 2).
- UNIX System V Release 2.0 Programming Guide (April 1984 - Issue 2).

System V Release 4.2

Operating System API Reference, UNIX SVR4.2 (1992) (ISBN: 0-13-017658-3).

Introduction

1.1 Scope

The scope of IEEE Std 1003.1-2001 is described in the Base Definitions volume of IEEE Std 1003.1-2001.

1.2 Conformance

Conformance requirements for IEEE Std 1003.1-2001 are defined in the Base Definitions volume of IEEE Std 1003.1-2001, Chapter 2, Conformance.

1.3 Normative References

Normative references for IEEE Std 1003.1-2001 are defined in the Base Definitions volume of IEEE Std 1003.1-2001.

1.4 Change History

Change history is described in the Rationale (Informative) volume of IEEE Std 1003.1-2001, and in the CHANGE HISTORY section of reference pages.

1.5 Terminology

This section appears in the Base Definitions volume of IEEE Std 1003.1-2001, but is repeated here for convenience:

For the purposes of IEEE Std 1003.1-2001, the following terminology definitions apply:

can

Describes a permissible optional feature or behavior available to the user or application. The feature or behavior is mandatory for an implementation that conforms to IEEE Std 1003.1-2001. An application can rely on the existence of the feature or behavior.

implementation-defined

Describes a value or behavior that is not defined by IEEE Std 1003.1-2001 but is selected by an implementor. The value or behavior may vary among implementations that conform to IEEE Std 1003.1-2001. An application should not rely on the existence of the value or behavior. An application that relies on such a value or behavior cannot be assured to be portable across conforming implementations.

The implementor shall document such a value or behavior so that it can be used correctly by an application.

legacy

Describes a feature or behavior that is being retained for compatibility with older applications, but which has limitations which make it inappropriate for developing portable

applications. New applications should use alternative means of obtaining equivalent functionality.

may

Describes a feature or behavior that is optional for an implementation that conforms to IEEE Std 1003.1-2001. An application should not rely on the existence of the feature or behavior. An application that relies on such a feature or behavior cannot be assured to be portable across conforming implementations.

To avoid ambiguity, the opposite of *may* is expressed as *need not*, instead of *may not*.

shall

For an implementation that conforms to IEEE Std 1003.1-2001, describes a feature or behavior that is mandatory. An application can rely on the existence of the feature or behavior.

For an application or user, describes a behavior that is mandatory.

should

For an implementation that conforms to IEEE Std 1003.1-2001, describes a feature or behavior that is recommended but not mandatory. An application should not rely on the existence of the feature or behavior. An application that relies on such a feature or behavior cannot be assured to be portable across conforming implementations.

For an application, describes a feature or behavior that is recommended programming practice for optimum portability.

undefined

Describes the nature of a value or behavior not defined by IEEE Std 1003.1-2001 which results from use of an invalid program construct or invalid data input.

The value or behavior may vary among implementations that conform to IEEE Std 1003.1-2001. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

unspecified

Describes the nature of a value or behavior not specified by IEEE Std 1003.1-2001 which results from use of a valid program construct or valid data input.

The value or behavior may vary among implementations that conform to IEEE Std 1003.1-2001. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

1.6 Definitions

Concepts and definitions are defined in the Base Definitions volume of IEEE Std 1003.1-2001.

1.7 Relationship to Other Formal Standards

Great care has been taken to ensure that this volume of IEEE Std 1003.1-2001 is fully aligned with the following standards:

ISO C (1999)

ISO / IEC 9899: 1999, Programming Languages — C.

Parts of the ISO / IEC 9899: 1999 standard (hereinafter referred to as the ISO C standard) are referenced to describe requirements also mandated by this volume of IEEE Std 1003.1-2001. Some functions and headers included within this volume of IEEE Std 1003.1-2001 have a version in the ISO C standard; in this case CX markings are added as appropriate to show where the ISO C standard has been extended (see Section 1.8.1). Any conflict between this volume of IEEE Std 1003.1-2001 and the ISO C standard is unintentional.

This volume of IEEE Std 1003.1-2001 also allows, but does not require, mathematics functions to support IEEE Std 754-1985 and IEEE Std 854-1987.

1.8 Portability

Some of the utilities in the Shell and Utilities volume of IEEE Std 1003.1-2001 and functions in the System Interfaces volume of IEEE Std 1003.1-2001 describe functionality that might not be fully portable to systems meeting the requirements for POSIX conformance (see the Base Definitions volume of IEEE Std 1003.1-2001, Chapter 2, Conformance).

Where optional, enhanced, or reduced functionality is specified, the text is shaded and a code in the margin identifies the nature of the option, extension, or warning (see Section 1.8.1). For maximum portability, an application should avoid such functionality.

1.8.1 Codes

Margin codes and their meanings are listed in the Base Definitions volume of IEEE Std 1003.1-2001, but are repeated here for convenience:

ADV Advisory Information

The functionality described is optional. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the ADV margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the ADV margin legend.

AIO Asynchronous Input and Output

The functionality described is optional. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the AIO margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the AIO margin legend.

BAR Barriers

The functionality described is optional. The functionality described is also an extension to the

107		ISO C standard.
108		Where applicable, functions are marked with the BAR margin legend in the SYNOPSIS section.
109		Where additional semantics apply to a function, the material is identified by use of the BAR
110		margin legend.
111	BE	Batch Environment Services and Utilities
112		The functionality described is optional.
113		Where applicable, utilities are marked with the BE margin legend in the SYNOPSIS section.
114		Where additional semantics apply to a utility, the material is identified by use of the BE margin
115		legend.
116	CD	C-Language Development Utilities
117		The functionality described is optional.
118		Where applicable, utilities are marked with the CD margin legend in the SYNOPSIS section.
119		Where additional semantics apply to a utility, the material is identified by use of the CD margin
120		legend.
121	CPT	Process CPU-Time Clocks
122		The functionality described is optional. The functionality described is also an extension to the
123		ISO C standard.
124		Where applicable, functions are marked with the CPT margin legend in the SYNOPSIS section.
125		Where additional semantics apply to a function, the material is identified by use of the CPT
126		margin legend.
127	CS	Clock Selection
128		The functionality described is optional. The functionality described is also an extension to the
129		ISO C standard.
130		Where applicable, functions are marked with the CS margin legend in the SYNOPSIS section.
131		Where additional semantics apply to a function, the material is identified by use of the CS
132		margin legend.
133	CX	Extension to the ISO C standard
134		The functionality described is an extension to the ISO C standard. Application writers may make
135		use of an extension as it is supported on all IEEE Std 1003.1-2001-conforming systems.
136		With each function or header from the ISO C standard, a statement to the effect that “any
137		conflict is unintentional” is included. That is intended to refer to a direct conflict.
138		IEEE Std 1003.1-2001 acts in part as a profile of the ISO C standard, and it may choose to further
139		constrain behaviors allowed to vary by the ISO C standard. Such limitations are not considered
140		conflicts.
141		Where additional semantics apply to a function or header, the material is identified by use of the
142		CX margin legend.
143	FD	FORTTRAN Development Utilities
144		The functionality described is optional.
145		Where applicable, utilities are marked with the FD margin legend in the SYNOPSIS section.
146		Where additional semantics apply to a utility, the material is identified by use of the FD margin
147		legend.
148	FR	FORTTRAN Runtime Utilities
149		The functionality described is optional.

150		Where applicable, utilities are marked with the FR margin legend in the SYNOPSIS section.
151		Where additional semantics apply to a utility, the material is identified by use of the FR margin
152		legend.
153	FSC	File Synchronization
154		The functionality described is optional. The functionality described is also an extension to the
155		ISO C standard.
156		Where applicable, functions are marked with the FSC margin legend in the SYNOPSIS section.
157		Where additional semantics apply to a function, the material is identified by use of the FSC
158		margin legend.
159	IP6	IPV6
160		The functionality described is optional. The functionality described is also an extension to the
161		ISO C standard.
162		Where applicable, functions are marked with the IP6 margin legend in the SYNOPSIS section.
163		Where additional semantics apply to a function, the material is identified by use of the IP6
164		margin legend.
165	MC1	Advisory Information and either Memory Mapped Files or Shared Memory Objects
166		The functionality described is optional. The functionality described is also an extension to the
167		ISO C standard.
168		This is a shorthand notation for combinations of multiple option codes.
169		Where applicable, functions are marked with the MC1 margin legend in the SYNOPSIS section.
170		Where additional semantics apply to a function, the material is identified by use of the MC1
171		margin legend.
172		Refer to the Base Definitions volume of IEEE Std 1003.1-2001, Section 1.5.2, Margin Code
173		Notation.
174	MC2	Memory Mapped Files, Shared Memory Objects, or Memory Protection
175		The functionality described is optional. The functionality described is also an extension to the
176		ISO C standard.
177		This is a shorthand notation for combinations of multiple option codes.
178		Where applicable, functions are marked with the MC2 margin legend in the SYNOPSIS section.
179		Where additional semantics apply to a function, the material is identified by use of the MC2
180		margin legend.
181		Refer to the Base Definitions volume of IEEE Std 1003.1-2001, Section 1.5.2, Margin Code
182		Notation.
183	MF	Memory Mapped Files
184		The functionality described is optional. The functionality described is also an extension to the
185		ISO C standard.
186		Where applicable, functions are marked with the MF margin legend in the SYNOPSIS section.
187		Where additional semantics apply to a function, the material is identified by use of the MF
188		margin legend.
189	ML	Process Memory Locking
190		The functionality described is optional. The functionality described is also an extension to the
191		ISO C standard.
192		Where applicable, functions are marked with the ML margin legend in the SYNOPSIS section.
193		Where additional semantics apply to a function, the material is identified by use of the ML

194		margin legend.
195	MLR	Range Memory Locking
196		The functionality described is optional. The functionality described is also an extension to the
197		ISO C standard.
198		Where applicable, functions are marked with the MLR margin legend in the SYNOPSIS section.
199		Where additional semantics apply to a function, the material is identified by use of the MLR
200		margin legend.
201	MON	Monotonic Clock
202		The functionality described is optional. The functionality described is also an extension to the
203		ISO C standard.
204		Where applicable, functions are marked with the MON margin legend in the SYNOPSIS section.
205		Where additional semantics apply to a function, the material is identified by use of the MON
206		margin legend.
207	MPR	Memory Protection
208		The functionality described is optional. The functionality described is also an extension to the
209		ISO C standard.
210		Where applicable, functions are marked with the MPR margin legend in the SYNOPSIS section.
211		Where additional semantics apply to a function, the material is identified by use of the MPR
212		margin legend.
213	MSG	Message Passing
214		The functionality described is optional. The functionality described is also an extension to the
215		ISO C standard.
216		Where applicable, functions are marked with the MSG margin legend in the SYNOPSIS section.
217		Where additional semantics apply to a function, the material is identified by use of the MSG
218		margin legend.
219	MX	IEC 60559 Floating-Point Option
220		The functionality described is optional. The functionality described is also an extension to the
221		ISO C standard.
222		Where applicable, functions are marked with the MX margin legend in the SYNOPSIS section.
223		Where additional semantics apply to a function, the material is identified by use of the MX
224		margin legend.
225	OB	Obsolescent
226		The functionality described may be withdrawn in a future version of this volume of
227		IEEE Std 1003.1-2001. Strictly Conforming POSIX Applications and Strictly Conforming XSI
228		Applications shall not use obsolescent features.
229		Where applicable, the material is identified by use of the OB margin legend.
230	OF	Output Format Incompletely Specified
231		The functionality described is an XSI extension. The format of the output produced by the utility
232		is not fully specified. It is therefore not possible to post-process this output in a consistent
233		fashion. Typical problems include unknown length of strings and unspecified field delimiters.
234		Where applicable, the material is identified by use of the OF margin legend.
235	OH	Optional Header
236		In the SYNOPSIS section of some interfaces in the System Interfaces volume of
237		IEEE Std 1003.1-2001 an included header is marked as in the following example:

238	OH	<code>#include <sys/types.h></code>
239		<code>#include <grp.h></code>
240		<code>struct group *getgrnam(const char *name);</code>
241		The OH margin legend indicates that the marked header is not required on XSI-conformant
242		systems.
243	PIO	Prioritized Input and Output
244		The functionality described is optional. The functionality described is also an extension to the
245		ISO C standard.
246		Where applicable, functions are marked with the PIO margin legend in the SYNOPSIS section.
247		Where additional semantics apply to a function, the material is identified by use of the PIO
248		margin legend.
249	PS	Process Scheduling
250		The functionality described is optional. The functionality described is also an extension to the
251		ISO C standard.
252		Where applicable, functions are marked with the PS margin legend in the SYNOPSIS section.
253		Where additional semantics apply to a function, the material is identified by use of the PS
254		margin legend.
255	RS	Raw Sockets
256		The functionality described is optional. The functionality described is also an extension to the
257		ISO C standard.
258		Where applicable, functions are marked with the RS margin legend in the SYNOPSIS section.
259		Where additional semantics apply to a function, the material is identified by use of the RS
260		margin legend.
261	RTS	Realtime Signals Extension
262		The functionality described is optional. The functionality described is also an extension to the
263		ISO C standard.
264		Where applicable, functions are marked with the RTS margin legend in the SYNOPSIS section.
265		Where additional semantics apply to a function, the material is identified by use of the RTS
266		margin legend.
267	SD	Software Development Utilities
268		The functionality described is optional.
269		Where applicable, utilities are marked with the SD margin legend in the SYNOPSIS section.
270		Where additional semantics apply to a utility, the material is identified by use of the SD margin
271		legend.
272	SEM	Semaphores
273		The functionality described is optional. The functionality described is also an extension to the
274		ISO C standard.
275		Where applicable, functions are marked with the SEM margin legend in the SYNOPSIS section.
276		Where additional semantics apply to a function, the material is identified by use of the SEM
277		margin legend.
278	SHM	Shared Memory Objects
279		The functionality described is optional. The functionality described is also an extension to the
280		ISO C standard.
281		Where applicable, functions are marked with the SHM margin legend in the SYNOPSIS section.
282		Where additional semantics apply to a function, the material is identified by use of the SHM

283		margin legend.
284	SIO	Synchronized Input and Output
285		The functionality described is optional. The functionality described is also an extension to the
286		ISO C standard.
287		Where applicable, functions are marked with the SIO margin legend in the SYNOPSIS section.
288		Where additional semantics apply to a function, the material is identified by use of the SIO
289		margin legend.
290	SPI	Spin Locks
291		The functionality described is optional. The functionality described is also an extension to the
292		ISO C standard.
293		Where applicable, functions are marked with the SPI margin legend in the SYNOPSIS section.
294		Where additional semantics apply to a function, the material is identified by use of the SPI
295		margin legend.
296	SPN	Spawn
297		The functionality described is optional. The functionality described is also an extension to the
298		ISO C standard.
299		Where applicable, functions are marked with the SPN margin legend in the SYNOPSIS section.
300		Where additional semantics apply to a function, the material is identified by use of the SPN
301		margin legend.
302	SS	Process Sporadic Server
303		The functionality described is optional. The functionality described is also an extension to the
304		ISO C standard.
305		Where applicable, functions are marked with the SS margin legend in the SYNOPSIS section.
306		Where additional semantics apply to a function, the material is identified by use of the SS
307		margin legend.
308	TCT	Thread CPU-Time Clocks
309		The functionality described is optional. The functionality described is also an extension to the
310		ISO C standard.
311		Where applicable, functions are marked with the TCT margin legend in the SYNOPSIS section.
312		Where additional semantics apply to a function, the material is identified by use of the TCT
313		margin legend.
314	TEF	Trace Event Filter
315		The functionality described is optional. The functionality described is also an extension to the
316		ISO C standard.
317		Where applicable, functions are marked with the TEF margin legend in the SYNOPSIS section.
318		Where additional semantics apply to a function, the material is identified by use of the TEF
319		margin legend.
320	THR	Threads
321		The functionality described is optional. The functionality described is also an extension to the
322		ISO C standard.
323		Where applicable, functions are marked with the THR margin legend in the SYNOPSIS section.
324		Where additional semantics apply to a function, the material is identified by use of the THR
325		margin legend.
326	TMO	Timeouts
327		The functionality described is optional. The functionality described is also an extension to the

328		ISO C standard.
329		Where applicable, functions are marked with the TMO margin legend in the SYNOPSIS section.
330		Where additional semantics apply to a function, the material is identified by use of the TMO
331		margin legend.
332	TMR	Timers
333		The functionality described is optional. The functionality described is also an extension to the
334		ISO C standard.
335		Where applicable, functions are marked with the TMR margin legend in the SYNOPSIS section.
336		Where additional semantics apply to a function, the material is identified by use of the TMR
337		margin legend.
338	TPI	Thread Priority Inheritance
339		The functionality described is optional. The functionality described is also an extension to the
340		ISO C standard.
341		Where applicable, functions are marked with the TPI margin legend in the SYNOPSIS section.
342		Where additional semantics apply to a function, the material is identified by use of the TPI
343		margin legend.
344	TPP	Thread Priority Protection
345		The functionality described is optional. The functionality described is also an extension to the
346		ISO C standard.
347		Where applicable, functions are marked with the TPP margin legend in the SYNOPSIS section.
348		Where additional semantics apply to a function, the material is identified by use of the TPP
349		margin legend.
350	TPS	Thread Execution Scheduling
351		The functionality described is optional. The functionality described is also an extension to the
352		ISO C standard.
353		Where applicable, functions are marked with the TPS margin legend for the SYNOPSIS section.
354		Where additional semantics apply to a function, the material is identified by use of the TPS
355		margin legend.
356	TRC	Trace
357		The functionality described is optional. The functionality described is also an extension to the
358		ISO C standard.
359		Where applicable, functions are marked with the TRC margin legend in the SYNOPSIS section.
360		Where additional semantics apply to a function, the material is identified by use of the TRC
361		margin legend.
362	TRI	Trace Inherit
363		The functionality described is optional. The functionality described is also an extension to the
364		ISO C standard.
365		Where applicable, functions are marked with the TRI margin legend in the SYNOPSIS section.
366		Where additional semantics apply to a function, the material is identified by use of the TRI
367		margin legend.
368	TRL	Trace Log
369		The functionality described is optional. The functionality described is also an extension to the
370		ISO C standard.
371		Where applicable, functions are marked with the TRL margin legend in the SYNOPSIS section.
372		Where additional semantics apply to a function, the material is identified by use of the TRL

373		margin legend.
374	TSA	Thread Stack Address Attribute
375		The functionality described is optional. The functionality described is also an extension to the
376		ISO C standard.
377		Where applicable, functions are marked with the TSA margin legend for the SYNOPSIS section.
378		Where additional semantics apply to a function, the material is identified by use of the TSA
379		margin legend.
380	TSF	Thread-Safe Functions
381		The functionality described is optional. The functionality described is also an extension to the
382		ISO C standard.
383		Where applicable, functions are marked with the TSF margin legend in the SYNOPSIS section.
384		Where additional semantics apply to a function, the material is identified by use of the TSF
385		margin legend.
386	TSH	Thread Process-Shared Synchronization
387		The functionality described is optional. The functionality described is also an extension to the
388		ISO C standard.
389		Where applicable, functions are marked with the TSH margin legend in the SYNOPSIS section.
390		Where additional semantics apply to a function, the material is identified by use of the TSH
391		margin legend.
392	TSP	Thread Sporadic Server
393		The functionality described is optional. The functionality described is also an extension to the
394		ISO C standard.
395		Where applicable, functions are marked with the TSP margin legend in the SYNOPSIS section.
396		Where additional semantics apply to a function, the material is identified by use of the TSP
397		margin legend.
398	TSS	Thread Stack Address Size
399		The functionality described is optional. The functionality described is also an extension to the
400		ISO C standard.
401		Where applicable, functions are marked with the TSS margin legend in the SYNOPSIS section.
402		Where additional semantics apply to a function, the material is identified by use of the TSS
403		margin legend.
404	TYM	Typed Memory Objects
405		The functionality described is optional. The functionality described is also an extension to the
406		ISO C standard.
407		Where applicable, functions are marked with the TYM margin legend in the SYNOPSIS section.
408		Where additional semantics apply to a function, the material is identified by use of the TYM
409		margin legend.
410	UP	User Portability Utilities
411		The functionality described is optional.
412		Where applicable, utilities are marked with the UP margin legend in the SYNOPSIS section.
413		Where additional semantics apply to a utility, the material is identified by use of the UP
414		legend.
415	XSI	Extension
416		The functionality described is an XSI extension. Functionality marked XSI is also an extension to
417		the ISO C standard. Application writers may confidently make use of an extension on all

systems supporting the X/Open System Interfaces Extension.

If an entire SYNOPSIS section is shaded and marked XSI, all the functionality described in that reference page is an extension. See the Base Definitions volume of IEEE Std 1003.1-2001, Section 3.439, XSI.

XSR XSI STREAMS

The functionality described is optional. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the XSR margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the XSR margin legend.

1.9 Format of Entries

The entries in Chapter 3 are based on a common format as follows. The only sections relating to conformance are the SYNOPSIS, DESCRIPTION, RETURN VALUE, and ERRORS sections.

NAME

This section gives the name or names of the entry and briefly states its purpose.

SYNOPSIS

This section summarizes the use of the entry being described. If it is necessary to include a header to use this function, the names of such headers are shown, for example:

```
#include <stdio.h>
```

DESCRIPTION

This section describes the functionality of the function or header.

RETURN VALUE

This section indicates the possible return values, if any.

If the implementation can detect errors, “successful completion” means that no error has been detected during execution of the function. If the implementation does detect an error, the error is indicated.

For functions where no errors are defined, “successful completion” means that if the implementation checks for errors, no error has been detected. If the implementation can detect errors, and an error is detected, the indicated return value is returned and *errno* may be set.

ERRORS

This section gives the symbolic names of the error values returned by a function or stored into a variable accessed through the symbol *errno* if an error occurs.

“No errors are defined” means that error values returned by a function or stored into a variable accessed through the symbol *errno*, if any, depend on the implementation.

EXAMPLES

This section is informative.

This section gives examples of usage, where appropriate. In the event of conflict between an example and a normative part of this volume of IEEE Std 1003.1-2001, the normative material is to be taken as correct.

459 APPLICATION USAGE

460 This section is informative.

461 This section gives warnings and advice to application writers about the entry. In the
462 event of conflict between warnings and advice and a normative part of this volume of
463 IEEE Std 1003.1-2001, the normative material is to be taken as correct.

464 RATIONALE

465 This section is informative.

466 This section contains historical information concerning the contents of this volume of
467 IEEE Std 1003.1-2001 and why features were included or discarded by the standard
468 developers.

469 FUTURE DIRECTIONS

470 This section is informative.

471 This section provides comments which should be used as a guide to current thinking;
472 there is not necessarily a commitment to adopt these future directions.

473 SEE ALSO

474 This section is informative.

475 This section gives references to related information.

476 CHANGE HISTORY

477 This section is informative.

478 This section shows the derivation of the entry and any significant changes that have
479 been made to it.

General Information

480

481 This chapter covers information that is relevant to all the functions specified in Chapter 3 and
 482 the Base Definitions volume of IEEE Std 1003.1-2001, Chapter 13, Headers.

483 2.1 Use and Implementation of Functions

484 Each of the following statements shall apply unless explicitly stated otherwise in the detailed
 485 descriptions that follow:

- 486 1. If an argument to a function has an invalid value (such as a value outside the domain of
 487 the function, or a pointer outside the address space of the program, or a null pointer), the
 488 behavior is undefined.
- 489 2. Any function declared in a header may also be implemented as a macro defined in the
 490 header, so a function should not be declared explicitly if its header is included. Any macro
 491 definition of a function can be suppressed locally by enclosing the name of the function in
 492 parentheses, because the name is then not followed by the left parenthesis that indicates
 493 expansion of a macro function name. For the same syntactic reason, it is permitted to take
 494 the address of a function even if it is also defined as a macro. The use of the C-language
 495 **#undef** construct to remove any such macro definition shall also ensure that an actual
 496 function is referred to.
- 497 3. Any invocation of a function that is implemented as a macro shall expand to code that
 498 evaluates each of its arguments exactly once, fully protected by parentheses where
 499 necessary, so it is generally safe to use arbitrary expressions as arguments. Likewise, those
 500 function-like macros described in the following sections may be invoked in an expression
 501 anywhere a function with a compatible return type could be called.
- 502 4. Provided that a function can be declared without reference to any type defined in a header,
 503 it is also permissible to declare the function explicitly and use it without including its
 504 associated header.
- 505 5. If a function that accepts a variable number of arguments is not declared (explicitly or by
 506 including its associated header), the behavior is undefined.

507 2.2 The Compilation Environment

508 2.2.1 POSIX.1 Symbols

509 Certain symbols in this volume of IEEE Std 1003.1-2001 are defined in headers (see the Base
 510 Definitions volume of IEEE Std 1003.1-2001, Chapter 13, Headers). Some of those headers could
 511 also define symbols other than those defined by IEEE Std 1003.1-2001, potentially conflicting
 512 with symbols used by the application. Also, IEEE Std 1003.1-2001 defines symbols that are not
 513 permitted by other standards to appear in those headers without some control on the visibility
 514 of those symbols.

515 Symbols called “feature test macros” are used to control the visibility of symbols that might be
 516 included in a header. Implementations, future versions of IEEE Std 1003.1-2001, and other
 517 standards may define additional feature test macros.

In the compilation of an application that **#defines** a feature test macro specified by IEEE Std 1003.1-2001, no header defined by IEEE Std 1003.1-2001 shall be included prior to the definition of the feature test macro. This restriction also applies to any implementation-provided header in which these feature test macros are used. If the definition of the macro does not precede the **#include**, the result is undefined.

Feature test macros shall begin with the underscore character ('_').

2.2.1.1 The `_POSIX_C_SOURCE` Feature Test Macro

A POSIX-conforming application should ensure that the feature test macro `_POSIX_C_SOURCE` is defined before inclusion of any header.

When an application includes a header described by IEEE Std 1003.1-2001, and when this feature test macro is defined to have the value 200112L:

1. All symbols required by IEEE Std 1003.1-2001 to appear when the header is included shall be made visible.
2. Symbols that are explicitly permitted, but not required, by IEEE Std 1003.1-2001 to appear in that header (including those in reserved name spaces) may be made visible.
3. Additional symbols not required or explicitly permitted by IEEE Std 1003.1-2001 to be in that header shall not be made visible, except when enabled by another feature test macro.

Identifiers in IEEE Std 1003.1-2001 may only be undefined using the **#undef** directive as described in Section 2.1 (on page 13) or Section 2.2.2. These **#undef** directives shall follow all **#include** directives of any header in IEEE Std 1003.1-2001.

Note: The POSIX.1-1990 standard specified a macro called `_POSIX_SOURCE`. This has been superseded by `_POSIX_C_SOURCE`.

2.2.1.2 The `_XOPEN_SOURCE` Feature Test Macro

XSI An XSI-conforming application should ensure that the feature test macro `_XOPEN_SOURCE` is defined with the value 600 before inclusion of any header. This is needed to enable the functionality described in Section 2.2.1.1 and in addition to enable the XSI extension.

Since this volume of IEEE Std 1003.1-2001 is aligned with the ISO C standard, and since all functionality enabled by `_POSIX_C_SOURCE` set equal to 200112L is enabled by `_XOPEN_SOURCE` set equal to 600, there should be no need to define `_POSIX_C_SOURCE` if `_XOPEN_SOURCE` is so defined. Therefore, if `_XOPEN_SOURCE` is set equal to 600 and `_POSIX_C_SOURCE` is set equal to 200112L, the behavior is the same as if only `_XOPEN_SOURCE` is defined and set equal to 600. However, should `_POSIX_C_SOURCE` be set to a value greater than 200112L, the behavior is unspecified.

2.2.2 The Name Space

XSI All identifiers in this volume of IEEE Std 1003.1-2001, except *environ*, are defined in at least one of the headers, as shown in the Base Definitions volume of IEEE Std 1003.1-2001, Chapter 13, Headers. When `_XOPEN_SOURCE` or `_POSIX_C_SOURCE` is defined, each header defines or declares some identifiers, potentially conflicting with identifiers used by the application. The set of identifiers visible to the application consists of precisely those identifiers from the header pages of the included headers, as well as additional identifiers reserved for the implementation. In addition, some headers may make visible identifiers from other headers as indicated on the relevant header pages.

Implementations may also add members to a structure or union without controlling the visibility of those members with a feature test macro, as long as a user-defined macro with the same name cannot interfere with the correct interpretation of the program. The identifiers reserved for use by the implementation are described below:

1. Each identifier with external linkage described in the header section is reserved for use as an identifier with external linkage if the header is included.
2. Each macro described in the header section is reserved for any use if the header is included.
3. Each identifier with file scope described in the header section is reserved for use as an identifier with file scope in the same name space if the header is included.

The prefixes `posix_`, `POSIX_`, and `_POSIX_` are reserved for use by IEEE Std 1003.1-2001 and other POSIX standards. Implementations may add symbols to the headers shown in the following table, provided the identifiers for those symbols begin with the corresponding reserved prefixes in the following table, and do not use the reserved prefixes `posix_`, `POSIX_`, or `_POSIX_`.

	Header	Prefix	Suffix	Complete Name
575				
576				
577				
578 AIO	< aio.h>	aio_, lio_, AIO_, LIO_		
579	<arpa/inet.h>	in_, inet_		
580	<ctype.h>	to[a-z], is[a-z]		
581	<dirent.h>	d_		
582	<errno.h>	E[0-9], E[A-Z]		
583	<fcntl.h>	l_		
584	<glob.h>	gl_		
585	<grp.h>	gr_		
586	<inttypes.h>			int[0-9a-z-]*_t, uint[0-9a-z-]*_t
587				
588	<limits.h>		_MAX, _MIN	
589	<locale.h>	LC_[A-Z]		
590 MSG	<mqueue.h>	mq_, MQ_		
591 XSI	<ndbm.h>	dbm_		
592	<netdb.h>	h_, n_, p_, s_		
593	<net/if.h>	if_		
594	<netinet/in.h>	in_, ip_, s_, sin_		
595 IP6		in6_, s6_, sin6_		
596 XSI	<poll.h>	pd_, ph_, ps_		
597	<pthread.h>	pthread_, PTHREAD_		
598	<pwd.h>	pw_		
599	<regex.h>	re_, rm_		
600 PS	<sched.h>	sched_, SCHED_		
601 SEM	<semaphore.h>	sem_, SEM_		
602	<signal.h>	sa_, uc_, SIG[A-Z], SIG_[A-Z]		
603 XSI		ss_, sv_		
604 RTS		si_, SI_, sigev_, SIGEV_, sival_		
605 XSI	<stropts.h>	bi_, ic_, l_, sl_, str_		
606	<stdint.h>			int[0-9a-z-]*_t, uint[0-9a-z-]*_t
607				
608	<stdlib.h>	str[a-z]		
609	<string.h>	str[a-z], mem[a-z], wcs[a-z]		
610 XSI	<sys/ipc.h>	ipc_		key, pad, seq
611 MF	<sys/mman.h>	shm_, MAP_, MCL_, MS_, PROT_		
612 XSI	<sys/msg.h>	msg		msg
613 XSI	<sys/resource.h>	rlim_, ru_		
614	<sys/select.h>	fd_, fds_, FD_		
615 XSI	<sys/sem.h>	sem		sem
616 XSI	<sys/shm.h>	shm		
617	<sys/socket.h>	ss_, sa_, if_, ifc_, ifru_, infu_, ifra_, msg_, cmsg_, l_		
618				
619	<sys/stat.h>	st_		
620 XSI	<sys/statvfs.h>	f_		
621	<sys/time.h>	fds_, it_, tv_, FD_		
622	<sys/times.h>	tms_		