

Implementation of ISO/IEC 9945-1 : 1990

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

**Part 1: System Application Program
Interface (API) [C Language]**

National foreword

This British Standard reproduces verbatim ISO/IEC 9945-1 : 1990 and implements it as the UK national standard.

This British Standard is published under the direction of the Information Systems Technology Standards Policy Committee whose Technical Committee IST/5 has the responsibility to:

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

NOTE. International and European Standards, as well as overseas standards, are available from BSI Sales Department, BSI, Linford Wood, Milton Keynes, MK14 6LE.

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

Amendment No. 0
published and effective from 15 September 1993
to BS ISO/IEC 9945-1 : 1990

**Information technology – Portable Operating System Interface
(POSIX) –
Part 1: System Application Program Interface (API)
[C Language]**

NOTE. The European Committee for Standardization (CEN) has accepted ISO/IEC 9945-1 : 1990 as a European Standard designated as EN 29945-1 : 1993. This amendment effects the implementation of EN 29945-1 as a British Standard in the BS EN series.

Implementation of European Standard

Front cover

Delete the existing cover page and substitute the attached.

AMD 7934/September 1993

National foreword

Between paragraphs 1 and 2 insert the following new paragraph.

'In 1993 the European Committee for Standardization (CEN) accepted ISO/IEC 9945-1 : 1990 as European Standard EN 29945-1 : 1993. As a consequence of implementing the European Standard this British Standard is renumbered as BS EN 29945-1 and any reference to BS ISO/IEC 9945-1 should be read as a reference to BS EN 29945-1.'

AMD 7934/September 1993

New EN title page and foreword

Insert the attached new EN title page and foreword page immediately after the national foreword.

AMD 7934/September 1993

Information technology — Portable Operating System Interface (POSIX) —

Part 1. System Application Program Interface (API) [C Language]

The European Standard EN 29945-1 : 1993 has the status of a
British Standard

Technologies de l'information – Interface pour la
portabilité des systèmes (POSIX) –
Partie 1. Interface programme de systèmes
d'application (API) [Langage C]

Informationstechnik –
Betriebssystem-Schnittstelle für die
Austauschbarkeit von Systemen (POSIX) –
Teil 1. Schnittstelle zwischen System und
Anwendungsprogramm (API) [Sprache C]

UDC 681.3.06

NO COPYING IN ANY FORM WITHOUT WRITTEN PERMISSION FROM BSI



International Standard ISO/IEC 9945-1: 1990

IEEE Std 1003.1-1990

(Revision of IEEE Std 1003.1-1988)

Information technology—Portable Operating System Interface (POSIX) Part 1: System Application Program Interface (API) [C Language]

Sponsor

Technical Committee on Operating Systems
and Application Environments
of the
IEEE Computer Society

Approved September 28, 1990

IEEE Standards Board

Approved 1990 by the

International Organization for Standardization
and by the
International Electrotechnical Commission

Abstract: ISO/IEC 9945-1: 1990 (IEEE Std 1003.1-1990), *Information technology—Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API) [C Language]* is part of the POSIX series of standards for applications and user interfaces to open systems. It defines the applications interface to basic system services for input/output, file system access, and process management. It also defines a format for data interchange. This standard is stated in terms of its C binding.

Keywords: API, application portability, C (programming language), data processing, information interchange, open systems, operating system, portable application, POSIX, programming language, system configuration computer interface



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



Published by
The Institute of Electrical and Electronics Engineers, Inc.

Foreword

On the proposal of the CEN Central Secretariat, the Technical Board has decided to submit the international standard:

'Information technology – Portable Operating System Interface (POSIX) – Part 1 : System Application Program Interface (API) [C language] (ISO/IEC 9945-1 : 1990)' to the formal vote.

The result of the formal vote was positive.

For the time being, this document exists only in English.

This European Standard shall be given the status of a national standard, either by publication of an identical text or by endorsement, at the latest by November 1993, and conflicting national standards shall be withdrawn at the latest by November 1993.

In accordance with the CEN/CENELEC Internal Regulations, the following countries are bound to implement this European Standard: Austria, Belgium, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.



S

INTERNATIONAL STANDARD

**ISO/IEC
9945-1**

**IEEE
Std 1003.1**

First edition
1990-12-07

Information technology — Portable Operating System Interface (POSIX) —

Part 1 : System Application Program Interface (API) [C Language]

*Technologies de l'information — Interface pour la portabilité des systèmes (POSIX) —
Partie 1 : Interface programme de systèmes d'application (API) [Langage C]*



Reference number
ISO/IEC 9945-1 : 1990 (E)
IEEE Std 1003.1-1990



Documents
Document Sharing Hub

ISBN 1-55937-061-0

Library of Congress Catalog Number 90-084554

Quote in 8.1.2.3 on Returns is taken from X3.159-1989,
developed under the auspices of the American National Standards
Accredited Committee X3 Technical Committee X3J11, Computer and
Business Equipment Manufacturers Association (CBEMA),
311 First St, N.W., Suite 500, Washington, DC 20001.

©Copyright 1990 by

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017, USA

*No part of this publication may be reproduced in any form,
in an electronic retrieval system or otherwise,
without the prior written permission of the publisher.*

December 7, 1990

International Standard ISO/IEC 9945-1: 1990

IEEE Std 1003.1-1990

(Revision of IEEE Std 1003.1-1988)

Information technology—Portable Operating System Interface (POSIX) Part 1: System Application Program Interface (API) [C Language]

Sponsor

Technical Committee on Operating Systems
and Application Environments
of the
IEEE Computer Society

Approved September 28, 1990

IEEE Standards Board

Approved 1990 by the

International Organization for Standardization
and by the
International Electrotechnical Commission

Abstract: ISO/IEC 9945-1: 1990 (IEEE Std 1003.1-1990), *Information technology—Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API) [C Language]* is part of the POSIX series of standards for applications and user interfaces to open systems. It defines the applications interface to basic system services for input/output, file system access, and process management. It also defines a format for data interchange. This standard is stated in terms of its C binding.

Keywords: API, application portability, C (programming language), data processing, information interchange, open systems, operating system, portable application, POSIX, programming language, system configuration computer interface



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



Published by
The Institute of Electrical and Electronics Engineers, Inc.

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. 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.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

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 all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, the IEEE and the members of its technical 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 on standards and requests for interpretations should be addressed to:

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

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.

Contents

	PAGE
Foreword	viii
Introduction	ix
Section 1: General	1
1.1 Scope	1
1.2 Normative References	2
1.3 Conformance	2
Section 2: Terminology and General Requirements	9
2.1 Conventions	9
2.2 Definitions	10
2.3 General Concepts	21
2.4 Error Numbers	23
2.5 Primitive System Data Types	27
2.6 Environment Description	27
2.7 C Language Definitions	29
2.8 Numerical Limits	34
2.9 Symbolic Constants	37
Section 3: Process Primitives	41
3.1 Process Creation and Execution	41
3.1.1 Process Creation	41
3.1.2 Execute a File	42
3.2 Process Termination	46
3.2.1 Wait for Process Termination	47
3.2.2 Terminate a Process	49
3.3 Signals	51
3.3.1 Signal Concepts	51
3.3.2 Send a Signal to a Process	56
3.3.3 Manipulate Signal Sets	57
3.3.4 Examine and Change Signal Action	58
3.3.5 Examine and Change Blocked Signals	60
3.3.6 Examine Pending Signals	62
3.3.7 Wait for a Signal	62
3.4 Timer Operations	63
3.4.1 Schedule Alarm	63
3.4.2 Suspend Process Execution	64
3.4.3 Delay Process Execution	65
Section 4: Process Environment	67
4.1 Process Identification	67
4.1.1 Get Process and Parent Process IDs	67

	PAGE
4.2 User Identification	68
4.2.1 Get Real User, Effective User, Real Group, and Effective Group IDs	68
4.2.2 Set User and Group IDs	68
4.2.3 Get Supplementary Group IDs	70
4.2.4 Get User Name	71
4.3 Process Groups	72
4.3.1 Get Process Group ID	72
4.3.2 Create Session and Set Process Group ID	72
4.3.3 Set Process Group ID for Job Control	73
4.4 System Identification	74
4.4.1 Get System Name	74
4.5 Time	75
4.5.1 Get System Time	75
4.5.2 Get Process Times	76
4.6 Environment Variables	77
4.6.1 Environment Access	77
4.7 Terminal Identification	78
4.7.1 Generate Terminal Pathname	78
4.7.2 Determine Terminal Device Name	79
4.8 Configurable System Variables	80
4.8.1 Get Configurable System Variables	80
Section 5: Files and Directories	83
5.1 Directories	83
5.1.1 Format of Directory Entries	83
5.1.2 Directory Operations	83
5.2 Working Directory	86
5.2.1 Change Current Working Directory	86
5.2.2 Get Working Directory Pathname	87
5.3 General File Creation	88
5.3.1 Open a File	88
5.3.2 Create a New File or Rewrite an Existing One	91
5.3.3 Set File Creation Mask	91
5.3.4 Link to a File	92
5.4 Special File Creation	94
5.4.1 Make a Directory	94
5.4.2 Make a FIFO Special File	95
5.5 File Removal	96
5.5.1 Remove Directory Entries	96
5.5.2 Remove a Directory	98
5.5.3 Rename a File	99
5.6 File Characteristics	101
5.6.1 File Characteristics: Header and Data Structure	101
5.6.2 Get File Status	103
5.6.3 Check File Accessibility	104
5.6.4 Change File Modes	106
5.6.5 Change Owner and Group of a File	107

	PAGE
5.6.6 Set File Access and Modification Times	108
5.7 Configurable Pathname Variables	110
5.7.1 Get Configurable Pathname Variables	110
Section 6: Input and Output Primitives	113
6.1 Pipes	113
6.1.1 Create an Inter-Process Channel	113
6.2 File Descriptor Manipulation	114
6.2.1 Duplicate an Open File Descriptor	114
6.3 File Descriptor Deassignment	115
6.3.1 Close a File	115
6.4 Input and Output	116
6.4.1 Read from a File	116
6.4.2 Write to a File	118
6.5 Control Operations on Files	121
6.5.1 Data Definitions for File Control Operations	121
6.5.2 File Control	121
6.5.3 Reposition Read/Write File Offset	127
Section 7: Device- and Class-Specific Functions	129
7.1 General Terminal Interface	129
7.1.1 Interface Characteristics	129
7.1.1.1 Opening a Terminal Device File	129
7.1.1.2 Process Groups	129
7.1.1.3 The Controlling Terminal	130
7.1.1.4 Terminal Access Control	130
7.1.1.5 Input Processing and Reading Data	131
7.1.1.6 Canonical Mode Input Processing	132
7.1.1.7 Noncanonical Mode Input Processing	132
7.1.1.8 Writing Data and Output Processing	133
7.1.1.9 Special Characters	133
7.1.1.10 Modem Disconnect	135
7.1.1.11 Closing a Terminal Device File	135
7.1.2 Parameters That Can Be Set	135
7.1.2.1 termios Structure	135
7.1.2.2 Input Modes	136
7.1.2.3 Output Modes	137
7.1.2.4 Control Modes	138
7.1.2.5 Local Modes	139
7.1.2.6 Special Control Characters	140
7.1.2.7 Baud Rate Values	141
7.1.3 Baud Rate Functions	141
7.1.3.1 Synopsis	141
7.1.3.2 Description	142
7.1.3.3 Returns	142
7.1.3.4 Errors	142
7.1.3.5 Cross-References	142
7.2 General Terminal Interface Control Functions	143
7.2.1 Get and Set State	143

	PAGE
7.2.2 Line Control Functions	145
7.2.3 Get Foreground Process Group ID	147
7.2.4 Set Foreground Process Group ID	148
Section 8: Language-Specific Services for the C Programming	
Language	151
8.1 Referenced C Language Routines	151
8.1.1 Extensions to Time Functions	152
8.1.2 Extensions to <i>setlocale()</i> Function	154
8.2 C Language Input/Output Functions	155
8.2.1 Map a Stream Pointer to a File Descriptor	156
8.2.2 Open a Stream on a File Descriptor	157
8.2.3 Interactions of Other FILE -Type C Functions	158
8.2.4 Operations on Files — the <i>remove()</i> Function	162
8.3 Other C Language Functions	162
8.3.1 Nonlocal Jumps	162
8.3.2 Set Time Zone	162
Section 9: System Databases	165
9.1 System Databases	165
9.2 Database Access	166
9.2.1 Group Database Access	166
9.2.2 User Database Access	167
Section 10: Data Interchange Format	169
10.1 Archive/Interchange File Format	169
10.1.1 Extended tar Format	169
10.1.2 Extended cpio Format	173
10.1.3 Multiple Volumes	177
Annex A (informative) Bibliography	179
A.1 Related Open Systems Standards	179
A.2 Other Standards	181
A.3 Historical Documentation and Introductory Texts	182
Annex B (informative) Rationale and Notes	185
B.1 Scope and Normative References	185
B.2 Definitions and General Requirements	196
B.3 Process Primitives	226
B.4 Process Environment	246
B.5 Files and Directories	253
B.6 Input and Output Primitives	264
B.7 Device- and Class-Specific Functions	273
B.8 Language-Specific Services for the C Programming Language	283
B.9 System Databases	293
B.10 Data Interchange Format	294
Annex C (informative) Header Contents Samples	301

	PAGE
Annex D (informative) Profiles	313
D.1 Definitions	313
D.2 Options in This Part of ISO/IEC 9945	314
D.3 Related Standards	315
D.4 Related Activities	315
D.5 Relationship to IEEE Draft Project 1003.0	315
Annex E (informative) Sample National Profile	317
E.1 (Example) Profile for Denmark	318
Identifier Index	321
Alphabetic Topical Index	327

TABLES

Table 2-1 – Primitive System Data Types	27
Table 2-2 – Reserved Header Symbols	31
Table 2-3 – Minimum Values	35
Table 2-4 – Run-Time Increasable Values	35
Table 2-5 – Run-Time Invariant Values (Possibly Indeterminate)	36
Table 2-6 – Pathname Variable Values	36
Table 2-7 – Invariant Value	37
Table 2-8 – Symbolic Constants for the <i>access()</i> Function	38
Table 2-9 – Symbolic Constants for the <i>lseek()</i> Function	38
Table 2-10 – Compile-Time Symbolic Constants	38
Table 2-11 – Execution-Time Symbolic Constants	39
Table 3-1 – Required Signals	52
Table 3-2 – Job Control Signals	52
Table 4-1 – <i>uname()</i> Structure Members	75
Table 4-2 – Configurable System Variables	80

Table 5-1	– stat Structure	101
Table 5-2	– Configurable Pathname Variables	111
Table 6-1	– cmd Values for fcntl()	122
Table 6-2	– File Descriptor Flags Used for fcntl()	122
Table 6-3	– l_type Values for Record Locking With fcntl()	122
Table 6-4	– oflag Values for open()	122
Table 6-5	– File Status Flags Used for open() and fcntl()	122
Table 6-6	– File Access Modes Used for open() and fcntl()	123
Table 6-7	– Mask for Use With File Access Modes	123
Table 6-8	– flock Structure	125
Table 6-9	– fcntl() Return Values	126
Table 7-1	– termios Structure	136
Table 7-2	– termios c_iflag Field	136
Table 7-3	– termios c_cflag Field	138
Table 7-4	– termios c_lflag Field	139
Table 7-5	– termios c_cc Special Control Characters	140
Table 7-6	– termios Baud Rate Values	141
Table 9-1	– group Structure	166
Table 9-2	– passwd Structure	167
Table 10-1	– tar Header Block	170
Table 10-2	– Byte-Oriented cpio Archive Entry	174
Table 10-3	– Values for cpio c_mode Field	175
Table B-1	– Suggested Feature Test Macros	222

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) together form a system for worldwide standardization as a whole. 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 nongovernmental, 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. Draft International Standards adopted by the joint technical committee are circulated to national bodies for approval before their acceptance as International Standards. They are approved in accordance with procedures requiring at least 75% approval by the national bodies voting.

International Standard ISO/IEC 9945-1: 1990 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

ISO/IEC 9945 consists of the following parts, under the general title *Information technology—Portable operating system interface (POSIX)*:

- *Part 1: System application program interface (API) [C language]*
- *Part 2: Shell and utilities* (under development)
- *Part 3: System administration* (under development)

Annexes A to E of ISO/IEC 9945-1 are provided for information only.



International Organization for Standardization/International Electrotechnical Commission
Case postale 56 • CH-1211 Genève 20 • Switzerland

Introduction

(This Introduction is not a normative part of ISO/IEC 9945-1 Information technology—Portable operating system interface (POSIX)—Part 1: System application programming interface (API) [C Language], but is included for information only.)

The purpose of this part of ISO/IEC 9945 is to define a standard operating system interface and environment based on the UNIX¹⁾ Operating System documentation to support application portability at the source level. This is intended for systems implementors and applications software developers.

Initially,²⁾ the focus of this part of ISO/IEC 9945 is to provide standardized services via a C language interface. Future revisions are expected to contain bindings for other programming languages as well as for the C language. This will be accomplished by breaking this part of ISO/IEC 9945 into multiple portions—one defining core requirements independent of any programming language, and others composed of programming language bindings.

The core requirements portion will define a set of required services common to any programming language that can be reasonably expected to form a language binding to this part of ISO/IEC 9945. These services will be described in terms of functional requirements and will not define programming language-dependent interfaces. Language bindings will consist of two major parts. One will contain the programming language's standardized interface for accessing the core services defined in the programming language-independent core requirements section of this part of ISO/IEC 9945. The other will contain a standardized interface for language-specific services. Any implementation claiming conformance to this part of ISO/IEC 9945 with any language binding will be required to comply with both sections of the language binding.

Within this document, the term "POSIX.1" refers to this part of ISO/IEC 9945 itself.

Organization of This Part of ISO/IEC 9945

This part of ISO/IEC 9945 is divided into four elements:

- (1) Statement of scope and list of normative references (Section 1)
- (2) Definitions and global concepts (Section 2)
- (3) The various interface facilities (Sections 3 through 9)
- (4) Data interchange format (Section 10)

1) UNIX is a registered trademark of AT&T in the USA and other countries.

2) The vertical rules in the right margin depict technical or significant non-editorial changes from IEEE Std 1003.1-1988 to IEEE Std 1003.1-1990. A vertical rule beside an empty line indicates deleted text.

Most of the sections describe a single service interface. The C Language binding for the service interface is given in the subclause labeled Synopsis. The Description subclause provides a specification of the operation performed by the service interface. Some examples may be provided to illustrate the interfaces described. In most cases there are also Returns and Errors subclauses specifying return values and possible error conditions. References are used to direct the reader to other related sections. Additional material to complement sections in this part of ISO/IEC 9945 may be found in the Rationale and Notes, Annex B. This annex provides historical perspectives into the technical choices made by the developers of this part of ISO/IEC 9945. It also provides information to emphasize consequences of the interfaces described in the corresponding section of this part of ISO/IEC 9945.

Informative annexes are not part of the standard and are provided for information only. (There is a type of annex called “normative” that is part of a standard and imposes requirements, but there are currently no such normative annexes in this part of ISO/IEC 9945.) They are provided for guidance and to help understanding.

In publishing this part of ISO/IEC 9945, its developers simply intend to provide a yardstick against which various operating system implementations can be measured for conformance. It is *not* the intent of the developers to measure or rate any products, to reward or sanction any vendors of products for conformance or lack of conformance to this part of ISO/IEC 9945, or to attempt to enforce this part of ISO/IEC 9945 by these or any other means. The responsibility for determining the degree of conformance or lack thereof with this part of ISO/IEC 9945 rests solely with the individual who is evaluating the product claiming to be in conformance with this part of ISO/IEC 9945.

Base Documents

The various interface facilities described herein are based on the *1984 /usr/group Standard* derived and published by the UniForum (formerly /usr/group) Standards Committee. The *1984 /usr/group Standard* and this part of ISO/IEC 9945 are largely based on UNIX Seventh Edition, UNIX System III, UNIX System V, 4.2BSD, and 4.3BSD documentation,³⁾ but wherever possible, compatibility with other systems derived from the UNIX operating system, or systems compatible with that system, has been maintained.

Background

The developers of POSIX.1 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. In the course of their deliberations, the developers reviewed related American and international standards, both published and in progress.

Conceptually, POSIX.1 describes a set of fundamental services needed for the efficient construction of application programs. Access to these services has been

3) The IEEE is grateful to both AT&T and UniForum for permission to use their materials.

provided by defining an interface, using the C programming language, that establishes 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 IEEE Std 1003.1-1988, the name POSIX more correctly refers to a *family* of related standards: IEEE 1003.*n* and the parts of International Standard 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 ISO/IEC 9945 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 part of ISO/IEC 9945:

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 part of ISO/IEC 9945 codifies the common, existing definition of the UNIX system. There was no attempt to define a new system interface.

Interface, Not Implementation

This part of ISO/IEC 9945 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 Annex B). 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.

Source, Not Object, Portability

This part of ISO/IEC 9945 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 part of ISO/IEC 9945 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. However, few impediments were placed in the way of binary compatibility, and some remarks on this are found in Annex B. See B.1.3.1.1 and B.4.8.

The C Language

This part of ISO/IEC 9945 is written in terms of the standard C language as specified in the C Standard [2].⁵⁾ See B.1.3 and B.1.1.1.

No Super-User, No System Administration

There was no intention to specify all aspects of an operating system. System administration facilities and functions are excluded from POSIX.1, and functions usable only by the super-user have not been included. Annex B notes several such instances. Still, an implementation of the standard interface may also implement features not in this part of ISO/IEC 9945; see 1.3.1.1. This part of ISO/IEC 9945 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, POSIX.1 is as minimal as possible. For example, it usually specifies only one set of functions to implement a capability. Exceptions were made in some cases where long tradition and many existing applications included certain functions, such as *creat()*. In such cases, as throughout POSIX.1, redundant definitions were avoided: *creat()* is defined as a special case of *open()*. Redundant functions or implementations with less tradition were excluded.

Broadly Implementable

The developers of POSIX.1 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

⁵⁾ The number in braces corresponds to those of the references in 1.2 (or the bibliographic entry in Annex A if the number is preceded by the letter B).

- 155 (3) Emulations hosted on entirely different operating systems
156 (4) Networked systems
157 (5) Distributed systems
158 (6) Systems running on a broad range of hardware

159 No direct references to this goal appear in this part of ISO/IEC 9945, but
160 some results of it are mentioned in Annex B.

161 **Minimal Changes to Historical Implementations**

162 There are no known historical implementations that will not have to
163 change in some area to conform to this part of ISO/IEC 9945, and in a
164 few areas POSIX.1 does not exactly match any existing system interface
165 (for example, see the discussion of O_NONBLOCK in B.6). Nonetheless,
166 there is a set of functions, types, definitions, and concepts that form an
167 interface that is common to most historical implementations. POSIX.1
168 specifies that common interface and extends it in areas where there has
169 historically been no consensus, preferably

- 170 (1) By standardizing an interface like one in an historical implemen-
171 tation; e.g., directories, or;
172 (2) By specifying an interface that is readily implementable in terms
173 of, and backwards compatible with, historical implementations,
174 such as the extended tar format in 10.1.1, or;
175 (3) By specifying an interface that, when added to an historical imple-
176 mentation, will not conflict with it, like B.6.

177 Required changes to historical implementations have been kept to a
178 minimum, but they do exist, and Annex B points out some of them.

179 POSIX.1 is specifically not a codification of a particular vendor's product.
180 It is similar to the UNIX system, but it is not identical to it.

181 It should be noted that implementations will have different kinds of
182 extensions. Some will reflect "historical usage" and will be preserved for
183 execution of pre-existing applications. These functions should be con-
184 sidered "obsolescent" and the standard functions used for new applica-
185 tions. Some extensions will represent functions beyond the scope of
186 POSIX.1. These need to be used with careful management to be able to
187 adapt to future POSIX.1 extensions and/or port to implementations that
188 provide these services in a different manner.

189 **Minimal Changes to Existing Application Code**

190 A goal of POSIX.1 was to minimize additional work for the developers of |
191 applications. However, because every known historical implementation |
192 will have to change at least slightly to conform, some applications will
193 have to change. Annex B points out the major places where POSIX.1
194 implies such changes.

195 **Related Standards Activities**

196 Activities to extend this part of ISO/IEC 9945 to address additional requirements
197 are in progress, and similar efforts can be anticipated in the future.

198 The following areas are under active consideration at this time, or are expected to
199 become active in the near future:⁶⁾

- 200 (1) Language-independent service descriptions of this part of ISO/IEC 9945
- 201 (2) C, Ada, and FORTRAN Language bindings to (1)
- 202 (3) Shell and Utility facilities
- 203 (4) Verification testing methods
- 204 (5) Realtime facilities
- 205 (6) Secure/Trusted System considerations
- 206 (7) Network interface facilities
- 207 (8) System Administration
- 208 (9) Graphical User Interfaces
- 209 (10) Profiles describing application- or user-specific combinations of Open Sys-
210 tems standards for: supercomputing, multiprocessor, and batch exten-
211 sions; transaction processing; realtime systems; and multiuser systems
212 based on historical models
- 213 (11) An overall guide to POSIX-based or related Open Systems standards and
214 profiles

215 Extensions are approved as “amendments” or “revisions” to this document, follow-
216 ing the IEEE and ISO/IEC Procedures.

217 Approved amendments are published separately until the full document is
218 reprinted and such amendments are incorporated in their proper positions.

219 If you have interest in participating in the TCOS working groups addressing these
220 issues, please send your name, address, and phone number to the Secretary, IEEE
221 Standards Board, Institute of Electrical and Electronics Engineers, Inc., P.O. Box
222 1331, 445 Hoes Lane, Piscataway, NJ 08855-1331, and ask to have this forwarded
223 to the chairperson of the appropriate TCOS working group. If you have interest in
224 participating in this work at the international level, contact your ISO/IEC national
225 body.

6) A *Standards Status Report* that lists all current IEEE Computer Society standards projects is available from the IEEE Computer Society, 1730 Massachusetts Avenue NW, Washington, DC 20036-1903; Telephone: +1 202 371-0101; FAX: +1 202 728-9614. Working drafts of POSIX standards under development are also available from this office.

IEEE Std 1003.1-1990 was prepared by the 1003.1 Working Group, sponsored by the Technical Committee on Operating Systems and Application Environments of the IEEE Computer Society. At the time this standard was approved, the membership of the 1003.1 Working Group was as follows:

**Technical Committee on Operating Systems
and Application Environments (TCOS)**

Chair: Luis-Felipe Cabrera

Standards Subcommittee for TCOS

Chair: Jim Isaak
Treasurer: Quin Hahn
Secretary: Shane McCarron

1003.1 Working Group Officials

Chair: Donn Terry
Vice Chair: Keith Stuck
Editor: Hal Jespersen
Secretary: Keith Stuck

Working Group

Steve Bartels
Robert Bismuth
James Bohem
Kathy Bohrer
Keith Bostic
Jonathan Brown
Tim Carter
Myles Connors
Landon Curt Noll
Dave Decot
Mark Doran
Glenn Fowler

Greg Goddard
Andrew Griffith
Rand Hoven
Randall Howard
Mike Karels
Jeff Kimmel
David Korn
Bob Lenk
Shane McCarron
John Meyer
Martha Nalebuff

Neguine Navab
Paul Rabin
Seth Rosenthal
Lorne Schachter
Steve Schwarm
Paul Shaughnessy
Steve Sommars
Ravi Tavakley
Jeff Tofano
David Willcox
John Wu

The following persons were members of the 1003.1 Balloting Group that approved the standard for submission to the IEEE Standards Board:

David Chinn	<i>Open Software Foundation Institutional Representative</i>
Michael Lambert	<i>X/Open Institutional Representative</i>
Heinz Lycklama	<i>UniForum Institutional Representative</i>
Shane McCarron	<i>UNIX International Institutional Representative</i>

Helene Armitage	William Henderson	Martha Nalebuff
David Athersych	Lee A. Hollaar	Barry Needham
Timothy Baker	Terrence Holm	Alan F. Nugent
Geoff Baldwin	Randall Howard	Jim Oldroyd
Steven E. Barber	Irene Hu	Craig Partridge
Robert Barned	Andrew Huber	John Peace
John Barr	Richard Hughes-Rowlands	John C. Penney
James Bohem	Judith Hurwitz	P. Plauger
Kathryn Bohrer	Jim Isaak	Gerald Powell
Robert Borochoff	Dan Iuster	Scott E. Preece
Keith Bostic	Richard James	Joseph Ramus
James P. Bound	Hal Jespersen	Wendy Rauch
Joseph Boykin	Michael J. Karels	Carol Raye
Kevin Brady	Sol Kavy	Wayne B. Reed
Phyllis Eve Bregman	Lorraine C. Kevra	Christopher J. Riddick
Fred Lee Brown, Jr.	Jeffrey S. Kimmel	Andrew K. Roach
A. Winsor Brown	M. J. Kirk	Robert Sarr
Luis-Felipe Cabrera	Dale Kirkland	Lorne H. Schachter
Nicholas A. Camillone	John T. Kline	Norman Schneidewind
Clyde Camp	Kenneth Klingman	Stephen Schwarm
John Carson	Joshua Knight	Richard Scott
Steven Carter	Andrew R. Knipp	Leonard Seagren
Jerry Cashin	David Korn	Glen Seeds
Kilnam Chon	Don Kretsch	Karen Sheaffer
Anthony Cincotta	Takahiko Kuki	Charles Smith
Mark Colburn	Thomas Kwan	Steven Sommars
Donald W. Cragun	Robin B. Lake	Douglas H. Steves
Ana Maria DeAlvare	Mark Lamonds	James Tanner
Dave Decot	Doris Lebovits	Ravi Tavakley
Steven Deller	Maggie Lee	Marc Teitelbaum
Terence Dowling	Greger Leijonhufvud	Donn S. Terry
Stephen A. Dum	Robert Lenk	Gary F. Tom
John D. Earls	David Lennert	Andrew Twigger
Ron Elliott	Donald Lewine	Mark-Rene Uchida
David Emery	Kevin Lewis	L. David Umbaugh
Philip H. Enslow	F. C. Lim	Michael W. Vannier
Ken Faubel	James Lonjers	David John Wallace
Kester Fong	Warren E. Loper	Stephen Walli
Kenneth R. Gibb	Roger Martin	Larry Wehr
Michel Gien	Martin J. McGowan	Bruce Weiner
Gregory W. Goddard	Marshall McKusick	Robert Weissensee
Dave Grindeland	Robert McWhirter	P. J. Weyman
Judy Guist	Paul Merry	Andrew Wheeler, Jr.
James Hall	Doug Michels	David Willcox
Charles Hammons	Gary W. Miller	Randall F. Wright
Allen Hankinson	James Moe	Oren Yuen
Steve Head	James W. Moore	Jason Zions
Barry Hedquist		

When the IEEE Standards Board approved this standard on September 28, 1990, it had the following membership:

Marco W. Migliaro, *Chairman*

James M. Daly, *Vice Chairman*

Andrew G. Salem, *Secretary*

Dennis Bodson
Paul L. Borrill
Fletcher J. Buckley
Allen L. Clapp
Stephen R. Dillon
Donald C. Fleckenstein
Jay Forster*
Thomas L. Hannan

Kenneth D. Hendrix
John W. Horch
Joseph L. Koepfinger*
Irving Kolodny
Michael A. Lawler
Donald J. Loughry
John E. May, Jr.

Lawrence V. McCall
L. Bruce McClung
Donald T. Michael*
Stig Nilsson
Roy T. Oishi
Gary S. Robinson
Terrance R. Whitemore
Donald W. Zipse

*Member Emeritus

Information technology—Portable operating system interface (POSIX)—Part 1: System application programming interface (API) [C Language]

Section 1: General

1.1 Scope

This part of ISO/IEC 9945 defines a standard operating system interface and environment to support application portability at the source-code level. It is intended to be used by both application developers and system implementors.

This part of ISO/IEC 9945 comprises four major components:

- (1) Terminology, concepts, and definitions and specifications that govern structures, headers, environment variables, and related requirements
- (2) Definitions for system service interfaces and subroutines
- (3) Language-specific system services for the C programming language
- (4) Interface issues, including portability, error handling, and error recovery

The following areas are outside of the scope of this part of ISO/IEC 9945:

- (1) User interface (shell) and associated commands
- (2) Networking protocols and system call interfaces to those protocols
- (3) Graphics interfaces
- (4) Database management system interfaces
- (5) Record I/O considerations

(6) Object or binary code portability

(7) System configuration and resource availability

(8) The behavior of system services on systems supporting concurrency within a single process

This part of ISO/IEC 9945 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.

This part of ISO/IEC 9945 has been defined exclusively at the source-code level. The objective is that a Strictly Conforming POSIX.1 Application source program can be translated to execute on a conforming implementation.

1.2 Normative References

The following standards contain provisions which, through references in this text, constitute provisions of this part of ISO/IEC 9945. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

{1} ISO/IEC 646: 1983,¹⁾ *Information processing—ISO 7-bit coded character set for information interchange.*

{2} ISO/IEC 9899: ...,²⁾ *Information technology—Programming languages—C.*

1.3 Conformance

1.3.1 Implementation Conformance

1.3.1.1 Requirements

A conforming implementation shall meet all of the following criteria:

1) Under revision. (This notation is meant to explicitly reference the 1990 Draft International Standard version of ISO/IEC 646.)

ISO/IEC documents can be obtained from the ISO office, 1, rue de Varembe, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse.

2) To be approved and published.

- 49 (1) The system shall support all required interfaces defined within this part
50 of ISO/IEC 9945. These interfaces shall support the functional behavior
51 described herein.
- 52 (2) The system may provide additional functions or facilities not required by
53 this part of ISO/IEC 9945. Nonstandard extensions should be identified
54 as such in the system documentation. Nonstandard extensions, when
55 used, may change the behavior of functions or facilities defined by this
56 part of ISO/IEC 9945. The conformance document shall define an
57 environment in which an application can be run with the behavior
58 specified by the standard. In no case shall such an environment require
59 modification of a Strictly Conforming POSIX.1 Application.

60 1.3.1.2 Documentation

61 A conformance document with the following information shall be available for an
62 implementation claiming conformance to this part of ISO/IEC 9945. The confor-
63 mance document shall have the same structure as this part of ISO/IEC 9945, with
64 the information presented in the appropriately numbered sections, clauses, and
65 subclauses. The conformance document shall not contain information about
66 extended facilities or capabilities outside the scope of this part of ISO/IEC 9945.

67 The conformance document shall contain a statement that indicates the full
68 name, number, and date of the standard that applies. The conformance document
69 may also list international software standards that are available for use by a Con-
70 forming POSIX.1 Application. Applicable characteristics where documentation is
71 required by one of these standards, or by standards of government bodies, may
72 also be included.

73 The conformance document shall describe the limit values found in the
74 <limits.h> and <unistd.h> headers, stating values, the conditions under
75 which those values may change, and the limits of such variations, if any.

76 The conformance document shall describe the behavior of the implementation for
77 all implementation-defined features defined in this part of ISO/IEC 9945. This
78 requirement shall be met by listing these features and providing either a specific
79 reference to the system documentation or providing full syntax and semantics of
80 these features. The conformance document may specify the behavior of the imple-
81 mentation for those features where this part of ISO/IEC 9945 states that imple-
82 mentations may vary or where features are identified as undefined or unspecified.

83 No specifications other than those described in this part of ISO/IEC 9945 shall be
84 present in the conformance document.

85 The phrases “shall document” or “shall be documented” in this part of
86 ISO/IEC 9945 mean that documentation of the feature shall appear in the confor-
87 mance document, as described previously, unless the system documentation is
88 explicitly mentioned.

89 The system documentation should also contain the information found in the con-
90 formance document.

1.3.1.3 Conforming Implementation Options

The following symbolic constants, described in the subclauses indicated, reflect implementation options for this part of ISO/IEC 9945 that could warrant requirement by Conforming POSIX.1 Applications, or in specifications of conforming systems, or both:

{NGROUPS_MAX}	Multiple groups option (in 2.8.3)
{_POSIX_JOB_CONTROL}	Job control option (in 2.9.3)
{_POSIX_CHOWN_RESTRICTED}	Administrative/security option (in 2.9.4)

The remaining symbolic constants in 2.9.3 and 2.9.4 are useful for testing purposes and as a guide to applications on the types of behaviors they need to be able to accommodate. They do not reflect sufficient functional difference to warrant requirement by Conforming POSIX.1 Applications or to distinguish between conforming implementations.

In the cases where omission of an option would cause functions described by this part of ISO/IEC 9945 to not be defined, an implementation shall provide a function that is callable with the syntax defined in this part of ISO/IEC 9945, even though in an instance of the implementation the function may always do nothing but return an error.

1.3.2 Application Conformance

All applications claiming conformance to this part of ISO/IEC 9945 shall use only language-dependent services for the C programming language described in 1.3.3 and shall fall within one of the following categories:

1.3.2.1 Strictly Conforming POSIX.1 Application

A Strictly Conforming POSIX.1 Application is an application that requires only the facilities described in this part of ISO/IEC 9945 and the applicable language standards. Such an application shall accept any behavior described in this part of ISO/IEC 9945 as *unspecified* or *implementation-defined*, and for symbolic constants, shall accept any value in the range permitted by this part of ISO/IEC 9945. Such applications are permitted to adapt to the availability of facilities whose availability is indicated by the constants in 2.8 and 2.9.

1.3.2.2 Conforming POSIX.1 Application

1.3.2.2.1 ISO/IEC Conforming POSIX.1 Application

An ISO/IEC Conforming POSIX.1 Application is an application that uses only the facilities described in this part of ISO/IEC 9945 and approved Conforming Language bindings for any ISO or IEC standard. Such an application shall include a statement of conformance that documents all options and limit dependencies, and all other ISO or IEC standards used.

128 1.3.2.2.2 <National Body> Conforming POSIX.1 Application

129 A <National Body> Conforming POSIX.1 Application differs from an ISO/IEC Con-
130 forming POSIX.1 Application in that it also may use specific standards of a single
131 ISO/IEC member body referred to here as “<National Body>.” Such an application
132 shall include a statement of conformance that documents all options and limit
133 dependencies, and all other <National Body> standards used.

134 1.3.2.3 Conforming POSIX.1 Application Using Extensions

135 A Conforming POSIX.1 Application Using Extensions is an application that differs
136 from a Conforming POSIX.1 Application only in that it uses nonstandard facilities
137 that are consistent with this part of ISO/IEC 9945. Such an application shall fully
138 document its requirements for these extended facilities, in addition to the docu-
139 mentation required of a Conforming POSIX.1 Application. A Conforming POSIX.1
140 Application Using Extensions shall be either an ISO/IEC Conforming POSIX.1
141 Application Using Extensions or a <National Body> Conforming POSIX.1 Applica-
142 tion Using Extensions (see 1.3.2.2.1 and 1.3.2.2.2).

143 1.3.3 Language-Dependent Services for the C Programming Language

144 Parts of ISO/IEC 9899 {2} (hereinafter referred to as the “C Standard {2}”) will be |
145 referenced to describe requirements also mandated by this part of ISO/IEC 9945.
146 The sections of the C Standard {2} referenced to describe requirements for this
147 part of ISO/IEC 9945 are specified in Section 8. Section 8 also sets forth additions
148 and amplifications to the referenced sections of the C Standard {2}. Any imple-
149 mentation claiming conformance to this part of ISO/IEC 9945 with the C Language
150 Binding shall provide the facilities referenced in Section 8, along with any addi-
151 tions and amplifications Section 8 requires.

152 Although this part of ISO/IEC 9945 references parts of the C Standard {2} to
153 describe some of its own requirements, conformance to the C Standard {2} is
154 unnecessary for conformance to this part of ISO/IEC 9945. Any C language imple-
155 mentation providing the facilities stipulated in Section 8 may claim conformance; |
156 however, it shall clearly state that its C language does not conform to the |
157 C Standard {2}.

158 1.3.3.1 Types of Conformance

159 Implementations claiming conformance to this part of ISO/IEC 9945 with the C
160 Language Binding shall claim one of two types of conformance—conformance to |
161 POSIX.1, C Language Binding (C Standard Language-Dependent System Sup- |
162 port), or to POSIX.1, C Language Binding (Common-Usage C Language-Dependent |
163 System Support).

164 1.3.3.2 C Standard Language-Dependent System Support

165 Implementors shall meet the requirements of Section 8 using for reference the
166 C Standard {2}. Implementors shall clearly document the version of the
167 C Standard {2} referenced in fulfilling the requirements of Section 8.

Implementors seeking to claim conformance using the C Standard {2} shall claim conformance to POSIX.1, C Language Binding (C Standard Language-Dependent System Support).

1.3.3.3 Common-Usage C Language-Dependent System Support

Implementors, instead of referencing the C Standard {2}, shall provide the routines and support required in Section 8 using common usage as guidance. Implementors shall meet all the requirements of Section 8 except where references are made to the C Standard {2}. In places where the C Standard {2} is referenced, implementors shall provide equivalent support in a manner consistent with common usage of the C programming language. Implementors shall document, in Section 8 of the conformance document, all differences between the interface provided and the interface that would have been provided had the C Standard {2} been implemented instead of common usage. Implementors shall clearly document the version of the C Standard {2} referenced in documenting interface differences and should issue updates on differences for all new versions of the C Standard {2}.

Where a function has been introduced by the C Standard {2}, and thus there is no common-usage referent for it, if the function is implemented, it shall be implemented as described in the C Standard {2}. If the function is not implemented, it shall be documented as a difference from the C Standard {2} as required above.

1.3.4 Other C Language-Related Specifications

The following rules apply to the usage of C language library functions; each of the statements in this subclause applies to the detailed function descriptions in Sections 3 through 9, unless explicitly stated otherwise:

- (1) If an argument to a function has an invalid value (such as a value outside the domain of the function, or a pointer outside the address space of the program, or a NULL pointer when that is not explicitly permitted), the behavior is undefined.
- (2) Any function may also be implemented as a macro in a header. Applications should use #undef to remove any macro definition and ensure that an actual function is referenced. Applications should also use #undef prior to declaring any function in this part of ISO/IEC 9945.
- (3) Any invocation of a library function that is implemented as a macro shall expand to code that evaluates each of its arguments only once, fully protected by parentheses where necessary, so it is generally safe to use arbitrary expressions as arguments.
- (4) Provided that a library function can be declared without reference to any type defined in a header, it is also permissible to declare the function, either explicitly or implicitly, and use it without including its associated header.

- 209 (5) If a function that accepts a variable number of arguments is not declared
210 (explicitly or by including its associated header), the behavior is
211 undefined.

212 **1.3.5 Other Language-Related Specifications**

213 This part of ISO/IEC 9945 is currently specified in terms of the language defined
214 by the C Standard {2}. Bindings to other programming languages are being
215 developed.

216 If conformance to this part of ISO/IEC 9945 is claimed for implementation of any
217 programming language, the implementation of that language shall support the
218 use of external symbols distinct to at least 31 bytes in length in the source pro-
219 gram text. (That is, identifiers that differ at or before the thirty-first byte shall be
220 distinct.) If a national or international standard governing a language defines a
221 maximum length that is less than this value, the language-defined maximum
222 shall be supported. External symbols that differ only by case shall be distinct
223 when the character set in use distinguishes upper- and lowercase characters and
224 the language permits (or requires) upper- and lowercase characters to be distinct
225 in external symbols.

226 Subsequent sections of this part of ISO/IEC 9945 refer only to the C Language.

Section 2: Terminology and General Requirements

2.1 Conventions

This part of ISO/IEC 9945 uses the following typographic conventions:

(1) The *italic* font is used for:

- Cross references to defined terms within 1.3, 2.2.1, and 2.2.2; symbolic parameters that are generally substituted with real values by the application
- C language data types and function names (except in function Synopsis subclauses)
- Global external variable names

(2) The **bold** font is used with a word in all capital letters, such as

PATH

to represent an environment variable, as described in 2.6. It is also used for the term “**NULL** pointer.”

(3) The constant-width (Courier) font is used:

- For C language data types and function names within function Synopsis subclauses
- To illustrate examples of system input or output where exact usage is depicted
- For references to utility names and C language headers

(4) Symbolic constants returned by many functions as error numbers are represented as:

[ERRNO]

See 2.4.

(5) Symbolic constants or limits defined in certain headers are represented as:

{LIMIT}

See 2.8 and 2.9.

In some cases tabular information is presented “inline”; in others it is presented in a separately labeled table. This arrangement was employed purely for ease of typesetting and there is no normative difference between these two cases.

The conventions listed previously are for ease of reading only. Editorial inconsistencies in the use of typography are unintentional and have no normative meaning in this part of ISO/IEC 9945.

NOTEs provided as parts of labeled tables and figures are integral parts of this part of ISO/IEC 9945 (normative). Footnotes and notes within the body of the text are for information only (informative).

Numerical quantities are presented in international style: comma is used as a decimal sign and units are from the International System (SI).

2.2 Definitions

2.2.1 Terminology

For the purposes of this part of ISO/IEC 9945, the following definitions apply:

2.2.1.1 conformance document: A document provided by an implementor that contains implementation details as described in 1.3.1.2.

2.2.1.2 implementation defined: An indication that the implementation shall define and document the requirements for correct program constructs and correct data of a value or behavior.

2.2.1.3 may: An indication of an optional feature.

With respect to implementations, the word *may* is to be interpreted as an optional feature that is not required in this part of ISO/IEC 9945, but can be provided. With respect to Strictly Conforming POSIX.1 Applications, the word *may* means that the optional feature shall not be used.

2.2.1.4 obsolescent: An indication that a certain feature may be considered for withdrawal in future revisions of this part of ISO/IEC 9945.

Obsolescent features are retained in this version because of their widespread use. Their use in new applications is discouraged.

2.2.1.5 shall: An indication of a requirement on the implementation or on Strictly Conforming POSIX.1 Applications, where appropriate.

2.2.1.6 should:

- (1) With respect to implementations, an indication of an implementation recommendation, but not a requirement.
- (2) With respect to applications, an indication of a recommended programming practice for applications and a requirement for Strictly Conforming POSIX.1 Applications.

64 **2.2.1.7 supported:** A condition regarding optional functionality. |

65 Certain functionality in this part of ISO/IEC 9945 is optional, but the interfaces to
66 that functionality are always required. If the functionality is *supported*, the
67 interfaces work as specified by this part of ISO/IEC 9945 (except that they do not
68 return the error condition indicated for the unsupported case). If the functional-
69 ity is not *supported*, the interface shall always return the indication specified for
70 this situation.

71 **2.2.1.8 system documentation:** All documentation provided with an imple- |
72 mentation, except the conformance document. |

73 Electronically distributed documents for an implementation are considered part of |
74 the system documentation. |

75 **2.2.1.9 undefined:** An indication that this part of ISO/IEC 9945 imposes no por- |
76 tability requirements on an application's use of an indeterminate value or its |
77 behavior with erroneous program constructs or erroneous data. |

78 Implementations (or other standards) may specify the result of using that value or
79 causing that behavior. An application using such behaviors is using extensions,
80 as defined in 1.3.2.3.

81 **2.2.1.10 unspecified:** An indication that this part of ISO/IEC 9945 imposes no |
82 portability requirements on applications for correct program constructs or correct |
83 data regarding a value or behavior. |

84 Implementations (or other standards) may specify the result of using that value or
85 causing that behavior. An application requiring a specific behavior, rather than
86 tolerating any behavior when using that functionality, is using extensions, as
87 defined in 1.3.2.3.

88 **2.2.2 General Terms**

89 For the purposes of this part of ISO/IEC 9945, the following definitions apply: |

90 **2.2.2.1 absolute pathname:** See *pathname resolution* in 2.3.6.

91 **2.2.2.2 access mode:** A form of access permitted to a file.

92 **2.2.2.3 address space:** The memory locations that can be referenced by a
93 process.

94 **2.2.2.4 appropriate privileges:** An implementation-defined means of associat-
95 ing privileges with a process with regard to the function calls and function call
96 options defined in this part of ISO/IEC 9945 that need special privileges.

97 There may be zero or more such means.

2.2.2.5 background process: A process that is a member of a background process group.

2.2.2.6 background process group: Any process group, other than a foreground process group, that is a member of a session that has established a connection with a controlling terminal.

2.2.2.7 block special file: A file that refers to a device.

A block special file is normally distinguished from a character special file by providing access to the device in a manner such that the hardware characteristics of the device are not visible.

2.2.2.8 character: A sequence of one or more bytes representing a single graphic symbol.

NOTE: This term corresponds in the C Standard [2] to the term *multibyte character*, noting that a single-byte character is a special case of multibyte character. Unlike the usage in the C Standard [2], *character* here has no necessary relationship with storage space, and *byte* is used when storage space is discussed.

2.2.2.9 character special file: A file that refers to a device.

One specific type of character special file is a terminal device file, whose access is defined in 7.1. Other character special files have no structure defined by this part of ISO/IEC 9945, and their use is unspecified by this part of ISO/IEC 9945.

2.2.2.10 child process: See *process* in 2.2.2.62.

2.2.2.11 clock tick: An interval of time.

A number of these occur each second. Clock ticks are one of the units that may be used to express a value found in type *clock_t*.

2.2.2.12 controlling process: The session leader that established the connection to the controlling terminal.

Should the terminal subsequently cease to be a controlling terminal for this session, the session leader shall cease to be the controlling process.

2.2.2.13 controlling terminal: A terminal that is associated with a session.

Each session may have at most one controlling terminal associated with it, and a controlling terminal is associated with exactly one session. Certain input sequences from the controlling terminal (see 7.1) cause signals to be sent to all processes in the process group associated with the controlling terminal.

2.2.2.14 current working directory: See *working directory* in 2.2.2.89.

131 **2.2.2.15 device:** A computer peripheral or an object that appears to the applica-
132 tion as such.

133 **2.2.2.16 directory:** A file that contains directory entries.

134 No two directory entries in the same directory shall have the same name.

135 **2.2.2.17 directory entry [link]:** An object that associates a filename with a file.

136 Several directory entries can associate names with the same file.

137 **2.2.2.18 dot:** The filename consisting of a single dot character (.).

138 See *pathname resolution* in 2.3.6.

139 **2.2.2.19 dot-dot:** The filename consisting solely of two dot characters (.).

140 See *pathname resolution* in 2.3.6.

141 **2.2.2.20 effective group ID:** An attribute of a process that is used in determin-
142 ing various permissions, including file access permissions, described in 2.3.2.

143 See *group ID*. This value is subject to change during the process lifetime, as
144 described in 3.1.2 and 4.2.2.

145 **2.2.2.21 effective user ID:** An attribute of a process that is used in determining
146 various permissions, including file access permissions.

147 See *user ID*. This value is subject to change during the process lifetime, as
148 described in 3.1.2 and 4.2.2.

149 **2.2.2.22 empty directory:** A directory that contains, at most, directory entries
150 for dot and dot-dot.

151 **2.2.2.23 empty string [null string]:** A character array whose first element is a
152 null character.

153 **2.2.2.24 Epoch:** The time 0 hours, 0 minutes, 0 seconds, January 1, 1970, Coor-
154 dinated Universal Time.

155 See *seconds since the Epoch*.

156 **2.2.2.25 feature test macro:** A #defined symbol used to determine whether a
157 particular set of features will be included from a header.

158 See 2.7.1.

159 **2.2.2.26 FIFO special file [FIFO]:** A type of file with the property that data |
160 written to such a file is read on a first-in-first-out basis. |

Other characteristics of *FIFOs* are described in 5.3.1, 6.4.1, 6.4.2, and 6.5.3.

2.2.2.27 file: An object that can be written to, or read from, or both.

A file has certain attributes, including access permissions and type. File types include regular file, character special file, block special file, FIFO special file, and directory. Other types of files may be defined by the implementation.

2.2.2.28 file description: See *open file description* in 2.2.2.51.

2.2.2.29 file descriptor: A per-process unique, nonnegative integer used to identify an open file for the purpose of file access.

2.2.2.30 file group class: The property of a file indicating access permissions for a process related to the process's group identification.

A process is in the file group class of a file if the process is not in the file owner class and if the effective group ID or one of the supplementary group IDs of the process matches the group ID associated with the file. Other members of the class may be implementation defined.

2.2.2.31 file mode: An object containing the file permission bits and other characteristics of a file, as described in 5.6.1.

2.2.2.32 filename: A name consisting of 1 to {NAME_MAX} bytes used to name a file.

The characters composing the name may be selected from the set of all character values excluding the slash character and the null character. The filenames dot and dot-dot have special meaning; see *pathname resolution* in 2.3.6. A filename is sometimes referred to as a pathname component.

2.2.2.33 file offset: The byte position in the file where the next I/O operation begins.

Each open file description associated with a regular file, block special file, or directory has a file offset. A character special file that does not refer to a terminal device may have a file offset. There is no file offset specified for a pipe or FIFO.

2.2.2.34 file other class: The property of a file indicating access permissions for a process related to the process's user and group identification.

A process is in the file other class of a file if the process is not in the file owner class or file group class.

2.2.2.35 file owner class: The property of a file indicating access permissions for a process related to the process's user identification.

A process is in the file owner class of a file if the effective user ID of the process matches the user ID of the file.

196 **2.2.2.36 file permission bits:** Information about a file that is used, along with
197 other information, to determine if a process has read, write, or execute/search per-
198 mission to a file.

199 The bits are divided into three parts: owner, group, and other. Each part is used
200 with the corresponding file class of processes. These bits are contained in the file
201 mode, as described in 5.6.1. The detailed usage of the file permission bits in
202 access decisions is described in *file access permissions* in 2.3.2.

203 **2.2.2.37 file serial number:** A per-file system unique identifier for a file.

204 File serial numbers are unique throughout a file system.

205 **2.2.2.38 file system:** A collection of files and certain of their attributes.

206 It provides a name space for file serial numbers referring to those files.

207 **2.2.2.39 foreground process:** A process that is a member of a foreground pro-
208 cess group.

209 **2.2.2.40 foreground process group:** A process group whose member processes
210 have certain privileges, denied to processes in background process groups, when
211 accessing their controlling terminal.

212 Each session that has established a connection with a controlling terminal has
213 exactly one process group of the session as the foreground process group of that
214 controlling terminal. See 7.1.1.4.

215 **2.2.2.41 foreground process group ID:** The process group ID of the foreground
216 process group.

217 **2.2.2.42 group ID:** A nonnegative integer, which can be contained in an object of
218 type *gid_t*, that is used to identify a group of system users.

219 Each system user is a member of at least one group. When the identity of a group
220 is associated with a process, a group ID value is referred to as a real group ID, an
221 effective group ID, one of the (optional) supplementary group IDs, or an (optional)
222 saved set-group-ID.

223 **2.2.2.43 job control:** A facility that allows users to selectively stop (suspend)
224 the execution of processes and continue (resume) their execution at a later point.

225 The user typically employs this facility via the interactive interface jointly sup-
226 plied by the terminal I/O driver and a command interpreter. Conforming imple-
227 mentations may optionally support job control facilities; the presence of this
228 option is indicated to the application at compile time or run time by the definition
229 of the `{_POSIX_JOB_CONTROL}` symbol; see 2.9.

230 **2.2.2.44 link:** See *directory entry* in 2.2.2.17.

231 **2.2.2.45 link count:** The number of directory entries that refer to a particular
232 file.

233 **2.2.2.46 login:** The unspecified activity by which a user gains access to the
234 system.

235 Each login shall be associated with exactly one login name.

236 **2.2.2.47 login name:** A user name that is associated with a login.

237 **2.2.2.48 mode:** A collection of attributes that specifies a file's type and its access
238 permissions.

239 See *file access permissions* in 2.3.2.

240 **2.2.2.49 null string:** See *empty string* in 2.2.2.23.

241 **2.2.2.50 open file:** A file that is currently associated with a file descriptor.

242 **2.2.2.51 open file description:** A record of how a process or group of processes
243 are accessing a file.

244 Each file descriptor shall refer to exactly one open file description, but an open file
245 description may be referred to by more than one file descriptor. A file offset, file
246 status (see Table 6-5), and file access modes (see Table 6-6) are attributes of an
247 open file description.

248 **2.2.2.52 orphaned process group:** A process group in which the parent of
249 every member is either itself a member of the group or is not a member of the
250 group's session.

251 **2.2.2.53 parent directory:**

252 (1) When discussing a given directory, the directory that both contains a
253 directory entry for the given directory and is represented by the path-
254 name dot-dot in the given directory.

255 (2) When discussing other types of files, a directory containing a directory
256 entry for the file under discussion.

257 This concept does not apply to dot and dot-dot.

258 **2.2.2.54 parent process:** See *process* in 2.2.2.62.

259 **2.2.2.55 parent process ID:** An attribute of a new process after it is created by
260 a currently active process.

261 The parent process ID of a process is the process ID of its creator, for the lifetime
262 of the creator. After the creator's lifetime has ended, the parent process ID is the
263 process ID of an implementation-defined system process.

264 **2.2.2.56 path prefix:** A pathname, with an optional ending slash, that refers to
265 a directory.

266 **2.2.2.57 pathname:** A string that is used to identify a file.

267 A pathname consists of, at most, {PATH_MAX} bytes, including the terminating
268 null character. It has an optional beginning slash, followed by zero or more
269 filenames separated by slashes. If the pathname refers to a directory, it may also
270 have one or more trailing slashes. Multiple successive slashes are considered to
271 be the same as one slash. A pathname that begins with two successive slashes
272 may be interpreted in an implementation-defined manner, although more than
273 two leading slashes shall be treated as a single slash. The interpretation of the
274 pathname is described in 2.3.6.

275 **2.2.2.58 pathname component:** See *filename* in 2.2.2.32.

276 **2.2.2.59 pipe:** An object accessed by one of the pair of file descriptors created by
277 the *pipe()* function.

278 Once created, the file descriptors can be used to manipulate it, and it behaves
279 identically to a FIFO special file when accessed in this way. It has no name in the
280 file hierarchy.

281 **2.2.2.60 portable filename character set:** The set of characters from which
282 portable filenames are constructed.

283 For a filename to be portable across conforming implementations of this part of
284 ISO/IEC 9945, it shall consist only of the following characters:

285	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
286	a b c d e f g h i j k l m n o p q r s t u v w x y z
287	0 1 2 3 4 5 6 7 8 9 . _ -

288 The last three characters are the period, underscore, and hyphen characters,
289 respectively. The hyphen shall not be used as the first character of a portable
290 filename. Upper- and lowercase letters shall retain their unique identities
291 between conforming implementations. In the case of a portable pathname, the
292 slash character may also be used.

293 **2.2.2.61 privilege:** See *appropriate privileges* in 2.2.2.4.

294 **2.2.2.62 process:** An address space and single thread of control that executes
295 within that address space, and its required system resources.

296 A process is created by another process issuing the *fork()* function. The process
297 that issues *fork()* is known as the parent process, and the new process created by
298 the *fork()* is known as the child process.

299 **2.2.2.63 process group:** A collection of processes that permits the signaling of
300 related processes.

Each process in the system is a member of a process group that is identified by a process group ID. A newly created process joins the process group of its creator.

2.2.2.64 process group ID: The unique identifier representing a process group during its lifetime.

A process group ID is a positive integer that can be contained in a *pid_t*. It shall not be reused by the system until the process group lifetime ends.

2.2.2.65 process group leader: A process whose process ID is the same as its process group ID.

2.2.2.66 process group lifetime: A period of time that begins when a process group is created and ends when the last remaining process in the group leaves the group, due either to the end of the last process's process lifetime or to the last remaining process calling the *setsid()* or *setpgid()* functions.

2.2.2.67 process ID: The unique identifier representing a process.

A process ID is a positive integer that can be contained in a *pid_t*. A process ID shall not be reused by the system until the process lifetime ends. In addition, if there exists a process group whose process group ID is equal to that process ID, the process ID shall not be reused by the system until the process group lifetime ends. A process that is not a system process shall not have a process ID of 1.

2.2.2.68 process lifetime: The period of time that begins when a process is created and ends when its process ID is returned to the system.

After a process is created with a *fork()* function, it is considered active. Its thread of control and address space exist until it terminates. It then enters an inactive state where certain resources may be returned to the system, although some resources, such as the process ID, are still in use. When another process executes a *wait()* or *waitpid()* function for an inactive process, the remaining resources are returned to the system. The last resource to be returned to the system is the process ID. At this time, the lifetime of the process ends.

2.2.2.69 read-only file system: A file system that has implementation-defined characteristics restricting modifications.

2.2.2.70 real group ID: The attribute of a process that, at the time of process creation, identifies the group of the user who created the process.

See *group ID* in 2.2.2.42. This value is subject to change during the process lifetime, as described in 4.2.2.

2.2.2.71 real user ID: The attribute of a process that, at the time of process creation, identifies the user who created the process.

See *user ID* in 2.2.2.87. This value is subject to change during the process lifetime, as described in 4.2.2.

338 **2.2.2.72 regular file:** A file that is a randomly accessible sequence of bytes, with
339 no further structure imposed by the system.

340 **2.2.2.73 relative pathname:** See *pathname resolution* in 2.3.6.

341 **2.2.2.74 root directory:** A directory, associated with a process, that is used in
342 pathname resolution for pathnames that begin with a slash.

343 **2.2.2.75 saved set-group-ID:** An attribute of a process that allows some flexibil-
344 ity in the assignment of the effective group ID attribute, when the saved set-user-
345 ID option is implemented, as described in 3.1.2 and 4.2.2.

346 **2.2.2.76 saved set-user-ID:** An attribute of a process that allows some flexibil-
347 ity in the assignment of the effective user ID attribute, when the saved set-user-ID
348 option is implemented, as described in 3.1.2 and 4.2.2.

349 **2.2.2.77 seconds since the Epoch:** A value to be interpreted as the number of
350 seconds between a specified time and the Epoch.

351 A Coordinated Universal Time name (specified in terms of seconds (*tm_sec*),
352 minutes (*tm_min*), hours (*tm_hour*), days since January 1 of the year (*tm_yday*),
353 and calendar year minus 1900 (*tm_year*) is related to a time represented as
354 seconds since the Epoch, according to the expression below.

355 If the year < 1970 or the value is negative, the relationship is undefined. If the
356 year ≥ 1970 and the value is nonnegative, the value is related to a Coordinated
357 Universal Time name according to the expression:

358
$$tm_sec + tm_min*60 + tm_hour*3\,600 + tm_yday*86\,400 +$$

359
$$(tm_year-70)*31\,536\,000 + ((tm_year-69)/4)*86\,400$$

360 **2.2.2.78 session:** A collection of process groups established for job control
361 purposes.

362 Each process group is a member of a session. A process is considered to be a
363 member of the session of which its process group is a member. A newly created
364 process joins the session of its creator. A process can alter its session membership
365 (see 4.3.2). Implementations that support the *setpgid()* function (see 4.3.3) can
366 have multiple process groups in the same session.

367 **2.2.2.79 session leader:** A process that has created a session (see 4.3.2).

368 **2.2.2.80 session lifetime:** The period between when a session is created and the
369 end of the lifetime of all the process groups that remain as members of the
370 session.

371 **2.2.2.81 signal:** A mechanism by which a process may be notified of, or affected
372 by, an event occurring in the system.

Examples of such events include hardware exceptions and specific actions by processes. The term *signal* is also used to refer to the event itself.

2.2.2.82 slash: The literal character “/”.

This character is also known as *solidus* in ISO 8859-1 [B34].

2.2.2.83 supplementary group ID: An attribute of a process used in determining file access permissions.

A process has up to {NGROUPS_MAX} supplementary group IDs in addition to the effective group ID. The supplementary group IDs of a process are set to the supplementary group IDs of the parent process when the process is created. Whether a process's effective group ID is included in or omitted from its list of supplementary group IDs is unspecified.

2.2.2.84 system: An implementation of this part of ISO/IEC 9945.

2.2.2.85 system process: An object, other than a process executing an application, that is defined by the system and has a process ID.

2.2.2.86 terminal [terminal device]: A character special file that obeys the specifications of 7.1.

2.2.2.87 user ID: A nonnegative integer, which can be contained in an object of type *uid_t*, that is used to identify a system user.

When the identity of a user is associated with a process, a user ID value is referred to as a real user ID, an effective user ID, or an (optional) saved set-user-ID.

2.2.2.88 user name: A string that is used to identify a user, as described in 9.1.

2.2.2.89 working directory [current working directory]: A directory, associated with a process, that is used in pathname resolution for pathnames that do not begin with a slash.

2.2.3 Abbreviations

For the purposes of this part of ISO/IEC 9945, the following abbreviations apply:

2.2.3.1 C Standard: ISO/IEC 9899, *Information technology—Programming languages—C* [2].

2.2.3.2 IRV: The International Reference Version coded character set described in ISO/IEC 646 [1].