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

AVA Documents
Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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

SH13630

Documents
Document Sharing Hub

Full document details: ISO_IEC 9945-1_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.

Documents
Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

Documents
Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

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.

AVA Documents
Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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

Documents
Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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

Documents
Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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 ***setlocale***() 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 ***remove***() 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

Documents
Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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 ***access***() Function 38

Table 2-9 – Symbolic Constants for the ***lseek***() 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 – ***uname***() Structure Members 75

Table 4-2 – Configurable System Variables 80

Documents
Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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

AVA Documents
Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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

Documents
Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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 UNIX[1] 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.

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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.

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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.

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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).

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

(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 part of ISO/IEC 9945, but some results of it are mentioned in Annex B.

Minimal Changes to Historical Implementations

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

(1) By standardizing an interface like one in an historical implementation; e.g., directories, or;

(2) By specifying an interface that is readily implementable in terms of, and backwards compatible with, historical implementations, such as the extended `tar` format in 10.1.1, or;

(3) By specifying an interface that, when added to an historical implementation, will not conflict with it, like B.6.

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

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

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 POSIX.1. These need to be used with careful management to be able to adapt to future POSIX.1 extensions and/or port to implementations that provide these services in a different manner.

Minimal Changes to Existing Application Code

A goal of POSIX.1 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. Annex B points out the major places where POSIX.1 implies such changes.

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

Related Standards Activities

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

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

(1) Language-independent service descriptions of this part of ISO/IEC 9945

(2) C, Ada, and FORTRAN Language bindings to (1)

(3) Shell and Utility facilities

(4) Verification testing methods

(5) Realtime facilities

(6) Secure/Trusted System considerations

(7) Network interface facilities

(8) System Administration

(9) Graphical User Interfaces

(10) Profiles describing application- or user-specific combinations of Open Systems standards for: supercomputing, multiprocessor, and batch extensions; transaction processing; realtime systems; and multiuser systems based on historical models

(11) An overall guide to POSIX-based or related Open Systems standards and profiles

Extensions are approved as "amendments" or "revisions" to this document, following the IEEE and ISO/IEC Procedures.

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

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

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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

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

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

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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. Whittemore
Donald W. Zipse

*Member Emeritus

Documents Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

Documents
Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

(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 Varembé, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse.

2) To be approved and published.

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

(1) The system shall support all required interfaces defined within this part of ISO/IEC 9945. These interfaces shall support the functional behavior described herein.

(2) The system may provide additional functions or facilities not required by this part of ISO/IEC 9945. Nonstandard extensions should be identified as such in the system documentation. Nonstandard extensions, when used, may change the behavior of functions or facilities defined by this part of ISO/IEC 9945. The conformance document shall define an environment in which an application can be run with the behavior specified by the standard. In no case shall such an environment require modification of a Strictly Conforming POSIX.1 Application.

1.3.1.2 Documentation

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

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

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

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

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

The phrases "shall document" or "shall be documented" in this part of ISO/IEC 9945 mean that documentation of the feature shall appear in the conformance document, as described previously, unless the system documentation is explicitly mentioned.

The system documentation should also contain the information found in the conformance document.

AVA Documents
Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

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.

AVA Documents Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

1.3.2.2.2 <National Body> Conforming POSIX.1 Application

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

1.3.2.3 Conforming POSIX.1 Application Using Extensions

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

1.3.3 Language-Dependent Services for the C Programming Language

Parts of ISO/IEC 9899 {2} (hereinafter referred to as the "C Standard {2}") will be referenced to describe requirements also mandated by this part of ISO/IEC 9945. The sections of the C Standard {2} referenced to describe requirements for this part of ISO/IEC 9945 are specified in Section 8. Section 8 also sets forth additions and amplifications to the referenced sections of the C Standard {2}. Any implementation claiming conformance to this part of ISO/IEC 9945 with the C Language Binding shall provide the facilities referenced in Section 8, along with any additions and amplifications Section 8 requires.

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

1.3.3.1 Types of Conformance

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

1.3.3.2 C Standard Language-Dependent System Support

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

Documents
Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

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.

AVA Documents Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

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

1.3.5 Other Language-Related Specifications

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

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

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

AVA Documents
Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

Documents
Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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.

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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.

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

2.2.1.7 supported: A condition regarding optional functionality.

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

2.2.1.8 system documentation: All documentation provided with an implementation, except the conformance document.

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

2.2.1.9 undefined: An indication that this part of ISO/IEC 9945 imposes no portability requirements on an application's use of an indeterminate value or its behavior with erroneous program constructs or erroneous data.

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

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

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

2.2.2 General Terms

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

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

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

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

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

There may be zero or more such means.

AVA Documents
Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

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.

Documents
Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

2.2.2.15 device: A computer peripheral or an object that appears to the application as such.

2.2.2.16 directory: A file that contains directory entries.

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

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

Several directory entries can associate names with the same file.

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

See *pathname resolution* in 2.3.6.

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

See *pathname resolution* in 2.3.6.

2.2.2.20 effective group ID: An attribute of a process that is used in determining various permissions, including file access permissions, described in 2.3.2.

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

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

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

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

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

2.2.2.24 Epoch: The time 0 hours, 0 minutes, 0 seconds, January 1, 1970, Coordinated Universal Time.

See *seconds since the Epoch*.

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

See 2.7.1.

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

Documents
Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

Other characteristics of *FIFO*s 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.

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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

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

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

File serial numbers are unique throughout a file system.

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

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

2.2.2.39 foreground process: A process that is a member of a foreground process group.

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

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

2.2.2.41 foreground process group ID: The process group ID of the foreground process group.

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

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

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

The user typically employs this facility via the interactive interface jointly supplied by the terminal I/O driver and a command interpreter. Conforming implementations may optionally support job control facilities; the presence of this option is indicated to the application at compile time or run time by the definition of the {_POSIX_JOB_CONTROL} symbol; see 2.9.

2.2.2.44 link: See *directory entry* in 2.2.2.17.

AVA Documents
Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

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

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

Each login shall be associated with exactly one login name.

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

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

See *file access permissions* in 2.3.2.

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

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

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

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

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

2.2.2.53 parent directory:

(1) When discussing a given directory, the directory that both contains a directory entry for the given directory and is represented by the pathname dot-dot in the given directory.

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

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

2.2.2.54 parent process: See *process* in 2.2.2.62.

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

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

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

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

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

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

2.2.2.58 pathname component: See *filename* in 2.2.2.32.

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

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

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

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

```
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
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
0 1 2 3 4 5 6 7 8 9 . _ -
```

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

2.2.2.61 privilege: See *appropriate privileges* in 2.2.2.4.

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

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

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

AVA Documents Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

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.

Documents
Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

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

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

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

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

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

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

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

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

$$tm\_sec + tm\_min*60 + tm\_hour*3\,600 + tm\_yday*86\,400 + (tm\_year-70)*31\,536\,000 + ((tm\_year-69)/4)*86\,400$$

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

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

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

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

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

AVA Documents
Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

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}.

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

404 **2.2.3.3 POSIX.1:** This part of ISO/IEC 9945.

405 2.3 General Concepts

406 **2.3.1 extended security controls:** The access control (see *file access permis-*
407 *sions*) and privilege (see *appropriate privileges* in 2.2.2.4) mechanisms have been
408 defined to allow implementation-defined extended security controls. These permit
409 an implementation to provide security mechanisms to implement different secu-
410 rity policies than described in this part of ISO/IEC 9945. These mechanisms shall
411 not alter or override the defined semantics of any of the functions in this part of
412 ISO/IEC 9945.

413 **2.3.2 file access permissions:** The standard file access control mechanism uses
414 the file permission bits, as described below. These bits are set at file creation by
415 *open*(), *creat*(), *mkdir*(), and *mkfifo*() and are changed by *chmod*(). These bits are
416 read by *stat*() or *fstat*().

417 Implementations may provide *additional* or *alternate* file access control mechan-
418 isms, or both. An additional access control mechanism shall only further restrict
419 the access permissions defined by the file permission bits. An alternate access
420 control mechanism shall:

421 (1) Specify file permission bits for the file owner class, file group class, and
422 file other class of the file, corresponding to the access permissions, to be
423 returned by *stat*() or *fstat*().

424 (2) Be enabled only by explicit user action, on a per-file basis by the file
425 owner or a user with the appropriate privilege.

426 (3) Be disabled for a file after the file permission bits are changed for that
427 file with *chmod*(). The disabling of the alternate mechanism need not
428 disable any additional mechanisms defined by an implementation.

429 Whenever a process requests file access permission for read, write, or
430 execute/search, if no additional mechanism denies access, access is determined as
431 follows:

432 (1) If a process has the appropriate privilege:

433 (a) If read, write, or directory search permission is requested, access is
434 granted.

435 (b) If execute permission is requested, access is granted if execute per-
436 mission is granted to at least one user by the file permission bits or
437 by an alternate access control mechanism; otherwise, access is
438 denied.

439 (2) Otherwise:

440 (a) The file permission bits of a file contain read, write, and
441 execute/search permissions for the file owner class, file group class,
442 and file other class.

443 (b) Access is granted if an alternate access control mechanism is not
444 enabled and the requested access permission bit is set for the class

AVA Documents Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

(file owner class, file group class, or file other class) to which the process belongs, or if an alternate access control mechanism is enabled and it allows the requested access; otherwise, access is denied.

2.3.3 file hierarchy: Files in the system are organized in a hierarchical structure in which all of the nonterminal nodes are directories and all of the terminal nodes are any other type of file. Because multiple directory entries may refer to the same file, the hierarchy is properly described as a "directed graph."

2.3.4 filename portability: Filenames should be constructed from the portable filename character set because the use of other characters can be confusing or ambiguous in certain contexts.

2.3.5 file times update: Each file has three distinct associated time values: *st_atime*, *st_mtime*, and *st_ctime*. The *st_atime* field is associated with the times that the file data is accessed; *st_mtime* is associated with the times that the file data is modified; and *st_ctime* is associated with the times that file status is changed. These values are returned in the file characteristics structure, as described in 5.6.1.

Any function in this part of ISO/IEC 9945 that is required to read or write file data or change the file status indicates which of the appropriate time-related fields are to be "marked for update." If an implementation of such a function marks for update a time-related field not specified by this part of ISO/IEC 9945, this shall be documented, except that any changes caused by pathname resolution need not be documented. For the other functions in this part of ISO/IEC 9945 (those that are not explicitly required to read or write file data or change file status, but that in some implementations happen to do so), the effect is unspecified.

An implementation may update fields that are marked for update immediately, or it may update such fields periodically. When the fields are updated, they are set to the current time and the update marks are cleared. All fields that are marked for update shall be updated when the file is no longer open by any process, or when a *stat*() or *fstat*() is performed on the file. Other times at which updates are done are unspecified. Updates are not done for files on read-only file systems.

2.3.6 pathname resolution: Pathname resolution is performed for a process to resolve a pathname to a particular file in a file hierarchy. There may be multiple pathnames that resolve to the same file.

Each filename in the pathname is located in the directory specified by its predecessor (for example, in the pathname fragment "`a/b`", file "`b`" is located in directory "`a`"). Pathname resolution fails if this cannot be accomplished. If the pathname begins with a slash, the predecessor of the first filename in the pathname is taken to be the root directory of the process (such pathnames are referred to as absolute pathnames). If the pathname does not begin with a slash, the predecessor of the first filename of the pathname is taken to be the current working directory of the process (such pathnames are referred to as "relative pathnames").

The interpretation of a pathname component is dependent on the values of {NAME_MAX} and {_POSIX_NO_TRUNC} associated with the path prefix of that component. If any pathname component is longer than {NAME_MAX}, and

AVA Documents Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

{_POSIX_NO_TRUNC} is in effect for the path prefix of that component (see 5.7.1), the implementation shall consider this an error condition. Otherwise, the implementation shall use the first {NAME_MAX} bytes of the pathname component.

The special filename, dot, refers to the directory specified by its predecessor. The special filename, dot-dot, refers to the parent directory of its predecessor directory. As a special case, in the root directory, dot-dot may refer to the root directory itself.

A pathname consisting of a single slash resolves to the root directory of the process. A null pathname is invalid.

2.4 Error Numbers

Most functions provide an error number in the external variable *errno*, which is defined as:

```
extern int errno;
```

The value of this variable shall be defined only after a call to a function for which it is explicitly stated to be set and until it is changed by the next function call. The variable *errno* should only be examined when it is indicated to be valid by a function's return value. No function defined in this part of ISO/IEC 9945 sets *errno* to zero to indicate an error.

If more than one error occurs in processing a function call, this part of ISO/IEC 9945 does not define in what order the errors are detected; therefore, any one of the possible errors may be returned.

Implementations may support additional errors not included in this clause, may generate errors included in this clause under circumstances other than those described in this clause, or may contain extensions or limitations that prevent some errors from occurring. The Errors subclause in each function description specifies which error conditions shall be detected by all implementations and which may be optionally detected by an implementation. Each implementation shall document, in the conformance document, situations in which each of the optional conditions are detected. If no error condition is detected, the action requested shall be successful. Implementations may contain extensions or limitations that prevent some specified errors from occurring.

Implementations may generate error numbers listed in this clause under circumstances other than those described, if and only if all those error conditions can always be treated identically to the error conditions as described in this part of ISO/IEC 9945. Implementations may support additional errors not listed in this clause, but shall not generate a different error number from one required by this part of ISO/IEC 9945 for an error condition described in this part of ISO/IEC 9945.

The following symbolic names identify the possible error numbers, in the context of functions specifically defined in this part of ISO/IEC 9945; these general descriptions are more precisely defined in the Errors subclauses of functions that return them. Only these symbolic names should be used in programs, since the actual value of an error number is unspecified. All values listed in this clause shall be unique. The values for these names shall be found in the header <errno.h>.

Documents
Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

The actual values are unspecified by this part of ISO/IEC 9945.

[E2BIG] Arg list too long
The sum of the number of bytes used by the new process image's argument list and environment list was greater than the system-imposed limit of {ARG_MAX} bytes.

[EACCES] Permission denied
An attempt was made to access a file in a way forbidden by its file access permissions.

[EAGAIN] Resource temporarily unavailable
This is a temporary condition, and later calls to the same routine may complete normally.

[EBADF] Bad file descriptor
A file descriptor argument was out of range, referred to no open file, or a read (write) request was made to a file that was only open for writing (reading).

[EBUSY] Resource busy
An attempt was made to use a system resource that was not available at the time because it was being used by a process in a manner that would have conflicted with the request being made by this process.

[ECHILD] No child processes
A *wait*() or *waitpid*() function was executed by a process that had no existing or unwaited-for child processes.

[EDEADLK] Resource deadlock avoided
An attempt was made to lock a system resource that would have resulted in a deadlock situation.

[EDOM] Domain error
Defined in the C Standard {2}; an input argument was outside the defined domain of the mathematical function.

[EEXIST] File exists
An existing file was specified in an inappropriate context; for instance, as the new link name in a *link*() function.

[EFAULT] Bad address
The system detected an invalid address in attempting to use an argument of a call. The reliable detection of this error is implementation defined; however, implementations that do detect this condition shall use this value.

[EFBIG] File too large
The size of a file would exceed an implementation-defined maximum file size.

[EINTR] Interrupted function call
An asynchronous signal (such as SIGINT or SIGQUIT; see the description of header `<signal.h>` in 3.3.1) was caught by the process during the execution of an interruptible function. If the

AVA Documents Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

signal handler performs a normal return, the interrupted function call may return this error condition.

[EINVAL] Invalid argument
Some invalid argument was supplied. [For example, specifying an undefined signal to a *signal*() or *kill*() function].

[EIO] Input/output error
Some physical input or output error occurred. This error may be reported on a subsequent operation on the same file descriptor. Any other error-causing operation on the same file descriptor may cause the [EIO] error indication to be lost.

[EISDIR] Is a directory
An attempt was made to open a directory with write mode specified.

[EMFILE] Too many open files
An attempt was made to open more than the maximum number of {OPEN_MAX} file descriptors allowed in this process.

[EMLINK] Too many links
An attempt was made to have the link count of a single file exceed {LINK_MAX}.

[ENAMETOOLONG] Filename too long
The size of a pathname string exceeded {PATH_MAX}, or a pathname component was longer than {NAME_MAX} and {_POSIX_NO_TRUNC} was in effect for that file.

[ENFILE] Too many open files in system
Too many files are currently open in the system. The system reached its predefined limit for simultaneously open files and temporarily could not accept requests to open another one.

[ENODEV] No such device
An attempt was made to apply an inappropriate function to a device; for example, trying to read a write-only device such as a printer.

[ENOENT] No such file or directory
A component of a specified pathname did not exist, or the pathname was an empty string.

[ENOEXEC] Exec format error
A request was made to execute a file that, although it had the appropriate permissions, was not in the format required by the implementation for executable files.

[ENOLCK] No locks available
A system-imposed limit on the number of simultaneous file and record locks was reached, and no more were available at that time.

[ENOMEM] Not enough space
The new process image required more memory than was allowed

AVA Documents
Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

by the hardware or by system-imposed memory management constraints.

[ENOSPC] No space left on device
During a *write*() function on a regular file, or when extending a directory, there was no free space left on the device.

[ENOSYS] Function not implemented
An attempt was made to use a function that is not available in this implementation.

[ENOTDIR] Not a directory
A component of the specified pathname existed, but it was not a directory, when a directory was expected.

[ENOTEMPTY] Directory not empty
A directory with entries other than dot and dot-dot was supplied when an empty directory was expected.

[ENOTTY] Inappropriate I/O control operation
A control function was attempted for a file or a special file for which the operation was inappropriate.

[ENXIO] No such device or address
Input or output on a special file referred to a device that did not exist, or made a request beyond the limits of the device. This error may also occur when, for example, a tape drive is not online or a disk pack is not loaded on a drive.

[EPERM] Operation not permitted
An attempt was made to perform an operation limited to processes with appropriate privileges or to the owner of a file or other resource.

[EPIPE] Broken pipe
A write was attempted on a pipe or FIFO for which there was no process to read the data.

[ERANGE] Result too large
Defined in the C Standard {2}; the result of the function was too large to fit in the available space.

[EROFS] Read-only file system
An attempt was made to modify a file or directory on a file system that was read-only at that time.

[ESPIPE] Invalid seek
An *lseek*() function was issued on a pipe or FIFO.

[ESRCH] No such process
No process could be found corresponding to that specified by the given process ID.

[EXDEV] Improper link
A link to a file on another file system was attempted.

AVA Documents Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990

2.5 Primitive System Data Types

Some data types used by the various system functions are not defined as part of this part of ISO/IEC 9945, but are defined by the implementation. These types are then defined in the header <sys/types.h>, which contains definitions for at least the types shown in Table 2-1.

Table 2-1 – Primitive System Data Types

Defined Type	Description
dev_t	Used for device numbers.
gid_t	Used for group IDs.
ino_t	Used for file serial numbers.
mode_t	Used for some file attributes, for example file type, file access permissions.
nlink_t	Used for link counts.
off_t	Used for file sizes.
pid_t	Used for process IDs and process group IDs.
size_t	As defined in the C Standard {2}.
ssize_t	Used by functions that return a count of bytes (memory space) or an error indication.
uid_t	Used for user IDs.

All of the types listed in Table 2-1 shall be arithmetic types; *pid_t*, *ssize_t*, and *off_t* shall be signed arithmetic types. The type *ssize_t* shall be capable of storing values in the range from –1 to {SSIZE_MAX}, inclusive. The types *size_t* and *ssize_t* shall also be defined in the header <unistd.h>.

Additional unspecified type symbols ending in *_t* may be defined in any header specified by POSIX.1. The visibility of such symbols need not be controlled by any feature test macro other than _POSIX_SOURCE.

2.6 Environment Description

An array of strings called the *environment* is made available when a process begins. This array is pointed to by the external variable *environ*, which is defined as:

```
extern char **environ;
```

These strings have the form "*name*=*value*"; *name*s shall not contain the character '='. There is no meaning associated with the order of the strings in the environment. If more than one string in a process's environment has the same *name*, the consequences are undefined. The following names may be defined and have the indicated meaning if they are defined:

HOME The name of the user's initial working directory from the user database (see the description of the header <pwd.h> in 9.2.2).

Documents
Document Sharing Hub
Full document details: ISO_IEC 9945-1_1990

LANG The name of the locale to use for locale categories when both **LC_ALL** and the corresponding environment variable (beginning with "**LC_**") do not specify a locale.

LC_ALL The name of the locale to be used to override any values for locale categories specified by the setting of **LANG** or any environment variables beginning with "**LC_**".

LC_COLLATE The name of the locale for collation information.

LC_CTYPE The name of the locale for character classification.

LC_MONETARY The name of the locale containing monetary-related numeric editing information.

LC_NUMERIC The name of the locale containing numeric editing (i.e., radix character) information.

LC_TIME The name of the locale for date/time formatting information.

LOGNAME The login name associated with the current process. The value shall be composed of characters from the portable filename character set.

NOTE: An application that requires, or an installation that actually uses, characters outside the portable filename character set would not strictly conform to this part of ISO/IEC 9945. However, it is reasonable to expect that such characters would be used in many countries (recognizing the reduced level of interchange implied by this), and applications or installations should permit such usage where possible. No error is defined by this part of ISO/IEC 9945 for violation of this condition.

PATH The sequence of path prefixes that certain functions apply in searching for an executable file known only by a filename (a pathname that does not contain a slash). The prefixes are separated by a colon (:). When a nonzero-length prefix is applied to this filename, a slash is inserted between the prefix and the filename. A zero-length prefix is a special prefix that indicates the current working directory. It appears as two adjacent colons (::), as an initial colon preceding the rest of the list, or as a trailing colon following the rest of the list. The list is searched from beginning to end until an executable program by the specified name is found. If the pathname being sought contains a slash, the search through the path prefixes is not performed.

TERM The terminal type for which output is to be prepared. This information is used by commands and application programs wishing to exploit special capabilities specific to a terminal.

Documents
Document Sharing Hub

Full document details: ISO_IEC 9945-1_1990