Recognized as an
American National Standard (ANSI)

IEEE
Std 1003.1b-1993
(Formerly known as IEEE P1003.4)
(Includes IEEE Std 1003.1-1990)

IEEE Standard for Information Technology—Portable Operating Sytem Interface (POSIX®)—Part 1: System Application Program Interface (API)—Amendment 1: Realtime Extension [C Language]

Sponsor

Portable Applications Standards Committee
of the
IEEE Computer Society

Approved September 15, 1993

IEEE Standards Board

Approved April 14, 1994

American National Standards Institute

Abstract: This amendment 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. When options specified in the Realtime Extension are included, the standard also defines interfaces appropriate for realtime applications.

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

POSIX is a trademark of the Institute of Electrical and Electronics Engineers, Inc.

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Copyright © 1994 by the
Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1994.
Printed in the United States of America.

ISBN 1-55937-375-X

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.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.1b-1993

THIS PAGE WAS
BLANK IN THE ORIGINAL

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

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 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 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
USA

IEEE standards documents may involve the use of patented technology. Their approval by the Institute of Electrical and Electronics Engineers, Inc. does not mean that using such technology for the purpose of conforming to such standards is authorized by the patent owner. It is the obligation of the user of such technology to obtain all necessary permissions.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.1b-1993

Contents

PAGE

Introduction x
Organization of This Standard x
Base Documents xi
Background xi
Audience xii
Purpose xii
Related Standards Activities xvi

Section 1: General 1
1.1 Scope 1
1.2 Normative References 2
1.3 Conformance 3

Section 2: Terminology and General Requirements 9
2.1 Conventions 9
2.2 Definitions 10
2.3 General Concepts 25
2.4 Error Numbers 28
2.5 Primitive System Data Types 32
2.6 Environment Description 33
2.7 C Language Definitions 34
2.8 Numerical Limits 40
2.9 Symbolic Constants 45

Section 3: Process Primitives 49
3.1 Process Creation and Execution 49
3.1.1 Process Creation 49
3.1.2 Execute a File 51
3.2 Process Termination 55
3.2.1 Wait for Process Termination 56
3.2.2 Terminate a Process 58
3.3 Signals 60
3.3.1 Signal Concepts 60
3.3.2 Send a Signal to a Process 68
3.3.3 Manipulate Signal Sets 69
3.3.4 Examine and Change Signal Action 70
3.3.5 Examine and Change Blocked Signals 73
3.3.6 Examine Pending Signals 75
3.3.7 Wait for a Signal 75
3.3.8 Synchronously Accept a Signal 76
3.3.9 Queue a Signal to a Process 78
3.4 Timer Operations 79
3.4.1 Schedule Alarm 79

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

3.4.2 Suspend Process Execution 80
3.4.3 Delay Process Execution 81

Section 4: Process Environment 83
4.1 Process Identification 83
4.1.1 Get Process and Parent Process IDs 83
4.2 User Identification 84
4.2.1 Get Real User, Effective User, Real Group, and Effective Group IDs 84
4.2.2 Set User and Group IDs 84
4.2.3 Get Supplementary Group IDs 86
4.2.4 Get User Name 87
4.3 Process Groups 88
4.3.1 Get Process Group ID 88
4.3.2 Create Session and Set Process Group ID 88
4.3.3 Set Process Group ID for Job Control 89
4.4 System Identification 90
4.4.1 Get System Name 90
4.5 Time 91
4.5.1 Get System Time 91
4.5.2 Get Process Times 92
4.6 Environment Variables 93
4.6.1 Environment Access 93
4.7 Terminal Identification 94
4.7.1 Generate Terminal Pathname 94
4.7.2 Determine Terminal Device Name 95
4.8 Configurable System Variables 95
4.8.1 Get Configurable System Variables 95

Section 5: Files and Directories 99
5.1 Directories 99
5.1.1 Format of Directory Entries 99
5.1.2 Directory Operations 99
5.2 Working Directory 102
5.2.1 Change Current Working Directory 102
5.2.2 Get Working Directory Pathname 103
5.3 General File Creation 104
5.3.1 Open a File 104
5.3.2 Create a New File or Rewrite an Existing One 107
5.3.3 Set File Creation Mask 108
5.3.4 Link to a File 108
5.4 Special File Creation 110
5.4.1 Make a Directory 110
5.4.2 Make a FIFO Special File 111
5.5 File Removal 113
5.5.1 Remove Directory Entries 113
5.5.2 Remove a Directory 114
5.5.3 Rename a File 115
5.6 File Characteristics 117
5.6.1 File Characteristics: Header and Data Structure 117

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

5.6.2 Get File Status . . . 120
5.6.3 Check File Accessibility . . . 121
5.6.4 Change File Modes . . . 122
5.6.5 Change Owner and Group of a File . . . 124
5.6.6 Set File Access and Modification Times . . . 126
5.6.7 Truncate a File to a Specified Length . . . 127
5.7 Configurable Pathname Variables . . . 129
5.7.1 Get Configurable Pathname Variables . . . 129

Section 6: Input and Output Primitives . . . 133
6.1 Pipes . . . 133
6.1.1 Create an Inter-Process Channel . . . 133
6.2 File Descriptor Manipulation . . . 134
6.2.1 Duplicate an Open File Descriptor . . . 134
6.3 File Descriptor Deassignment . . . 135
6.3.1 Close a File . . . 135
6.4 Input and Output . . . 137
6.4.1 Read from a File . . . 137
6.4.2 Write to a File . . . 139
6.5 Control Operations on Files . . . 142
6.5.1 Data Definitions for File Control Operations . . . 142
6.5.2 File Control . . . 143
6.5.3 Reposition Read/Write File Offset . . . 147
6.6 File Synchronization . . . 149
6.6.1 Synchronize the State of a File . . . 149
6.6.2 Synchronize the Data of a File . . . 150
6.7 Asynchronous Input and Output . . . 151
6.7.1 Data Definitions for Asynchronous Input and Output . . . 151
6.7.2 Asynchronous Read . . . 154
6.7.3 Asynchronous Write . . . 156
6.7.4 List Directed I/O . . . 158
6.7.5 Retrieve Error Status of Asynchronous I/O Operation . . . 161
6.7.6 Retrieve Return Status of Asynchronous I/O Operation . . . 162
6.7.7 Cancel Asynchronous I/O Request . . . 163
6.7.8 Wait for Asynchronous I/O Request . . . 164
6.7.9 Asynchronous File Synchronization . . . 166

Section 7: Device- and Class-Specific Functions . . . 169
7.1 General Terminal Interface . . . 169
7.1.1 Interface Characteristics . . . 169
7.1.1.1 Opening a Terminal Device File . . . 169
7.1.1.2 Process Groups . . . 169
7.1.1.3 The Controlling Terminal . . . 170
7.1.1.4 Terminal Access Control . . . 170
7.1.1.5 Input Processing and Reading Data . . . 171
7.1.1.6 Canonical Mode Input Processing . . . 172
7.1.1.7 Noncanonical Mode Input Processing . . . 172
7.1.1.8 Writing Data and Output Processing . . . 173
7.1.1.9 Special Characters . . . 173
7.1.1.10 Modem Disconnect . . . 175

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

7.1.1.11 Closing a Terminal Device File 175
7.1.2 Parameters That Can Be Set 175
7.1.2.1 *termios* Structure 175
7.1.2.2 Input Modes 176
7.1.2.3 Output Modes 177
7.1.2.4 Control Modes 178
7.1.2.5 Local Modes 179
7.1.2.6 Special Control Characters 180
7.1.2.7 Baud Rate Values 181
7.1.3 Baud Rate Functions 181
7.1.3.1 Synopsis 181
7.1.3.2 Description 182
7.1.3.3 Returns 182
7.1.3.4 Errors 182
7.1.3.5 Cross-References 182
7.2 General Terminal Interface Control Functions 183
7.2.1 Get and Set State 183
7.2.2 Line Control Functions 185
7.2.3 Get Foreground Process Group ID 187
7.2.4 Set Foreground Process Group ID 188

Section 8: Language-Specific Services for the C Programming Language . 191
8.1 Referenced C Language Routines 191
8.1.1 Extensions to Time Functions 192
8.1.2 Extensions to *setlocale*() Function 194
8.2 C Language Input/Output Functions 195
8.2.1 Map a Stream Pointer to a File Descriptor 196
8.2.2 Open a Stream on a File Descriptor 197
8.2.3 Interactions of Other *FILE*-Type C Functions 198
8.2.4 Operations on Files — the *remove*() Function 202
8.3 Other C Language Functions 202
8.3.1 Nonlocal Jumps 202
8.3.2 Set Time Zone 202

Section 9: System Databases 205
9.1 System Databases 205
9.2 Database Access 206
9.2.1 Group Database Access 206
9.2.2 User Database Access 207

Section 10: Data Interchange Format 209
10.1 Archive/Interchange File Format 209
10.1.1 Extended `tar` Format 209
10.1.2 Extended `cpio` Format 213
10.1.3 Multiple Volumes 217

Section 11: Synchronization 219
11.1 Semaphore Characteristics 219
11.2 Semaphore Functions 219
11.2.1 Initialize an Unnamed Semaphore 219

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

11.2.2 Destroy an Unnamed Semaphore 220
11.2.3 Initialize/Open a Named Semaphore 221
11.2.4 Close a Named Semaphore 224
11.2.5 Remove a Named Semaphore 225
11.2.6 Lock a Semaphore 226
11.2.7 Unlock a Semaphore 227
11.2.8 Get the Value of a Semaphore 229

Section 12: Memory Management 231
12.1 Memory Locking Functions 232
12.1.1 Lock/Unlock the Address Space of a Process 232
12.1.2 Lock/Unlock a Range of Process Address Space 234
12.2 Memory Mapping Functions 236
12.2.1 Map Process Addresses to a Memory Object 236
12.2.2 Unmap Previously Mapped Addresses 240
12.2.3 Change Memory Protection 241
12.2.4 Memory Object Synchronization 243
12.3 Shared Memory Functions 245
12.3.1 Open a Shared Memory Object 245
12.3.2 Remove a Shared Memory Object 247

Section 13: Execution Scheduling 249
13.1 Scheduling Parameters 249
13.2 Scheduling Policies 249
13.2.1 SCHED_FIFO 250
13.2.2 SCHED_RR 251
13.2.3 SCHED_OTHER 251
13.3 Process Scheduling Functions 252
13.3.1 Set Scheduling Parameters 252
13.3.2 Get Scheduling Parameters 253
13.3.3 Set Scheduling Policy and Scheduling Parameters . . . 254
13.3.4 Get Scheduling Policy 256
13.3.5 Yield Processor 257
13.3.6 Get Scheduling Parameter Limits 258

Section 14: Clocks and Timers 261
14.1 Data Definitions for Clocks and Timers 261
14.1.1 Time Value Specification Structures 261
14.1.2 Timer Event Notification Control Block 262
14.1.3 Type Definitions 262
14.1.4 Manifest Constants 262
14.2 Clock and Timer Functions 263
14.2.1 Clocks 263
14.2.2 Create a Per-Process Timer 264
14.2.3 Delete a Per-Process Timer 266
14.2.4 Per-Process Timers 267
14.2.5 High Resolution Sleep 269

Section 15: Message Passing 271
15.1 Data Definitions for Message Queues 271

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

15.1.1 Data Structures 271
15.2 Message Passing Functions 272
15.2.1 Open a Message Queue 272
15.2.2 Close a Message Queue 275
15.2.3 Remove a Message Queue 276
15.2.4 Send a Message to a Message Queue 277
15.2.5 Receive a Message From a Message Queue 279
15.2.6 Notify Process that a Message is Available on a Queue . . 280
15.2.7 Set Message Queue Attributes 281
15.2.8 Get Message Queue Attributes 283

Annex A (informative) Bibliography 285
A.1 Related Open Systems Standards 285
A.2 Other Standards 287
A.3 Historical Documentation and Introductory Texts 288
A.4 Other Sources of Information 290

Annex B (informative) Rationale and Notes 291
B.1 Scope and Normative Cross-References 291
B.2 Definitions and General Requirements 302
B.3 Process Primitives 334
B.4 Process Environment 362
B.5 Files and Directories 369
B.6 Input and Output Primitives 380
B.7 Device- and Class-Specific Functions 397
B.8 Language-Specific Services for the C Programming Language . . 407
B.9 System Databases 417
B.10 Data Interchange Format 418
B.11 Synchronization 425
B.12 Memory Management 427
B.13 Execution Scheduling 440
B.14 Clocks and Timers 444
B.15 Message Passing 451

Annex C (informative) Header Contents Samples 457

Annex D (informative) Profiles 475
D.1 Definitions 475
D.2 Options in This Part of ISO/IEC 9945 477
D.3 Related Standards 477
D.4 Related Activities 477
D.5 Relationship to IEEE Draft Project IEEE P1003.0 478

Annex E (informative) Sample National Profile 479
E.1 (Example) Profile for Denmark 480

Annex F (informative) Portability Considerations 483
F.1 User Requirements 483
F.2 Portability Capabilities 485
F.3 Profiling Considerations 490

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

Annex G (informative) Performance Metrics 497
G.1 Performance Measurement Documentation 497
G.2 Signals 498
G.3 Synchronized Input and Output 499
G.4 Asynchronous Input and Output 501
G.5 Semaphores 503
G.6 Process Memory Locking 504
G.7 Shared Memory 505
G.8 Execution Scheduling 505
G.9 Clocks and Timers 507
G.10 Message Passing 510

Annex H (informative) Realtime Files 511
H.1 Data Definitions for Realtime Files 512
H.2 Realtime File Functions 518
H.3 Realtime Files 530
H.4 Rationale for Realtime Files 532
H.5 Realtime File Functions 535

Identifier Index 541

Alphabetic Topical Index 549

TABLES

Table 2-1 – Primitive System Data Types 32
Table 2-2 – Reserved Header Symbols 36
Table 2-3 – Minimum Values 41
Table 2-4 – Run-Time Increasable Values 42
Table 2-5 – Run-Time Invariant Values (Possibly Indeterminate) . . 43
Table 2-6 – Pathname Variable Values 44
Table 2-7 – Invariant Value 44
Table 2-7a – Maximum Values 44
Table 2-8 – Symbolic Constants for the *access*() Function 45
Table 2-9 – Symbolic Constants for the *lseek*() Function 45
Table 2-10 – Compile-Time Symbolic Constants 46
Table 2-11 – Execution-Time Symbolic Constants 47
Table 3-1 – Required Signals 61
Table 3-2 – Job Control Signals 61
Table 3-3 – Memory Protection Signals 62
Table 4-1 – *uname*() Structure Members 91
Table 4-2 – Configurable System Variables 96
Table 5-1 – *stat* Structure 118
Table 5-2 – Configurable Pathname Variables 130
Table 6-1 – *cmd* Values for *fcntl*() 142
Table 6-2 – File Descriptor Flags Used for *fcntl*() 142
Table 6-3 – *l_type* Values for Record Locking With *fcntl*() 142

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

Table 6-4 – *oflag* Values for *open*() 142
Table 6-5 – File Status Flags Used for *open*() and *fcntl*() 143
Table 6-6 – File Access Modes Used for *open*() and *fcntl*() 143
Table 6-7 – Mask for Use With File Access Modes 143
Table 6-8 – *flock* Structure 145
Table 6-9 – *fcntl*() Return Values 146
Table 7-1 – *termios* Structure 176
Table 7-2 – *termios c_iflag* Field 176
Table 7-3 – *termios c_cflag* Field 178
Table 7-4 – *termios c_lflag* Field 179
Table 7-5 – *termios c_cc* Special Control Characters 180
Table 7-6 – *termios* Baud Rate Values 181
Table 9-1 – *group* Structure 206
Table 9-2 – *passwd* Structure 207
Table 10-1 – `tar` Header Block 210
Table 10-2 – Byte-Oriented `cpio` Archive Entry 214
Table 10-3 – Values for `cpio` *c_mode* Field 215
Table 12-1 – Memory Protection Values 237
Table B-1 – Suggested Feature Test Macros 330
Table F-1 – Portability Capability Summary 486

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

Introduction

(This Introduction is not a normative part of IEEE Std 1003.1b-1993, IEEE Standard for Information Technology—Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API)—Amendment 1: Realtime Extension [C Language])

The purpose of this standard 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 standard 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 standard 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 standard. 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 standard. The other will contain a standardized interface for language-specific services. Any implementation claiming conformance to this standard 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 standard itself.

Organization of This Standard

This standard is divided into four elements:

(1) Statement of scope and list of normative references (Section 1)

(2) Definitions and global concepts (Section 2)

---

1) UNIX is a registered trademark in the USA and other countries licensed exclusively through X/Open.

2) The vertical rules in the right margin depict technical or significant non-editorial changes from IEEE Std 1003.1-1990 to IEEE Std 1003.1b-1993. A vertical rule beside an empty line indicates deleted text. The unmodified text of IEEE Std 1003.1-1990 is included for convenience and is believed to be correct. If there are technical differences between the text of this document not marked as part of IEEE Std 1003.1b-1993 and the corresponding separately published text of IEEE Std 1003.1-1990, the text of the separately published IEEE Std 1003.1-1990 prevails.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

(3) The various interface facilities (Sections 3 through 9 and 11 through 15)

(4) Data interchange format (Section 10)

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 standard 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 standard. It also provides information to emphasize consequences of the interfaces described in the corresponding section of this standard.

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 standard.) They are provided for guidance and to help understanding.

In publishing this standard, 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 standard, or to attempt to enforce this standard by these or any other means. The responsibility for determining the degree of conformance or lack thereof with this standard rests solely with the individual who is evaluating the product claiming to be in conformance with this standard.

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 standard 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.

---

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

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

Conceptually, POSIX.1 describes a set of fundamental services needed for the efficient construction of application programs. Access to these services has been provided by defining an interface, using the C programming language, 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 standard:

Application Oriented

The basic goal was to promote portability of application programs across UNIX system environments by developing a clear, consistent, and unambiguous standard for the interface specification of a portable operating system based on the UNIX system documentation. This standard codifies the common, existing definition of the UNIX system. There was no attempt to define a new system interface.

Interface, Not Implementation

This standard defines an interface, not an implementation. No distinction is made between library functions and system calls: both are referred to as functions. No details of the implementation of any function are given (although historical practice is sometimes indicated in

---

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.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

Annex B). Symbolic names are given for constants (such as signals and error numbers) rather than numbers.

Source, Not Object, Portability

This standard has been written so that a program written and translated for execution on one conforming implementation may also be translated for execution on another conforming implementation. This standard does not guarantee that executable (object or binary) code will execute under a different conforming implementation than that for which it was translated, even if the underlying hardware is identical. 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 standard 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 standard; see 1.3.1.1. This standard is also not concerned with hardware constraints or system maintenance.

Minimal Interface, Minimally Defined

In keeping with the historical design principles of the UNIX system, 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)

---

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).

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

(2) Compatible systems that are not derived from the original UNIX system code

(3) Emulations hosted on entirely different operating systems

(4) Networked systems

(5) Distributed systems

(6) Systems running on a broad range of hardware

No direct references to this goal appear in this standard, but some results of it are mentioned in 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 standard, 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.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

Realtime Extension

This supplement defines optional sets of systems interfaces to support the source portability of applications with realtime requirements. The interfaces included in this standard were the set required to make POSIX.1 minimally usable to realtime applications on single processor systems. The scope is to take existing realtime operating system practice and add it to the base standard.

The definition of *realtime* used in defining the scope of this standard is:

> "Realtime in operating systems: the ability of the operating system to provide a required level of service in a bounded response time."

The key elements of defining the scope are

(1) Defining a sufficient set of functionality to cover a significant part of the realtime application program domain, and

(2) Defining sufficient performance constraints and performance-related functions to allow a realtime application to achieve deterministic response from the system.

Specifically within the scope is to define interfaces that do not preclude high-performance implementations on traditional uniprocessor realtime systems.

Wherever possible, the requirements of other application environments were included in this interface definition. The specific areas are noted in the scope overviews of each of the interface areas given below. It is beyond the scope of these interfaces to support networking or multiprocessor functionality.

The specific functional areas included in this standard and their scope includes:

— *Semaphores*: A minimum synchronization primitive to serve as a basis for more complex synchronization mechanisms to be defined by the application program.

— *Process memory locking*: A performance improvement facility to bind application programs into the high-performance random access memory of a computer system. This avoids potential latencies introduced by the operating system in storing parts of a program that were not recently referenced on secondary memory devices.

— *Memory mapped files and shared memory*: A performance improvement facility to allow for programs to access files as part of the program images and for separate application programs to have portions of their program image commonly accessible.

— *Priority scheduling*: A performance and determinism improvement facility to allow applications to determine the order in which processes that are ready to run are granted access to processor resources.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

— *Realtime signal extension*: A determinism improvement facility that augments the signals mechanism of historical implementations to enable asynchronous signal notifications to an application to be queued without impacting compatibility with the existing signals interface.

— *Timers*: A functionality and determinism improvement facility to increase the resolution and capabilities of the time-base interface.

— *Interprocess communication*: A functionality enhancement to add a high-performance, deterministic interprocess communication facility for local communication. Network transparency is beyond the scope of this interface.

— *Synchronized input and output*: A determinism and robustness improvement mechanism to enhance the data input and output mechanisms so that an application can insure that the data being manipulated is physically present on secondary mass storage devices.

— *Asynchronous input and output*: A functionality enhancement to allow an application process to queue data input and output commands with asynchronous notification of completion. This facility includes in its scope the requirements of supercomputer applications.

Related Standards Activities

Activities to extend this standard 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 standard

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

(3) Shell and utility facilities

(4) Verification testing methods

(5) Multi-threaded process facilities

(6) Secure/Trusted system considerations

(7) Network interface facilities

(8) System administration

(9) Graphical user interfaces

---

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.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

(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 IEEE Standards procedures.

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

If you have an interest in participating in the PASC 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 PASC working group. If you have an interest in participating in this work at the international level, contact your ISO/IEC national body.

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

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

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

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

David Chinn	*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

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

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

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

Marco W. Migliaro, *Chair* **James M. Daly,** *Vice Chair*
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

IEEE Std 1003.1b-1993 was prepared by the P1003.4 working group, sponsored by the Portable Applications Standards Committee of the IEEE Computer Society. At the time this standard was approved, the membership of the P1003.4 working group was as follows:

Portable Applications Standards Committee

Chair:	Jim Isaak
Vice Chairs:	Hal Jespersen
	Lorraine Kevra
	Barry Needham
Treasurer:	Peter Smith
Secretary:	Charles Severance

1003.4 Working Group Officials

Chair:	Bill Corwin
Vice Chairs:	Mike Cossey (1985-1991)
	Bill Gallmeister (1991-1993)
Editors:	Ed Blackmond (1987-1988)
	John Zolnowsky (1988-1993)
Secretary:	Lee Schermerhorn

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

Technical Reviewers

Bill Corwin
Bill Gallmeister
Alan Kiecker
Steve Kleiman
Doug Locke
Brian McCarthy
Lee Schermerhorn
John Zolnowsky

Working Group

Miguel Abdo
Bill Allen
Ted Baker
John Barr
Greg Batti
Pascal Beyls
Andrew Bishop
Nawaf Bitar
Peter Bixby
Ed Blackmond
Jim Blondeau
Kathy Bohrer
Bill Bone
Keith Bostic
Jim Bound
Elliot Brebner
Sven Brehmer
Giovanni Brignolo
Steve Brodeur
Mark Brown
Mitchell Bunnell
L. W. Burns III
Steve Buroff
Paul Cantrell
James Capps
John Carmichael
Stephean Carpenter
Christine Carroll
John Carson
Tim Carter
Cheng-Chung Chen
Philip Christopher
Richard Clark
Jeff Cleveland
Dave Cline
David Cohen
Bob Conti
Bill Corwin
Mike Cossey
Bernard Cox
Denise Cully
Charles Curley
Ajit Dandapani
David Dodge
Mark Doran
Terence Dowling
Larry Dwyer
Ron Elliott
Philip Erickson
Lemeul Eubanks
Fran Fadden
Allen Farris
Kester Fong
Martin Fouts
Dan Frank
Art Fritzson
Mitchell Fuchs
Bill Gallmeister
Bob Gambrel
John Gertwagen
Greg Goddard
W. E. Goebel
Andrew Gollan
Rup Grafendorfer
David Greenstein
Rick Greer
Jerry Gross
Dan Grostick
Mesut Gunduc
Bob Hairfield
John Hanley
Richard Hart
Terry Hayes
Stephen Head
Mats Hellstrom
Morris Herbert
Russ Holt
Ian Hopper
Jim Houston
John Howard
Bill Hullsiek
James Isaak
Doug Jewett
Mike Jones
Christopher Juillet
Vassilios Kalfakakos
Peter Kao
Tom Kapish
Lars Karlsson
Sol Kavy
Gregg Kellogg
Doug Kevorkian
Sandeep Khanna
Alan Kiecker
Jeff Kimmel
Dale Kirlkand
Y. K. Kiu
Bob Kleinschumidf
John Kline
Robert Knighten
Joseph Korty
K. Kothari
Eva Kristensson
Peter Krupp
Jeff Lee
Bob Lenk
Kin Leung
Doug Locke
Juliam Lomberg
Michel Lortie
Brian Lucas
Robert Luken
Dave Lunger
Ron Mabe
Rod MacDonald
Sukan Makmuri
Mark Manasse
Mike Manley
Steve Marcie
Mike Marciniszyn
Michael McBride
Brian McCarthy
Dora Merris
John Meyer
Laura Micks
Stephen Miller
Cliff Moore
Jim Moseman
Naren Nachiappan
Martha Nalebuff
Bret Needle
Mark Nudelman
Greg Nuss
Bob Nystrom
Robin O'Neal
Hagai Ohel
John Parker
Robert Parlock
Simon Patience
Offer Pazy
Donald Peterson
Prayoon Phathayakorn
Gilbert Pile Jr
Dave Plauger
Wendy Rauch-Hindin
Dave Rorke
Barry Ruzek
Doris Ryan
Ashok Saxena
Rich Schaaf

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

Norm Scherer
Lee Schermerhorn
Curt Schimmel
Mike Schultz
Ed Schwartz
Tom Scott
Karen Sheaffer
Thomas Shonk
Bruce Sigmon
Inder Singh
Roger Sippl
Val Skalabrin
Carl Smith
James Soddy
K. Y. Srinivasan
Daniel Steinberg
Audrey Strathmeyer
Keith Stuck
Garret Swart
Ravi Tavakley
Marc Teller
Jack Test
Dan Tiernan
Barry Traylor
Michael Turner
David Uhrlaub
Peter vanderLinden
Luis Varges
Joel Wagner
Alan Weaver
Perry Weller
Andrew Wheeler, Jr.
Gary Whisenhunt
Jack White
Dwight Wilcox
John Williams
John Wu
Margaret Yang
Tohru Yoneda
Steve Zanoni
Fred Zlotnick
John Zolnowsky

The following persons were members of the P1003.4 balloting group:

Shane McCarron *Unix International Organizational Representative*

Miguel Abdo
Steven Albert
Bill Allen
Helene Armitage
David Athersych
Theodore P. Baker
Steven E. Barber
Robert Barned
John Barr
Edward Benson
Robert Bismuth
Nawaf Bitar
Jim Blondeau
James Bohem
Kathryn Bohrer
William Bone
Robert Borochoff
Keith Bostic
James P. Bound
Kevin Brady
Sven Brehmer
Charles Brooks
David Brower
Gretchen Brown
Ray Bryant
Luis-Felipe Cabrera
Nicholas A. Camillone
Clyde Camp
George S. Carson
Steven Carter
Jerry Cashin
John Caywood
Kilnam Chon
Philip Christopher
Anthony Cincotta
Robert L. Claeson
Mark Colburn
R. Cornelius
William M. Corwin
Mike Cossey
William Cox
Donald Cragun
Russell Davis
Ana Maria De Alvare
Virgil Decker
Dave Decot
Larry Diegel
David Dodge
Terence Dowling
Stephen A. Dum
John D. Earls
Ron Elliott
Richard W. Elwood
David Emery
Philip H. Enslow
Philip Erickson
Catherine Fitzpatrick
Kester Fong
Daniel M. Frank
Bill Gallmeister
Michel Gien
Jean Gilmore
Richard Greer
Tom Griest
Dave Grindeland
Robert C. Groman
Judy Guist
Carl Hall
Charles Hammons
Allen L. Hankinson
Michael J. Hannah
Carol J. Harkness
Craig Harmer
Steve Head
Myron Hecht
Barry Hedquist
William Hefley
Hans H. Heilborn
Ralph Henkaus
Karl Heubaum
Jim Hightower
David F. Hinnant
John Hogan
Terrence Holm
Rand Hoven
Randall Howard
Irene Hu
Andrew R. Huber
Richard Hughes-Rowlands
William Hullsiek
Judith Hurwitz
Jim Isaak
Dan Iuster
Hal Jespersen
Michael Jones
Greg Jones
James E. Jordan
Mike Kamrad
Michael J. Karels
Sol Kavy
Michael Kearney
Joseph Keenan
Jerry Keselman
Lorraine C. Kevra
Sandeep Khanna
Alan W. Kiecker
Jeffrey S. Kimmel
Martin J. Kirk
Dale Kirkland
Steve Kleiman
Philip Klimbal

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

John T. Kline
Kenneth Klingman
Robert Knighten
Takashi Kojo
David Korn
D. Richard Kuhn
Takahiko Kuki
Joan Kundig
Thomas Kwan
Robin B. Lake
Mike Lambert
Mark Lamonds
Sue Le Grand
Doris Lebovits
Maggie Lee
Greger Leijonhufvud
Robert Lenk
David Lennert
Donald Lewine
Kevin Lewis
Kin Fun Li
F. C. Lim
John Litke
C. Douglass Locke
James P. Lonjers
Warren E. Loper
Joseph F. P. Luhukay
Robert D. Luken
Dave Lunger
Heinz Lycklama
Wafa Makhlouf
Robert Makowski
William Mar
Roger Martin
Joberto S. B. Martins
Yoshihiro Matsumoto
Martin J. McGowan
Marshall McKusick
Robert McWhirter
Doug Michels
Gary W. Miller
David Mitchell
James Moe
Martha Nalebuff
Barry Needham
Landon Curt Noll
Fred Noz
Mark Nudelman
Dennis O'Neill
Jim Oldroyd
Offer Pazy
John Peace
Prayoon Phathayakorn
Dave Plauger
P. Plauger
Gerald Powell
Scott E. Preece
John S. Quarterman
Roger Racine
Joseph Ramus
Wendy Rauch
Carol Raye
Michael Ressler
Christopher J. Riddick
Arnold Robbins
David Rorke
Harry Rowe
Arthur Sabsevitz
Peter Salus
Robert Sarr
Richard Schaaf
Lorne H. Schachter
Norman Schneidewind
Mike Schultz
Wolfgang Schwabl
Stephen C. Schwarn
Thomas Scott
Richard L. Scott
Leonard W. Seagren
Glen Seeds
Richard Seibel
Karen L. Sheaffer
Dan Shia
Lee Silverthorn
Inder Singh
Rajiv Sinha
Jacob Slonim
Charles Smith
Warren Smith
Steven Sommars
Rob Spake
K. Srinivasaiv
Dennis Steinauer
Daniel Stock
Brian Sullivan
Scott Sutter
James G. Tanner
Ravi Tavakley
Donn S. Terry
Gary F. Tom
Andrew T. Twigger
Mark-Rene Uchida
L. David Umbaugh
Michael W. Vannier
Dalibor F. Vrsalovic
David John Wallace
Stephen R. Walli
Neal Walters
Tom Watson
Alan Weaver
Larry A. Wehr
Bruce Weiner
Brian Weis
Robert Weissensee
Andrew E. Wheeler, Jr.
Alex White
Nicholas Ray Wilkes
David Willcox
John R. Williams
Randall F. Wright
Tohru Yoneda
Oren Yuen
Janusz Zalewski
Alaa Zeineldine
Jason Zions
John Zolnowsky
Glen Zorn

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

When the IEEE Standards Board approved this standard on September 15, 1993, it had the following membership:

Wallace S. Read, *Chair* **Donald C. Loughry,** *Vice Chair*
Andrew G. Salem, *Secretary*

Gilles A. Baril	Jim Isaak	Don T. Michael*
Jose A. Berrios de la Paz	Ben C. Johnson	Marco W. Migliaro
Clyde R. Camp	Walter J. Karplus	L. John Rankine
Donald C. Fleckenstein	Lorraine C. Kevra	Arthur K. Reilly
Jay Forster*	E. G. "Al" Kiener	Ronald H. Reimer
David F. Franklin	Ivor N. Knight	Gary S. Robinson
Ramiro Garcia	Joseph L. Koepfinger*	Leonard L. Tripp
Donald N. Heirman	D. N. "Jim" Logothetis	Donald W. Zipse

* Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal
James Beall
Richard B. Engelman
David E. Soffrin
Stanley I. Warshaw

Mary Lynne Nielsen
IEEE Standards Project Editor

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

THIS PAGE WAS
BLANK IN THE ORIGINAL

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

IEEE Standard for Information Technology—Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API)—Amendment 1: Realtime Extension [C Language]

Section 1: General

1.1 Scope

This standard 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 standard 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 standard:

(1) User interface (shell) and associated commands

(2) Networking protocols and system call interfaces to those protocols

(3) Graphics interfaces

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

(4) Database management system interfaces

(5) Record I/O considerations

(6) Object or binary code portability

(7) System configuration and resource availability

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

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

The interfaces included were the set required to make this standard minimally usable to realtime applications on single processor systems. The scope is to take existing realtime operating system practice and add it to the standard. The definition of *realtime* used in defining the scope of this standard is:

> "Realtime in operating systems: the ability of the operating system to provide a required level of service in a bounded response time."

The key elements defining the scope are

(1) Defining a sufficient set of functionality to cover a significant part of the realtime application program domain, and

(2) Defining sufficient performance constraints and performance-related functions to allow a realtime application to achieve deterministic response from the system.

Specifically within the scope is to define interfaces that do not preclude high-performance implementations on traditional uniprocessor realtime systems. Wherever possible, the requirements of other application environments are included in this interface definition. It is beyond the scope of these interfaces to support networking or multiprocessor functionality.

This standard 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. Additionally, although the interfaces will be portable, some of the numeric parameters used by an implementation may have hardware dependencies.

1.2 Normative References

The following standards contain provisions which, through references in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

{1} ISO/IEC 646: 1991,[1] *Information processing—ISO 7-bit coded character set for information interchange.*

{2} ISO/IEC 9899: 1990, *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) The system shall support all required interfaces defined within this standard. These interfaces shall support the functional behavior described herein.

(2) The system may provide additional functions or facilities not required by this standard. 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 standard. 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 standard. The conformance document shall have the same structure as this standard, 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 standard.

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

---

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

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

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 standard. 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 standard states that implementations may vary or where features are identified as undefined or unspecified.

No specifications other than those described in this standard shall be present in the conformance document.

The phrases "shall document" or "shall be documented" in this standard 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.

1.3.1.3 Conforming Implementation Options

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

{NGROUPS_MAX}	Supplementary Groups option (in 2.8.3)
{_POSIX_ASYNCHRONOUS_IO}	Asynchronous Input and Output option (in 2.9.3)
{_POSIX_CHOWN_RESTRICTED}	Change File Owner Restriction (in 2.9.4)
{_POSIX_FSYNC}	File Synchronization option (in 2.9.3)
{_POSIX_JOB_CONTROL}	Job Control option (in 2.9.3)
{_POSIX_MAPPED_FILES}	Memory Mapped Files option (in 2.9.3)
{_POSIX_MEMLOCK}	Process Memory Locking option (in 2.9.3)
{_POSIX_MEMLOCK_RANGE}	Range Memory Locking option (in 2.9.3)
{_POSIX_MEMORY_PROTECTION}	Memory Protection option (in 2.9.3)
{_POSIX_MESSAGE_PASSING}	Message Passing option (in 2.9.3)
{_POSIX_PRIORITIZED_IO}	Prioritized Input and Output option (in 2.9.3)
{_POSIX_PRIORITY_SCHEDULING}	Process Scheduling option (in 2.9.3)
{_POSIX_REALTIME_SIGNALS}	Realtime Signals Extension (in 2.9.3)
{_POSIX_SEMAPHORES}	Semaphores option (in 2.9.3)
{_POSIX_SHARED_MEMORY_OBJECTS}	Shared Memory Objects option (in 2.9.3)

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

{_POSIX_SYNCHRONIZED_IO}	Synchronized Input and Output option (in 2.9.3)
{_POSIX_TIMERS}	Timers option (in 2.9.3)

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 standard to not be defined, an implementation shall provide a function that is callable with the syntax defined in this standard, 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 standard 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 standard and the applicable language standards. Such an application shall accept any behavior described in this standard as *unspecified* or *implementation-defined*, and for symbolic constants, shall accept any value in the range permitted by this standard. Such applications are permitted to adapt to the availability of facilities whose availability is indicated by the constants in 2.8.2 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 standard 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.

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.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

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 standard. 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 standard. The sections of the C Standard {2} referenced to describe requirements for this standard 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 standard with the C Language Binding shall provide the facilities referenced in Section 8, along with any additions and amplifications Section 8 requires.

Although this standard 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 standard. 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 standard 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.

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).

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

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 and 11 through 15, 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 standard.

(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.

(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.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

1.3.5 Other Language-Related Specifications

This standard 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 standard 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 standard refer only to the C Language.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

Section 2: Terminology and General Requirements

2.1 Conventions

This standard 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.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

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 standard.

NOTEs provided as parts of labeled tables and figures are integral parts of this standard (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 standard, 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 standard, 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 standard.

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.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

2.2.1.7 supported: A condition regarding optional functionality.

Certain functionality in this standard is optional, but the interfaces to that functionality are always required. If the functionality is *supported*, the interfaces work as specified by this standard (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 standard 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 standard 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 standard, the following definitions apply:

2.2.2.1 absolute pathname: See pathname resolution in B.2.3.7.

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 implementation defined process with regard to the function calls and function call options defined in this standard that need special privileges.

There may be zero or more such means.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

2.2.2.5 arm (a timer): To start a timer measuring the passage of time, enabling notifying a process when the specified time or time interval has passed.

2.2.2.6 asynchronous I/O operation: An I/O operation that does not of itself cause the process requesting the I/O to be blocked from further use of the processor.

This implies that the process and the I/O operation may be running concurrently.

2.2.2.7 asynchronous I/O completion: For an asynchronous read or write operation, when a corresponding synchronous read or write would have completed and when any associated status fields have been updated.

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

2.2.2.9 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.

2.2.2.10 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.

2.2.2.11 blocked process: A process that is waiting for some condition (other than the availability of a processor) to be satisfied before it can continue execution.

2.2.2.12 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.

2.2.2.13 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 standard, and their use is unspecified by this standard.

2.2.2.14 child process: See *process* in 2.2.2.83.

2.2.2.15 clock: An object that measures the passage of time.

The current value of the time measured by a clock can be queried and, possibly, set to a value within the legal range of the clock.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

2.2.2.16 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*.

2.2.2.17 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.

2.2.2.18 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.

2.2.2.19 current working directory: See working directory in 2.2.2.132.

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

2.2.2.21 directory: A file that contains directory entries.

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

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

Several directory entries can associate names with the same file.

2.2.2.23 direct I/O: An operation that attempts to circumvent a system performance optimization for the optimization of the individual I/O operation.

2.2.2.24 disarm (a timer): To stop a timer from measuring the passage of time, disabling any future process notifications (until the timer is armed again).

2.2.2.25 drift rate (of a clock): The rate at which the time measured by a clock deviates from the actual passage of real time.

A positive drift rate causes a clock to gain time with respect to real time; a negative drift rate causes a clock to lose time with respect to real time.

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

See pathname resolution in B.2.3.7.

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

See pathname resolution in B.2.3.7.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

2.2.2.28 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.

2.2.2.29 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.

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

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

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

See seconds since the Epoch.

2.2.2.33 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.

2.2.2.34 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.

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

2.2.2.35 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.

2.2.2.36 file description: See *open file description* in 2.2.2.66.

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

2.2.2.38 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 groups of the process matches the group ID associated with the file. Other members of the class may be implementation defined.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

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

2.2.2.40 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 B.2.3.7. A filename is sometimes referred to as a pathname component.

2.2.2.41 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.

2.2.2.42 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.

2.2.2.43 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.

2.2.2.44 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.

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

File serial numbers are unique throughout a file system.

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

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

2.2.2.47 first open (of a file): When a process opens a file that is not currently an open file within any process.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

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

2.2.2.49 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.

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

2.2.2.51 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.

2.2.2.52 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.

2.2.2.53 last close (of a file): When a process closes a file, resulting in the file not being an open file within any process.

2.2.2.54 link: See directory entry in 2.2.2.22.

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

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

Each login shall be associated with exactly one login name.

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

2.2.2.58 map: To create an association between a page-aligned range of the address space of a process and a range of physical memory or some memory

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

object, such that a reference to an address in that range of the address space results in a reference to the associated physical memory or memory object.

The mapped memory or memory object is not necessarily memory-resident.

2.2.2.59 memory object: Either a file or shared memory object.

When used in conjunction with *mmap*(), a memory object will appear in the address space of the calling process.

2.2.2.60 memory-resident: Managed by the implementation in such a way as to provide an upper bound on memory access times.

2.2.2.61 message: Information that can be transferred between processes by being added to and removed from a message queue.

A message consists of a fixed-size message buffer.

2.2.2.62 message queue: An object to which messages can be added and removed.

Messages may be removed in the order in which they were added or in priority order. Characteristics and interfaces associated with message queues are defined in Section 15.

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

See file access permissions in 2.3.2.

2.2.2.64 null string: See empty string in 2.2.2.31.

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

2.2.2.66 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.

2.2.2.67 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.

2.2.2.68 page: The granularity of process memory mapping or locking.

Physical memory and memory objects can be mapped into the address space of a process on page boundaries and in integral multiples of pages. Process address

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

space can be locked into memory—made memory-resident—on page boundaries and in integral multiples of pages.

2.2.2.69 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.

2.2.2.70 parent process: See *process* in 2.2.2.83.

2.2.2.71 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.

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

2.2.2.73 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 B.2.3.7.

2.2.2.74 pathname component: See filename in 2.2.2.40.

2.2.2.75 persistence: A mode for semaphores, shared memory, and message queues requiring that the object and its state (including data, if any) are preserved after the object is no longer referenced by any process.

Persistence of an object does not imply that the state of the object is maintained across a system crash or a system reboot.

2.2.2.76 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

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

identically to a FIFO special file when accessed in this way. It has no name in the file hierarchy.

2.2.2.77 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 standard, 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.

2.2.2.78 preallocation: The reservation of resources in a system for a particular use.

Preallocation does not imply that the resources are immediately allocated to that use, but merely indicates that they are guaranteed to be available in bounded time when needed.

2.2.2.79 preempted process: A running process whose execution is suspended due to another process becoming runnable at a higher priority.

2.2.2.80 priority: A nonnegative integer associated with each process, whose value is constrained to a range defined by the applicable scheduling policy.

Numerically higher values represent higher priorities.

2.2.2.81 priority-based scheduling: Scheduling in which the selection of a running process is determined by the priority of the runnable processes.

2.2.2.82 privilege: See appropriate privileges in 2.2.2.4.

2.2.2.83 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.

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

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

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

2.2.2.85 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.

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

2.2.2.87 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.

2.2.2.88 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.

2.2.2.89 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.

2.2.2.90 process list: An ordered set of runnable processes that all have the same ordinal value for their priority.

The ordering of processes on the list is determined by a scheduling policy or policies. The set of process lists includes all runnable processes in the system.

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

2.2.2.92 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 2.2.2.51. This value is subject to change during the process lifetime, as described in 4.2.2.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

2.2.2.93 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 2.2.2.130. This value is subject to change during the process lifetime, as described in 4.2.2.

2.2.2.94 referenced shared memory object: A shared memory object that is open or has one or more mappings defined on it.

2.2.2.95 region:

(1) As relates to the address space of a process, a sequence of addresses.

(2) As relates to a file, a sequence of offsets.

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

2.2.2.97 relative pathname: See pathname resolution in B.2.3.7.

2.2.2.98 (time) resolution: The minimum time interval that a clock can measure or whose passage a timer can detect.

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

2.2.2.100 runnable process: A process that is capable of being a running process, but for which no processor is available.

2.2.2.101 running process: A process currently executing on a processor.

There may be more than one such process in a system at a time in a system with multiple processors.

2.2.2.102 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.

2.2.2.103 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.

2.2.2.104 scheduling: The application of a policy to select a runnable process to become a running process, or to alter one or more of the process lists.

2.2.2.105 scheduling policy: A set of rules that is used to determine the order of execution of processes to achieve some goal.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

In the context of this standard, a scheduling policy affects process ordering

(1) When a process is a running process and it becomes a blocked process

(2) When a process is a running process and it becomes a preempted process

(3) When a process is a blocked process and it becomes a runnable process

(4) When a running process calls a function that can change the priority or scheduling policy of a process

(5) In other scheduling-policy-defined circumstances

Conforming implementations shall define the manner in which each of the scheduling policies may modify the priorities or otherwise affect the ordering of processes at each of the occurrences listed above. Additionally, conforming implementations shall define at what other circumstances and in what manner each scheduling policy may modify the priorities or affect the ordering of processes.

2.2.2.106 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:

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

2.2.2.107 semaphore: A shareable resource that has a nonnegative integral value.

When the value is zero, there is a (possibly empty) set of processes awaiting the availability of the semaphore.

2.2.2.108 semaphore lock operation: An operation that is applied to a semaphore.

If, prior to the operation, the value of the semaphore is zero, the semaphore lock operation shall cause the calling process to be blocked and added to the set of processes awaiting the semaphore. Otherwise, the value is decremented. See 2.2.2.109.

2.2.2.109 semaphore unlock operation: An operation that is applied to a semaphore.

If, prior to the operation, there are any processes in the set of processes awaiting the semaphore, then some process from that set shall be removed from the set and become unblocked. Otherwise, the semaphore value is incremented. See 2.2.2.108.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

2.2.2.110 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.

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

2.2.2.112 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.

2.2.2.113 shared memory object: An object that represents memory that can be mapped concurrently into the address space of more than one process.

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

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

2.2.2.115 slash: The literal character "/".

This character is also known as *solidus* in ISO 8859-1 {B34}.

2.2.2.116 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.

2.2.2.117 successfully transferred: For a write operation to a regular file, when the system ensures that all data written is readable on any subsequent open of the file (even one that follows a system or power failure) in the absence of a failure of the physical storage medium.

For a read operation, when an image of the data on the physical storage medium is available to the requesting process.

2.2.2.118 synchronized I/O completion: The state of an I/O operation that has either been successfully transferred or diagnosed as unsuccessful.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993

2.2.2.119 synchronized I/O data integrity completion:

(1) For read, when the operation has been completed or diagnosed if unsuccessful. The read is complete only when an image of the data has been successfully transferred to the requesting process. If there were any pending write requests affecting the data to be read at the time that the synchronized read operation was requested, these write requests shall be successfully transferred prior to reading the data.

(2) For write, when the operation has been completed or diagnosed if unsuccessful. The write is complete only when the data specified in the write request is successfully transferred and all file system information required to retrieve the data is successfully transferred.

File attributes that are not necessary for data retrieval (access time, modification time, status change time) need not be successfully transferred prior to returning to the calling process.

2.2.2.120 synchronized I/O file integrity completion: Identical to a synchronized I/O data integrity completion with the addition that all file attributes relative to the I/O operation (including access time, modification time, status change time) shall be successfully transferred prior to returning to the calling process.

2.2.2.121 synchronized I/O operation: An I/O operation performed on a file that provides the application assurance of the integrity of its data and files.

2.2.2.122 synchronous I/O operation: An I/O operation that causes the process requesting the I/O to be blocked from further use of the processor until that I/O operation completes.

NOTE: A synchronous I/O operation does not perforce imply synchronized I/O data integrity completion or synchronized I/O file integrity completion.

2.2.2.123 system: An implementation of this standard.

2.2.2.124 system crash: An interval initiated by an unspecified circumstance that causes all processes (possibly other than special system processes) to be terminated in an undefined manner, after which any changes to the state and contents of files created or written to by a Conforming POSIX.1 Application prior to the interval are undefined, except as required elsewhere in this standard.

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

2.2.2.126 system reboot: An implementation defined sequence of events that may result in the loss of transitory data, i.e., data that is not saved in permanent storage.

This includes message queues, shared memory, semaphores, and processes.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:17:57 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.1b-1993