

IEEE Std 1003.5c-1998
(Includes IEEE Std 1003.5-1992
and IEEE Std 1003.5b-1996)

IEEE Standard for Information Technology— POSIX[®] Ada Language Interfaces— Part 1: Binding for System Application Program Interface (API)— Amendment 2: Protocol-Independent Interfaces

Sponsor
Portable Applications Standards Committee
of the
IEEE Computer Society

Approved 8 December 1998
IEEE Standards Board

Abstract: This standard is part of the POSIX series of standards for applications and user interfaces to open systems. It defines the Ada language bindings as package specifications and accompanying textual descriptions of the application program interface (API). This standard supports application portability at the source code level through the binding between ISO 8652:1995 (Ada) and ISO/IEC 9945-1:1996 (IEEE Std 1003.1-1996) (POSIX), as amended by IEEE P1003.1g/D6.6. Terminology and general requirements, process primitives, the process environment, files and directories, input and output primitives, device- and class-specific functions, language-specific services for Ada, system databases, synchronization, memory management, execution scheduling, clocks and timers, message passing, task management, the XTI and sockets detailed network interfaces, event management, network support functions, and protocol-specific mappings are covered. It also specifies behavior to support the binding that must be provided by the Ada compilation system.

Keywords: Ada, API, application portability, computer language bindings, information exchange, interprocess communication, networks, open systems, operating systems, portable application, POSIX, POSIX language bindings, protocol-specific, protocol-independent, real-time, sockets, thread, XTI.

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

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 1999 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 3 December 1999. Printed in the United States of America.

Print: ISBN 0-7381-1539-8 SH94710
PDF: ISBN 0-7381-1540-1 SS94710

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.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) 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 societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

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

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

Authorization to photocopy portions of any individual standard for internal or personal use is granted by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; (978) 750-8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Contents

	PAGE
Introduction	vi
Section 1: General	1
1.1 Scope	1
1.2 Normative References	3
1.3 Conformance	4
Section 2: Terminology and General Requirements	9
2.1 Editorial Conventions	9
2.2 Definitions	10
2.3 General Concepts	38
2.4 Package POSIX	42
2.5 Package POSIX_Options	76
2.6 Package POSIX_Limits	82
2.7 Package Ada_Streams	87
2.8 Package System	88
2.9 Package System_Storage_Elements	89
2.10 Package POSIX_Page_Alignment	90
2.11 Environment Description	91
Section 3: Process Primitives	95
3.1 Package POSIX_Process_Primitives	95
3.2 Package POSIX_Unsafe_Process_Primitives	108
3.3 Package POSIX_Signals	115
Section 4: Process Environment	149
4.1 Package POSIX_Process_Identification	149
4.2 Package POSIX_Process_Times	156
4.3 Package POSIX_Process_Environment	157
4.4 Package POSIX_Calendar	164
4.5 Package POSIX_Configurable_System_Limits	167
Section 5: Files and Directories	175
5.1 Package POSIX_Permissions	175
5.2 Package POSIX_Files	178
5.3 Package POSIX_File_Status	190
5.4 Package POSIX_Configurable_File_Limits	194
Section 6: I/O Primitives	205
6.1 Package POSIX_IO	205

6.2	Package POSIX_File_Locking	232
6.3	Package POSIX_Asynchronous_IO	234
Section 7: Device- and Class-Specific Functions		253
7.1	General Terminal Interface	253
7.2	Package POSIX_Terminal_Functions	260
Section 8: Language-Specific Services for Ada		279
8.1	Interoperable Ada I/O Services	279
8.2	Package POSIX_Supplement_to_Ada_IO	283
Section 9: System Databases		287
9.1	Package POSIX_User_Database	287
9.2	Package POSIX_Group_Database	289
Section 10: Data Interchange Format		293
Section 11: Synchronization		295
11.1	Package POSIX_Semaphores	295
11.2	Package POSIX_Mutexes	304
11.3	Package POSIX_Condition_Variables	314
Section 12: Memory Management		323
12.1	Package POSIX_Memory_Locking	324
12.2	Package POSIX_Memory_Range_Locking	326
12.3	Package POSIX_Memory_Mapping	328
12.4	Package POSIX_Shared_Memory_Objects	337
12.5	Package POSIX_Generic_Shared_Memory	341
Section 13: Execution Scheduling		349
13.1	Scheduling Concepts and Terminology	349
13.2	Package POSIX_Process_Scheduling	349
13.3	Task Scheduling	354
13.4	Synchronization Scheduling	356
Section 14: Clocks and Timers		357
14.1	Package POSIX_Timers	357
14.2	High Resolution Delay	366
Section 15: Message Passing		367
15.1	Package POSIX_Message_Queues	367
Section 16: Task Management		383
16.1	Package Ada_Task_Identification	383
Section 17: Detailed Network Interface - XTI		385
17.1	Introduction	385
17.2	States and Events	391
17.3	The Use of Options	399
17.4	Package POSIX_XTI	410

Section 18: Detailed Network Interface - Socket	487
18.1 Introduction	487
18.2 Events and States	489
18.3 Use of Options	498
18.4 Package POSIX_Sockets	498
Section 19: Event Management	535
19.1 Package POSIX_Event_Management	535
Annex A (informative) Bibliography	547
Annex B (informative) Rationale and Notes	549
B.1 General	549
B.2 Terminology and General Requirements	563
B.3 Process Primitives	575
B.4 Process Environment	592
B.5 Files and Directories	598
B.6 Input and Output Primitives	602
B.7 Device- and Class-Specific Functions	610
B.8 Language-Specific Services for Ada	610
B.9 System Databases	621
B.10 Data Interchange Format	623
B.11 Synchronization	623
B.12 Memory Management	629
B.13 Execution Scheduling	634
B.14 Clocks and Timers	639
B.15 Message Passing	642
B.16 Task Identification	643
B.17 Thread-Specific Data	643
B.18 Detailed Network Interface - XTI	643
B.19 Detailed Network Interface - Socket	646
B.20 Network Support Functions	647
B.21 Protocol Mappings Annex	648
Annex C (informative) Ada/C Cross-References	651
C.1 Ada-to-C Cross-Reference	651
C.2 C-to-Ada Cross-Reference	681
Annex D (normative) Protocol Mappings	713
D.1 Sockets Protocol Mappings	713
D.2 XTI Protocol Mappings	759
Alphabetic Topical Index	819
TABLES	
Table1.1 – Sockets and XTI Package Renaming	5
Table2.1 – Typographical Conventions	9

Table2.2 – Constant and Subtype Correspondences	52
Table2.3 – Option Set Comparisons	60
Table2.4 – Static Subtypes and Options	81
Table2.5 – Portable Constants and Limits	86
Table2.6 – Static Subtypes and Limits	88
Table3.1 – Default Actions for Job Control Signals	127
Table4.1 – Functions for System-Wide Options	172
Table4.2 – Configurable System Limits	174
Table6.1 – Standard File Descriptors	209
Table6.2 – Error Codes and AIO Status Values	247
Table7.1 – Terminal_Characteristics Components	263
Table7.2 – Terminal_Modes Values for Input Control	265
Table7.3 – Terminal_Modes Values for Output Control	267
Table7.4 – Terminal_Modes Values for Hardware Control	267
Table7.5 – Terminal_Modes Values for Local Control Modes	269
Table7.6 – Special Control Character Usage	272
Table17.1 – Events and Look	389
Table17.2 – Classification of the XTI Functions	392
Table17.3 – Communication Interface States	394
Table17.4 – Initialization/De-initialization State Table	396
Table17.5 – Data Transfer State Table for Connectionless-Mode Service	396
Table17.6 – Connection/Release/Data Transfer State Table for Connection- Mode Service	397
Table17.7 – Event_Requires_Attention Error Indications	399
Table18.1 – Socket Events	490
Table18.2 – Socket States	491
TableB.1 – Correspondence of File Creation Flags	615
TableD.1 – Port Number Re-Use	744
TableD.2 – Communications_Provider_Info Returned by Get_Info and Open, mOSI	774
TableD.3 – XTI and ACSE/Presentation Services	777
TableD.4 – XTI mOSI Connection-Mode Data Transfer Services	779
TableD.5 – XTI and Association Release Services	779
TableD.6 – XTI Connectionless-Mode ACSE Services	780
TableD.7 – Communications_Provider_Info Returned by Get_Info and Open, ISO	801
TableD.8 – Communications_Provider_Info Returned by Get_Info and Open, Internet	810

Introduction

(This introduction is not a part of IEEE Std 1003.5c-1998, IEEE Standard for Information Technology – POSIX[®] Ada Language Interfaces – Part 1: Binding for System Application Program Interface (API) – Amendment 2: Protocol-Independent Interfaces, but is included for information only.)

This standard is an amended version of IEEE Std 1003.5b-1996. The basic goal of this standard is to provide an Ada application program interface for the language-independent services made accessible to C-language applications programs by the interfaces defined in ISO/IEC 9945-1:1996 (IEEE Std 1003.1-1996) {2} as amended by IEEE P1003.1g {B14}.

The intent is to support portability of Ada applications via a standard binding to the services provided by a POSIX-conforming operating system. POSIX is defined by the standard C-language interfaces cited above. Therefore, much of the work in producing this standard was deciding what features of those C-language interfaces represented POSIX functionality, as opposed to C-language-specific features.

This standard provides package specifications and accompanying textual description for a set of Ada packages that represent the POSIX system. This standard also specifies behavior to support the binding that must be provided by the Ada compilation system, and further defines behavior specified as implementation defined in the Ada language standard (particularly in the area of Text_IO) for use in a POSIX environment.

The emphasis in POSIX is on application program portability, so the interfaces in this standard are not intended to be sufficient to implement an Ada compilation system or a POSIX shell as defined in IEEE Standard 1003.2 {B16}. For an application, the intent is that a Strictly Conforming POSIX.5 Application (one that uses only the facilities in this standard and that does not depend on implementation-defined behavior) can be ported to any Conforming Implementation of these interfaces and that the binding makes it easy to identify where a program is not strictly conforming and makes such programs easier to port.

Organization of This Standard

The standard is divided into three parts:

- Statement of scope, list of normative references, and conformance information (Section 1)
- Definitions and global concepts (Section 2)
- The various interface facilities (Sections 3 through 19)

The content of the sections parallels that of the correspondingly numbered sections of ISO/IEC 9945-1:1996 and IEEE P1003.1g/D6.6, with a few changes required to accommodate differences between the Ada and C-language interfaces. This standard

has no Section 10, since there is no Ada binding for that Section 10 (Data Interchange Formats) of ISO/IEC 9945-1:1996.

This introduction, any footnotes, notes accompanying the text, and the informative annexes are not considered part of this standard.

Related Standards Activities

Activities to extend this standard to address additional requirements can be anticipated in the future¹⁾.

Extensions are approved as amendments or revisions to this standard, following IEEE and ISO/IEC procedures.

Anyone interested in participating in the PASC working groups addressing these issues should send his or her 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, USA, and ask to have this information forwarded to the chair of the appropriate PASC working group. A person who is interested in participating in this work at the international level should contact his or her ISO/IEC national body.

c

1) 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, USA; Telephone: +1 202 371-0101; FAX: +1 202 728-9614.

IEEE Std 1003.5-1992 was prepared by the 1003.5 Working Group, sponsored by the Technical Committee on Operating Systems and Applications Environments of the IEEE Computer Society. At the time IEEE Std 1003.5-1992 was approved the membership of the IEEE P1003.5 working group was as follows:

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

Chair: Jehan-François Pâris

TCOS Standards Subcommittee

Chair: Jim Isaak
Vice Chairs: Ralph Barker
Hal Jespersen
Lorraine Kevra
Pete Meier
Andrew Twigger
Treasurer: Peter Smith
Secretary: Shane McCarron

P1003.5 Working Group Officials

Chair: James P. Lonjers
Steven Deller (1989-1991)
Major Terrence Fong (1988-1989)
Vice Chairs: James P. Lonjers (1990-1991)
Major Terrence Fong (1989-1990)
Stowe Boyd (1988-1989)
Editors: David Emery
Hal Jespersen
Steven Deller (1988-1989)
Rationale Editor: Mitch Gart
Secretary: C. Jayne Baker
David Emery (1988-1989)

Technical Reviewers

Ted Baker	David Emery	Jim Lonjers
Steven Deller	Mitch Gart	Jim Moore
Dennis Doubleday		Stephen Schwarm

Working Group

Ted Baker	Michael Gillam	Sue LeGrand
Stowe Boyd	Al Globus	James Lonjers
Bevin Brett	Mars Gralia	James Moore
Charles Brown	Jayne Guyse	Mark Ruddock
Robert Brown	Ken Harvey	Stephen Schwarm
Bhavesb Damania	Ruth Hirt	Michael Shapiro
Steven Deller	Jeff Hooley	Brian Sullivan
Dennis Doubleday	Michael Kjolsrud	Del Swanson
David Emery	Peter Krupp	Robert Voigt
Terry Fong	James Leathrum	Olle Wikstrom
Mitchell Gart		John Zenor

The following persons were members of the balloting group for IEEE Std 1003.5-1992.

Harold C. Adams	Andrew Chung	Allen L. Grau
John S. Adams	Brad Clark	Charles R. Grauling
Omar Ahmed	Lori A. Clarke	Daniel Green
David Allen	Norman H. Cohen	Tom Griest
Charles J. Antonelli	Edward Colbert	F. Grize
B. Ardary	Phillippe Collard	Ernesto Guerrieri
David Athersych	Robert A. Conti	Lawrence M. Gunther
Randall Atkinson	William M. Corwin	R. N. Hagen
Randal J. August	Mike Cossey	Charles Hammons
Kenneth A. Austin	John Courtney	Peter A. Hansen
Carolyn J. Baker	Donald Cragun	Sam Harbaugh
Robert L. Baker	Richard A. Crawford	Samuel Harbison
Ted Baker	Jim Creegan	David S. Hardin
James Baldo	Phyllis Crill	Charles Harkey
Brad Balfour	John J. Cupak	Loren L. Hart
Gary E. Barnes	Charles Dana	Thomas S. Hawker
Mitchell C. Barnhart	William H. Dashiell	Clark M. Hay
Randall Barron	David Davis	Ralph Hayward
Steven Barryte	Rich DeBernardo	John Craig Heberle
Barbara K. Beauchamp	David DeFanti	William Hefley
Gary Beerman	Mike Dean	A. Marlow Henne
E. Jerome Bell	Dave Decot	Donald C. Hill
Donald Bennett	Steven Deller	Norman Hines
Peter A. Berggren	Jorge Diaz-Herrera	C. Michael Holloway
Mark Biggar	Michael B. Dillencourt	Jeffrey Hooley
Robert Bismuth	James H. Dobbins	Joseph P. Hoolihan
Alex Blakemore	Audrey Dorofee	Tom Housman
Stephen Blanchette, Jr.	Terence Dowling	Richard Howard
Pieter Botman	Diptendu Dutta	Norman R. Howes
Stowe Boyd	Eugene Edelstein	Lynne M. Hubbs
Carl Brandon	Theodore F. Elbert	David K. Hughes
Philip Brashear	Richard W. Elwood	Richard G. Hull
Joseph P. Brazy	David Emery	Jeremy James
Mark S. Breckenridge	Arny Engelson	Hal Jespersen
Ronald F. Brender	Philip H. Enslow	Darryl N. Johnson
Jim Briggs	William Eventoff	Bruce Johnston
Thomas C. Brooke	Gary Falacara	Alain Jouchoux
Jerry R. Brookshire	John H. Fauerby	Juern Juergens
Charles O. Brown	Charles A. Finnell	Steven Kahn
Elizabeth B. Brown	Jeffery Fischer	Fumimiko Kamijo
Jane C. Bryan	Shayne Flint	Alan Kaminsky
Gary L. Burt	Terence Fong	Ling Kan
Christopher Byrnes	Edward J. Forbes, Jr.	Karl Kelley
David Calloway	Roy S. Freedman	Robert H. C. Kemp
Nicholas A. Camillone	Randal S. Freier	Judy S. Kerner
Kenneth W. Campbell	Dale J. Gaumer	James J. Keys
Rick Carle	Larry Gearhart	Paul J. King
David J. Carlson	K. M. George	Hans R. Klay
Dana Carson	Gregory A. Gicca	Kenneth Kloss
Jeffrey R. Carter	Robert T. Goettge	Robert Knighten
Jerry Cashin	Phillip Goldstein	Joseph B. Kolb
H. L. Catala	Roger Golliver	John C. Krasnowski
Larry Chandler	William N. Goolsby	Lak Ming Lam
Andy Cheese	William J. Goulet	Rudolf C. Landwehr
James Chelini	Mars J. Gralia	Charles F. Lanman

Gary Lauther	James K. Parrish	Ronald Skoog
Patricia K. Lawlis	Thomas Parrish	Thomas J. Smith
Scott A. Leschke	Offer Pazy	Charles Snyder
M. Levitz	Walt Penney	Jon S. Squire
Stephen H. Levy	Guido Persch	Jeff Stevenson
F. C. Lim	Flavio Petersen	Brian Sullivan
Timothy E. Lindquist	Thomas A. Peterson	Del Swanson
J. J. Logan	George W. Petznick	S. Tucker Taft
James P. Lonjers	Hane W. Polzer	Ravi Tavakley
Warren E. Loper	J. Pottmyer	Donn S. Terry
Mark Loveland	Charles Pow	John A. Thalhamer
George A. Ludgate	Eileen Quann	William J. Thomas
Sonny Lundahl	Paul Rabin	Peter L. Thompson
Wesley Mackey	John Reddan	James L. Troy
Austin J. Maher	W. Scott Redmon	Roger Tubby
James Maloney	Gregg Reed	Mark-Rene Uchida
Roger Martin	Carl Reinert	L. David Umbaugh
Robert Mathis	Judith Richardson	Robert B. Urling
Fred Maymir-Ducharme	Richard A. Rink	Evelyn M. Uzzle
Catherine McDonald	Clyde Roby	Frances Van Scoy
Robert L. McGarvey	C. Allan Rofer	Leonard Vanek
Daniel L. McNamee	Hyman Rosen	Michael W. Vannier
Robert McWhirter	Jerome D. Rosen	Uwe Wacker
Nancy R. Mead	Frederick M. Rysz	Robert N. Wagoner
Geoff Mendal	Agnes M. Sardi	Mary Wall
Jay Michael	Robert J. Satnik	Stephen R. Walli
Gary W. Miller	Allen Saxton	Neal Walters
Robert E. Miller	Lorne H. Schachter	Kenneth Wasmundt
Judah Mogilensky	F. P. Schauer	William Webster
Al Mok	Alfred H. Scholldorf	J. Richard Weger
Charles S. Mooney	Ron Schroeder	Brian Weis
James D. Mooney	Mike Schultz	Robert Weissensee
Freeman Moore	W. L. Schultz	Michael K. Welter
James W. Moore	Fritz Schulz	Stephen Wersan
Jerry A. Moore	Leonard Seagren	Thomas Wheeler
John I. Moore, Jr.	Richard Seibel	William Whitaker
Duncan Morrill	Lawrence H. Shafer	Bruce Wieand
M. W. Morron	Michael D. Shapiro	David Willcox
Gary Mrenak	John G. Shea	David C. Willet
David G. Mullens	Nagy M. Shehad	David Williamson
Richard E. Nese	Dan Shia	Paul A. Willis
Sai Lun Ng	Thomas E. Shields	David H. Winfield
Daniel Nissen	Keith Shillington	David C. Wolfe
Karl Nyberg	David Shochat	Paul A. Wolfgang
James O'Day	Stephen Schwarm	Michal Young
Evelyn Obaid	Robert Charles Shock	Oren Yuan
Patricia Oberndorf	Jerome L. Sibol, Jr.	Janusz Zalewski
Kurt M. Olender	Lee Silverthorn	K. M. Zemrowski
S. Ron Oliver		John Zolnowsky

When the IEEE Standards Board approved IEEE Std 1003.5-1992 on 18 June 1992, it had the following membership:

Marco W. Migliaro, *Chair*

Donald C. Loughry, *Vice Chair*

Andrew G. Salem, *Secretary*

Dennis Bodson	Donald N. Heirman	T. Don Michael*
Paul L. Borrill	Ben C. Johnson	John L. Rankins
Clyde R. Camp	Walter J. Karplus	Wallace S. Read
Donald C. Fleckenstein	Ivor N. Knight	Ronald H. Reimer
Jay Forster *	Joseph L. Koepfinger*	Gary S. Robinson
David F. Franklin	Irving Kolodny	Martin V. Schneider
Ramiro Garcia	D.N. "Jim" Logothetis	Terrance R. Whittemore
Thomas L. Hannan	Lawrence V. McCall	Donald W. Zipse

*Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal
 James Beall
 Richard B. Engleman
 David E. Soffrin
 Stanley Warshaw
 Mary Lynne Nielsen
IEEE Standards Project Editor

IEEE Std 1003.5b-1996 was prepared by the P1003.5 working group, sponsored by the Portable Applications Standards Committee of the IEEE Computer Society. At the time IEEE Std 1003.5b-1996 was approved the membership of the P1003.5 working group was as follows:

Portable Applications Standards Committee (PASC)

Chair: Jehan-François Pâris

PASC Standards Subcommittee

Chair: Lowell Johnson
 Vice Chair: Charles Severance
 Functional Chairs: Barry Needham
 John Spencer
 Jay Ashford
 Andrew Josey
 Treasurer: Peter Smith
 Secretary: Charles Severance

IEEE P1003.5 Working Group Officials

Chair: James P. Lonjers (1991-1994)
 Stephen Schwarm (1995-1996)
 Ted Baker (1996-1997)
 Vice Chairs: Stephen Schwarm (1991-1995)
 Randy Greene (1995-1996)
 David Emery (1003.5 Interpretations)
 Editor: Ted Baker (P1003.5b)
 Rationale Editor: Lee Lucas
 Secretary: C. Jayne Guyse (1991-1993)
 Peter Obermayer (1994-1996)

Technical Reviewers

Ted Baker	Lee Lucas	Henry H. Robbins
Mark Faulk	Peter Obermayer	Stephen Schwarm
Ted Giering	Offer Pazy	Del Swanson
Randy Greene	Ruth A. Peek	Laurent Visconti
	Ed Posnak	

Working Group

Theodore P. (Ted) Baker	David K. Hughes	Ray Ricco
Bevin Brett	James Lonjers	Henry H. Robbins
Steven Deller	Lee Lucas	Stephen Schwarm
David Emery	Peter Obermayer	Jim Smith
Mark Faulk	James T. Oblinger	Del Swanson
Randy Greene	Offer Pazy	Laurent Visconti
C. Jayne Guyse	Ruth A. Peek	John Zenor

The following persons were members of the balloting group for IEEE Std 1003.5b-1996:

Alejandro A. Alonso	Norman R. Howes	Dave Plauger
Theodore P. Baker	David K. Hughes	Arlan Pool
Robert Barned	Judy Kerner	Henry H. Robbins
Andy Bihain	Philippe Kruchten	Stephen Schwarm
William M. Corwin	Thomas M. Kurihara	Leonard W. Seagren
Steven Deller	Arthur Licht	Robert Alan Siegel
David Emery	C. Douglass Locke	Dennis C. Stewart
Philip H. Enslow	James P. Lonjers	Alfred Strohmeier
Michael Gonzalez	Lee W. Lucas	Del Swanson
C. Jayne Guyse	Roland McGrath	Mark-René Uchida
Joe Gwinn	Paul Murdock	USENIX
Patrick Hebert	James T. Oblinger	Victor Fay-Wolfe
Steven Howell	Offer Pazy	John Zenor

When the IEEE Standards Board approved IEEE Std 1003.5b-1996 on 20 June 1996, it had the following membership:

Donald C. Loughry, *Chair*

Richard J. Holleman, *Vice Chair*

Andrew G. Salem, *Secretary*

Gilles A. Baril	E. G. "Al" Kiener	Jose R. Ramos
Clyde R. Camp	Joseph L. Koepfinger*	Arthur K. Reilly
Joseph A. Cannatelli	Stephen R. Lambert	Ronald H. Reimer
Stephen L. Diamond	Lawrence V. McCall	Gary S. Robinson
Harold E. Epstein	Bruce McClung	Ingo Rüsck
Donald C. Fleckenstein	Marco W. Migliaro	John S. Ryan
Jay Forster *	Mary Lou Padgett	Chee Kiow Tan
Donald N. Heirman	John W. Pope	Leonard L. Tripp
Ben C. Johnson		Howard L. Wolfman

*Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal
Alan H. Cookson
Chester C. Taylor

Lisa S. Young
IEEE Standards Project Editor

IEEE Std 1003.5c-1998 was prepared by the P1003.5 working group, sponsored by the Portable Applications Standards Committee of the IEEE Computer Society. At the time IEEE Std 1003.5c-1998 was approved the membership of the P1003.5 working group was as follows:

PASC Standards Subcommittee

Chair:	Lowell Johnson
Vice Chair:	Joe Gwinn
Functional Chairs:	Curtis Royster
	Jason Zions
	Jay Ashford
	Andrew Josey
Secretary:	Nick Stoughton

IEEE P1003.5 Working Group Officials

Chair:	Ted Baker
Vice Chair:	Linda Harowicz
Editors:	Craig Meyer (P1003.5c Editor)

Working Group

Ted Baker	Greg Bussiere	Craig Meyer
	Linda Harowicz	

The following persons were voting members of the balloting group for IEEE Std 1003.5c-1998:

Ted Baker	Mars J. Gralia	Craig Meyer
Bob Barned	Linda Harowicz	Stephen Michell
Carl Brandon	Matthew Heaney	Howard E. Neely
Greg Bussiere	Niklas Holsti	Peter E. Obermayer
Jorge L. Diaz-Herrera	David C. Hoos	James T. Oblinger
Victor Giddings	Michael J. Kamrad	Jan Pukite
Michael Gonzalez	Mark Lundquist	Curtis Royster

The following persons were nonvoting members of the balloting group for IEEE Std 1003.5c-1998:

Robert E. Allen	Robert A. Duff	Robert C. Leif
A. Barnes	W. Douglas Findly	B. Craig Meyers
Ronald Bjornseth	Anthony Gargaro	James W. Moore
Stephen E. Blake	David Gross	Tushar Pokle
Chad Bremmon	Maretta Holden	Bill Pritchett
Vincent Celier	Harry Joiner	Michael Rohan
Hans O. Danielsson	Rush Kester	David Shochat
John Davies	Jim Kroening	Lynn Stuckey
Peter Dencker	Mark Larsen	Terry J. Westley
Guido Duerinckx		Stephen Whiting

c

When the IEEE Standards Board approved IEEE Std 1003.5c-1998 on 8 December 1998, it had the following membership:

Richard J. Holleman, *Chair*

Donald N. Heirman, *Vice Chair*

Judith Gorman, *Secretary*

James H. Gurney
Satish K. Aggarwal
Clyde R. Camp
Gary R. Engman
Harold E. Epstein
Jay Forster*
Thomas F. Garrity
Ruben D. Garzon

Jim D. Isaak
Lowell G. Johnson
Robert Kennelly
E. G. "Al" Kiener
Joseph L. Koepfinger*
Stephen R. Lambert
Jim Logothetis
Donald C. Loughry

L. Bruce McClung
Louis-François Pau Ronald C.
Petersen
Gerald H. Peterson
John B. Posey
Gary S. Robinson
Hans E. Weinrich
Donald W. Zipse

*Member Emeritus

Yvette Ho Sang
IEEE Standards Project Editor

c

IEEE Standard for Information Technology— POSIX[®] Ada Language Interfaces— Part 1: Binding for System Application Program Interface (API)— Amendment 2: Protocol Independent Interfaces

c

Section 1: General

1.1 Scope

This standard defines a set of system application program interfaces to operating system services. These interfaces provide access via the Ada programming language to the same operating system services for which C-language interfaces are specified in ISO/IEC 9945-1:1996 {2} ¹⁾²⁾ and IEEE P1003.1g {B14}.

c

The purpose of this standard is to support application portability at the Ada source code level. This standard is intended to be used by both application developers and system implementors.

This standard is intended to be compatible with implementations of the 1995 revision to the Ada language standard (ISO/IEC 8652:1995 {1}). Fall-back approaches compatible with implementations of the original Ada language standard (ISO/IEC 8652:1987 {B5}) are also provided (see 1.3).

-
- 1) Plain numbers in curly braces correspond to those of the normative references in 1.2. Numbers preceded by a “B” in curly braces correspond to those of the bibliography in Annex A. See 2.1 for the description of this and the other typographical conventions followed in this document.
 - 2) A language-independent definitions of this standard was once under development, but work on that project was suspended.

c

This standard is intended to contain no specifications that conflict with “Year 2000” requirements.

|c

This standard comprises three major components:

- Definitions for terminology and concepts, and definitions and specifications that govern program structures, language-system interaction, and related requirements.
- Definitions of the specific Ada interfaces to the system services defined by the POSIX standards, presented in the form of Ada packages.
- Interpretations of Ada semantics with respect to the POSIX standards.

The following areas are outside the scope of this standard:

- (1) User interface (shell) and commands associated with Ada program development.
- (2) Ada bindings to the archive/interchange file formats for *tar* and *cpio*.
- (3) Network protocols.
- (4) Graphics and windowing interfaces.
- (5) Database management system interfaces.
- (6) Object or binary code portability.
- (7) System configuration and resource availability.
- (8) Interfaces to the Ada runtime system.

When the XTI Detailed Network Interface option and/or the Sockets Detailed Network Interface option are supported, then a set of DNI's (see 2.2.3.26) are also within the scope of this standard. A DNI is intended to provide access to protocol-specific features of the underlying network for highly portable applications that need access to sophisticated network features. The DNI's are based on the SPG4 XTI and 4.4 BSD socket specifications.

The following areas are outside of the scope of the DNI's:

- Interface to manipulate underlying protocol implementations
- Network management interface
- Interface to manipulate performance-specific features
- Definition for protocol address formats

|c

This standard describes the external characteristics and facilities that are of importance to applications developers, rather than the implementation approaches that may be employed to achieve them. Special emphasis is placed on those facilities and capabilities needed for the broad spectrum of applications.

This standard has been defined exclusively at the source code level. The objective is that a Strictly Conforming POSIX.5 Application can be compiled to execute on any conforming implementation, within the portability of the application Ada code itself.

1.2 Normative References

The following standards contain provisions that, 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. Members of IEC and ISO maintain registers of currently valid International Standards.

- {1} ISO/IEC 8652:1995³⁾, *Information technology—Programming languages—Ada* [Revision of first edition (ISO/IEC 8652:1987)], 15 February 1995.
 - {2} ISO/IEC 9945-1:1996 (IEEE Std 1003.1-1996⁴⁾), *Information technology—Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API) [C Language]*. This edition incorporates the extensions for realtime applications (POSIX.1b, POSIX.1i) and threads (POSIX.1c).
 - {3} ISO/IEC 8072:1996 (CCIT X.214:1988⁵⁾), *Information technology—Open systems interconnection—Transport service definition*.
 - {4} ISO/IEC 8073:1992 (CCITT X.224:1992), *Information technology—Telecommunications and information exchange between systems—Open systems interconnection—Protocol for providing the connection-mode transport service*.
 - {5} ISO/IEC 8208:1995, *Information technology—Data communications—X.25 Packet layer protocol for data terminal equipment*.
 - {6} ISO/IEC 8348:1996, *Information technology—Open systems interconnection—Network service definition*.
 - {7} ISO/IEC 8473-1:1994, *Information technology—Protocol for providing the connectionless-mode network service: Protocol specification*.
 - {8} ISO/IEC 8473-3:1995, *Information technology—Protocol for providing the connectionless-mode network service: Provision of the underlying service by an X.25 subnetwork*.
 - {9} ISO/IEC 8602:1995, *Information Technology—Protocol for providing the OSI connectionless-mode transport service*.
 - {10} ISO/IEC 8878:1992, *Information technology—Telecommunications and information exchange between systems—Use of X.25 to provide the OSI connection-mode network service*.
-
- 3) ISO/IEC publications can be obtained from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembe, CH-1211, Genève 20, Switzerland/Suisse (<http://www.iso.ch>) or from the Sales Department of the International Electrotechnical Commission, Case Postale 131, 3 rue de Varembe, CH-1211, Genève 20, Switzerland/Suisse (<http://www.iec.ch/>). ISO/IEC publications can also be obtained in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA (<http://www.ansi.org>).
 - 4) IEEE standards publications are available from the IEEE Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA (<http://www.standards.ieee.org>).
 - 5) CCITT documents can be obtained from the CCIT General Secretariat, International Telecommunications Union, Sales Section, Place des Nations, CH-1211, Genève 20, Switzerland/Suisse.

c

- {11} ISO/IEC ISP 11188-3:1996, *Information Technology—International standardization profile—Common upper layer requirements—Part 3: Minimal OSI upper layer facilities.*
- {12} IETF RFC 768:1980⁶⁾, *User Datagram Protocol.*
- {13} IETF RFC 791:1981, *Internet Protocol DARPA Internet Program Protocol Specification.*
- {14} IETF RFC 793:1981, *Transmission Control Protocol DARPA Internet Program Protocol Specification.*
- {15} IETF RFC 919:1984, *Broadcasting Internet Datagrams.*
- {16} IETF RFC 922:1984, *Broadcasting Internet Datagrams in the Presence of Subnets.*
- {17} IETF RFC 1006:1987, *ISO Transport Service on Top of the TCP, Version: 3.*
- {18} IETF RFC 1122:1989, *Requirements for Internet Hosts—Communication Layers.*

NOTE: Abbreviations for the above standards are defined in 2.2.3.

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 in this standard.
- (2) For packages and subprograms that are defined to be dependent on implementation options, either the runtime behavior shall be as defined by this standard or references to the name shall be rejected at compile or link time.
- (3) The system may provide additional facilities not required by this standard. Nonstandard extensions shall be identified as such in the system documentation. Nonstandard extensions, when used, may change the behavior of functions or facilities defined by this standard, but only if the application *activates an extension*, by reference to an implementation-defined extension package in a **with** clause, calling a subprogram in an implementation-defined extension package, or using an implementation-defined configuration pragma (see 10.1.5 (8) of {1}). The conformance document shall define an environment in which an application can be run with the behavior specified by this standard. In no case except package name conflicts shall such an environment require modification of a Strictly Conforming POSIX.5 Application. An implementation shall not change package specifications in this standard except by the following:

6) IETF documents can be obtained in printed form from the Network Information Center, Network Solutions, 14200 Park Meadow Drive, Suite 200, Chantilly, VA 22021, USA, or in electronic form via FTP over the Internet from nic.ddn.mil.

- (a) Adding **with** clauses, pragmas, representation specifications, and comments.
- (b) Replacing instances of the words *implementation-defined* with appropriate value(s).
- (c) Optionally changing the type `POSIX_Character` (see 2.4.2).
- (d) Adding or changing private parts.
- (e) Making any other changes that are lexically transparent to Ada compilers.
- (f) Adding specific declarations for which permission is explicitly granted in the text of this standard.
- (g) For Ada 95 implementations, changing packages with names of the form `POSIX_XXX` to child packages of the package `POSIX` with names of the form `POSIX.XXX` and adding library-level renaming declarations of the form:

```
with POSIX.XXX;
package POSIX_XXX renames POSIX.XXX;
```

*NOTE: The implementor does not have permission to change underscores to dots in the **with** clauses of the package specifications defined by this standard.*

Packages defined for the sockets and XTI interfaces shall be renamed as indicated in Table 1.1.

Table 1.1 – Sockets and XTI Package Renaming

Ada 83 Package Name	Ada 95 Package Name
<code>POSIX_Event_Management</code>	<code>POSIX.Event_Management</code>
<code>POSIX_Sockets</code>	<code>POSIX.Sockets</code>
<code>POSIX_Sockets_Local</code>	<code>POSIX.Sockets.Local</code>
<code>POSIX_Sockets_Internet</code>	<code>POSIX.Sockets.Internet</code>
<code>POSIX_Sockets_ISO</code>	<code>POSIX.Sockets.ISO</code>
<code>POSIX_XTI</code>	<code>POSIX.XTI</code>
<code>POSIX_XTI_Internet</code>	<code>POSIX.XTI.Internet</code>
<code>POSIX_XTI_ISO</code>	<code>POSIX.XTI.ISO</code>
<code>POSIX_XTI_mOSI</code>	<code>POSIX.XTI.mOSI</code>

- (h) For implementations that only support the original Ada language standard {B5}, changing Ada 95 general access types to ordinary access types.
- (i) Where a subprogram has a parameter with a default value expression in the final position, omitting the default value expression and providing an equivalent overloaded definition of the same function name, without the last parameter, if this change results in a gain in efficiency. For example,

```
procedure P(X: T:= E);
```

may be replaced by

```
procedure P(X: T);
procedure P;
```

Calling the version of procedure `P` without the parameter `X` shall have the same effect as calling `P(X=>E)`.

- (4) Only the following types may be implemented using Ada access types:

- POSIX.POSIX_String_List
- POSIX_Asynchronous_IO.AIO_Descriptor
- POSIX_Condition_Variables.Attributes
- POSIX_Condition_Variables.Condition
- POSIX_Condition_Variables.Condition_Descriptor
- POSIX_Event_Management.File_Descriptor_Set
- POSIX_Message_Queues.Message_Queue_Descriptor
- POSIX_Mutexes.Attributes
- POSIX_Mutexes.Mutex
- POSIX_Mutexes.Mutex_Descriptor
- POSIX_Process_Environment.Environment
- POSIX_Process_Primitives.Process_Template
- POSIX_Semaphores.Semaphore
- POSIX_Semaphores.Semaphore_Descriptor
- POSIX_Sockets.IO_Vector_Array_Pointer
- POSIX_Sockets.Socket_Address_Info_List
- POSIX_Sockets_Internet.Database_Array_Pointer
- POSIX_Sockets_Internet.Internet_Socket_Address_Pointer
- POSIX_Sockets_ISO.ISO_Socket_Address_Pointer
- POSIX_Sockets_Local.Local_Socket_Address_Pointer
- POSIX_XTI.Octet_Buffer_Pointer
- POSIX_XTI.Protocol_Option_List
- POSIX_XTI.Protocol_Option_List_Pointer
- POSIX_XTI_Internet.Internet_XTI_Address_Pointer
- POSIX_XTI_Internet.Database_Array_Pointer
- POSIX_XTI_ISO.ISO_XTI_Address_Pointer
- POSIX_XTI_mOSI.mOSI_XTI_Address_Pointer
- POSIX_XTI_mOSI.Presentation_Context_List
- POSIX_XTI_mOSI.Syntax_Object_List

— Any other type specifically defined as an Ada access type in this standard

No other types defined in this standard shall be implemented as access types or types with components of an access type.

- (5) A POSIX implementation shall not raise `Program_Error` on elaboration of a POSIX package or on execution of a POSIX subprogram, due to elaboration order dependencies in the POSIX implementation.
- (6) Except as explicitly provided for in this standard, `POSIX.POSIX_Error` and `Standard.Storage_Error` are the only exceptions that shall be raised by operations declared in this standard.
- (7) Wherever this standard states a requirement on an implementation, without any other explicit qualification, the requirement applies to all conforming implementations.

1.3.1.2 Documentation

A conformance document 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 equivalently 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 software standards approved by ISO/IEC or any ISO/IEC member body that are available for use by a ISO/IEC Strictly Conforming POSIX.5 Application (see 1.3.2.2.1). Applicable characteristics whose documentation is required by one of these standards, or by standards of government bodies, may also be included.

The document shall describe the contents of the packages `POSIX`, `POSIX_Limits`, `POSIX_Options`, `POSIX_Configurable_File_Limits`, and `POSIX_Configurable_System_Limits` and shall state values, the conditions under which those values may change, and the limits of such variations, if any.

The conformance document shall describe the behavior of the implementation for all implementation-defined features defined in this standard. This requirement shall be met by listing these features and either providing 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 subclause shall be present in the conformance document.

The phrase “shall be documented” in this standard means 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 symbolic constants in the packages `POSIX`, `POSIX_Limits`, and `POSIX_Options` provide declarations for a range of variation in the standard that is implementation defined. The packages `POSIX_Configurable_System_Limits` and `POSIX_Configurable_File_Limits` contain subprograms that may further constrain the implementation-defined ranges.

An implementation is required to define all of the subprograms for all of the operations defined in this standard, including those whose implementation is optional. If an unimplemented feature is used, the exception `POSIX.POSIX_Error` shall be raised, with the error code `POSIX.Operation_Not_Implemented`.

1.3.2 Application Conformance

All applications claiming conformance to this standard shall fall within one of the categories defined in the following subclauses.

1.3.2.1 Strictly Conforming POSIX.5 Application

A Strictly Conforming POSIX.5 Application is an application that requires only the facilities described in this standard and {1}. Such an application shall be designed to work for all behaviors and values allowed by this standard, including behaviors that are unspecified or implementation defined. Such applications are permitted to adapt to the availability of facilities whose availability is indicated by the values from the packages POSIX, POSIX_Limits, POSIX_Options, POSIX_Configurable_File_Limits, and POSIX_Configurable_System_Limits.

1.3.2.2 Conforming POSIX.5 Application

There are three categories of *Conforming POSIX.5 Applications*:

1.3.2.2.1 ISO/IEC Conforming POSIX.5 Application

An ISO/IEC Conforming POSIX.5 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.5 Application

A <National Body> Conforming POSIX.5 Application differs from an ISO/IEC Conforming POSIX.5 Application in that it also may use specific standards of a single ISO/IEC member body, referred to here as <National Body>. Such an application shall include a statement of conformance that documents all options and limit dependencies and all other <National Body> standards used.

1.3.2.2.3 Conforming POSIX.5 Application Using Extensions

A Conforming POSIX.5 Application Using Extensions is an application that differs from other Conforming POSIX.5 Applications 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.5 Application. A Conforming POSIX.5 Application Using Extensions shall be either an ISO/IEC Conforming POSIX.5 Application Using Extensions or a <National Body> Conforming POSIX.5 Application Using Extensions. (See 1.3.2.2.1 and 1.3.2.2.2.)

Section 2: Terminology and General Requirements

2.1 Editorial Conventions

Text with no marginal bars is carried over unchanged from POSIX.5b.

Text with grey marginal bars tagged with the letter “c” is new or substantially changed between POSIX.5b and POSIX.5c.

A summary of typographical conventions followed by this standard is shown in Table 2.1.

Table 2.1 – Typographical Conventions

Reference	Example
Ada Identifier	<code>POSIX_String_List</code>
Ada Implementation Defined Element	<i>implementation-defined</i>
Ada Reserved Word	generic
Bibliographic Citation	{B1}
C-Language Data Type	<i>long</i>
C-Language Error Number	[EINTR]
C-Language Function	<i>system()</i>
C-Language Global External	<i>errno</i>
C-Language Header	<code><sys/stat.h></code>
Citation of Normative Reference	ISO/IEC 8602 {9}
Conceptual Attribute	Message Length
Cross-Reference: Section	Section 2
Cross-Reference: Annex	Annex C
Cross-Reference: Clause	2.1
Cross-Reference: Subclause	2.1.2, 2.1.2.3, etc.
Environment Variable	PATH
Figure Reference	Figure 7-3
File Name	/tmp
Implementation Limit	Realtime Signals Maximum
Implementation Option	Job Control option
Table Reference	Table 6-1

Long Ada identifiers are occasionally hyphenated for typographic purposes only; none of the identifiers in this standard have names that include a hyphen.

Qualified name notation, with the package name, is generally used for names that are not defined within the section where the name is used. The exceptions to this rule are names that are defined by the Ada language and certain names defined in the package `POSIX` that are used so frequently that the fully qualified form would be merely cumbersome (e.g., `POSIX_String` is used instead of `POSIX.POSIX_String`).

Numbers within braces, such as “{1},” represent citations. If the number within braces appears alone, it corresponds to that of one of the normative references (1.2). A number preceded by a B, such as “{B1}”, corresponds to one of bibliographic entries listed in Annex A. Bibliographic entries are for information only.

Defined terms are shown in three styles, depending on context:

- (1) Terms defined in 2.2.1 and 2.2.2 are expressed as subclause titles. Alternative forms of the terms appear in [brackets].
- (2) The initial appearances of other terms, applying to a limited portion of the text, are in *italics*.
- (3) Subsequent appearances of the term are in Roman font.

Filenames and pathnames are shown in **Courier**. When a pathname is shown starting with “**\$HOME**/”, this indicates the remaining components of the pathname are to be related to the directory named by the **HOME** environment variable.

Defined names that are usually in lowercase are never used at the beginning of a sentence or anywhere else that regular English usage would require capitalization.

Parenthetical expressions within normative text also contain normative information. The general typographic hierarchy of parenthetical expressions is:

{ [()] }

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

Annexes marked as *normative* are parts of the standard that pose requirements, exactly the same as the numbered sections, but they follow the body of the standard for clarity of exposition. *Informative* annexes are for information only and pose no requirements. The index, and all material preceding page 1 of this standard (the front matter) are also only informative.

Notes that appear in a small italic font have two different meanings, depending on their location:

- When they are within the normal text of the document, they are the same as footnotes—informative, posing no requirements on implementations or applications.
- When they are attached to tables or figures, they are normative, posing requirements.

Text marked as examples (including the use of “*e.g.*”) is for information only.

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

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.(Quoted from POSIX.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. (Quoted from POSIX.1 {2}.)

2.2.1.3 may: An indication of an optional feature.

- (1) 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.
- (2) With respect to Strictly Conforming POSIX.5 Applications, the word *may* means that the optional feature shall not be used.

(Paraphrased from POSIX.1 {2}.)

2.2.1.4 obsolescent: An indication that a certain feature may be considered for withdrawal in future revisions of this standard. (Quoted from POSIX.1 {2}.) Use of obsolescent features is deprecated.

NOTE: Obsolescent features are retained for upward compatibility from POSIX.5, or to maintain symmetry with certain aspects of the C-language binding of POSIX.1. Their use in Ada applications is discouraged.

2.2.1.5 protocol-specific: An indication that the implementation, or Strictly Conforming POSIX.5 Applications where appropriate, shall behave in a manner that is specified in D.1 and D.2 .

2.2.1.6 shall: An indication of a requirement on the implementation or on Strictly Conforming POSIX.5 Applications, where appropriate. (Paraphrased from POSIX.1 {2}.)

2.2.1.7 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.5 Applications.

(Paraphrased from POSIX.1 {2}.)

2.2.1.8 supported: A condition regarding optional functionality.(Quoted from POSIX.1 {2}.)

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 raise the exception indicating the unsupported case). If the functionality is not supported, the interface shall raise the exception indicating the unsupported case.

(Paraphrased from POSIX.1 {2}.)

2.2.1.9 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. (Quoted from POSIX.1 {2}.)

2.2.1.10 undefined: An indication that this standard imposes no portability requirements on the use by an application of an indeterminate value or the behavior of that application 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 1.3.2.2.3.

(Paraphrased from POSIX.1 {2}.)

2.2.1.11 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 1.3.2.2.3.

(Paraphrased from POSIX.1 {2}.)

2.2.2 General Terms

For the purposes of this standard, the following definitions apply:

2.2.2.1 abort deferred operation: An operation that always continues to completion without being affected by an abort. (Paraphrased from 9.8 (5) of the Ada RM {1}.)

Certain operations are required by the Ada language to be abort deferred. It is implementation defined whether other operations defined by this standard are abort deferred.

2.2.2.2 abort completion point: A point at which the execution of an aborted construct must complete. (Quoted from 9.8 (15) of the Ada RM {1}.)

2.2.2.3 abortive release: An abrupt termination of a network connection that may result in the loss of data.

2.2.2.4 absolute pathname: See the explanation of *pathname resolution* (2.3.11).

2.2.2.5 access mode: A form of access to a file, or an attribute of a file indicating the kind of operations that may be performed on the file. For more detail see the explanation of file access permissions in 2.3.7.

2.2.2.6 Ada I/O: The I/O operations defined in Annex A of the Ada RM {1} and further defined in Section 8 of this standard.

2.2.2.7 address space: The memory locations that can be referenced by a process. (Quoted from POSIX.1 {2}.)

2.2.2.8 ancillary data: Optional, protocol-specific or local-system-specific information. The information can be both local or end-to-end significant. It can be header information or part of the data portion. It can be protocol-specific and implementation- or system-specific.

2.2.2.9 application: A computer program that performs some desired function. For the purpose of standard, an application is a program that uses the interfaces defined in this standard.

2.2.2.10 application entity title: In OSI, a title that unambiguously identifies an application entity. An application entity title is composed of an application process title and an application entity qualifier (ISO/IEC 7498 {B3}).

2.2.2.11 application entity qualifier: In OSI, a component of an application entity title that is unambiguous within the scope of the application process (ISO/IEC 7498 {B3}).

2.2.2.12 application process title: In OSI, a title that unambiguously identifies an application process. An application process title is a single name, which, for convenience, may be structured internally (ISO/IEC 7498 {B3}).

2.2.2.13 appropriate privileges: An implementation-defined means of associating privileges with an implementation-defined process with regard to the subprogram calls and options defined in this standard that need special privileges.

There may be zero or more such means.

(Paraphrased from POSIX.1 {2}.)

2.2.2.14 arm a timer: To start a timer measuring the passage of time, enabling the notification of a process when the specified time or time interval has passed. (Quoted from POSIX.1 {2}.)

2.2.2.15 asynchronous events: Events that occur independently of the execution of the application.

2.2.2.16 asynchronous I/O [AIO] operation: An I/O operation that does not of itself cause the task requesting the I/O to be blocked. An asynchronous I/O operation and the requesting task may be running concurrently. (Paraphrased from POSIX.1 {2}.)

2.2.2.17 asynchronous I/O [AIO] completion: The state of an asynchronous read or write operation when a corresponding synchronous read or write would have completed and any associated status attributes have been updated. (Paraphrased from POSIX.1 {2}.)

2.2.2.18 asynchronously generated signal: An occurrence of a signal that is generated by some mechanism external to the task receiving the signal; for example, via a call to `POSIX_Signals.Send_Signal` by another process or via the keyboard. Being asynchronous is a property of how an occurrence of the signal was generated and not a property of the signal. All signals may be generated asynchronously. The effect of an asynchronously generated occurrence of a *reserved signal* (3.3.1) on an Ada implementation is undefined.

NOTE: Only asynchronously generated signal occurrences are visible to an Ada application as signals. All signal occurrences that are not generated asynchronously are translated into exceptions.

Whether a signal occurrence is generated asynchronously is not to be confused with whether it is delivered or accepted. The difference between delivery and acceptance of a signal is defined in 3.3.1.

2.2.2.19 background process: A process that is a member of a background process group. (Quoted from POSIX.1 {2}.)

2.2.2.20 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. (Quoted from POSIX.1 {2}.)

2.2.2.21 bind: To assign a network address to an endpoint. |_c

2.2.2.22 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.

(Quoted from POSIX.1 {2}.)

2.2.2.23 blocking: Executing with `POSIX_IO.Non_Blocking` not set. See also *nonblocking* (2.2.2.102). |_c

2.2.2.24 blocking behavior: The effect on other tasks in the same partition when a task is blocked by a POSIX operation. Certain POSIX operations are required to block the calling task under defined conditions. For implementation-defined reasons a blocked task may prevent other tasks from executing. Names for blocking behaviors are defined in 2.4.1.5.

2.2.2.25 blocked task: An Ada task that is not running or ready to run. A task is either blocked or ready to run. While ready, a task competes for the available execution resources that it requires to run. An operation that causes a task to become blocked is said to *block* the task, and an operation that causes a task to no longer be blocked is said to *unblock* the task. (Paraphrased from Section 9 (10) of the Ada RM {1}.)

2.2.2.26 broadcast: The transfer of data from one endpoint to several endpoints, as it is described in RFC 919 {15} and RFC 922 {16}. |_c

2.2.2.27 byte: The measurement unit for data used in this standard. By common usage the term *byte* usually refers to eight bits of data, but within the context of this standard the size of a byte is implementation defined subject to the constraints given in 2.4.1.3. The size of a byte is given by the constant `POSIX.Byte_Size`.

NOTE: In the context of serial I/O (e.g., see Section 7), transmitting a byte of data may require the transmission of more bits than the size of a byte in memory, since, for example, stop bits and parity bits might be included.

2.2.2.28 canonical input processing: The processing of terminal input in the form of text lines. For more detail see the explanation of canonical mode input processing in `secdefCanonical Mode Input Processing`.

2.2.2.29 character: A sequence of one or more values of type `POSIX.POSIX_Character`.

NOTE: This definition of the term character applies when it is used by itself. It does not apply to qualified phrases containing the word character, such as POSIX character (2.2.2.126), graphic character (2.2.2.80), Ada character (2.3.3), and character special file (2.2.2.30).

2.2.2.30 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.

(Paraphrased from POSIX.1 {2}.)

2.2.2.31 character special file for use with XTI calls: A file of a particular type that is used for process-to-process communication as described in Section 17 of this standard. A character special file for use with XTI calls corresponds to a communications endpoint that uses a specified family of communications protocols.

2.2.2.32 child process: See *POSIX process* (2.2.2.128).

2.2.2.33 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.

(Quoted from POSIX.1 {2}.)

2.2.2.34 communication provider: A component of the system that provides the communications service through an endpoint.

2.2.2.35 communications endpoint: See *endpoint* (2.2.2.56).

2.2.2.36 communications user: An application that uses process-to-process communication services.

2.2.2.37 completion of a call: The execution of a construct or entity is complete when the end of that execution has been reached, or when a transfer of control causes it to be abandoned. Completion due to reaching the end of execution, or due to the transfer of control of an **exit**, **return**, **goto**, **requeue**, or of the selection of a **terminate** alternative is normal completion. Completion is abnormal when control is transferred out of a construct due to abort or the raising of an exception. (Quoted from 7.6.1(2) of the Ada RM {1}.)

2.2.2.38 condition variable: A synchronization object that allows a task to become blocked until it is unblocked by some event. The unblocking may occur spontaneously or as a result of a timeout or another task performing a condition-signalling operation on the condition variable. In use, condition variables are always associated with mutexes. See also 2.2.2.98.

2.2.2.39 connection: An association established between two or more endpoints for the transfer of data.⁷⁾

2.2.2.40 connection mode: The transfer of data in the context of a connection.⁸⁾ See also *connectionless mode* (2.2.2.41).

2.2.2.41 connectionless mode: The transfer of data other than in the context of a connection.⁸⁾ See also *connection mode* (2.2.2.40) and *datagram* (2.2.2.45).

2.2.2.42 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.

(Quoted from POSIX.1 {2}.)

2.2.2.43 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 (7.1) cause signals to be sent to all processes in the process group associated with the controlling terminal.

(Quoted from POSIX.1 {2}.)

2.2.2.44 current working directory: See *working directory* (2.2.2.198).

2.2.2.45 datagram: A unit of data transferred from one endpoint to another in connectionless mode service.

2.2.2.46 device: A computer peripheral or an object that appears to the application as such. (Quoted from POSIX.1 {2}.)

7) Connection establishment generally involves communication between the endpoints.

8) Definition is based on ISO/IEC 7498-1/ISO/IEC 7498 {B3}.

2.2.2.47 directory: A file that contains directory entries.

No two entries in a directory shall have the same filename.

(Quoted from POSIX.1 {2}.)

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

Several directory entries can associate different filenames with the same file.

(Quoted from POSIX.1 {2}.)

2.2.2.49 disarm a timer: To stop a timer from measuring the passage of time, thereby disabling any future process notifications (until the timer is armed again).
(Quoted from POSIX.1 {2}.)

2.2.2.50 dot: The filename consisting of a single dot character (.). (Quoted from POSIX.1 {2}.)

For the significance of dot see the explanation of pathname resolution in 2.3.11.

2.2.2.51 dot-dot: The filename consisting of (. .). (Quoted from POSIX.1 {2}.)

For the significance of dot-dot see the explanation of pathname resolution in 2.3.11.

2.2.2.52 effective group ID: An attribute of a process that is used in determining various permissions, including the file access permissions described in 2.3.7.

This value is subject to change during the process lifetime, as described in 4.1 and 3.1). See also *group ID* (2.2.2.81).

(Paraphrased from POSIX.1 {2}.)

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

This value is subject to change during the process lifetime, as described in 4.1 and 3.1). See also *user ID* (2.2.2.196).

(Paraphrased from POSIX.1 {2}.)

2.2.2.54 empty directory: A directory that contains, at most, entries for dot and dot-dot. (Quoted from POSIX.1 {2}.)

2.2.2.55 empty string [null string]: A zero-length array whose components are of some character type.

2.2.2.56 endpoint: An object that is created and maintained by a communications provider and used by applications for sending and receiving data; endpoints are used by the communications providers to identify the sources and destinations of data.

c

2.2.2.57 environment task: The anonymous task whose execution elaborates the library items of the declarative part of an active partition, and then calls the main subprogram, if there is one. (Paraphrased from 10.2 (8) of the Ada RM {1}.)

2.2.2.58 Epoch: A base reference time defined as 0 hours, 0 minutes, 0.0 seconds, 1 January 1970, Universal Coordinated Time (see Blair {B2}). (Paraphrased from POSIX.1 {2}.)

2.2.2.59 erroneous execution: In this standard as defined in 1.1.5 of the Ada RM {1}.

2.2.2.60 error code of a task: An attribute of a task that ordinarily specifies information about the most recent error that caused `POSIX_Error` to be raised. (See 2.4.6.)

2.2.2.61 event management: The mechanism that enables applications to register for and be made aware of external events such as data becoming available for reading. | c

2.2.2.62 Exec family of operations: The collection of operations that cause a new program to be executed, *i.e.*, the `Exec` and `Exec_Search` procedures in `POSIX_Unsafe_Process_Primitives` and the `Start_Process` and `Start_Process_Search` procedures in `POSIX_Process_Primitives`.

2.2.2.63 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. (Quoted from POSIX.1 {2}.)

Other characteristics of FIFOs are described in 5.2 and 6.1.

2.2.2.64 file: An object that can be written to, or read from, or both. (Quoted from POSIX.1 {2}.)

A file has certain attributes, including access permissions and type. File types include regular file, character special file, block special file, FIFO special file, socket, character special file for use with XTI calls, and directory. Other types of files may be defined by the implementation. (Paraphrased from POSIX.1 {2}.) | c

2.2.2.65 file description: See *open file description* (2.2.2.108).

2.2.2.66 file descriptor: A per-process unique nonnegative integer value used to identify an open file for the purpose of file access. (Quoted from POSIX.1 {2}.)

2.2.2.67 file group class: A property of a file indicating access permissions for a process related to the group identification of the process.

A process is in the file group class of a file if the process is not in the file owner class and if the effective group ID or one of the supplementary group IDs of the process matches the group ID associated with the file. Other members of the class may be implementation defined. (Paraphrased from POSIX.1 {2}.)

2.2.2.68 filename: A nonempty string that is used to name a file.

A filename consists of, at most, `POSIX_Limits.Filename_Maxima` 'Last components of type `POSIX.POSIX_Character`. The characters composing the name may be selected from the set of all the character values excluding the slash character and the null character. The filenames dot and dot-dot have special meaning, as specified in the explanation of *pathname resolution* (2.3.11). A filename is sometimes referred to as a pathname component.

(Paraphrased from POSIX.1 {2}.)

2.2.2.69 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.

(Quoted from POSIX.1 {2}.)

2.2.2.70 file other class: A property of a file indicating access permissions for a process related to the user and group information of the process.

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.

(Quoted from POSIX.1 {2}.)

2.2.2.71 file owner class: A property of a file indicating access permissions for a process related to the user identification of the process.

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.

(Quoted from POSIX.1 {2}.)

2.2.2.72 file permission: Information about a file that is used, along with other information, to determine whether a process has read, write, or execute/search permission to a file. (Quoted from POSIX.1 {2}.)

The file permission information is divided into three parts: owner, group, and other. Each part is used with the corresponding file class of processes. These permissions are contained in the *file mode*, as described in 5.1. The detailed usage of the file permission information in access decisions is described in *file access permissions* in 2.3.7. (Paraphrased from POSIX.1 {2}.)

2.2.2.73 file serial number: A per-file-system unique value for a file.

File serial numbers are unique throughout a file system.

(Quoted from POSIX.1 {2}.)

2.2.2.74 file system: A collection of files, together with certain of their attributes. (Quoted from POSIX.1 {2}.)

Each file system provides a separate binding of file serial numbers to files. A given file serial number is associated with at most one file in a file system, but it may refer to distinct files in distinct file systems. In other words, each file system defines a new *name space*, giving meaning to the *names* (file serial numbers) that designate files.

2.2.2.75 first open, of a file: The act when a process opens a file, message queue, or shared memory object that is not currently open within any process. (Paraphrased from POSIX.1 {2}.)

2.2.2.76 flow control: The mechanism employed by a communications provider that constrains a sending entity to wait until the receiving entities can safely receive additional data without loss.

2.2.2.77 foreground process: A process that is a member of a foreground process group. (Quoted from POSIX.1 {2}.)

2.2.2.78 foreground process group: A group of processes that 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 7.1.0.7.

(Quoted from POSIX.1 {2}.)

2.2.2.79 foreground process group ID: The process group ID of the foreground process group. (Quoted from POSIX.1 {2}.)

2.2.2.80 graphic character: A sequence of one or more POSIX.POSIX_Characters representing a single graphic symbol.

2.2.2.81 group ID: A value identifying a group of system users.

Each system user is a member of at least one group. A group ID is defined in the package POSIX_Process_Identification(4.1). 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.

(Quoted from POSIX.1 {2}.)

2.2.2.82 host byte order: The native representation of an integer: unsigned integer m is the representation in *host byte order* of bit string $b_n b_{n-1} \dots b_0$ (where b_n is the *most significant*, or *highest order* bit, and b_0 is the *least significant* or *lowest order* bit) if $m = 2^n * b_n + 2^{n-1} * b_{n-1} + \dots + 2^0 * b_0$.

2.2.2.83 job control: A facility that allows users to stop (suspend) selectively the execution of processes and continue (resume) their execution at a later time. (Quoted from POSIX.1 {2}.)

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 by the subtype `Job_Control_Support` in package `POSIX_Options` or at run time by the value returned by the function `Job_Control_Is_Supported` in package `POSIX_Configurable_System_Limits`; see 2.4 and 4.5. (Paraphrased from POSIX.1 {2}.)

2.2.2.84 last close: The act of a process closing a file, message queue, or shared memory object that results in the file, message queue, or shared memory object no longer being open within any process. (Paraphrased from POSIX.1 {2}.)

2.2.2.85 linger: To wait for a period of time before terminating a connection to allow outstanding data to be transferred. | c

2.2.2.86 link: See *directory entry* (2.2.2.48).

2.2.2.87 link count: The number of directory entries that refer to a particular file. (Quoted from POSIX.1 {2}.)

2.2.2.88 local interprocess communication [local IPC]: The transfer of data between processes in the same system. | c

2.2.2.89 login: The unspecified (by this standard) activity by which a user gains access to the system. (Paraphrased from POSIX.1 {2}.)

Each login shall be associated with exactly one login name.

(Paraphrased from POSIX.1 {2}.)

2.2.2.90 login name: A user name that is associated with a login. (Quoted from POSIX.1 {2}.)

2.2.2.91 map a range of addresses: To create an association process's address space and a range of physical memory or some memory 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.

(Quoted from POSIX.1 {2}.)

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

When used in conjunction with `Map_Memory`, `Open_And_Map_Shared_Memory`, or `Open_Or_Create_And_Map_Shared_Memory`, a memory object will appear in the address space of the calling process.

(Paraphrased from POSIX.1 {2}.)

2.2.2.93 memory-resident: Managed by the implementation in such a way as to provide an upper bound on memory access times. (Quoted from POSIX.1 {2}.)

2.2.2.94 message: Information that can be transferred among tasks (possibly in different processes) by being added to and removed from a message queue.

A message queue consists of a fixed-size buffer.

(Quoted from POSIX.1 {2}.)

2.2.2.95 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.

(Quoted from POSIX.1 {2}.)

2.2.2.96 message queue descriptor: A per-process unique value used to identify an open message queue.

2.2.2.97 mode: A collection of attributes that specifies the type of a file and its access permissions. (Quoted from POSIX.1 {2}.)

For more detail see the explanation of file access permissions in 2.3.7.

2.2.2.98 mutex: A synchronization object used to allow multiple tasks (possibly in different processes) to serialize their access to shared data or other shared resources.

The name derives from the capability it provides, namely, mutual exclusion.

(Paraphrased from POSIX.1 {2}.)

2.2.2.99 mutex owner: The task that last locked a mutex, until that same task unlocks the mutex.

2.2.2.100 network address: A network-visible identifier used to designate specific endpoints in a network. Specific endpoints on host systems shall have addresses, and host systems may also have addresses.

2.2.2.101 network byte order: An implementation-defined way of representing an integer so that, when transmitted over a network via a network endpoint, the integer shall be transmitted as an appropriate number of octets with the most significant octet first. This term has been used in historical systems and base documents to denote this representation. This representation is used in many networking protocols, including Internet and OSI. However, it should not be assumed this representation is useful with all network protocols.

2.2.2.102 nonblocking: Executing with `POSIX_IO.Non_Blocking` set. When executing with `POSIX_IO.Non_Blocking` set, functions do not wait for protocol events (e.g., acknowledgments) to occur before returning control. See also *blocking* (2.2.2.23).

c

2.2.2.103 noncanonical input processing: The processing of terminal input as uninterpreted characters. (See Section 7.)

2.2.2.104 null character: The value `POSIX_Character'Value(0)`, if defined. The use of a null character in filenames or environment variable names or values produces undefined results. Implementations shall not return null characters to applications in filenames or environment variable names or values.

2.2.2.105 null string: See *empty string* (2.2.2.55).

2.2.2.106 octet: Unit of data representation that consists of eight contiguous bits. |_c

2.2.2.107 open file: A file that is currently associated with a file descriptor. (Quoted from POSIX.1 {2}.)

2.2.2.108 open file description: A record of how a process or group of processes is 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 6.1.7), and file access modes are attributes of an open file description.

(Quoted from POSIX.1 {2}.)

2.2.2.109 orderly release: The graceful termination of a network connection with no loss of data. |_c

2.2.2.110 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 session of the group. (Paraphrased from POSIX.1 {2}.)

An orphaned process group is no longer a member of the session of the process that created it. A process group can become orphaned when all other members of the session exit.

2.2.2.111 page: The granularity of memory mapping and locking, *i.e.*, a fixed-length contiguous range of the address space of a process. 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 space can be locked into memory (*i.e.*, made memory-resident) on page boundaries and in integral multiples of pages. (Paraphrased from POSIX.1 {2}.)

NOTE: There is no implied requirement that usage of the term page in this interface for memory mapping necessarily be the same as the term might be used in a virtual memory implementation.

2.2.2.112 page size: The number of storage units in a page.

NOTE: Unlike POSIX.1, in this standard the page size is given in storage units rather than bytes. Storage units are used in this standard for consistency with the units used in the address arithmetic provided by `System.Storage_Elements`, in case the system storage unit is not one byte.

2.2.2.113 parent directory:

- (1) When discussing a given directory, the directory that contains a directory entry for the given directory. The parent directory 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.

(Quoted from POSIX.1 {2}.)

2.2.2.114 parent process: See *POSIX process* (2.2.2.128).

2.2.2.115 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 lifetime of the creator has ended, the parent process ID is the process ID of an implementation-defined system process.

(Quoted from POSIX.1 {2}.)

2.2.2.116 partition: A partition is a program or part of a program that can be invoked from outside the Ada implementation. Each partition may run in a separate address space, possibly on a separate computer. An *active partition* is a partition that contains at least one task. Every active partition has an *environment task*, on which all the other tasks of that partition depend. (Paraphrased from 10.2 (2) of the Ada RM {1}.) An active partition corresponds to a POSIX process.

2.2.2.117 path prefix: A pathname, with an optional ending slash, that refers to a directory. (Quoted from POSIX.1 {2}.)

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

A pathname consists of, at most, `POSIX_Limits.Pathname_Maxima`' Last components of type `POSIX.POSIX_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 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 exactly the same as a single slash. The interpretation of the pathname is explained in 2.3.11.

(Paraphrased from POSIX.1 {2}.)

2.2.2.119 pathname component: See *filename* (2.2.2.68).

2.2.2.120 permission: See *file permission* (2.2.2.72).

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

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

(Paraphrased from POSIX.1 {2}.)

2.2.2.122 pipe: An object accessed by one of the pair of file descriptors created by the `POSIX_IO.Create_Pipe` procedure.

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

(Paraphrased from POSIX.1 {2}.)

2.2.2.123 polling: A scheduling scheme whereby the local process periodically checks until the prespecified events (*e.g.*, read, write) have occurred.

2.2.2.124 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.

(Paraphrased from POSIX.1 {2}.)

2.2.2.125 portable pathname character set: The set of characters from which portable pathnames are constructed. The set contains all the characters of the portable filename set, plus the character slash (/).

2.2.2.126 POSIX character: A value of the type `POSIX_Character`. An array of POSIX characters, of type `POSIX_String` is called a *POSIX string*.

2.2.2.127 POSIX I/O: The I/O operations defined by this standard, except for Section 8.

2.2.2.128 POSIX process: A conceptual object, having an associated address space, one or more threads of control executing within that address space, a collection of system resources required for execution, and certain other attributes. A

POSIX process is said to perform an action if any of the conceptual threads of control within it performs the action. See also 2.3.1.

Many of the *system resources* defined by this standard are shared among all of the threads of control within a process. These resources include (but are not limited to) the process ID; the parent process ID; process group ID; session membership; real, effective, and saved set-user-ID; real, effective, and saved set-group-ID; supplementary group IDs; current working directory; root directory; file mode creation mask; file descriptors; timers; file locks; mutexes; environment variables; process argument list; and possibly, a process signal mask. For more detail on the POSIX signal model see 3.3.1.

A process is created by another process with procedures `POSIX_Process_Primitives.Start_Process`, `POSIX_Process_Primitives.Start_Process_Search`, or the function `POSIX_Unsafe_Process_Primitives.Fork`. The process that issues `Start_Process`, `Start_Process_Search`, or `Fork` is known as the parent process. The newly created process is the child process.

2.2.2.129 potentially blocking operation: An operation that is not allowed within a protected action, because it may be required to block the calling task. (Paraphrased from 9.5.1 (8) of the Ada RM {1}.) Certain operations are defined by the Ada language to be potentially blocking. In addition, every operation defined by this standard whose effect is defined as blocking the calling task under any circumstance is also a potentially blocking operation.

2.2.2.130 priority: The general term for an integer-valued attribute of processes, tasks, messages, and asynchronous I/O operations, whose value is used in selecting among entities of the same kind. Numerically higher values represent higher priorities and are given preference for selection over lower priorities.

The *priority of an Ada task* is an integer that indicates a degree of urgency and is the basis for resolving competing demands of tasks for resources. Unless otherwise specified, whenever tasks compete for processors or other implementation-defined resources, the resources are allocated to the task with the highest priority value. The *base priority* of a task is the priority with which it was created, or to which it was later set by `Dynamic_Priorities.Set_Priority` (defined in D.5 of the Ada RM {1}). At all times, a task also has an *active priority*, which generally reflects its base priority as well as any priority it inherits from other sources. *Priority inheritance* is the process by which the priority of a task or other entity (e.g., a protected object, defined in D.3 of the Ada RM {1}) is used in the evaluation of the active priority of another task. At any time, the active priority of a task is the maximum of all the priorities the task is inheriting at that instant. (Paraphrased from D.1 (20) of the Ada RM {1}.)

2.2.2.131 privilege: See *appropriate privileges* (2.2.2.13).

2.2.2.132 process: See *POSIX process* (2.2.2.128).

2.2.2.133 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 its process group ID. A newly created process joins the process group of its creator.

(Quoted from POSIX.1 {2}.)

2.2.2.134 process group ID: A unique value identifying a process group during its lifetime.

A process group ID shall not be reused by the system until the process group lifetime ends.

(Quoted from POSIX.1 {2}.)

2.2.2.135 process group leader: The unique process, within a process group, that created the process group. (Quoted from POSIX.1 {2}.)

2.2.2.136 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 process lifetime of the last process or to the last remaining process calling the `Set_Process_Group_ID` procedure. (Paraphrased from POSIX.1 {2}.)

2.2.2.137 process ID: A unique value identifying a process during its lifetime.

The process ID is a value of the type `Process_ID` defined in the package `POSIX_Process_Identification` (4.1). A process ID shall not be reused by the system until the process lifetime ends. In addition, if a process group exists where the process ID of the process group leader is equal to that process ID, that process ID shall not be reused by the system until the process group lifetime ends.

An implementation shall reserve a value of process ID for use by system processes. A process that is not a system process shall not have this process ID.

2.2.2.138 process lifetime: A 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, it is considered active. Its threads 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_For_Child_Process` procedure 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.

(Paraphrased from POSIX.1 {2}.)

2.2.2.139 process-to-process communication: The transfer of data between processes.

2.2.2.140 program: A set of partitions, which can execute in parallel with one another, possibly in a separate address space and possibly on a separate computer. (Paraphrased from 10.2 (1) of the Ada RM {1}.)

2.2.2.141 protocol: A set of semantic and syntactic rules for exchanging information.

2.2.2.142 protocol independent interface: An interface that enables the application to be insulated from the specifics of the underlying protocol stack which provides the communication services. Protocol independent interfaces allow the application to be written so that it can be ported to various protocol stacks.

2.2.2.143 protocol engine: A component of the DNI implementation model that is a conceptual machine that implements a particular communications protocol profile.

2.2.2.144 protocol event: In the DNI implementation model, an event that is generated by a protocol engine and queued for attention by the event handler.

2.2.2.145 protocol profile: A set of one or more protocol definitions and, where applicable, the identification of chosen classes, subsets, option and parameters of those definitions, necessary for accomplishing a particular function. In the context of this standard, a protocol profile can be thought of as a vertical slice through a layered set of communications protocols.

2.2.2.146 read-only file system: A file system that has implementation-defined characteristics restricting modifications. (Quoted from POSIX.1 {2}.)

2.2.2.147 ready task: A task that is not blocked. The ready tasks include those that are running as well as those that are waiting for a processor. See also *blocked task* (2.2.2.25).

NOTE: For consistency, this standard uses the definition of ready from the Ada RM {1}, which includes running tasks among those that are ready. Thus the meaning of ready is close to the meaning of runnable as used in POSIX.1, but the two terms differ in that ready includes running threads. This difference in terminology requires differences in several descriptions between this Ada binding and the base POSIX standards to preserve compatible semantics.

2.2.2.148 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. (Quoted from POSIX.1 {2}.)

This value is subject to change during the process lifetime, as described in 4.1. See also *group ID* (2.2.2.81).

2.2.2.149 real user ID: The attribute of a process that, at the time of process creation, identifies the user who created the process. (Quoted from POSIX.1 {2}.)

This value is subject to change during the process lifetime, as described in 4.1. See also *user ID* (2.2.2.196).

2.2.2.150 record: A collection of related data units or words that itself is treated as a unit.

2.2.2.151 referenced shared memory object: A shared memory object that is open or has one or more mappings defined on it. (Quoted from POSIX.1 {2}.)

2.2.2.152 region:

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

(Quoted from POSIX.1 {2}.)

2.2.2.153 regular file: A file that is a randomly accessible sequence of bytes, with no further structure imposed by the system. (Quoted from POSIX.1 {2}.)

2.2.2.154 relative pathname: See the explanation of *pathname resolution* (2.3.11).

2.2.2.155 reserved signal: Signals that the application cannot accept and for which the application cannot modify the signal action or masking because the signals are reserved for use by the Ada language implementation. The reserved signals defined by this standard are `Signal_Abort`, `Signal_Alarm`, `Signal_Floating_Point_Error`, `Signal_Illegal_Instruction`, `Signal_Segmentation_Violation`, `Signal_Bus_Error`. If the implementation supports any signals besides those defined by this standard, the implementation may also reserve some of those.

2.2.2.156 resolution [time]: The minimum time interval that a clock can measure or whose passage a timer can detect. (Quoted from POSIX.1 {2}.)

2.2.2.157 root directory: A directory, associated with a process, that is used in pathname resolution for pathnames that begin with a slash. (Quoted from POSIX.1 {2}.)

2.2.2.158 running task: The task currently being executed by a processor.

2.2.2.159 saved set-group-ID: When the Saved IDs option is implemented, an attribute of a process that allows some flexibility in the assignment of the effective group ID attribute, as described for `Set_Group_ID`. For more detail see 2.3.7 and Section 4. (Paraphrased from POSIX.1 {2}.)

2.2.2.160 saved set-user-ID: When the Saved IDs option is implemented, an attribute of a process that allows some flexibility in the assignment of the effective user ID attribute, as described for `Set_User_ID`. For more detail see 2.3.7 and Section 4. (Paraphrased from POSIX.1 {2}.)

2.2.2.161 semaphore: A shareable resource that has a nonnegative integer value. When the value is zero, is a (possibly empty) set of tasks is awaiting the availability of the semaphore.

2.2.2.162 semaphore decrement operation: An operation that decrements the value of a semaphore, blocking until this is possible.

If, prior to the operation, the value of the semaphore is zero, the semaphore decrement operation shall cause the calling task to be blocked and added to the set of tasks (possibly in different processes) awaiting the semaphore. Otherwise, the semaphore value is decremented. For more detail see 11.1.7.

2.2.2.163 semaphore increment operation: An operation that increments the value of a semaphore or unblocks a waiting task.

If, prior to the operation, any tasks are awaiting the semaphore, then some task from that set shall be removed from the set and be unblocked. Otherwise, the semaphore value shall be incremented. For more detail see 11.1.8. `indexx[blocking]semaphore`

2.2.2.164 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 by the procedure `Create_Session` in the package `POSIX_Process_Identification`. Implementations that support `Set_Process_Group_ID` can have multiple process groups in the same session.

(Paraphrased from POSIX.1 {2}.)

2.2.2.165 session leader: A process that has created a session. (Quoted from POSIX.1 {2}.) For more detail on the role of session leader see 4.1.2.

2.2.2.166 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. (Quoted from POSIX.1 {2}.)

2.2.2.167 shared memory object: An object that represents memory that can be mapped concurrently into the address space of more than one process. (Quoted from POSIX.1 {2}.)

These named regions of storage may be independent of the file system and can be mapped into the address space of one or more processes to allow them to share the associated memory.

2.2.2.168 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.

(Quoted from POSIX.1 {2}.)

2.2.2.169 signal-driven mode: A mode of operation in which the signal `POSIX_Signals.Signal_IO` is sent to the owner of a socket whenever an I/O operation becomes possible on that socket. In this mode, `POSIX_Signals.Signal_IO` is sent when additional data could be sent on the socket, when new data arrives to be received on a socket, or a state transition occurs that would allow a send or receive call to return status without blocking. Signal-driven mode is enabled by setting the `POSIX_IO.Signal_When_Socket_Ready` flag on the socket and disabled by resetting the `POSIX_IO.Signal_When_Socket_Ready` flag. The default mode for signal driven mode is disabled.

2.2.2.170 signal queueing: When queueing is enabled for a signal, occurrences of that signal are queued in FIFO order and information is included if the signal is from a source that supplies information. Otherwise, the signal queue may be only one occurrence deep and it is implementation defined whether the data are included. Support for signal queueing is governed by the Realtime Signals option. Not all signals may support queueing.

2.2.2.171 slash: The literal character `"/"`.

This character is also known as *solidus* in ISO 8859-1 ().

(Quoted from POSIX.1 {2}.)

2.2.2.172 socket: A file of a particular type that is used as a communications endpoint for process-to-process communication as described in Section 18.

2.2.2.173 socket address: An address associated with a socket or remote endpoint. The address may include multiple parts, such as a network address associated with a host system and an identifier for a specific endpoint.

2.2.2.174 storage unit: The length of an addressable element of storage in the machine, measured in bits. (Every storage element has the same size.) This term is used here as defined in 13.7 (31) of the Ada RM {1}, where it is declared as `System.Storage_Unit`.

NOTE: The storage unit is very likely to be one byte, but this is not a requirement. For example, it might be 32 or 64 bits.

2.2.2.175 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.

(Quoted from POSIX.1 {2}.)

2.2.2.176 supplementary group ID: An attribute of a process, used in determining file access permissions. (Quoted from POSIX.1 {2}.)

A process has group IDs in addition to the effective group ID. The size of this list of supplementary group IDs is specified at compile time by `Groups_Maxima` in package `POSIX_Limits`, or at run time by the value of the function `Groups_Maximum` in package `POSIX_Configurable_System_Limits`. The supplementary group IDs of a process are set to the supplementary group IDs of the parent process when the process is created. Whether the effective group ID of a process is included in or omitted from its list of supplementary group IDs is unspecified.

2.2.2.177 synchronized I/O completion: The state of an I/O operation that has either been successfully transferred or diagnosed as unsuccessful. (Quoted from POSIX.1 {2}.)

2.2.2.178 synchronized I/O data integrity completion: A degree of completion for an I/O operation that occurs when:

- (1) For read, the operation has been completed or diagnosed as unsuccessful. The read is complete only when an image of the data has been successfully transferred to the requesting task. 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, the operation has been completed or diagnosed as 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 (Last Access Time, Last Modification Time, Last Status Change Time) need not be successfully transferred prior to returning to the calling task.

(Paraphrased from POSIX.1 {2}.)

2.2.2.179 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 Last Access Time, Last Modification Time, Last Status Change Time) shall be successfully transferred prior to returning to the calling task. (Paraphrased from POSIX.1 {2}.)

2.2.2.180 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. (Quoted from POSIX.1 {2}.) See also *synchronized I/O file integrity completion* (2.2.2.178) and *synchronized I/O data integrity completion* (2.2.2.179).

2.2.2.181 synchronous I/O operation: An I/O operation that causes the task requesting the I/O to be blocked from further use of the processor until that I/O operation completes. (Quoted from POSIX.1 {2}.)

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