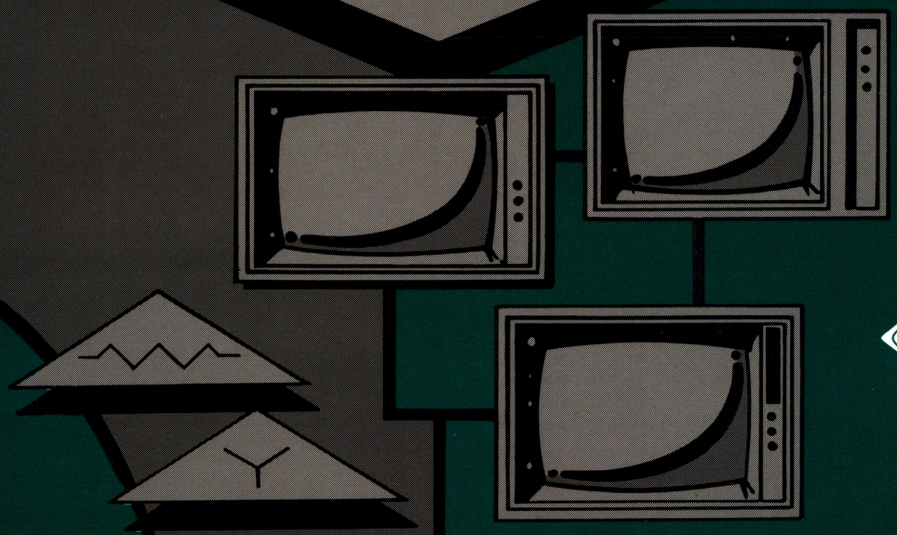
IEEE
Std 1003.5-1992

IEEE Standard for Information Technology

POSIX Ada

LANGUAGE INTERFACES

Part 1: Binding for System Application Program Interface (API)



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

AVA Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.5-1992

IEEE
Std 1003.5-1992

IEEE Standard for Information Technology– POSIX Ada Language Interfaces— Part 1: Binding for System Application Program Interface (API)

Sponsor

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

Approved June 18, 1992

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 to the application program interface (API). This standard supports application portability at the source level through the binding between ISO 8652: 1987 (Ada) and ISO/IEC 9945-1: 1990 (IEEE Std 1003.1-1990) (POSIX). 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, and system databases are covered. It also specifies behavior to support the binding that must be provided by the Ada compilation system. The emphasis in POSIX is on application programs, so the interfaces in this document are not sufficient to implement an Ada compilation system or a POSIX shell, nor is this standard meant to be sufficient for realtime applications.

Keywords: Ada, API, application portability, computer language bindings, open systems, operating system, portable application, POSIX, POSIX language bindings

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

Copyright © 1992 by the
Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1992
Printed in the United States of America

ISBN 1-55937-229-X

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

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

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

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

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

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

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

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

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

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.5-1992

Contents

PAGE

Introduction vi

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

Section 2: Terminology and General Requirements 7
2.1 Conventions 7
2.2 Definitions 9
2.3 General Concepts 20
2.4 Package `POSIX` 24
2.4.1 Constants and Static Subtypes 29
2.4.2 Characters and Strings 33
2.4.3 Option Sets 36
2.4.4 Error Codes and Exceptions 37
2.4.5 System Identification 41
2.5 Environment Description 41

Section 3: Process Primitives 45
3.1 Package `POSIX_Process_Primitives` 45
3.1.1 Process Template 45
3.1.2 Process Creation 50
3.1.3 Process Exit 52
3.1.4 Termination Status 54
3.1.5 Wait for Process Termination 55
3.2 Package `POSIX_Unsafe_Process_Primitives` 57
3.2.1 Process Creation 58
3.2.2 File Execution 59
3.3 Package `POSIX_Signals` 62
3.3.1 Signal Type 62
3.3.2 Signal Model 66
3.3.3 Standard Signals 68
3.3.4 Default Signal Actions 70
3.3.5 Signal Sets 72
3.3.6 Sending a Signal 73
3.3.7 Blocking and Unblocking Signals 74
3.3.8 Ignoring Signals 75
3.3.9 Controlling Delivery of `Signal_Child` Signal 76
3.3.10 Examining Pending Signals 77

Section 4: Process Environment 79
4.1 Package `POSIX_Process_Identification` 79

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.5-1992

4.1.1 Process Identification Operations . . . 79
4.1.2 Process Group Identification . . . 82
4.1.3 User Identification . . . 84
4.1.4 User and Group Identification . . . 85
4.2 Package POSIX_Process_Times . . . 86
4.2.1 Process Time Accounting . . . 86
4.3 Package POSIX_Process_Environment . . . 88
4.3.1 Argument List . . . 90
4.3.2 Environment Variables . . . 91
4.3.3 Process Working Directory . . . 94
4.4 Package POSIX_Calendar . . . 95
4.4.1 Obtaining Time Information From the System . . . 95
4.4.2 Operations on POSIX Times . . . 97
4.5 Package POSIX_Configurable_System_Limits . . . 98
4.5.1 Configurable System Limits . . . 99

Section 5: Files and Directories . . . 101
5.1 Package POSIX_Permissions . . . 101
5.1.1 File Permissions . . . 101
5.1.2 The Process Permission Set . . . 103
5.2 Package POSIX_Files . . . 104
5.2.1 Creating and Removing Files . . . 104
5.2.2 Inquiries on File Types . . . 108
5.2.3 Modifying File Pathnames . . . 109
5.2.4 Directory Iteration . . . 112
5.2.5 Updating File Status Information . . . 113
5.2.6 Check File Accessibility . . . 114
5.3 Package POSIX_File_Status . . . 116
5.3.1 Accessing File Status . . . 117
5.3.2 Accessing Status Information . . . 118
5.4 Package POSIX_Configurable_File_Limits . . . 119
5.4.1 File Limits . . . 119
5.4.2 File Restrictions . . . 123

Section 6: Input and Output Primitives . . . 125
6.1 Package POSIX_IO . . . 125
6.1.1 Open, Open_Or_Create, Is_Open, Close, Duplicate, Create_Pipe . . . 125
6.1.2 Read, Write . . . 131
6.1.3 Seek . . . 133
6.1.4 Is_A_Terminal, Get_Terminal_Name . . . 134
6.1.5 File Control . . . 135
6.2 Package POSIX_File_Locking . . . 136
6.2.1 Locking Files . . . 136

Section 7: Device- and Class-Specific Functions . . . 139
7.1 General Terminal Interface . . . 139
7.1.1 Interface Characteristics . . . 139
7.1.1.1 Opening a Terminal Device File . . . 139
7.1.1.2 Process Groups . . . 139

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.5-1992

7.1.1.3 The Controlling Terminal . . . 140
7.1.1.4 Terminal Access Control . . . 140
7.1.1.5 Input Processing and Reading Data . . . 141
7.1.1.6 Canonical Mode Input Processing . . . 142
7.1.1.7 Noncanonical Mode Input Processing . . . 142
7.1.1.8 Writing Data and Output Processing . . . 144
7.1.1.9 Special Characters . . . 144
7.1.1.10 Modem Disconnect . . . 146
7.1.1.11 Closing a Terminal Device File . . . 146
7.2 Package POSIX_Terminal_Functions . . . 146
7.2.1 Terminal Characteristics . . . 150
7.2.2 Input Modes . . . 152
7.2.3 Output Modes . . . 154
7.2.4 Control Modes . . . 154
7.2.5 Local Modes . . . 155
7.2.6 Retrieving and Defining Terminal Modes and Bits per Character . . . 157
7.2.7 Special Control Characters . . . 158
7.2.8 Noncanonical Controls . . . 159
7.2.9 Baud Rate Subprograms . . . 160
7.2.10 Line Control Functions . . . 160
7.2.11 Foreground Process Group ID . . . 162
7.2.12 Generating Terminal Pathname . . . 163

Section 8: Language-Specific Services for Ada . . . 165
8.1 Interoperable Ada I/O Services . . . 165
8.1.1 Requirements . . . 165
8.1.2 Additional Interpretation of the POSIX.1 {2} Standard . . . 170
8.2 Package POSIX_Supplement_to_Ada_IO . . . 170
8.2.1 Parsing Form Values . . . 170
8.2.2 Flushing Files . . . 172

Section 9: System Databases . . . 175
9.1 Package POSIX_User_Database . . . 175
9.1.1 Accessing Contents of a User Database Item . . . 175
9.1.2 Accessing User Database Items . . . 177
9.2 Package POSIX_Group_Database . . . 177
9.2.1 Accessing Contents of a Group Database Item . . . 177
9.2.2 Accessing Elements of the Group Item List of Members . . . 179
9.2.3 Accessing Group Database Items . . . 179

Annex A (informative) Bibliography . . . 181

Annex B (informative) Rationale and Notes . . . 183
B.1 General Rationale . . . 183
B.2 Terminology and General Requirements . . . 199
B.3 Process Primitives . . . 201
B.4 Process Environment . . . 217
B.5 Files and Directories . . . 223
B.6 Input and Output Primitives . . . 227

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.5-1992

B.7 Device- and Class-Specific Functions 231
B.8 Language-Specific Services for Ada 232
B.9 System Databases 243

Annex C (informative) Ada/C Cross-References 247
C.1 Ada-to-C Cross-Reference 247
C.2 C-to-Ada Cross-Reference 259

Alphabetic Topical Index 271

FIGURES

Figure 2-1 – POSIX 24
Figure 3-1 – POSIX_Process_Primitives 46
Figure 3-2 – POSIX_Unsafe_Process_Primitives 57
Figure 3-3 – POSIX_Signals 63
Figure 4-1 – POSIX_Process_Identification 80
Figure 4-2 – POSIX_Process_Times 87
Figure 4-3 – POSIX_Process_Environment 89
Figure 4-4 – POSIX_Calendar 96
Figure 4-5 – POSIX_Configurable_System_Limits 98
Figure 5-1 – POSIX_Permissions 102
Figure 5-2 – POSIX_Files 105
Figure 5-3 – POSIX_File_Status 116
Figure 5-4 – POSIX_Configurable_File_Limits 120
Figure 6-1 – POSIX_IO 126
Figure 6-2 – POSIX_File_Locking 136
Figure 7-1 – POSIX_Terminal_Functions 147
Figure 8-1 – POSIX_Supplement_to_Ada_IO 171
Figure 9-1 – POSIX_User_Database 176
Figure 9-2 – POSIX_Group_Database 178
Figure B-1 – Directory Iterator Example 225
Figure B-2 – Example of Form Parameter 235
Figure B-3 – Example of Access to Standard Error 242

TABLES

Table 2-1 – Typographical Conventions 7
Table 7-1 – Terminal_Characteristics Components 150
Table 7-2 – Terminal_Modes Values for Input Control 152
Table 7-3 – Terminal_Modes Values for Output Control 154
Table 7-4 – Terminal_Modes Values for Hardware Control 154
Table 7-5 – Terminal_Modes Values for Local Control Modes . . . 156
Table 7-6 – Special Control Character Usage 158

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.5-1992

Introduction

(This Introduction is not a normative part of IEEE Std 1003.5-1992, IEEE Standard for Information Technology—POSIX Ada Language Interfaces—Part 1: Binding for System Application Program Interface (API), but is included for information only.)

The basic goal of this standard is to provide an Ada binding for the interfaces specified in ISO/IEC 9945-1: 1990 (IEEE Std 1003.1-1990; referred to in this standard as "POSIX.1 {2}"[1]). The intent is to support Ada applications portability via a standard binding to the services provided by a POSIX-conforming operating system. Currently POSIX is defined by the C-language binding represented in POSIX.1; a language-independent definition of the interface is under development. Therefore, much of the work in producing this document was deciding what features of POSIX.1 represented POSIX functionality, as opposed to features of the C-language binding.

This standard provides package specifications and accompanying textual description for a set of Ada packages that represent the POSIX system. This document 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 ISO 8652: 1987, the Ada Language standard (particularly in the area of `Text_IO`) for use in a POSIX environment.

The emphasis in POSIX is on application programs, so the interfaces in this document are not sufficient to implement an Ada compilation system or a POSIX shell (as defined in IEEE Std 1003.2-1992). Nor is this standard meant to be sufficient for realtime applications. The IEEE P1003.4 working group is currently studying extensions to POSIX to support realtime. For an applications program, the intent is that a Strictly Conforming POSIX.5 Application (one that uses only the facilities in this document 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 9)

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

---

1) The numbers in braces indicate the place in the reference list for a normative reference (or bibliographic entry if the number is preceded by the letter B).

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.5-1992

Related Standards Activities

Activities to extend this standard to address additional requirements are in progress, and similar efforts can be anticipated in the future.

The following areas are under active consideration at this time, or are expected to become active in the near future:[2)]

(1) Language-independent service descriptions of POSIX.1 {2}

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

(3) Shell and utility facilities

(4) Verification testing methods

(5) Realtime facilities

(6) Secure/Trusted system considerations

(7) Network interface facilities

(8) System administration

(9) Graphical user interfaces

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

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

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

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

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

---

2) 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.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.5-1992

IEEE Std 1003.5-1992 was prepared by the P1003.5 working group, sponsored by the Technical Committee on Operating Systems and Application Environments of the IEEE Computer Society. At the time this standard was approved, the membership of the 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 Terence Fong (1988-1989)
Vice Chairs:	James P. Lonjers (1990-1991)
	Major Terence Fong (1989-1990)
	Stowe Boyd (1988-1989)
Editors:	David Emery
	Hal Jespersen
	Steven Deller (1988-1989)
Rationale Editor:	Mitch Gart
Secretary:	Jayne Baker
	David Emery (1988-1989)

Technical Reviewers

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

Working Group

Ted Baker	Michael Gillam	James Leathrum
Stowe Boyd	Al Globus	James Lonjers
Bevin Brett	Mars Gralia	James Moore
Charles Brown	Jayne Guyse	Mark Ruddock
Robert Brown	Ken Harvey	Steven Schwarm
Bhavesh 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	Sue LeGrand	Olle Wikstrom
Mitchell Gart		John Zenor

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.5-1992

The following persons were members of the 1003.5 balloting group that approved the standard for submission to the IEEE Standards Board:

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

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

172 Gary Lauther
173 Patricia K. Lawlis
174 Scott A. Leschke
175 M. Levitz
176 Stephen H. Levy
177 F. C. Lim
178 Timothy E. Lindquist
179 J. J. Logan
180 James P. Lonjers
181 Warren E. Loper
182 Mark Loveland
183 George A. Ludgate
184 Sonny Lundahl
185 Wesley Mackey
186 Austin J. Maher
187 James Maloney
188 Roger Martin
189 Robert Mathis
190 Fred Maymir-Ducharme
191 Catherine McDonald
192 Robert L. McGarvey
193 Daniel L. McNamee
194 Robert McWhirter
195 Nancy R. Mead
196 Geoff Mendal
197 Jay Michael
198 Gary W. Miller
199 Robert E. Miller
200 Judah Mogilensky
201 Al Mok
202 Charles S. Mooney
203 James D. Mooney
204 Freeman Moore
205 James W. Moore
206 Jerry A. Moore
207 John I. Moore, Jr.
208 Duncan Morrill
209 M. W. Morron
210 Gary Mrenak
211 David G. Mullens
212 Richard E. Nesee
213 Sai Lun Ng
214 Daniel Nissen
215 Karl Nyberg
216 James O'Day
217 Evelyn Obaid
218 Patricia Oberndorf
219 Kurt M. Olender
220 S. Ron Oliver

James K. Parrish
Thomas Parrish
Offer Pazy
Walt Penney
Guido Persch
Flavio Petersen
Thomas A. Peterson
George W. Petznick
Hane W. Polzer
J. Pottmyer
Charles Pow
Eileen Quann
Paul Rabin
John Reddan
W. Scott Redmon
Gregg Reed
Carl Reinert
Judith Richardson
Richard A. Rink
Clyde Roby
C. Allan Rofer
Hyman Rosen
Jerome D. Rosen
Frederick M. Rysz
Agnes M. Sardi
Robert J. Satnik
Allen Saxton
Lorne H. Schachter
F. P. Schauer
Alfred H. Scholldorf
Ron Schroeder
Mike Schultz
W. L. Schultz
Fritz Schulz
Stephen C. Schwarm
Leonard Seagren
Richard Seibel
Lawrence H. Shafer
Michael D. Shapiro
John G. Shea
Nagy M. Shehad
Dan Shia
Thomas E. Shields
Keith Shillington
David Shochat
Robert Charles Shock
Jerome L. Sibol, Jr.
Lee Silverthorn

Ronald Skoog
Thomas J. Smith
Charles Snyder
Jon S. Squire
Jeff Stevenson
Brian Sullivan
Del Swanson
S. Tucker Taft
Ravi Tavakley
Donn S. Terry
John A. Thalhamer
William J. Thomas
Peter L. Thompson
James L. Troy
Roger Tubby
Mark-Rene Uchida
L. David Umbaugh
Robert B. Urling
Evelyn M. Uzzle
Frances Van Scoy
Leonard Vanek
Michael W. Vannier
Uwe Wacker
Robert N. Wagoner
Mary Wall
Stephen R. Walli
Neal Walters
Kenneth Wasmundt
William Webster
J. Richard Weger
Brian Weis
Robert Weissensee
Michael K. Welter
Stephen Wersan
Thomas Wheeler
William Whitaker
Bruce Wieand
David Willcox
David C. Willet
David Williamson
Paul A. Willis
David H. Winfield
David C. Wolfe
Paul A. Wolfgang
Michal Young
Oren Yuan
Janusz Zalewski
K. M. Zemrowski
John Zolnowsky

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

When the IEEE Standards Board approved this standard on June 18, 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. Rankine
Clyde Camp	Walter J. Karplus	Wallace S. Read
Donald C. Fleckenstein	Ivor N. Knight	Ronald H. Reimer
Jay Forster*	Joseph 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. Engelman
David E. Soffrin
Stanley Warshaw

Mary Lynne Nielsen
IEEE Standards Project Editor

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.5-1992

IEEE Standard for Information Technology—POSIX Ada Language Interfaces—Part 1: Binding for System Application Program Interface (API)

Section 1: General

1.1 Scope

This standard defines a standard Ada language binding to ISO/IEC 9945-1: 1990 {2}, hereinafter referred to as "POSIX.1 {2}," to support application portability at the source code level. It is intended to be used by both application developers and system implementors.

The focus of this standard is the Ada interface to the POSIX.1 {2} standard. The POSIX.1 {2} standard was initially defined using the C programming language. A language-independent definition of POSIX.1 {2} is under development, but at this time has not achieved a level of definition suitable as a starting point for this document.

This standard is comprised of 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 POSIX.1 {2} system services presented in the form of Ada packages

— Interpretations of Ada semantics with respect to POSIX.1 {2}

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.5-1992

The following areas are outside the scope of this standard:

(1) User interface (shell) and commands associated with Ada program development[1)]

(2) Ada bindings to the POSIX.1 {2} archive/interchange file formats for *tar* and *cpio*[1)]

(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

(9) Realtime programming

(10) Multiprocessing

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.

{1} ISO 8652: 1987,[2)] *Programming languages—Ada* (Endorsement of ANSI/MIL-STD 1815A-1983). Hereinafter, this is referred to as Ada RM {1}.

{2} ISO/IEC 9945-1: 1990 (IEEE Std 1003.1-1990),[3)] *Information Technology—Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API) [C Language].* Hereinafter, this is referred to as POSIX.1 {2}.

---

1) These interfaces will be defined in the Ada binding to ISO/IEC 9945-2:..., Information technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities.

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

3) This standard is available from the IEEE Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA. Telephone: 1 (800) 678-IEEE or +1 (908) 981-1393 (outside US).

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.5-1992

1.3 Conformance

1.3.1 Implementation Conformance

1.3.1.1 Requirements

A conforming implementation shall meet all of the following criteria:

(1) The system shall support all required interfaces defined within this standard. These interfaces shall support the functional behavior described herein.

(2) The system may provide additional 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. The conformance document shall define an environment in which an application can be run with the behavior specified by the 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

(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

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

— `POSIX.POSIX_String_List`

— `POSIX_Process_Primitives.Process_Template`

— `POSIX_Process_Environment.Environment`

No other types defined in this standard shall be implemented using Ada access types.

(4) 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.

(5) Except as explicitly provided for in this standard, `POSIX.POSIX_Error` or `Standard.Storage_Error` are the only exceptions that shall be raised by operations declared in this standard.

(6) When executed, an implementation of this standard shall not be erroneous, as defined by the Ada RM {1}.

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Documents
Document Sharing Hub
Full document details: IEEE 1003.5-1992

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 Conforming POSIX.5 Application. 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_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 providing either a specific reference to the system documentation or providing full syntax and semantics of these features. The conformance document may specify the behavior of the implementation for those features where this standard states that implementations may vary or where features are identified as undefined or unspecified.

No specifications other than those described in this 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 package `POSIX` 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, and `POSIX.Get_Error_Code` shall return `POSIX.Operation_Not_Implemented`.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

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 the Ada RM {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_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 a Conforming POSIX.5 Application only in that it uses nonstandard facilities that are consistent with this standard. Such an application shall fully document its requirements for these extended facilities, in addition to the documentation required of a Conforming POSIX.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.)

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Documents
Document Sharing Hub
Full document details: IEEE 1003.5-1992

Section 2: Terminology and General Requirements

2.1 Conventions

2.1.1 Editorial Conventions

This standard uses the following editorial and typographical conventions. A summary of typographical conventions is shown in Table 2-1.

Table 2-1 – Typographical Conventions

Reference	Example
Ada Identifier	`POSIX_String_List`
Ada Implementation-Defined Element	***<implementation defined>***
Ada Reserved Word	**`generic`**
C-Language Data Type	*long*
C-Language Error Number	[EINTR]
C-Language Function	*system*()
C-Language Global External	*errno*
C-Language Header	`<sys/stat.h>`
Cross-Reference: Annex	Annex A
Cross-Reference: Clause	2.3
Cross-Reference: Other Standard	ISO/IEC 9999-1 {*n*}
Cross-Reference: Section	Section 2
Cross-Reference: Subclause	2.3.4, 2.3.4.5, 2.3.4.5.6
Definition Citation	[POSIX.1 {2}]
Environment Variable	**PATH**
Figure Reference	Figure 7-3
File Name	`/tmp`
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.

Numbers within braces, such as "POSIX.1 {2}," represent cross references to the normative references clause (see 1.2). If the number is preceded by a B, it represents a bibliographic entry (see 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]. At the conclusion of this type of definition, the designation of a standard within [brackets]

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Documents
Document Sharing Hub
Full document details: IEEE 1003.5-1992

indicates that the text of the definition was copied from that standard, with only editorial changes, if any.

(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 the 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 inline; 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. All material preceding page 1 of this standard (the "front matter") and the two indexes at the end are also only informative.

Notes that appear in a smaller point size and are indented 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. Editorial inconsistencies in the use of typography are unintentional and have no normative meaning in this standard.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 07:07:07 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

2.2 Definitions

2.2.1 Terminology

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

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

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

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.

2.2.1.4 obsolescent: An indication that a certain feature may be considered for withdrawal in future revisions of this standard.

NOTE: Obsolescent features are retained in this version to maintain symmetry with certain aspects of the C-language binding to POSIX.1 {2}. Their use in Ada applications is discouraged.

2.2.1.5 shall: An indication of a requirement on the implementation or on Strictly Conforming POSIX.5 Applications, where appropriate.

2.2.1.6 should:

(1) With respect to implementations, an indication of an implementation recommendation, but not a requirement.

(2) With respect to applications, an indication of a recommended programming practice for applications and a requirement for Strictly Conforming POSIX.5 Applications.

2.2.1.7 supported: A condition regarding optional functionality.

Certain functionality in this standard is optional, but the interfaces to that functionality are always required. If the functionality is *supported*, the interfaces work as specified by this standard (except that they do not raise the exception indicating the unsupported case). If the functionality is not supported, the interface shall raise the exception indicating the unsupported case.

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

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

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

2.2.1.9 undefined: An indication that this standard imposes no portability requirements on 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.

2.2.1.10 unspecified: An indication that this standard imposes no portability requirements on applications for correct program constructs or correct data regarding a value or behavior.

Implementations (or other standards) may specify the result of using that value or causing that behavior. An application requiring a specific behavior, rather than tolerating any behavior when using that functionality, is using extensions, as defined in 1.3.2.2.3.

2.2.2 General Terms

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

2.2.2.1 absolute pathname: See *pathname resolution* in 2.3.10.

2.2.2.2 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.

See *file access permissions* in 2.3.6.

2.2.2.3 Ada I/O: The input/output operations defined in Chapter 14 of the Ada RM {1} and further defined in Section 8 of this standard.

2.2.2.4 address space: The memory locations that can be referenced by a process. [POSIX.1 {2}]

2.2.2.5 appropriate privileges: An implementation-defined means of associating privileges with a 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. [POSIX.1 {2}]

2.2.2.6 background process: A process that is a member of a background process group. [POSIX.1 {2}]

2.2.2.7 background process group: Any process group that is a member of a session that has established a connection with a controlling terminal and is not in the foreground process group.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2025 at 07:07:07 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

2.2.2.8 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. [POSIX.1 {2}]

2.2.2.9 byte: A sequence of bits transmitted over a serial line (see Section 7).

2.2.2.10 canonical input processing: The processing of terminal input in the form of text lines. (See 7.1.1.6.)

2.2.2.11 character: A sequence of one or more values of type `POSIX.POSIX_Character` representing a single graphic symbol.

2.2.2.12 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. [POSIX.1 {2}]

2.2.2.13 child process: See *process* in 2.2.2.68.

2.2.2.14 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. [POSIX.1 {2}]

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

Each session may have at most one controlling terminal associated with it, and a controlling terminal is associated with exactly one session. Certain input sequences from the controlling terminal (see 7.1) cause signals to be sent to all processes in the process group associated with the controlling terminal. [POSIX.1 {2}]

2.2.2.16 current working directory: See *working directory* in 2.2.2.94.

2.2.2.17 device: A computer peripheral or an object that appears to the application as such. [POSIX.1 {2}]

2.2.2.18 directory: A file that contains directory entries.

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

2.2.2.19 directory entry [link]: A relation that associates a filename with a file.

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

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Documents
Document Sharing Hub
Full document details: IEEE 1003.5-1992

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

See *pathname resolution* in 2.3.10. [POSIX.1 {2}]

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

See *pathname resolution* in 2.3.10. [POSIX.1 {2}]

2.2.2.22 effective group ID: An attribute of a process that is used in determining various permissions, including file access permissions, described in 2.3.6. [POSIX.1 {2}]

See *group ID* in 2.2.2.42. This value is subject to change during the process lifetime, as described in packages `POSIX_Process_Identification` (see 4.1) and `POSIX_Process_Primitives` (see 3.1).

2.2.2.23 effective user ID: An attribute of a process that is used in determining various permissions, including file access permissions. [POSIX.1 {2}]

See *user ID* in 2.2.2.92. This value is subject to change during the process lifetime, as described in packages `POSIX_Process_Identification` (see 4.1) and `POSIX_Process_Primitives` (see 3.1).

2.2.2.24 empty directory: A directory that contains, at most, directory entries for dot and dot-dot. [POSIX.1 {2}]

2.2.2.25 empty string [null string]: A character array of zero length.

2.2.2.26 erroneous: Erroneous is used in this standard as defined in Section 1.6 of the Ada RM {1}.

2.2.2.27 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. [POSIX.1 {2}]

Other characteristics of FIFOs are described in packages `POSIX_Files` (see 5.2) and `POSIX_IO` (see 6.1).

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

A file has certain attributes, including access permissions and type. File types include regular file, character special file, block special file, FIFO special file, and directory. Other types of files may be defined by the implementation. [POSIX.1 {2}]

2.2.2.29 file description: See *open file description* in 2.2.2.53.

2.2.2.30 file descriptor: A value used to identify an open file for the purpose of file access.

File descriptors are unique within a process.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:30 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

2.2.2.31 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. [POSIX.1 {2}]

2.2.2.32 filename: A name consisting of 1 to `POSIX.Filename_Limit'Maxima'Last` `POSIX.POSIX_Characters` used to name a file.

The characters composing the name may be selected from the set of all character values excluding the slash character and the null character. The filenames dot and dot-dot have special meaning; see *pathname resolution* in 2.3.10. A filename is sometimes referred to as a pathname component.

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

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

2.2.2.34 file other class: 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. [POSIX.1 {2}]

2.2.2.35 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. [POSIX.1 {2}]

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

The 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 the package `POSIX_Permissions` (see 5.1). The detailed usage of the file permission information in access decisions is described in *file access permissions* in 2.3.6.

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

File serial numbers are unique throughout a file system. [POSIX.1 {2}]

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

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

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. That is, each file system defines a new *name space*, giving meaning to the *names* (file serial numbers) that designate files.

2.2.2.39 foreground process: A process that is a member of a foreground process group. [POSIX.1 {2}]

2.2.2.40 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 7.1.1.4. [POSIX.1 {2}]

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

2.2.2.42 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` (see 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.

2.2.2.43 job control: A facility that allows users to stop (suspend) selectively the execution of processes and continue (resume) their execution at a later time. [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` or at run time by the value returned by the function `Job_Control_Supported` in package `POSIX_Configurable_System_Limits`; see 2.4 and 4.5.

2.2.2.44 link: See *directory entry* in 2.2.2.19.

2.2.2.45 link count: The number of directory entries that refer to a particular file. [POSIX.1 {2}]

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

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2025 at 04:35:31 UTC from IEEE Xplore. Restrictions apply.

Documents
Document Sharing Hub
Full document details: IEEE 1003.5-1992

Each login shall be associated with exactly one login name.

2.2.2.47 login name: A user name that is associated with a login. [POSIX.1 {2}]

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

See *file access permissions* in 2.3.6. [POSIX.1 {2}]

2.2.2.49 noncanonical input processing: The processing of terminal input as uninterpreted *characters* (see Section 7).

2.2.2.50 null character: The `POSIX_Character` 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.

2.2.2.51 null string: See *empty string* in 2.2.2.25.

2.2.2.52 open file: A file that is currently associated with a file descriptor. [POSIX.1 {2}]

2.2.2.53 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 6.1.5), and file access modes are attributes of an open file description. [POSIX.1 {2}]

2.2.2.54 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. [POSIX.1 {2}]

The process group is no longer a member of the session of the process that created it. This can occur when all other members of the session exit.

2.2.2.55 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. [POSIX.1 {2}]

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

2.2.2.56 parent process: See *process* in 2.2.2.68.

2.2.2.57 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. [POSIX.1 {2}]

2.2.2.58 path prefix: A pathname, with an optional ending slash, that refers to a directory. [POSIX.1 {2}]

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

A pathname consists of, at most, `POSIX.Pathname_Limit_Maxima'Last` components of type `POSIX.POSIX_Characters`. 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 described in 2.3.10.

2.2.2.60 pathname component: See *filename* in 2.2.2.32.

2.2.2.61 permission: See *file permission* in 2.2.2.36.

2.2.2.62 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.

2.2.2.63 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.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2025 at 08:07:07 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

2.2.2.64 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 and the character slash (/).

2.2.2.65 POSIX Character: The smallest addressable entity in a file.

A graphic character is a sequence of one or more `POSIX.POSIX_Characters`. It shall be possible to represent all data read and written in `POSIX.POSIX_Characters` or concatenations of `POSIX.POSIX_Characters`.

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

2.2.2.67 privilege: See *appropriate privileges* in 2.2.2.5.

2.2.2.68 process: An address space, and the program (including any Ada tasks contained within the program) executing within that address space, and its required system resources.

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, and the newly created process is the child process.

2.2.2.69 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. [POSIX.1 {2}]

2.2.2.70 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. [POSIX.1 {2}]

2.2.2.71 process group leader: A process whose process ID matches the process ID of the process group leader for all members of the process group.

2.2.2.72 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.

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

The process ID is defined in the package `POSIX_Process_Identification`. See 4.1. A process ID shall not be reused by the system until the process lifetime

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

ends. In addition, if there exists a process group 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.

2.2.2.74 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.

2.2.2.75 read-only file system: A file system that has implementation-defined characteristics restricting modifications. [POSIX.1 {2}]

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

See *group ID* in 2.2.2.42. This value is subject to change during the process lifetime, as described in 4.1. [POSIX.1 {2}]

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

See *user ID* in 2.2.2.92. This value is subject to change during the process lifetime, as described in 4.1. [POSIX.1 {2}]

2.2.2.78 regular file: A file that is a randomly accessible sequence of `POSIX.POSIX_Characters`, with no further structure imposed by the system.

2.2.2.79 relative pathname: See *pathname resolution* in 2.3.10.

2.2.2.80 root directory: A directory, associated with a process, that is used in pathname resolution for pathnames that begin with a slash. [POSIX.1 {2}]

2.2.2.81 saved set-group-ID: When the `Saved_IDs_Support` 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`.

See 2.3.6 and Section 4.

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

2.2.2.82 saved set-user-ID: When the `Saved_IDs_Support` 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`.

See 2.3.6 and Section 4.

2.2.2.83 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.

2.2.2.84 session leader: A process that has created a session (see `POSIX_Process_Identification.Create_Session`).

2.2.2.85 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. [POSIX.1 {2}]

2.2.2.86 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. [POSIX.1 {2}]

2.2.2.87 slash: The literal character "/".

This character is also known as *solidus* in ISO 8859-1 {B2}. [POSIX.1 {2}]

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

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`, 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.

2.2.2.89 system: An implementation of this standard. [POSIX.1 {2}]

2.2.2.90 system process: An object, other than a process executing an application, that is defined by the system and has a process ID. [POSIX.1 {2}]

An implementation shall reserve at least one process ID for system processes.

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Documents
Document Sharing Hub
Full document details: IEEE 1003.5-1992

2.2.2.91 terminal [terminal device]: A character special file that obeys the specifications of 7.1. [POSIX.1 {2}]

2.2.2.92 user ID: A value identifying a system user.

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

2.2.2.93 user name: A value of `POSIX_String` that is used to identify a user, as described in 9.1.

2.2.2.94 working directory [current working directory]: A directory, associated with a process, that is used in pathname resolution for pathnames that do not begin with a slash. [POSIX.1 {2}]

2.2.3 Abbreviations

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

2.2.3.1 Ada RM: The Ada Language Reference Manual: ISO 8652, *Programming languages—Ada* (Endorsement of ANSI/MIL-STD 1815A-1983) {1}.

2.2.3.2 POSIX.1: ISO/IEC 9945-1:1990, *Information technology—Portable operating system interface (POSIX)—Part 1: System Application Program Interface (API) [C Language]* {2}.

2.2.3.3 POSIX.5: This standard.

2.3 General Concepts

2.3.1 program/process equivalence: An Ada program in execution shall behave as if it is a single POSIX process throughout this standard. Thus, all the tasks of the program shall execute within the environment of the same POSIX process. Tasks of an Ada program may be implemented as separate processes as long as the environment is shared and all these processes can be viewed as a single POSIX process by the other processes in the system.

2.3.2 Ada character differences: The Ada RM {1} defines the representation of characters in the Ada language. This representation does not allow for extended character sets for localization, graphics, or non-ASCII-based character sets. In anticipation of POSIX support for character sets other than ASCII, the Ada character set has been separated from the one supported by the POSIX system. This standard does not prohibit the character sets from being identical.

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2025 at 08:07:07 UTC from IEEE Xplore. Restrictions apply.
Documents Document Sharing Hub
Full document details: IEEE 1003.5-1992

2.3.3 POSIX signals as interrupts: The signal handling mechanism supported in POSIX is mapped onto the Ada interrupt entry mechanism. POSIX Ada signals are defined as interrupt entry points to a task in the Ada program.

2.3.4 system call exception errors: Ada supports an exception mechanism that is used in this binding to report errors from system calls. The details of the error are reported by an error code that is accessed via a function.

2.3.5 extended security controls: The access control (see *file access permissions*) and privilege (see *appropriate privileges*) mechanisms have been defined to allow implementation-defined *extended security controls*. These permit an implementation to provide security mechanisms to implement different security policies than are described in this standard. These mechanisms shall be implemented and mapped onto the POSIX interfaces in such a way as to appear to be an extension to the POSIX security system.

2.3.6 file access permissions: The standard file access control mechanism uses the file permission set, as described below. This permission set is determined at file creation by the procedures `POSIX_IO.Open_Or_Create`, `POSIX_Files.Create_FIFO`, or `POSIX_Files.Create_Directory` and are changed by `POSIX_Files.Change_Permissions`. The permission set can be read by the function `POSIX_File_Status.Permission_Set_Of`.

Implementations may provide *additional* or *alternate* file access control mechanisms, or both. An additional access control mechanism shall only further restrict the access permissions defined by the file permission set. An alternate access control mechanism shall

(1) Specify file permission set for the owner class, group class, and other class of the file, corresponding to the access permissions.

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

(3) Be disabled for a file after the file permission set is changed for that file. The disabling of the alternate mechanism need not disable any additional mechanisms defined by an implementation.

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

(1) If a process has the appropriate privilege:

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

  (b) If execute permission is requested, access is granted if execute permission is granted to at least one user by the file permission set or by an alternate access control mechanism; otherwise, access is denied.

(2) Otherwise:

  (a) The file permission set of a file contains read, write, and execute/search permissions for the file owner class, file group class, and file other class.

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

(b) Access is granted if an alternate access control mechanism is not enabled and the requested access permission is set for the class to which the process belongs, or if an alternate access control mechanism is enabled and it allows the requested access; otherwise, access is denied.

Unless otherwise stated, the user ID is the effective user ID of the process and the group ID is the effective group ID of the process. The effective user ID and the effective group ID can be saved in the set-user-ID or set-group-ID if `POSIX_Configurable_System_Limits.Saved_IDs_Supported` returns `True`. (See Section 4.)

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

2.3.8 filename portability: Filenames should be constructed from the portable filename character set because the use of other characters can be confusing or ambiguous in certain contexts. The function `Is_Portable_Filename` (see 2.4) is provided for checking a `POSIX_String` to verify if it is a portable filename. The function `Is_Portable_Pathname` (see 2.4) is provided to check for portable pathnames.

2.3.9 file times update: Each file has three associated time values that are updated when file data has been accessed, file data has been modified, or file status has been changed, respectively. These values are accessed via functions defined in the package `POSIX_File_Status`.

For each function or procedure in this standard that reads or writes file data or changes the file status, the appropriate time-related fields are noted as *marked for update*. If an implementation of such a function marks for update a time-related field not specified by this standard, this shall be documented, except that any changes caused by pathname resolution need not be documented. For the other functions in this standard (those that are not explicitly required to read or write file data or change file status, but that in some implementations happen to do so), the effect is unspecified.

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

2.3.10 pathname resolution: Pathname resolution is performed for a process to resolve a pathname to a particular file in a file hierarchy. There may be multiple pathnames that resolve to the same file. (See 5.2.1.)

Each filename in the pathname is located in the directory specified by its predecessor (for example, in the pathname fragment "`a/b`", file "`b`" is located in directory "`a`"). Pathname resolution fails if this cannot be accomplished. If the

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 07:07:07 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.5-1992

pathname begins with a slash, the predecessor of the first filename in the pathname is taken to be the root directory of the process (such pathnames are referred to as absolute pathnames). If the pathname does not begin with a slash, the predecessor of the first filename of the pathname is taken to be the current working directory of the process (such pathnames are referred to as relative pathnames). If the pathname begins with two leading slashes, the predecessor of the first filename in the pathname may be interpreted in an implementation-defined manner.

The interpretation of a pathname component is dependent on the values of the functions `Filename_Limit` and `Filename_Is_Truncated` applied to the path prefix of the component. If the length of a pathname component exceeds the value of `Filename_Limit` and `Filename_Is_Truncated` is `False`, the implementation shall raise the exception `POSIX_Error` and set the error code `Filename_Too_Long`. Otherwise, the implementation shall ignore all the characters after the initial substring whose length is the value returned by `Filename_Limit` for the pathname component. (See Section 5.)

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

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

2.3.11 allocated storage: In some places the POSIX packages may have to allocate storage. One example is string lists. In any place where an access type is declared in the interface, or where heap storage is likely to be dynamically allocated, storage retrieval procedures are declared. This allows an application to free any dynamic storage that is likely to be allocated by a POSIX package.

IEEE

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Documents
Document Sharing Hub
Full document details: IEEE 1003.5-1992

2.4 Package POSIX

```
package POSIX is

    -- Symbolic subtypes and constants

      -- Optional facilities

    subtype Job_Control_Support is Boolean range
                <implementation defined> .. <implementation defined>;
    subtype Saved_IDs_Support is Boolean range
                <implementation defined> .. <implementation defined>;
    subtype Change_Owner_Restriction is Boolean range
                <implementation defined> .. <implementation defined>;
    subtype Filename_Truncation is Boolean range
                <implementation defined> .. <implementation defined>;

    System_POSIX_Version : constant := 1990_09;
    POSIX_Ada_Version    : constant := 1992_06;

      -- I/O count

    type IO_Count is <optional parent type>
            range 0..<implementation defined>;
    subtype IO_Count_Maxima is IO_Count range 32767..IO_Count'Last;

      -- System limits

    Portable_Groups_Maximum           : constant Natural := 0;
    subtype Groups_Maxima is Natural
            range <implementation defined> .. <implementation defined>;

    Portable_Argument_List_Maximum    : constant Natural := 4096;
    subtype Argument_List_Maxima is Natural
            range <implementation defined> .. <implementation defined>;

    Portable_Child_Processes_Maximum  : constant Natural := 6;
    subtype Child_Processes_Maxima is Natural
            range <implementation defined> .. <implementation defined>;

    Portable_Open_Files_Maximum       : constant Natural := 16;
    subtype Open_Files_Maxima is Natural
            range <implementation defined> .. <implementation defined>;

    Portable_Stream_Maximum           : constant Natural := 8;
    subtype Stream_Maxima is Natural
            range <implementation defined> .. <implementation defined>;

    Portable_Time_Zone_String_Maximum : constant Natural := 3;
    subtype Time_Zone_String_Maxima is Natural
            range <implementation defined> .. <implementation defined>;

      -- Pathname variable values

    Portable_Link_Limit_Maximum       : constant Natural := 8;
    subtype Link_Limit_Maxima is Natural
            range <implementation defined> .. <implementation defined>;
```

Figure 2-1 – POSIX

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:53:38 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

```
   Portable_Input_Line_Limit_Maximum  : constant IO_Count := 255;
   subtype Input_Line_Limit_Maxima is IO_Count
           range <implementation defined> .. <implementation defined>;

   Portable_Input_Queue_Limit_Maximum : constant IO_Count := 255;
   subtype Input_Queue_Limit_Maxima is IO_Count
           range <implementation defined> .. <implementation defined>;

   Portable_Filename_Limit_Maximum    : constant Natural := 14;
   subtype Filename_Limit_Maxima is Natural
           range <implementation defined> .. <implementation defined>;

   Portable_Pathname_Limit_Maximum    : constant Natural := 255;
   subtype Pathname_Limit_Maxima is Natural
           range <implementation defined> .. <implementation defined>;

   Portable_Pipe_Limit_Maximum        : constant IO_Count := 512;
   subtype Pipe_Limit_Maxima is IO_Count
           range <implementation defined> .. <implementation defined>;

     -- Blocking behavior values

   type Blocking_Behavior is (Tasks, Program);
   subtype Text_IO_Blocking_Behavior is Blocking_Behavior
           range <implementation defined> .. <implementation defined>;

   IO_Blocking_Behavior               : constant Blocking_Behavior
                                      := <implementation defined>;
   File_Lock_Blocking_Behavior        : constant Blocking_Behavior
                                      := <implementation defined>;
   Wait_For_Child_Blocking_Behavior   : constant Blocking_Behavior
                                      := <implementation defined>;

   -- Signal masking

   type Signal_Masking is (No_Signals, RTS_Signals, All_Signals);

   -- Characters and strings

   type POSIX_Character is
           (
           -- ' ','0','1','2','3','4','5','6','7','8','9',
           -- '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',
           -- '.','_','-','/','"', '#', '&', '''', '(', ')',
           -- '*', '+', ',', ':', ';', '<', '=', '>', '|'
            <other characters are implementation defined>);

   type POSIX_String is array (Positive range <>) of POSIX_Character;

   function To_POSIX_String (Str : string)
       return POSIX_String;
   function To_String (Str : POSIX_String)
       return string;
```

Figure 2-1 – POSIX (*continued*)

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Documents Document Sharing Hub
Full document details: IEEE 1003.5-1992

```
684
685     subtype Filename is POSIX_String;
686     subtype Pathname is POSIX_String;
687     function Is_Filename (Str : POSIX_String)
688         return Boolean;
689     function Is_Pathname (Str : POSIX_String)
690         return Boolean;
691     function Is_Portable_Filename (Str : POSIX_String)
692         return Boolean;
693     function Is_Portable_Pathname (Str : POSIX_String)
694         return Boolean;
695     -- String lists
696     type POSIX_String_List is limited private;
697     Empty_String_List : constant POSIX_String_List;
698     procedure Make_Empty (List : in out POSIX_String_List);
699     procedure Append (List     : in out POSIX_String_List;
700                       Str      : in     POSIX_String);
701     generic
702         with procedure Action
703                        (Item : in POSIX_String;
704                         Quit : in out Boolean);
705     procedure For_Every_Item (List : in     POSIX_String_List);
706     function Length (List : POSIX_String_List)
707         return Natural;
708     function Value
709             (List  : POSIX_String_List;
710              Index : Positive)
711         return POSIX_String;
712     -- Option sets
713     type Option_Set is private;
714     function Empty_Set
715         return Option_Set;
716     function "+" (L, R : Option_Set)
717         return Option_Set;
718     function "-" (L, R : Option_Set)
719         return Option_Set;
720     -- Exceptions and error codes
721     POSIX_Error: exception;
722     type Error_Code is <implementation-defined integer>;
723     function Get_Error_Code
724         return Error_Code;
725
```

726 **Figure 2-1 – POSIX (*continued*)**

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

```
    procedure Set_Error_Code (Error: in   Error_Code);

    function Is_POSIX_Error (Error: Error_Code)
        return Boolean;

    function Image (Error: Error_Code)
        return String;

    No_Error                      : constant Error_Code
                                  := 0;
    Argument_List_Too_Long        : constant Error_Code
                                  := <implementation-defined static expression>;
    Bad_File_Descriptor           : constant Error_Code
                                  := <implementation-defined static expression>;
    Broken_Pipe                   : constant Error_Code
                                  := <implementation-defined static expression>;
    Directory_Not_Empty           : constant Error_Code
                                  := <implementation-defined static expression>;
    Exec_Format_Error             : constant Error_Code
                                  := <implementation-defined static expression>;
    File_Exists                   : constant Error_Code
                                  := <implementation-defined static expression>;
    File_Too_Large                : constant Error_Code
                                  := <implementation-defined static expression>;
    Filename_Too_Long             : constant Error_Code
                                  := <implementation-defined static expression>;
    Improper_Link                 : constant Error_Code
                                  := <implementation-defined static expression>;
    Inappropriate_IO_Control_Operation: constant Error_Code
                                  := <implementation-defined static expression>;
    Input_Output_Error            : constant Error_Code
                                  := <implementation-defined static expression>;
    Interrupted_Operation         : constant Error_Code
                                  := <implementation-defined static expression;>
    Invalid_Argument              : constant Error_Code
                                  := <implementation-defined static expression>;
    Invalid_Seek                  : constant Error_Code
                                  := <implementation-defined static expression>;
    Is_A_Directory                : constant Error_Code
                                  := <implementation-defined static expression>;
    No_Child_Process              : constant Error_Code
                                  := <implementation-defined static expression>;
    No_Locks_Available            : constant Error_Code
                                  := <implementation-defined static expression>;
    No_Space_Left_On_Device       : constant Error_Code
                                  := <implementation-defined static expression>;
    No_Such_Operation_On_Device   : constant Error_Code
                                  := <implementation-defined static expression>;
    No_Such_Device_Or_Address     : constant Error_Code
                                  := <implementation-defined static expression>;
    No_Such_File_Or_Directory     : constant Error_Code
                                  := <implementation-defined static expression;>
```

Figure 2-1 – POSIX (*continued*)

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

```
779  ---------------------------------------------------------------------
780      No_Such_Process                 : constant Error_Code
781                                      := <implementation-defined static expression>;
782      Not_A_Directory                 : constant Error_Code
783                                      := <implementation-defined static expression>;
784      Not_Enough_Space                : constant Error_Code
785                                      := <implementation-defined static expression>;
786      Operation_Not_Implemented       : constant Error_Code
787                                      := <implementation-defined static expression>;
788      Operation_Not_Permitted         : constant Error_Code
789                                      := <implementation-defined static expression>;
790      Permission_Denied               : constant Error_Code
791                                      := <implementation-defined static expression>;
792      Read_Only_File_System           : constant Error_Code
793                                      := <implementation-defined static expression>;
794      Resource_Busy                   : constant Error_Code
795                                      := <implementation-defined static expression>;
796      Resource_Deadlock_Avoided       : constant Error_Code
797                                      := <implementation-defined static expression>;
798      Resource_Temporarily_Unavailable : constant Error_Code
799                                      := <implementation-defined static expression>;
800      Too_Many_Links                  : constant Error_Code
801                                      := <implementation-defined static expression>;
802      Too_Many_Open_Files             : constant Error_Code
803                                      := <implementation-defined static expression>;
804      Too_Many_Open_Files_In_System: constant Error_Code
805                                      := <implementation-defined static expression>;

806      -- System identification

807      function System_Name
808          return POSIX_String;

809      function Node_Name
810          return POSIX_String;

811      function Release
812          return POSIX_String;

813      function Version
814          return POSIX_String;

815      function Machine
816          return POSIX_String;

817  private
818      type POSIX_String_List is <implementation defined>;
819      Empty_String_List       : constant POSIX_String_List
820                              := <implementation defined>;
821      type Option_Set is <implementation defined>;
822  end POSIX;
823  ---------------------------------------------------------------------
```

824 **Figure 2-1 – POSIX (*concluded*)**

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 07:07:07 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.5-1992
Documents Document Sharing Hub

2.4.1 Constants and Static Subtypes

2.4.1.1 Optional Facilities

Constants: `System_POSIX_Version`, `POSIX_Ada_Version`

Types: `Job_Control_Support`, `Saved_IDs_Support`, `Change_Owner_Restriction`, `Filename_Truncation`

The package `POSIX` defines `Boolean` subtypes that indicate whether or not optional features are potentially supported. The range of the subtype shall be static. The subtypes in the package `POSIX` and the correspondingly named functions in the packages `POSIX_Configurable_System_Limits` and `POSIX_Configurable_File_Limits` define one of four possible conditions:

(1) If the range is `False .. False`, the option is not supported. The corresponding function shall always return `False`.

(2) If the range is `False .. True` and the corresponding function returns `False`, the option is not supported.

(3) If the range is `False .. True` and the corresponding function returns `True`, the option is supported.

(4) If the range is `True .. True`, the corresponding function shall always return `True` and the option is supported.

The functions in the packages `POSIX_Configurable_System_Limits` and `POSIX_Configurable_File_Limits` shall return `False` if the range is `False .. False`. The functions in the packages `POSIX_Configurable_System_Limits` and `POSIX_Configurable_File_Limits` shall return `True` if the range is `True .. True`. If the range is `False .. True`, the application shall call the function in `POSIX_Configurable_System_Limits` or `POSIX_Configurable_File_Limits` to determine if the option is supported.

The meanings of the optional features are

`Job_Control_Support`
: If `True`, it indicates that the implementation supports job control. If `False`, the implementation does not support job control.

`Saved_IDs_Support`
: If `True`, each process shall have a saved set-user-ID and a saved set-group-ID. If `False`, there is neither a saved set-user-ID nor a saved set-group-ID.

`Change_Owner_Restriction`
: If `True`, the procedure `Change_Owner_And_Group` shall be restricted to changing the group ID of a file only to the effective group ID of the process or to one of its supplementary group IDs, and changing the owner ID shall be restricted to processes with appropriate privileges. If `False`, these restrictions do not apply.

`Filename_Truncation`
: If `True`, filenames shall be truncated to the maximum allowed value. If `False` and a filename is longer than allowed, an error shall result, as

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

defined by the subprograms that use filenames and pathnames.

The constant `System_POSIX_Version` shall provide the year and month of the POSIX standard on which this standard is based. The constant `POSIX_Ada_Version` shall provide the year and month of this standard.

2.4.1.2 System Limits

Types: `Groups_Maxima`, `Argument_List_Maxima`, `Child_Processes_Maxima`, `Open_Files_Maxima`, `Stream_Maxima`, `Time_Zone_String_Maxima` `Link_Limit_Maxima`, `Input_Line_Limit_Maxima`, `Input_Queue_Limit_Maxima`, `Filename_Limit_Maxima`, `Pathname_Limit_Maxima`, `Pipe_Limit_Maxima`

Constants: `Portable_Groups_Maximum`, `Portable_Argument_List_Maximum`, `Portable_Child_Processes_Maximum`, `Portable_Open_Files_Maximum`, `Portable_Stream_Maximum`, `Portable_Time_Zone_String_Maximum` `Portable_Link_Limit_Maximum`, `Portable_Input_Line_Limit_Maximum`, `Portable_Input_Queue_Limit_Maximum`, `Portable_Filename_Limit_Maximum`, `Portable_Pathname_Limit_Maximum`, `Portable_Pipe_Limit_Maximum`

The numeric constants beginning with `Portable_` define minimal upper bounds of values for a conforming implementation. An implementation may support values greater than the values specified, but a conforming application shall not require values greater than those specified. The subtypes define the range of values that the corresponding functions in the packages `POSIX_Configurable_System_Limits` and `POSIX_Configurable_File_Limits` shall return. The `'First` and `'Last` values of the subtype shall be static. The `'First` value of the subtype shall be greater than or equal to the portable constant. If the `'First` value of the subtype is equal to its `'Last` value, the corresponding functions shall always return that constant value.

A conforming application may adjust its execution based on the range of the subtypes defined in package `POSIX` as well as the values returned by the functions in the packages `POSIX_Configurable_System_Limits` and `POSIX_Configurable_File_Limits`.

2.4.1.2.1 Configurable System Limits

The values returned by the functions in the package `POSIX_Configurable_System_Limits` shall be greater than or equal to the portable constants and shall be in the range of the subtype. For a given instance of an implementation, the values returned by the functions shall not change during execution. The meanings are

`Argument_List_Maxima`
: The maximum total length of the argument lists, environment data, and any associated overhead (in POSIX characters) that can be passed to a

Authorized licensed use limited to: Tsinghua University. Downloaded on March 02,2026 at 08:58:47 UTC from IEEE Xplore. Restrictions apply.

Documents
Document Sharing Hub
Full document details: IEEE 1003.5-1992

child when it is started by the procedures `POSIX_Process_Primitives.Start_Process`, `POSIX_Process_Primitives.Start_Process_Search`, `POSIX_Unsafe_Process_Primitives.Exec`, or `POSIX_Unsafe_Process_Primitives.Exec_Search`.

`Child_Processes_Maxima`
The maximum number of simultaneous processes per real user ID.

`Groups_Maxima`
The minimum number of simultaneous supplementary group IDs per process. Zero means that this feature is not supported.

`Open_Files_Maxima`
The maximum number of files a process can have open at one time.

`Stream_Maxima`
For an implementation that also supports the C-language binding, the maximum number of C-language streams a process can have open at one time.

2.4.1.2.2 Pathname Variable Values

The subtypes in `POSIX` define the range of values returned by the functions in the package `POSIX_Configurable_File_Limits`. The values returned by the functions may vary from one pathname to another. The meanings are

`Link_Limit_Maxima`
The maximum value of the link count of a file.

`Input_Line_Limit_Maxima`
The maximum number of `POSIX_Characters` that can be in a canonical input line. (See 7.1.1.6.)

`Input_Queue_Limit_Maxima`
The maximum number of `POSIX_Characters` of space in a terminal input queue; therefore, the maximum number of `POSIX_Characters` a conforming application may require to be typed as input before reading them. (See 7.1.1.6.)

`Filename_Limit_Maxima`
The maximum number of `POSIX_Characters` in a filename.

`Pathname_Limit_Maxima`
The maximum number of `POSIX_Characters` in a pathname.

`Pipe_Limit_Maxima`
The maximum number of `POSIX_Characters` that can be written atomically when writing to a pipe.

2.4.1.3 I/O Counts

Types: `IO_Count`, `IO_Count_Maxima`

The type `IO_Count` is used to reference data in a regular file. An implementation may specify an optional parent type for `IO_Count`. The maximum file size allowed on a file system shall be no less than `IO_Count_Maxima'First` and can be as large as `IO_Count_Maxima'Last`. The file system may limit the size of a file due to

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

capacity limits.

2.4.1.4 Blocking Behavior Values

Constants: `IO_Blocking_Behavior`, `File_Lock_Blocking_Behavior`, `Wait_For_Child_Blocking_Behavior`

Types: `Blocking_Behavior`, `Text_IO_Blocking_Behavior`

Because Ada is a multitasking language, multiple threads of execution may appear to be executing concurrently. On some implementations, when a task executes a POSIX operation that blocks until it is complete, the blocking may occur for all the tasks in the program (`Blocking_Behavior=> Program`) or only for the task that executed the POSIX operation (`Blocking_Behavior=> Tasks`). Different classes of POSIX operation may have different `Blocking_Behaviors`. The constants and types below allow an application to tailor its execution based on the blocking behavior.

`Text_IO_Blocking_Behavior`
: If the range of the subtype is `Task .. Task`, the blocking behavior of Ada I/O operations shall be at the task level. If the range of the subtype is `Program .. Program`, the blocking behavior of Ada I/O operations shall be at the program level. If the range of the subtype is `Task .. Program`, the blocking behavior of Ada I/O operations shall be selectable via the `FORM` parameter. (See 8.1.)

`IO_Blocking_Behavior`
: Specifies the blocking behavior of POSIX I/O operations that may block. (See 6.1.)

`File_Lock_Blocking_Behavior`
: Specifies the blocking behavior of the file locking operations. (See 6.2.)

`Wait_For_Child_Blocking_Behavior`
: Specifies the blocking behavior of the wait for child termination operations. (See 3.1.)

2.4.1.5 Signal Masking

Types: `Signal_Masking`

During some POSIX operations, it is desirable to block the receipt of certain classes of signals. The type `Signal_Masking` specifies what class of signal is masked.

`No_Signals`
: No signals are masked during the POSIX operation.

`RTS_Signals`
: Only the signals reserved for the Ada language implementation runtime system are masked during the POSIX operation. (See 3.3.2.)

`All_Signals`
: All signals that can be masked are masked during the POSIX operation.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:30 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1003.5-1992

2.4.1.6 Implementation Requirements

An implementation shall allow programs with

(1) At least `Portable_Groups_Maximum` simultaneous supplementary group IDs

(2) At least `Portable_Argument_List_Maximum POSIX_Characters` as the length of the argument list, environment data, and their overhead

(3) At least `Portable_Child_Process_Maximum` as the number of simultaneous processes per real user ID

(4) At least `Portable_Open_Files_Maximum` as the number of simultaneously open files per process

(5) At least `Portable_Link_Limit_Maximum` as the value of the link count of a file

(6) At least `Portable_Line_Limit_Maximum POSIX_Characters` as the length of an input line

(7) At least `Portable_Input_Queue_Limit_Maximum POSIX_Characters` as the length of an terminal input queue

(8) At least `Portable_Filename_Limit_Maximum POSIX_Characters` as the length of a filename

(9) At least `Portable_Pathname_Limit_Maximum POSIX_Characters` as the length of a pathname

(10) At least `Portable_Pipe_Limit_Maximum POSIX_Characters` as the size of an atomic write to a pipe

(11) At least `Portable_Time_Zone_String_Maximum POSIX_Characters` as the size of the **TZ** environment variable

(12) At least `Portable_Stream_Maximum` open C-language streams, if the implementation also supports the C-language binding

2.4.2 Characters and Strings

Constants: `Empty_String_List`

Types: `POSIX_Character`, `POSIX_String`, `Filename`, `Pathname`, `POSIX_String_List`

Operations: `To_POSIX_String`, `To_String`, `Is_Filename`, `Is_Pathname`, `Is_Portable_Pathname`, `Make_Empty`, `Append`, `For_Every_Item`, `Length`, `Value`

2.4.2.1 Description

The implementation shall provide a correspondence between the Ada type `Standard.Character` and the underlying character set of the implementation, type `POSIX_Character`. This correspondence shall have the following characteristics:

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Documents
Document Sharing Hub
Full document details: IEEE 1003.5-1992

(1) The type `Standard.Character` shall be as defined in the Ada RM {1}.

(2) The type `POSIX_Character` shall include the entire character set supported by the implementation. A conforming application shall make no assumption as to ordering of the enumeration and the size of the representation. Therefore, a conforming application shall make no assumption that the results of executing "<", "<=", ">", ">=", `'Pred` and `'Succ` are preserved across conversions between `Standard.Character` and `POSIX_Character` and their respective string types.

(3) A core subset of `Standard.Character` shall be placed into an invertible mapping with the corresponding subset of `POSIX_Character`, such that characters that have the same enumeration literal are mapped to each other. The core subset shall include the following characters:

Digits:

```
0 1 2 3 4 5 6 7 8 9
```

Uppercase letters:

```
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
```

Lowercase letters:

```
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
```

Special characters:

space, ampersand (`&`), apostrophe (`'`), left parenthesis (`(`), right parenthesis (`)`), star (`*`), plus (`+`), comma (`,`), hyphen (–), period (`.`), slash (`/`), colon (`:`), semicolon (`;`), less than (<), equals (=), greater than (>), underscore (`_`).

Format effectors:

horizontal tabulation, vertical tabulation, carriage return, line feed, form feed

(4) All remaining characters of the type `Standard.Character` shall be mapped to some subset of the remaining characters within `POSIX_Character`. This standard does not require invertibility of this mapping.

(5) Any remaining characters of the type `POSIX_Character` would have no counterpart in `Standard.Character`.

The type `POSIX_String` is an array of `POSIX_Character`. This type is used as the argument to all POSIX procedures or functions that require a "string." All values of type `POSIX_String` returned by functions defined in this standard shall have the attribute `'First` = 1 unless otherwise noted.

The functions `To_POSIX_String` and `To_String` are used to convert between the `Standard.Character` and `POSIX_Character` representations. All characters in the core subset, defined above, shall be *convertible* and their conversion shall be *invertible*. The results of converting the other characters in the standard Ada character set is implementation defined. A character in one character set may translate into a sequence of characters in the other character set. An implementation may convert the string sequences back into the single character when doing the

Authorized licensed use limited to: Tsinghua University. Downloaded on March 02,2025 at 05:10:07 UTC from IEEE Xplore. Restrictions apply.

Documents
Document Sharing Hub
Full document details: IEEE 1003.5-1992

inverse translation.

The constant `Portable_Pathname_Limit_Maximum` is the maximum number of characters one can have in a `POSIX_String` and guarantee a portable pathname. The constant `Portable_Filename_Limit_Maximum` is the maximum number of characters one can have in a `POSIX_String` and guarantee a portable filename. The subtype `Pathname_Limit_Maxima` is the range of the number of characters that the current implementation shall allow in a valid pathname depending on the path and the configuration. The subtype `Filename_Limit_Maxima` is the range of the number of characters that the current implementation shall allow in a valid filename depending on the path and the configuration.

The function `Is_Filename` shall return the value `True` if the parameter has the correct syntax (or form) to be a filename for this implementation. The function `Is_Pathname` shall return the value `True` if the parameter has the correct syntax (or form) to be a pathname for this implementation. These functions shall check all constraints set on filename and pathname by the implementation that can be checked without accessing the file system directly. None of the pathname components need exist, and access rights are not checked. These functions shall not raise any exceptions.

The function `Is_Portable_Filename` shall return the value `True` if the parameter has the correct syntax (or form) to be a portable filename. The function `Is_Portable_Pathname` shall return the value `True` if the parameter has the correct syntax (or form) to be a portable pathname. These functions shall not raise any exceptions, and they shall check all constraints on portable filenames and pathnames. The filename need not exist, and access rights are not checked.

The type `POSIX_String_List` is used to process arbitrary lists of `POSIX_Strings`. The value `Empty_String_List` represents a `POSIX_String_List` that contains no `POSIX_Strings` and is used as a parameter to procedures where an empty `POSIX_String_List` is needed. All objects of type `POSIX_String_List` shall have the initial value `Empty_String_List`.

The procedure `Make_Empty` shall remove any `POSIX_String` values on the `POSIX_String_List` value supplied via the parameter, free any allocated storage associated with the supplied `POSIX_String_List`, and then set the supplied `POSIX_String_List` to the value `Empty_String_List`.

The procedure `Append` shall append the `POSIX_String` value supplied via `Str` to the end of the `POSIX_String_List` supplied via `List`. The `POSIX_String` value supplied via `Str` shall be copied by the implementation so that the supplied `POSIX_String` value can be modified without changing the value associated with the `POSIX_String_List`. The original bounds of the `POSIX_String` value supplied via `Str` shall be preserved.

The application program instantiates the generic procedure `For_Every_Item`, providing an actual procedure for the generic formal procedure `Action`. When invoked, the newly created instance of `For_Every_Item` shall invoke the actual procedure supplied for `Action` once for every `POSIX_String` value (as the parameter `Str`) in the `POSIX_String_List` value supplied via `List` in the invocation of `For_Every_Item`. The original bounds of the `POSIX_String` values shall be preserved. The actual procedure supplied for `Action` shall be able to force termination of the instance of `For_Every_Item` by setting the value of `Quit` to `True`.

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.

Documents
Document Sharing Hub
Full document details: IEEE 1003.5-1992

Prior to calling `Action`, `For_Every_Item` shall set `Quit` to `False`. If `Action` modifies `List` (as a side effect) the effect of calling the instantiation of `For_Every_Item` is undefined. Any exceptions raised by the actual procedure supplied for `Action` shall terminate the iteration of the instantiation of `For_Every_Member` and shall propagate to the caller of the instantiation.

The function `Length` shall return the number of `POSIX_String` values in the `POSIX_String_List`. This includes any null strings.

The function `Value` shall return the value of type `POSIX_String` in the `Index` position in the list. The original bounds of the `POSIX_String` shall be preserved. `Constraint_Error` shall be raised if `Index` is greater than the length of the list.

2.4.2.2 Error Handling

`Constraint_Error` shall be raised by the string conversion functions `To_POSIX_String` and `To_String` if the conversion is not possible.

`Constraint_Error` shall be raised by an invocation of the `Value` function if the parameter `Index` has a value greater than the list length.

No exceptions shall be raised by `Is_Filename`, `Is_Pathname`, and `Is_Portable_Pathname`. No exceptions are specified by this standard for `Make_Empty`, `Append`, `For_Every_Item`, or `Length`.

2.4.3 Option Sets

Types: `Option_Set`

Operations: `Empty_Set`, "+", "-"

2.4.3.1 Description

The type `Option_Set` shall be used to represent sets of options. All objects of this type shall have as their initial value the value returned by `Empty_Set`. The function `Empty_Set` shall return a value of type `Option_Set` with no options set. The operation "+" shall return an option set where all options that are set in the two operands are set in the result. This is the "union" operation on option sets. The binary operation "-" shall return an option set where options set in the right-hand operand are not set in the result, and all other options set in the left-hand operand are set in the result.

2.4.3.2 Error Handling

No exceptions shall be raised by these operations.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2021 at 04:25:01 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.5-1992

2.4.4 Error Codes and Exceptions

Types: `Error_Code`

Constants: `No_Error, Argument_List_Too_Long, Bad_File_Descriptor, Broken_Pipe, Directory_Not_Empty, Exec_Format_Error, File_Exists, File_Too_Large, Filename_Too_Long, Improper_Link, Inappropriate_IO_Control_Operation, Input_Output_Error, Interrupted_Operation, Invalid_Argument, Invalid_Seek, Is_A_Directory, No_Child_Process, No_Locks_Available, No_Space_Left_On_Device, No_Such_Operation_On_Device, No_Such_Device_Or_Address, No_Such_File_Or_Directory, No_Such_Process, Not_A_Directory, Not_Enough_Space, Operation_Not_Implemented, Operation_Not_Permitted, Permission_Denied, Read_Only_File_System, Resource_Busy, Resource_Deadlock_Avoided, Resource_Temporarily_Unavailable, Too_Many_Links, Too_Many_Open_Files, Too_Many_Open_Files_In_System`

Operations: `Get_Error_Code, Set_Error_Code, Is_POSIX_Error, Image`

Exceptions: `POSIX_Error`

2.4.4.1 Description

Most functions and procedures raise the exception `POSIX_Error` when an error or exceptional condition occurs during the processing of the call. An `Error_Code` shall be set when this exception is raised. If more than one error occurs in the processing, this standard does not define in what order the errors are detected; therefore, any one of the possible `Error_Codes` may be set. If the exception is not raised, the current error code value shall be unchanged for the current task.

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

Implementations may generate the `Error_Codes` listed in this subclause under circumstances other than those described if and only if all those error conditions can always be treated identically to the error conditions as described in this subclause. Implementations may support additional `Error_Codes` not listed in this subclause, but they shall not generate a different `Error_Code` than one required by this subclause for an error condition described in this subclause.

The following descriptions identify the possible *errors* that can cause the exception `POSIX_Error` to be raised, and the `Error_Code` to be set by operations defined

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2026 at 04:25:32 UTC from IEEE Xplore. Restrictions apply.
Documents Document Sharing Hub
Full document details: IEEE 1003.5-1992

in this standard; these general descriptions are more precisely defined in the Error Handling subclauses of the packages that raise them.

`No_Error`
: This code is not set by any POSIX operation. It is returned by some POSIX operations when status is requested and no error has occurred. It shall have the value zero.

`Argument_List_Too_Long`
: The sum of the number of `POSIX_Characters` used by the argument list and environment list of the new process image was greater than `POSIX_Configurable_System_Limits.Argument_List_Maximum`.

`Bad_File_Descriptor`
: A file descriptor argument was out of range, referred to no open file, or a read (write) request was made to a file that was only open for writing (reading).

`Broken_Pipe`
: A write was attempted on a pipe or FIFO for which there was no process to read the data.

`Directory_Not_Empty`
: A directory with entries other than dot and dot-dot was supplied when an empty directory was expected.

`Exec_Format_Error`
: A request was made to execute a file that, although it had the appropriate permissions, was not in the format required by the implementation for executable files.

`File_Exists`
: An existing file was specified in an inappropriate context.

`File_Too_Large`
: The size of a file would exceed an implementation-defined maximum file size.

`Filename_Too_Long`
: The size of a pathname string exceeded the value returned by `POSIX_Configurable_File_Limits.Pathname_Limit` for that pathname, or a pathname component was longer than the value returned by `POSIX_Configurable_File_Limits.Filename_Limit` for that pathname and `Filename_Is_Truncated` was not in effect for that component.

`Improper_Link`
: A link to a file on another file system was attempted.

`Inappropriate_IO_Control_Operation`
: A control function was attempted for a file or special file for which the operation was inappropriate.

`Input_Output_Error`
: Some physical input or output error occurred. This error may be reported on a subsequent operation on the same file descriptor. Any other error-causing operation on the same file descriptor may cause the `Input_Output_Error` to be lost.

AVA Documents Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 05,2025 at 08:11:11 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1003.5-1992