INTERNATIONAL STANDARD

ISO/IEC 14515-2

IEEE Std 2003.2-1996

First edition
2003-09-01

Information technology — Portable Operating System Interface (POSIX®) — Test methods for measuring conformance to POSIX —

Part 2: Shell and utilities

Technologies de l'information — Interface de système de fonctionnement portable (POSIX®) — Méthodes d'essai pour mesurer la conformité au POSIX —

Partie 2: Enveloppe et utilités

Reference number
ISO/IEC 14515-2:2003(E)
IEEE
Std 2003.2-1996 Edition

AVA Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

ISO
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Documents
Document Sharing Hub

Full document details: ISO IEC 14515-2-2003

IEEE Std 2003.2-1996

Information Technology — Test Methods for Measuring Conformance to POSIX® — Part 2: Shell and Utilities

Sponsor

Portable Application Standards Committee
of the
IEEE Computer Society

Approved 20 June 1996

IEEE Standards Board

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

Documents
Document Sharing Hub

Full document details: ISO IEC 14515-2-2003

Abstract: A definition of the requirements placed upon providers of a POSIX® Conformance Test Suite for the POSIX.2 standard (ISO/IEC 9945-2:1993, IEEE/ANSI Std. 1003.2-1992) is provided. These requirements consist of a list of assertions defining those aspects of POSIX.2 that are to be tested and the associated test methods that are to be used in performing those tests. This standard is primarily aimed at test suite providers but it also defines to POSIX.2 implementors those aspects of the standard that will be verified by a conformance test suite.

Keywords: conformance, POSIX, Shell and Utilities, test methods

---

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

Copyright © 2003 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1 September 2003. Printed in the United States of America.

Print: ISBN 0-7381-3800-2 SH95167
PDF: ISBN 0-7381-3801-0 SS95167

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

Full document details: ISO IEC 14515-2-2003

International Standard ISO/IEC 14515-2:2003(E)

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 14515 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 14515-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

ISO/IEC 14515 consists of the following parts, under the general title *Information technology — Portable Operating System Interface (POSIX®) — Test methods for measuring conformance to POSIX*:

— *Part 1: System interfaces*

— *Part 2: Shell and utilities*

Annexes A, B and C form a normative part of this part of ISO/IEC 14515. Annex D is for information only.

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

Documents
Document Sharing Hub

Full document details: ISO IEC 14515-2-2003

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied "**AS IS**."

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.

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

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

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

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

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

Authorization to photocopy portions of any individual standard for internal or personal use is granted by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Documents
Document Sharing Hub

Full document details: ISO IEC 14515-2-2003

Introduction

(This introduction is not a normative part of IEEE Std 1003.2-1996, IEEE Standard for Information Technology — Test Methods for Measuring Conformance to POSIX® — Part 2: Shell and Utilities Interfaces, but is included for information only.)

The primary purpose of this standard is to define test methods for POSIX.2 {9}. It is intended for systems implementors and verification software developers. It is complementary to POSIX.2 {9} (second in a family of "POSIX" standards), which specifies a standard interface and environment for application programs that require the services of a "shell" command language interpreter and a set of common utility programs. (See 1.1 for a full description of the relationship between the standards.)

The majority of this standard describes the assertions that a test suite shall provide for the execution of a conformance test for the POSIX.2 {9} standard. The standard also describes the various other test methods that are to be used in the development or execution of such a test suite, and it describes the complimentary procedures that are needed to assess the results of the execution of the test suite. This standard should be read in conjunction with ISO/IEC 13210:1994 (ANSI/IEEE Std 1003.3-1991) Standard for Information Technology — Test Methods for Measuring Conformance to POSIX {11}, which defines the general test method requirements applicable to POSIX Conformance Testing.

Organization of This Standard

The standard is divided into nine parts:

— General, including a statement of scope, normative references, a description of test suite conformance requirements, and a description of the test methods specific to POSIX.2 {9} testing (Section 1).

— Assertions for definitions, general requirements, and the environment available to applications (Section 2).

— Assertions for the shell command language interpreter (Section 3).

— Assertions for the utilities in the required "Execution Environment Utilities" (Section 4).

— Assertions for the utilities in the optional "User Portability Utilities" (Section 5).

— Assertions for the utilities in the optional "Software Development Utilities" (Section 6).

— Assertions for the utilities in the optional "C Language Development Utilities" (Annex A).

— Assertions for C Language Bindings option (Annex B).

— Assertions for the utilities in the optional "FORTRAN Development and Runtime Utilities" (Annex C).

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

This introduction, any footnotes, notes accompanying the text, and the *informative* annexes are not considered part of the standard. Annexes A through C are normative and the other annexes are informative.

Acknowledgments

We wish to thank the following organizations for donating significant computer, printing, and editing resources to the production of this standard: McCarron Advanced Computing Services, Inc., The POSIX Software Group, UniSoft Limited, and The Open Group.

Also we wish to thank the organizations employing the members of the working group and the balloting group for both covering the expenses related to attending and participating in meetings, and for donating the time required both in and out of meetings for this effort.

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:[1)]

(1) Language-independent service descriptions of POSIX.1 {8}

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

(3) Verification testing methods

(4) Realtime facilities

(5) Secure/Trusted system considerations

(6) Network interface facilities

(7) System administration

(8) Graphical user interfaces

(9) 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

(10) 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.

---

1) A *Standards Status Report* that lists all current IEEE Computer Society standards projects is available from the IEEE Computer Society, 1730 Massachusetts Avenue NW, Washington, DC 20036-1903; Telephone: +1 202 371-0101; FAX: +1 202 728-9614.

Documents
Document Sharing Hub

Full document details: ISO IEC 14515-2-2003

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

IEEE Std 2003.2-1996 was prepared by the P2003.2 Working Group, sponsored by the Portable Applications Standards Committee of the IEEE Computer Society. At the time this standard was approved, the membership of the P2003.2 Working Group was as follows:

Portable Applications Standards Committee (PASC)

Chair:	Lowell Johnson
Vice Chair:	Charles Severance
Functional Chairs:	Jay Ashford
	Andrew Josey
	Barry Needham
	Stephe Walli
Secretary:	Nicholas Stoughton

P2003.2 Working Group Officials

Chair:	Lowell Johnson
Vice Chair:	Krystyne Supplee
Editor:	Andrew Twigger
Secretary:	Steve Henderson

Working Group

Sanjay Agraharam	Yvonne Gee	Gerald Powell
Lynda Allen	Dave Grindeland	Bill Putman
Helene Armitage	Ken Harvey	Vella Raman
John Bissell	Barry Hedquist	Elliot Rappaport
Barry Books	Gail Holmes	John Reed
Keith Bostic	Eric Horner	David Rowley
Gary Bradley	Hal Jespersen	Piyanai Saowarattitada
Mark Brown	Greg Jones	Tom Shem
Dawn Burnett	David Korn	Keith Stobie
Suzanne Callaghan	Mark Lamonds	William Sudman
Kate Chen	Chuck Lanchester	Scott Sutter
Steven Church	Eric Lewine	Ravi Tavakley
Anthony Cincotta	Roger Martin	William Toth
Don Cragun	Eric Matsono	Sandy Waters
Patric Dempster	James Moe	Bruce Weiner
Kevin Dodson	Cindy Morris	N. Ray Wilkes
Stan Douglas	Anita Mundkur	David Williams
Shiela Frankel	Arnie Powell	Fred Zlotnick

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

The following persons were members of the balloting group:

Nick Stoughton *EurOpen Institutional Representative*
Robert Boucher *Uniforum Institutional Representative*

Lynda Allen
Ralph Barker
Andy R. Bihain
Robert Bismuth
Michael E. Browne
Dawn Burnett
Suzanne Callaghan
Stephan M. Chan
William Cox
Donald Cragun
Fred D. Crowner
Dave Decot
Stephen L. Diamond
Stanford Douglas
Sheila Frankel
Michel Gien
Dave Grindeland
Barry Hedquist
John L. Hill
Richard Hughes-Rowlands
Hal Jespersen
Lowell Johnson
Judy Kerner
Alan W. Kiecker
Thomas M. Kurihara
Greger Leijonhufvud
Kevin Lewis
Roger Martin
Roland McGrath
Pete Meier
Mark Modig
John S. Morris
Landon Curt Noll
A. W. Powell
Piyanai Saowarattitada
Keith Stobie
Krystyne Supplee
James G. Tanner
Ravi Tavakley
Donn S. Terry
Andrew T. Twigger
Mark-Rene Uchida
Stephen R. Walli
Bruce Weiner
Andrew E. Wheeler
Alex White
Peter Wishart
Oren Yuen
George R. Zerdian

When the IEEE Standards Board approved this standard on 20 June 1996, it had the following membership:

Donald C. Loughry, *Chair* **Richard J. Holleman,** *Vice Chair*
Andrew G. Salem, *Secretary*

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

*Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

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

Trademarks

POSIX® is a registered certification mark of the Institute of Electrical and Electronics Engineers, Inc.

UNIX® is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

X/Open® is a registered trademark and the "X" device is a trademark of X/Open Company Limited.

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

Contents

PAGE

Introduction iii

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

Section 2: Conformance, Terminology, and General Requirements . . . 5
2.1 Conventions 5
2.2 Definitions 9
2.3 POSIX.2 {9} Conformance, Definitions and Built-in Utilities . . 11
2.4 Character Set 15
2.5 Locale 17
2.6 Environment Variables 46
2.7 Required Files 48
2.8 Regular Expression Notation 48
2.9 Dependencies on Other Standards 74
2.10 Utility Conventions 77
2.11 Utility Description Defaults 78
2.12 File Format Notation 82
2.13 Configuration Values 82
2.14 Terminal Characteristics 82

Section 3: Shell Command Language 85
3.1 Shell Introduction 85
3.2 Quoting 85
3.3 Token Recognition 87
3.4 Reserved Words 89
3.5 Parameters and Variables 90
3.6 Word Expansions 92
3.7 Redirection 114
3.8 Exit Status and Errors 119
3.9 Shell Commands 121
3.10 Shell Grammar 127
3.11 Signals and Error Handling 128
3.12 Shell Execution Environment 128
3.13 Special Built-in Commands 135

Section 4: Execution Environment Utilities 147
4.1 `awk` — Pattern scanning and processing language 147
4.2 `basename` — return nondirectory portion of pathname 203
4.3 `bc` — Arbitrary-precision arithmetic language 207
4.4 `cat` — Concatenate and print files 221
4.5 `cd` — Change working directory 225

AVA Documents
Document Sharing Hub

Full document details: ISO IEC 14515-2-2003

4.6 chgrp — Change file group ownership 229
4.7 chmod — Change file modes 234
4.8 chown — Change file ownership 242
4.9 cksum — Display file checksums and block counts 247
4.10 cmp — Compare two files 251
4.11 comm — select or reject lines common to two files 256
4.12 command — Execute a simple command 262
4.13 cp — Copy files 268
4.14 cut — Cut out selected fields of each line of a file 281
4.15 date — Display the date and time 287
4.16 dd — Convert and copy a file 295
4.17 diff — Compare two files 303
4.18 dirname — Return directory portion of pathname 311
4.19 echo — Write arguments to standard output 315
4.20 ed — Text editor 317
4.21 env — Set environment for command execution 375
4.22 expr — Evaluate arguments as an expression 379
4.23 false — Return false value 391
4.24 find — Find files 393
4.25 fold — Filter for folding lines 414
4.26 getconf — Get POSIX configuration values 420
4.27 getopts — Parse utility options 429
4.28 grep — File pattern searcher 435
4.29 head — Copy the first part of files 463
4.30 id — Return user identity 467
4.31 join — Relational database operator 474
4.32 kill — Terminate or signal processes 484
4.33 ln — Link files 488
4.34 locale — Get locale-specific information 495
4.35 localedef — Get locale-specific information 499
4.36 logger — Log messages 508
4.37 logname — Return user's login name 511
4.38 lp — Send files to a printer 513
4.39 ls — List directory contents 518
4.40 mailx — Interactive message processing system. 530
4.41 mkdir — Make directories 598
4.42 mkfifo — Make FIFO special files 605
4.43 mv — Move files 612
4.44 nohup — Invoke a utility immune to hangups 623
4.45 od — Dump files in various formats 627
4.46 paste — Merge corresponding or subsequent lines of files . . . 636
4.47 pathchk — Check pathnames 641
4.48 pax — Portable archive interchange 646
4.49 pr — Print files 672
4.50 printf — Write formatted output 679
4.51 pwd — Return working directory name 687
4.52 read — Read a line from standard input 689
4.53 rm — Remove directory entries 693
4.54 rmdir — Remove directories 699
4.55 sed — Stream editor 704
4.56 sh — Shell, the standard command language interpreter . . . 728
4.57 sleep — Suspend execution for an interval 758

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

4.58 sort — Sort, merge or sequence check text files 760
4.59 stty — Set the options for a terminal 770
4.60 tail — Copy the last part of a file 778
4.61 tee — Duplicate standard input 785
4.62 test — Condition evaluation utility 790
4.63 touch — Change file access and modification times 798
4.64 tr — Translate Characters 806
4.65 true — Return true value 811
4.66 tty — Return user's terminal name 813
4.67 umask — Get or set file mode creation mask 816
4.68 uname — Return system name 820
4.69 uniq — Report or filter out repeated lines in a file 824
4.70 wait — Await process completion 831
4.71 wc — Word, line, and byte count 835
4.72 xargs — Construct argument list(s) and execute utility . . . 839

Section 5: User Portability Utilities Option 847
5.1 alias — Define or display attributes 847
5.2 at — Execute commands at a later time 851
5.3 batch — Execute commands when the system load permits . . 861
5.4 bg — Run jobs in background 864
5.5 crontab — Schedule periodic background work 869
5.6 csplit — Split files based on context 875
5.7 ctags — Create a tags file 883
5.8 df — Report free disk space 891
5.9 du — Estimate file space usage 896
5.10 ex — Text editor 901
5.11 expand — Convert tabs to spaces 979
5.12 fc — Process command history list 985
5.13 fg — Run jobs in the foreground 992
5.14 file — Determine file type 996
5.15 jobs — Display status of jobs in the current session 1002
5.16 man — Display system documentation 1008
5.17 mesg — Permit or deny messages 1011
5.18 more — Display files on a page-by-page basis 1015
5.19 newgrp — Change to a new group 1044
5.20 nice — Invoke utility with altered scheduling system priority . 1048
5.21 nm — Write the name list of an object file 1053
5.22 patch — Apply changes to files 1062
5.23 ps — Report process status 1071
5.24 renice — Set system scheduling priorities of running processes 1079
5.25 split — Split into pieces 1085
5.26 strings — Find printable strings in files 1090
5.27 tabs — Set terminal tabs 1094
5.28 talk — Talk to another user 1098
5.29 time — Time a simple command 1104
5.30 tput — Change terminal characteristics 1108
5.31 unalias — Remove alias definitions 1112
5.32 unexpand — Convert spaces to tabs 1115
5.33 uudecode — Decode a binary file 1121
5.34 uuencode — Encode a binary file 1125

AVA Documents
Document Sharing Hub

Full document details: ISO IEC 14515-2-2003

5.35 `vi` — Screen-oriented (visual) display editor 1129
5.36 `who` — Display who is on the system 1177
5.37 `write` — Write to another user 1180

Section 6: Software Development Utilities Option 1187
6.1 `ar` — Create and maintain library archives 1187
6.2 `make` — Maintain, update, and regenerate groups of programs . 1197
6.3 `strip` — Remove unnecessary information from executable files 1228

Annex A (normative) C Language Development Utilities Option . . . 1233
A.1 `c89` — Compile Standard C programs 1233
A.2 `lex` — Generate programs for lexical tasks 1247
A.3 `yacc` — Yet another compiler compiler 1266

Annex B (normative) C Language Bindings Option 1287
B.1 C Language Definitions 1287
B.2 C Numerical Limits 1288
B.3 C Binding for Shell Command Interface 1291
B.4 C Binding for Access Environment Variables 1295
B.5 C Binding for Regular Expression Matching 1295
B.6 C Binding for Match Filename or Pathname 1314
B.7 C Binding for Command Option Parsing 1319
B.8 C Binding for Generate Pathnames Matching a Pattern 1322
B.9 C Binding for Perform Word Expansions 1330
B.10 C Binding for Get POSIX Configurable Variables 1333
B.11 C Binding for Locale Control 1336

Annex C (normative) FORTRAN Development Utilities Option 1337
C.1 `asa` — Interpret carriage control characters 1337
C.2 `fort77` — FORTRAN compiler 1342

Annex D (informative) Bibliography 1355

Identifier Index 1359

Alphabetic Topical Index 1365

Acknowledgments 1401

Documents
Document Sharing Hub

Full document details: ISO IEC 14515-2-2003

Information Technology— Test Methods for Measuring Conformance to POSIX® — Part 2: Shell and Utilities Interfaces

Section 1: General

1.1 Scope

This document defines a standard for test methods for the interface to command interpretation, or "shell," services and common utility programs for application programs. These test methods are derived from the definitions contained in ISO/IEC 9945-2:1993 (IEEE Std 1003.2-1992) {9}, hereinafter referred to as "POSIX.2 {9}." The services and programs described in POSIX.2 {9} are complementary to those specified by ISO/IEC 9945-1:1990 (IEEE Std 1003.1-1990) {8}, hereinafter referred to as "POSIX.1 {8}."

This standard has been designed to be used by both verification suite authors and system implementors. However, it is intended to be a reference document and not a tutorial on the construction of verification suites or implementations.

1.2 Normative References

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

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

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

{2} ISO 4217:1995, *Codes for the representation of currencies and funds.*

{3} ISO/IEC 4873:1991, *Information technology — ISO 8-bit code for information interchange — Structure and rule for implementation.*

{4} ISO 8601:1988, *Data elements and interchange formats — Information interchange — Representation of dates and times.*

{5} ISO 8859-1:1987, *Information processing — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1.*

{6} ISO 8859-2:1987, *Information processing — 8-bit single-byte coded graphic character sets — Part 2: Latin alphabet No. 2.*

{7} ISO/IEC 9899:1990, *Programming languages — C.*

{8} ISO/IEC 9945-1:1990 (IEEE Std 1003-1990), *Information technology — Portable Operating System Interface (POSIX®) — Part 1: System Application Program Interface (API) [C Language].*

{9} ISO/IEC 9945-2:1993 (IEEE/ANSI Std 1003.2-1992), *Information technology — Portable Operating System Interface (POSIX®) — Part 2: Shell and utilities.*

{10} ANSI X3.9-1978 (Reaff. 1989),[2] *American National Standard — Programming languages — FORTRAN.*

{11} ISO/IEC 13210:1994 (ANSI/IEEE Std 1003.3-1991), *Information Technology — Test Methods for Measuring Conformance to POSIX®.*

{12} IEEE Std 2003.1-1992,[3] *IEEE Standard for Information Technology — Test Methods for Measuring Conformance to POSIX® — Part 1: System Interfaces.*

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

2) ANSI documents can be obtained from the Sales Department, American National Standards Institute, 1430 Broadway, New York, NY 10018, USA.

3) IEEE documents can be obtained from The Institute of Electrical and Electronics Engineers, Inc, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

Documents Document Sharing Hub

Full document details: ISO IEC 14515-2-2003

1.3 Conformance

1.3.1 Implementation Conformance

1.3.1.1 Requirements

A PCTS used to test the conformance of POSIX.2 {9} shall conform to the requirements and recommendations contained in this standard and to the requirements and recommendations contained in POSIX.3 {11}.

1.3.2 Application Conformance

1.3.2.1 Strictly Conforming POSIX.2 Application

This standard does not specify test methods to determine whether or not applications conform to POSIX.2 {9}. This includes Strictly Conforming POSIX.2 Applications, ISO/IEC Conforming POSIX.2 Applications, <National Body> Conforming POSIX.2 Applications, and Conforming POSIX.2 Applications using Extensions.

1.3.3 Documentation Audit Conformance

1.3.3.1 Requirements

A conforming documentation audit used to test the conformance of the POSIX.2 {9} PCD (also referred to as the PCD.2) shall conform to the requirements and recommendations contained in this standard and to the requirements and recommendations contained in POSIX.3 {11}.

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

(Blank page)

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

Section 2: Conformance, Terminology, and General Requirements

(The structure of this section mirrors that of POSIX.2 {9}. In some cases, subclauses in POSIX.2 {9} have no specific assertions. In those cases, the subclause heading is included in this standard, but there is no content for that specific subclause.)

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
C-Language Data Type	*long*
C-Language Function	*system*()
C-Language Function Argument	*arg1*
C-Language Global External	*errno*
C-Language Header	`<sys/stat.h>`
C-Language Keyword	`#define`
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
Defined Term	(see text)
Definition Citation	[]
Environment Variable	**PATH**
Error Number	[EINTR]
Example Input	`echo Hello, World`
Example Output	**`Hello, World`**
Figure Reference	Figure 7-1
File Name	`/tmp`
Parameter	*<directory pathname>*
Special Character	`<newline>`
Symbolic Constant, Limit	{_POSIX_VDISABLE}, {LINE_MAX}
Table Reference	Table 6-1
Utility Name	`awk`
Utility Operand	*file_name*
Utility Option	`-c`
Utility Option With Option-Argument	`-w` *width*

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

The **Bold** font is used to show brackets that denote optional arguments in a utility synopsis, as in

`cut` **[**`-c` *list*]**[***file_name*]

These brackets shall not be used by the application unless they are specifically mentioned as literal input characters by the utility description.

There are two types of symbols enclosed in angle brackets (< >):

C-Language Headers — The header name is in the Courier font, such as `<sys/stat.h>`. When coding C programs, the angle brackets are used as required by the language.

Parameters — Parameters, also called *metavariables*, are in italics, such as <*directory pathname*>. The entire symbol, including the brackets, is meant to be replaced by the value of the symbol described within the brackets.

Numbers within braces, such as "POSIX.1 {8}," 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 D). Bibliographic entries are for information only.

In some examples, the **`Bold Courier`** font is used to indicate the output of the system that resulted from some user input, shown in `Courier`.

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] 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.

Symbolic constants are shown in two styles: those within curly braces are intended to call the attention of the reader to values in `<limits.h>` and `<unistd.h>`; those without braces are usually defined by one or a few related functions. There is no semantic difference between these two forms of presentation.

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 of the user.

The style selected for some of the special characters, such as `<newline>`, matches the form of the input given to the `localedef` utility (see 2.5.2). Generally, the characters selected for this special treatment are those that are not visually distinct, such as the control characters `<tab>` or `<newline>`.

Literal characters and strings used as input or output are shown in various ways, depending on context:

Documents Document Sharing Hub

Full document details: ISO IEC 14515-2-2003

`%`, `begin` When no confusion would result, the character or string is rendered in the Courier font and used directly in the text.

`'c'` In some cases a character is enclosed in single-quote characters, similar to a C-language character constant. Unless otherwise noted, the quotes shall not be used as input or output.

`"string"` In some cases, a string is enclosed in double-quote characters, similar to a C-language string constant. Unless otherwise noted, the quotes shall not be used as input or output.

Defined names that are usually in lowercase, particularly function names, are never used at the beginning of a sentence or anywhere else that English usage would require them to be capitalized.

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

{ [( )] }

The square brackets are most frequently used to enclose a parenthetical expression that contains a function name [such as *waitpid*()] with its built-in parentheses.

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. They have been moved to near the end of the document 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 are also informative.

NOTES that appear in a smaller point size have one of 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 and can include requirements.

Text marked as examples (including the use of "e.g.") is for information only except for C-language programs and program fragments used to represent algorithms, as described in 2.1.3.

In some cases, certain characters are interpreted as special characters by the shell. In the normative portions of this standard, these characters are shown without escape characters or quoting (see 3.2). In all examples, however, quoting has been used, showing how sample commands (utility names combined with arguments) could be passed correctly to a shell (see `sh` in 4.56) or as a string to the *system*() function.

Clause and subclause numbers correspond to the equivalent clauses and subclauses in POSIX.2 {9}. Any exceptions state the equivalent POSIX.2 {9} clause or subclause in parentheses following the title.

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

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.

2.1.2 Grammatical Conventions

Portions of this standard are expressed in terms of a special grammar notation. It is used to portray the complex syntax of certain program input. The grammar is based on the syntax used by the `yacc` utility (see A.3). However, it does not represent fully functional `yacc` input, suitable for program use: the lexical processing and all semantic requirements are described only in textual form. The grammar is not based on source used in any traditional implementation and has not been tested with the semantic code that would normally be required to accompany it. Furthermore, there is no implication that the partial `yacc` code presented represents the most efficient, or only, means of supporting the complex syntax within the utility. Implementations may use other programming languages or algorithms, as long as the syntax supported is the same as that represented by the grammar.

The following typographical conventions are used in the grammar; they have no significance except to aid in reading.

— The identifiers for the reserved words of the language are shown with a leading capital letter. (These are terminals in the grammar. Examples: `While`, `Case`.)

— The identifiers for terminals in the grammar are all named with uppercase letters and underscores. Examples: `NEWLINE`, `ASSIGN_OP`, `NAME`.

— The identifiers for nonterminals are all lowercased.

2.1.3 Miscellaneous Conventions

This standard frequently uses the C language to express algorithms in terms of programs or program fragments. The following shall be considered in reading this code:

— The programs use the syntax and semantics described by the C Standard.

— The programs are merely examples and do not represent the most efficient, or only, means of coding the interface. Implementations may use other programming languages or algorithms, as long as the results are the same as those achieved by the programs in this standard.

— C-language comments are informative and pose no requirements.

Further conventions are presented in

— Utility Conventions, 2.10, describing utility and application command-line syntax.

— File Format Notation, 2.12, describing the notation used to represent utility input and output.

Documents Document Sharing Hub

Full document details: ISO IEC 14515-2-2003

2.2 Definitions

2.2.1 Terminology

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

2.2.1.1 application: When the User Portability Utilities Option is supported, requirements associated with the term *application* also shall be interpreted to include the actions of the user who is interacting with the system by entering shell command language statements from a terminal.

2.2.1.2 can: An indication of a permissible optional feature or behavior available to the application; the implementation shall support such features or behaviors as mandatory requirements.

2.2.1.3 implementation: An object providing to applications and users the services defined by this standard.

The word *implementation* is to be interpreted to mean that object, after it has been modified in accordance with the instructions of the manufacturer to

— Configure it for conformance with this standard;

— Select some of the various optional facilities described by this standard, through customization by local system administrators or operators.

An exception to this meaning occurs when discussing conformance documentation or using the term *implementation defined*.

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

When the value or behavior in the implementation is designed to be variable or customizable on each instantiation of the system, the implementation provider shall document the nature and permissible ranges of this variation.

2.2.1.5 may: An indication of an optional feature or behavior of the implementation that is not required by this standard, although there is no prohibition against providing it.

A Strictly Conforming POSIX.2 Application is permitted to use such features, but shall not rely on the actions of the implementation in such cases. To avoid ambiguity, the reverse sense of *may* is not expressed as *may not*, but as *need not*.

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

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

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

2.2.1.7 shall: An indication of a requirement on the implementation or on Strictly Conforming POSIX.2 Applications, where appropriate.

2.2.1.8 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.2 Applications.

NOTE: The use of the word "should" to make an implementation recommendation does not of itself make any requirement on the PCD.2.

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

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

2.2.1.10 undefined: An indication that this standard imposes no portability requirements on applications for erroneous program construction, erroneous data, or use of an indeterminate value.

Implementations (or other standards) may specify the result of using that value or causing that behavior.

2.2.1.11 unspecified: An indication that this standard imposes no portability requirements on applications for a correct program construction or correct data.

Implementations (or other standards) may specify the result of using that value or causing that behavior.

2.2.2 Abbreviations

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

2.2.2.1 C Standard: ISO/IEC 9899: 1990, *Information technology—Programming languages—C*.

2.2.2.2 ERE: An Extended Regular Expression, as defined in 2.8.4 of POSIX.2 {9}.

2.2.2.3 LC_*: An abbreviation used to represent all of the environment variables named in 2.6 whose names begin with the characters "LC_".

2.2.2.4 PCD.2: The POSIX Conformance Document required by POSIX.2 {9}.

2.2.2.5 POSIX.1: ISO/IEC 9945-1: 1990 (IEEE Std 1003.1-1990) {8}.

Documents Document Sharing Hub

Full document details: ISO IEC 14515-2-2003

2.2.2.6 POSIX.2: ISO/IEC 9945-2: 1993 (IEEE Std 1003.2-1992) {9}.

2.2.2.7 POSIX.3: ISO/IEC 13210: 1994 (IEEE Std 1003.3-1991) {11}.

2.2.2.8 POSIX.3.1: IEEE Std 2003.1-1992.

2.2.2.9 POSIX.3.2: This standard.

2.2.2.10 RE [BRE]: A Basic Regular Expression, as defined in 2.8.3 of POSIX.2 {9}.

2.3 POSIX.2 {9} Conformance, Definitions and Built-in Utilities

2.3.1 POSIX.2 {9} Conformance (1.3)

2.3.1.1 POSIX.2 {9} Implementation Conformance (1.3.1)

2.3.1.1.1 POSIX.2 {9} Requirements (1.3.1.1)

D01 If the implementation provides nonstandard extensions to utilities defined in POSIX.2 {9}, then the PCD.2 defines an environment in which an application can be run with the behavior specified by the standard.

2.3.1.1.2 POSIX.2 {9} Documentation Requirements (1.3.1.2)

D02 A PCD.2 with the same structure as POSIX.2 {9} is available.

D03 The PCD.2 does not contain information about extended facilities or capabilities outside the scope of POSIX.2 {9} that do not affect the behavior of a Strictly Conforming POSIX.2 Application.

D04 The PCD.2 contains a statement indicating the full name, number, and date of the standard that applies.

D05 The PCD.2 describes the symbolic values found in **Symbolic Limits** POSIX.2 {9} 2.13.2, stating values, the conditions under which those values can change, and the limits of such variations.

GA01 The PCD.2 describes the behavior of the implementation for all implementation-defined features defined in POSIX.2 {9}.

NOTES:

(1) This general assertion covers all of the documentation assertions relating to implementation-defined features. However, each of the documentation assertions is derived from the appropriate text in POSIX.2 {9} rather than from this statement, which is too general to warrant a reference to this general assertion being included.

(2) All documentation assertions in this standard that explicitly reference the PCD.2 implicitly refer to this general assertion.

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

2.3.1.1.3 POSIX.2 {9} Conforming Implementation Options. (1.3.1.3)

There are no assertions specific to this subclause.

2.3.1.2 Application Conformance. (1.3.2)

There are no assertions specific to this subclause.

2.3.2 POSIX.2 {9} Definitions (2.2)

2.3.2.1 POSIX.2 {9} Terminology (2.2.1)

There are no assertions specific to this subclause.

2.3.2.2 POSIX.2 {9} General Terms (2.2.2)

D06 The PCD.2 describes the method of associating privileges with a process with regard to those function calls and function call options defined in POSIX.1 {8} that need special privileges.

D07 The PCD.2 describes the number of contiguous sequences of bits that compose a byte.

GA02 When the collation sequence associated with the current locale assigns more than one weight to the collating elements, then each of these weights is used in turn to determine the order of strings that collate as equal using the previously assigned weights.

NOTES:

(1) This general assertion will only apply to commands and utilities that compare strings, for example, `sort`, `grep`, `awk`, and `sed`. These will generally be located by the reference to **LC_COLLATE** in the Environment Variables area of the command description.

(2) POSIX.2 {9} contains little on file access permissions and, where it does, assertions have been written in this standard concerning them.

The accessibility of a file by a particular utility is also affected by the privileges associated with that utility. POSIX.2 {9} does not restrict these privileges for a conforming implementation and it may, therefore, not be possible for a test suite to generate an environment in which a particular file cannot be accessed by a utility.

001(B) When a file is closed by the last process that had it open, then all time-related fields marked for update are updated.
See Reason 1 in Section 5 of POSIX.3 {11}.

D08 If the implementation uses additional criteria beyond those specified by POSIX.2 {9} to assign a process to the file group class of a file, and this is documented:

The details are contained in 2.2.2.68 of the PCD.2.

D09 If an implementation of a "function" that is required by POSIX.2 {9} to update time-related fields for reasons other than pathname resolution updates fields other than those required:

The details are contained in 2.2.2.80 of the PCD.2 or in the subclause of the PCD.2 that corresponds to the description subclause in

Documents Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

POSIX.2 {9} where the "function" is defined.

D10 If an implementation of a function that is not required by POSIX.2 {9} to update time-related fields happens to do so, and this is documented:

The details are contained in 2.2.2.80 of the PCD.2 or in the subclause of the PCD.2 that corresponds to the description subclause in POSIX.2 {9} where the "function" is defined.

002(B) No time-related fields are updated on read-only file systems.
See Reason 1 in Section 5 of POSIX.3 {11}.

GA03 When the job control job ID is specified as `%%`, then this is interpreted as a reference to the current job.

GA04 When the job control job ID is specified as `%+`, then this is interpreted as a reference to the current job.

GA05 When the job control job ID is specified as `%-`, then this is interpreted as a reference to the previous job.

GA06 When the job control job ID is specified as `%`*n*, where *n* is a decimal number, then this is interpreted as a reference to the job number *n*.

GA07 When the job control job ID is specified as `%`*string*, then this is interpreted as a reference to the job whose command commences with *string*.

GA08 When the job control job ID is specified as `%?`*string*, then this is interpreted as a reference to the job whose command contains *string*.

GA09 When the first filename component of the pathname is "." and the pathname does not begin with a slash, then the pathname is resolved by locating the second filename component (when specified) in the current working directory.

GA10 When the pathname begins with a slash, then the pathname is resolved by locating the first filename component of the pathname in the root directory of the process.

GA11 When the first filename component of the pathname is ".." and the pathname does not begin with a slash, and the current working directory is not the root directory of the process, then the pathname is resolved by locating the second filename component (when specified) in the parent directory of the current working directory.

GA12 When the pathname consists of a single component followed by a slash and the final component of the pathname is a directory, then the pathname is resolved to the directory specified by the single component of the pathname.

GA13 When the pathname consists of a single component followed by more than one slash and the single component of the pathname is a directory, then the pathname is resolved to the directory specified by the single component of the pathname.

GA14 When the pathname points to the string `F1/F2`, then the pathname is resolved to the file `F2` in the directory `F1` in the current directory.

GA15 When the pathname points to the string `F1/./F2`, then the pathname is resolved to the file `F2` in the directory `F1` in the current directory.

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

GA16 When the pathname points to the string `F1/../F1/F2`, then the pathname is resolved to the file `F2` in the directory `F1` in the current directory.

GA17 When the pathname points to the string `F1//F2`, then the pathname is resolved to the file `F2` in the directory `F1` in the current directory.

GA18 A pathname of no more than {PATH_MAX} bytes can be resolved.

GA19 If {_POSIX_NO_TRUNC} is not in effect for one or more path prefixes:
When a pathname component is longer than {NAME_MAX} bytes and {_POSIX_NO_TRUNC} is not in effect for the directory containing the file referred to by this pathname component, then the pathname is resolved with the corresponding pathname component truncated to {NAME_MAX} bytes.

GA20 If {_POSIX_NO_TRUNC} is in effect for one or more path prefixes:
When a pathname component is longer than {NAME_MAX} bytes and {_POSIX_NO_TRUNC} is in effect for the directory containing the file referred to by this pathname component, then pathname resolution results in an error.

003(B) A process group ID is not reused by the system until the process group lifetime ends.
See Reason 1 in Section 5 of POSIX.3 {11}.

004(B) A process ID is not reused by the system until the process lifetime ends.
See Reason 1 in Section 5 of POSIX.3 {11}.

D11 The PCD.2 describes the mechanism that the implementation provides for extended security controls (if any).

D12 The PCD.2 describes the manner in which parent process ID is assigned to a system process after the lifetime of the process creator has ended.

D13 The PCD.2 describes the manner in which the implementation interprets a pathname with exactly two leading slashes.

D14 The PCD.2 describes the characteristics of a read-only file system

R01 The assignment name=word assigns the value word to the variable name. (See assertion 001 in 3.1.)

R02 When name does not already exist, then name is created. (See assertion 002 in 3.1.)

R03 When word is not specified, then name is given a null value. (See assertion 003 in 3.1.)

2.3.2.3 POSIX.2 {9} Abbreviations (2.2.3)

There are no assertions specific to this subclause.

Documents Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

2.3.3 Built-in Utilities (2.3)

GA21 The implementation provides an `exec`-able version of each of the standard utilities, including regular built-ins, in POSIX.2 {9}.

Testing Requirements:

For each utility that refers to the above general assertion, the ability to invoke the utility directly from `env`, `find`, `nohup`, and `xargs` should be tested. In the case that the User Portability Utilities Option is supported, the ability to invoke the utility directly from `nice` and `time` should also be tested.

GA22 If the implementation conforms to POSIX.1 {8}:
Then each of the utilities in POSIX.2 {9} can be *exec*()ed.

GA23 If **POSIX2_C_BIND** is supported:
Then each of the utilities in POSIX.2 {9} can be executed using the *system*() or *popen*() interfaces.

2.4 Character Set

001(B) A conforming implementation supports at least one coded character set.
See Reason 2 in Section 5 of POSIX.3 {11}.

002(A) Each locale includes the portable character set as defined in Table 2-4 of POSIX.2 {9}, and each character specified in that table has a unique encoding value except where the glyphs are identical.

003(A) The null character, NUL, which has all bits set to zero, is in the set of characters.

004(A) The encoded values associated with the digits `'0'` to `'9'` are such that the value of each character after `'0'` is one greater than the value of the previous character.

D15 The system documentation describes the rules for the creation of additional character set description files.

2.4.1 Character Set Description File

D16 The PCD.2 describes whether or not users or applications can provide additional character set description files.

D17 The system documentation describes the rules for the creation of additional character set description files.

005(A) A character set description file exists for at least one coded character set supported by the implementation.

006(B) Each character set description file contains the symbolic names, and defines characteristics for the portable character set and the encoding for the characters specified by Table 2-4 of POSIX.2 {9}.
See Reason 1 in Section 5 of POSIX.3 {11}.

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

007(D) If the control characters specified in Table 2-5 of POSIX.2 {9} are supported by the implementation:

> Each character set description file includes the symbolic names for the control character set specified by Table 2-5 of POSIX.2 {9}.

See Reason 1 in Section 5 of POSIX.3 {11}.

008(B) Each coded character set uses symbolic names followed by the character values to define the characteristics and character values in the coded character set.
See Reason 1 in Section 5 of POSIX.3 {11}.

009(B) The character set characteristic `<mb_cur_max>` can be defined in each description file before the character set mappings are defined.
See Reason 3 in Section 5 of POSIX.3 {11}.

010(B) The default value for `<mb_cur_max>` is 1.
See Reason 3 in Section 5 of POSIX.3 {11}.

011(B) When the character set characteristic `<mb_cur_min>` is defined, then the value of `<mb_cur_min>` is less than or equal to the value of `<mb_cur_max>`.
See Reason 3 in Section 5 of POSIX.3 {11}.

012(B) The default value for `<mb_cur_min>` is the same as the value of `<mb_cur_max>`.
See Reason 3 in Section 5 of POSIX.3 {11}.

013(B) The character set characteristic `<escape_char>` can be defined in each description file before the character set mappings are defined.
See Reason 3 in Section 5 of POSIX.3 {11}.

014(B) The default value for `<escape_char>` is backslash (`\`).
See Reason 3 in Section 5 of POSIX.3 {11}.

015(B) The character set characteristic `<comment_char>` can be defined in each description file before the character set mappings are defined.
See Reason 3 in Section 5 of POSIX.3 {11}.

016(B) The default value for `<comment_char>` is the number sign (#).
See Reason 3 in Section 5 of POSIX.3 {11}.

017(B) The character set mapping definition is the set of lines between the CHARMAP and the END CHARMAP lines in the character set definition file.
See Reason 3 in Section 5 of POSIX.3 {11}.

018(B) Empty lines and lines commencing with `<comment_char>` in column one of a charmap line are ignored.
See Reason 3 in Section 5 of POSIX.3 {11}.

019(B) The format of each line of the character mapping set definition in the description file is in one of the following forms:

> "`%s %s %s\n`", `<symbolic-name>`, `<encoding>`, `<comments>`
>
> "`%s ... %s %s %s\n`", `<symbolic-name>`, `<symbolic_name>`, `<encoding>`, `<comments>`

where each byte of the `<encoding>` is expressed in one of the following

Documents Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

forms:

"%cd%d", <escape_char>, <decimal byte value>

"%cx%x", <escape_char>, <hexadecimal byte value>

"%c%o", <escape_char>, <octal byte value>

See Reason 3 in Section 5 of POSIX.3 {11}.

020(B) Each coded character value within a character set description file can be associated with more than one symbolic name.
See Reason 3 in Section 5 of POSIX.3 {11}.

021(B) The symbolic names used in each character set description file are unique within the context of that description file.
See Reason 3 in Section 5 of POSIX.3 {11}.

D18 The implementation documents the manner in which the constants concatenated for multibyte character values are represented in the character stored in the system.

022(B) When constants are concatenated for multibyte character values, they are of the same type and are interpreted in byte order from first to last with the least significant byte of the multibyte character specified by the last constant.
See Reason 3 in Section 5 of POSIX.3 {11}.

2.5 Locale

R01 The behavior of the utilities is based on the setting of the current locale. (Refer to assertions in 2.6).

D19 The PCD.2 defines the method used to locate the locale in the case that the value associated with a locale environment variable does not begin with a slash (/).

001(A) When the value of any of the environment variables associated with a locale category is set to the value of the *name* argument associated with a previous successful call to `localedef`, then the characteristics associated with that locale category are set to match the locale source definition used to generate *name*.

002(C) If {**POSIX2_LOCALEDEF**} is defined:

When the value of the locale environment variable begins with a slash (/), then it is interpreted as the pathname of the locale definition.

D20 The PCD.2 specifies the type of file (regular, directory, etc.) used to store the locale definition.

Documents Document Sharing Hub

Full document details: ISO IEC 14515-2-2003

2.5.1 POSIX Locale

There are no assertions specific to this subclause.

2.5.2 Locale Definition

D21 The PCD.2 describes the implementation-supplied locales in the format specified in subclause 2.5 of POSIX.2 {9}.

D22 The PCD.2 describes locale categories other than those specified in subclause 2.5 of POSIX.2 {9} that the implementation allows to appear in the locale definition file.

D23 The PCD.2 describes the additional keywords that are recognized by the implementation and the locale category to which each of these keywords applies.

003(C) If {**POSIX2_LOCALEDEF**} is defined:
A call to `localedef` accepts a locale definition file containing at least one locale category source definition.

004(D) If {**POSIX2_LOCALEDEF**} is defined:
A call to `localedef` accepts a locale definition file that contains more than one category source definition and that lists those definitions documented in subclause 2.5 of POSIX.2 {9} before any implementation-defined definitions.
See Reason 3 in Section 5 of POSIX.3 {11}.

005(C) If {**POSIX2_LOCALEDEF**} is defined:
A call to `localedef` accepts a locale definition file in which each category source definition contains either the definition of a category or a `copy` directive (as described in subclause 4.35.5.2 of POSIX.2 {9}).

006(C) If {**POSIX2_LOCALEDEF**} is defined:
A call to `localedef` accepts a category source definition containing a category header that consists of the character string naming the category and beginning with the characters “LC_”.

007(C) If {**POSIX2_LOCALEDEF**} is defined:
A call to `localedef` accepts a category source definition containing a category trailer that consists of the character string `END` followed by one or more `<blank>`s and the string used in the corresponding category header.

008(C) If {**POSIX2_LOCALEDEF**} is defined:
A call to `localedef` accepts a locale definition file in which no category source definition keyword contains a keyword that begins with the characters “LC_”.

009(C) If {**POSIX2_LOCALEDEF**} is defined:
A call to `localedef` accepts a locale definition file in which no category source definition keyword, other than `copy`, is repeated within that category source definition.

Documents Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

010(C) If {**POSIX2_LOCALEDEF**} is defined:
A call to `localedef` accepts a locale definition file in which an operand following an identifier in a category source definition is separated from the identifier by one or more `<blank>`s.

011(C) If {**POSIX2_LOCALEDEF**} is defined:
A call to `localedef` accepts a locale definition file in which operands that are a string of characters are enclosed in double quotes (").

012(C) If {**POSIX2_LOCALEDEF**} is defined:
A call to `localedef` accepts a locale definition file in which multiple operands following a keyword are separated by semicolons, optionally preceded and/or followed by 1 or more `<blank>`s.

013(C) If {**POSIX2_LOCALEDEF**} is defined:
When the <*comment character*> is not changed in a locale definition file, then a call to `localedef` ignores lines that begin in column 1 with the default comment character (#).

014(C) If {**POSIX2_LOCALEDEF**} is defined:
When the first category header in a locale definition file is preceded by a line of text having the format starting in column 1 "`comment_char %c\n`", <*comment character*>, then a call to `localedef` modifies the comment character to <*comment character*> and ignores lines that begin in column 1 with this character.

015(C) If {**POSIX2_LOCALEDEF**} is defined:
When lines of the locale definition file are comment lines, an escaped `<newline>` does not cause the line to be continued or accumulated.

016(C) If {**POSIX2_LOCALEDEF**} is defined:
A call to `localedef` ignores blank lines in the locale definition file.

017(C) If {**POSIX2_LOCALEDEF**} is defined:
When the <*escape character*> is not changed in the locale definition file, then a call to `localedef` considers the escape character to be the default of backslash (`\`).

018(C) If {**POSIX2_LOCALEDEF**} is defined:
When the first category header in a locale definition file is preceded by a line of text having the format starting in column 1 "`escape_char %c\n`", <*escape character*>, then a call to `localedef` modifies the escape character to be <*escape character*>.

019(C) If {**POSIX2_LOCALEDEF**} is defined:
When the <*escape character*> is used with an operand, irrespective of whether this operand is in double quotes, and the <*escape character*> is preceded by another <*escape character*>, then a call to `localedef` interprets the escape character as itself.

020(C) If {**POSIX2_LOCALEDEF**} is defined:
A call to `localedef` accepts a locale definition file containing lines of length up to and including {LINE_MAX} bytes.

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

021(C) If {**POSIX2_LOCALEDEF**} is defined:
When the last character of a line in a locale definition file is the escape character, then a call to `localedef` discards that character from the input and the next line is added to the previous one.

022(C) If {**POSIX2_LOCALEDEF**} is defined:
When lines of a locale definition file are accumulated using escape characters, then `localedef` accepts a continued line of length **LINE_MAX**+1 bytes.

023(D) If {**POSIX2_LOCALEDEF**} is defined:
When lines of a locale definition file are accumulated using escape characters, then `localedef` places no limit on the length of the accumulated line.
See Reason 3 in Section 5 of POSIX.3 {11}.

024(C) If {**POSIX2_LOCALEDEF**} is defined:
When `symbolic_name` matches a symbolic name defined in the character set description file specified via `-f` *charmap*, then a call to `localedef` accepts a sequence of characters in the format <*symbolic_name*> as an operand.

025(C) If {**POSIX2_LOCALEDEF**} is defined:
When a sequence of characters in the format <*symbolic_name*> is used as an operand in the categories LC_CTYPE or LC_COLLATE and `symbolic_name` does not match a symbolic name defined in the *charmap*, then a call to `localedef` treats this as a warning condition.

026(C) If {**POSIX2_LOCALEDEF**} is defined:
When a sequence of characters in the format `<symbolic_name>` is used as an operand in categories other than **LC_CTYPE** or **LC_COLLATE** and the `symbolic_name` does not match a symbolic name defined in the `charmap` file, then a call to `localedef` treats this as an error.

027(C) If {**POSIX2_LOCALEDEF**} is defined:
When a sequence of characters in the format <*symbolic_name*> is used as an operand in a `collating-element` or `collating-symbol` clause and the `symbolic_name` duplicates a symbolic name in the *charmap*, then a call to `localedef` treats this as a error.

028(C) If {**POSIX2_LOCALEDEF**} is defined:
When the `symbolic_name` of a <*symbolic_name*> operand contains the escape character or the right-angle bracket character without these characters being preceded by the escape character, then a call to `localedef` treats this as an error.

029(C) If {**POSIX2_LOCALEDEF**} is defined:
When the double quote character (") is used within an operand, irrespective of whether this operand is enclosed in double quotes, and the double-quote character is preceded by an escape character, then a call to `localedef` interprets the double-quote character as itself.

Documents Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

030(C) If {**POSIX2_LOCALEDEF**} is defined:
When any of the characters `,`, `;`, <, >, or *escape_char* is used in an operand outside strings enclosed by double-quote characters, and the character is preceded by the escape character, then a call to `localedef` interprets the character as itself. When a character other than ", `,`, `;`, <, >, or *escape_char* is used in an operand, then a call to `localedef` interprets the character as itself.

031(C) If {**POSIX2_LOCALEDEF**} is defined:
When the < character is used within an operand, irrespective of whether this operand is enclosed in double quotes, and the left-angle bracket character is preceded by an escape character, then a call to `localedef` interprets the < character as itself.

032(C) If {**POSIX2_LOCALEDEF**} is defined:
When an octal constant is used as an operand and is represented by the escape character followed by two or more octal digits, then a call to `localedef` interpets the octal constant as a byte value.

033(C) If {**POSIX2_LOCALEDEF**} is defined:
When a sequence of concatenated octal constants are used to represent a multibyte character operand, then a call to `localedef` interprets each of the octal constants as a byte value.

034(C) If {**POSIX2_LOCALEDEF**} is defined:
When an hex constant is used as an operand and is represented by the escape character followed by an `x` character followed by two or more hexadecimal digits, then a call to `localedef` interprets the hexadecimal constant as a byte value.

035(C) If {**POSIX2_LOCALEDEF**} is defined:
When a sequence of concatenated hexadecimal constants are used to represent a multibyte character operand, then a call to `localedef` interprets each of the hexadecimal constants as a byte value.

036(C) If {**POSIX2_LOCALEDEF**} is defined:
When a decimal constant is used as an operand and is represented by the escape character followed by a `d` character followed by two or more decimal digits, then a call to `localedef` interprets the decimal constant as a byte value.

037(C) If {**POSIX2_LOCALEDEF**} is defined:
When a sequence of concatenated decimal constants are used to represent a multibyte character operand, then a call to `localedef` interprets each of the decimal constants as a byte value.

038(D) If {**POSIX2_LOCALEDEF**} is defined:
When an operand uses characters not existing in the character set for which the locale definition is created, then a call to `localedef` treats this as an error.
See Reason 3 in Section 5 of POSIX.3 {11}.

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

2.5.2.1 LC_CTYPE

039(C) If {**POSIX2_LOCALEDEF**} is defined:
When an ellipsis symbol is used in the LC_CTYPE category source definition, and this is preceded by an encoded value and followed by a higher encoded value, then a call to `localedef` treats this as the inclusive sequence of values between these two encoded values.

040(D) If {**POSIX2_LOCALEDEF**} is defined:
When a call to `localedef` includes a missing character, then this character has either an encoded value dependent on the `charmap` file in effect or a value derived from an implementation-defined character mapping.
See Reason 3 in Section 5 of POSIX.3 {11}.

041(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains a `copy` keyword with an operand that is the name of an existing locale, then a call to `localedef` uses that locale as the source for the definition of this category.

042(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains a `copy` keyword and also contains other keywords, then a call to `localedef` treats this as an error.

043(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `upper` keyword, then a call to `localedef` interprets the operands as characters to be classified as uppercase letters.

044(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `upper` keyword, and operands containing characters specified for the keywords `cntrl`, `digit`, `punct`, or `space` are specified, then a call to `localedef` treats this as an error.

045(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `upper` keyword, and all or some of the uppercase letters A through Z as defined in Table 2-4 of POSIX.2 {9} are not specified as operands, then a call to `localedef` includes these unspecified characters in the `upper` classification.

046(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition omits the `upper` keyword, then a call to `localedef` includes the uppercase letters A through Z as defined in Table 2-4 of POSIX.2 {9} in the `upper` classification.

047(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `upper` keyword and the uppercase letters A through Z as defined in Table 2-4 of POSIX.2 {9} are specified as operands, then a call to `localedef` accepts them silently.

Documents Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

D24 The PCD.2 describes character values of the uppercase letters A through Z, as defined in Table 2-4 of POSIX.2 {9} used when the `upper` keyword is not specified.

048(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `lower` keyword, then a call to `localedef` interprets the operands as characters to be classified as lowercase letters.

049(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `lower` keyword and operands containing characters specified for the keywords `cntrl`, `digit`, `punct`, or `space` are specified, then a call to `localedef` treats this as an error.

050(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `lower` keyword and all or some of the lowercase letters a through z as defined in Table 2-4 of POSIX.2 {9} are not specified as operands, then a call to `localedef` includes these unspecified characters in the `lower` classification.

051(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition omits the `lower` keyword, then a call to `localedef` includes the lowercase letters a through z as defined in Table 2-4 of POSIX.2 {9} in the `lower` classification.

052(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `lower` keyword and the lowercase letters a through z as defined in POSIX.2 {9} 2.4 Table 2-4 are specified as operands, then a call to `localedef` accepts them silently.

D25 The PCD.2 describes character values of the lowercase letters a through z, as defined in Table 2-4 of POSIX.2 {9} used when the `lower` keyword is not specified.

053(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `alpha` keyword, then a call to `localedef` interprets the characters specified in the operands as alphabetic characters.

054(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `alpha` keyword and operands containing characters specified for the keywords `cntrl`, `digit`, `punct`, or `space` are specified, then a call to `localedef` treats this as an error.

055(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `alpha` keyword and the associated operands omit characters classified as `upper` or `lower`, then a call to `localedef` includes these omitted characters in the `alpha` classification.

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

056(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition omits the `alpha` keyword, then a call to `localedef` includes the characters classified as `upper` and `lower` in the `alpha` classification.

057(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `alpha` keyword and any characters classified as `upper` and `lower` are specified as operands, then a call to `localedef` accepts them silently.

058(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `digit` keyword with operands omiting some or all of the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9, then a call to `localedef` includes these omitted characters in the `digit` classification.

059(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition omits the `digit` keyword, then a call to `localedef` includes the numerical digits in the `digit` classification.

D26 The PCD.2 describes character values of the digits 0 through 9, as defined in Table 2-4 of POSIX.2 {9}, used when the `digit` keyword is not specified.

060(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `digit` keyword with operands defining the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9, in contiguous ascending sequence by numerical value, then a call to `localedef` accepts them silently.

061(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `digit` keyword with operands other than the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9, then a call to `localedef` treats this as an error.

062(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `space` keyword, then a call to `localedef` interprets the characters specified in the operands as white space characters.

063(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `space` keyword and operands containing characters specified for the keywords `upper`, `lower`, `alpha`, `digit`, `graph`, or `xdigit` are included, then a call to `localedef` treats this as an error.

064(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `space` keyword and operands omit some or all of the characters `<space>`, `<form-feed>`, `<newline>`, `<carriage-return>`, `<tab>`, or `<vertical-tab>`, and characters included in the class `blank`, then a call to `localedef` includes these omitted characters in the `space` character classification.

Documents Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

065(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contain the `space` keyword and operands contains the characters `<space>`, `<form-feed>`, `<newline>`, `<carriage-return>`, `<tab>`, or `<vertical-tab>`, and characters included in the class `blank`, then a call to `localedef` accepts them silently.

066(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition omits the `space` keyword, then a call to `localedef` includes the characters `<space>`, `<form-feed>`, `<newline>`, `<carriage-return>`, `<tab>`, and `<vertical-tab>` in the `space` character classification.

D27 The PCD.2 describes character values of the characters `<space>`, `<form-feed>`, `<newline>`, `<carriage-return>`, `<tab>`, and `<vertical-tab>`, as defined in Table 2-4 of POSIX.2 {9}, used when the `space` keyword is not specified.

067(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `cntrl` keyword, then a call to `localedef` interprets the characters specified in the operands as control characters.

068(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `cntrl` keyword and operands containing characters specified for the keywords `upper`, `lower`, `alpha`, `digit`, `punct`, `graph`, or `xdigit` are included, then a call to `localedef` treats this as an error.

069(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `punct` keyword, then a call to `localedef` interprets the characters specified in the operands as punctuation characters.

070(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `punct` keyword and operands containing characters specified for the keywords `upper`, `lower`, `alpha`, `digit`, `graph`, or `xdigit` are included or the `<space>` character is included, then a call to `localedef` treats this as an error.

071(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `graph` keyword, then a call to `localedef` interprets the characters specified in the operands as printable characters.

072(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `graph` keyword and operands containing characters specified for the keywords `cntrl` are included, then a call to `localedef` treats this as an error.

073(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `graph` keyword and the operands omit some or all of the characters

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

specified in classes `upper`, `lower`, `alpha`, `digit`, `xdigit`, and `punct`, then a call to `localedef` includes the characters specified in these omitted classes in the `graph` character classification.

074(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition omits the `graph` keyword, then a call to `localedef` includes the characters specified in classes `upper`, `lower`, `alpha`, `digit`, `xdigit`, and `punct` in the `graph` character classification.

075(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `graph` keyword and the operands contain the characters specified in classes `upper`, `lower`, `alpha`, `digit`, `xdigit`, and `punct`, then a call to `localedef` accepts them silently.

076(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `print` keyword, then a call to `localedef` interprets the characters specified in the operands as printable characters.

077(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `print` keyword and operands containing characters specified for the keywords `cntrl` are included, then a call to `localedef` treats this as an error.

078(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `print` keyword, and the operands omit some or all of the characters specified in classes `upper`, `lower`, `alpha`, `digit`, `xdigit`, and `punct` and the `<space>` character, then a call to `localedef` includes these omitted characters specified in these classes in the `print` classification.

079(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition omits the `print` keyword, then a call to `localedef` includes the characters specified in classes `upper`, `lower`, `alpha`, `digit`, `xdigit`, and `punct`, and the `<space>` character, in the `print` character classification.

080(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `print` keyword and the operands contain the characters specified in classes `upper`, `lower`, `alpha`, `digit`, `xdigit`, and `punct`, and the `<space>` character, then a call to `localedef` accepts them silently.

081(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `xdigit` keyword, then a call to `localedef` interprets the operands as hexadecimal digits.

082(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `xdigit` keyword with operands defining the digits 0, 1, 2, 3, 4, 5, 6, 7, 8,

Documents Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

or 9, and the letters A, B, C, D, E, F, a, b, c, d, e, or f in contiguous ascending sequence by numerical value, then a call to `localedef` accepts them silently.

083(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `xdigit` keyword with operands other than the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9, and the letters A, B, C, D, E, F, a, b, c, d, e, or f, then a call to `localedef` treats this as an error.

084(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category contains the `xdigit` keyword and the associated operands omit some or all characters classified as hexadecimal digits, then a call to `localedef` includes these omitted characters in the `xdigit` classification.

085(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition omits the `xdigit` keyword, then a call to `localedef` includes these omitted characters in the `xdigit` classification.

D28 The PCD.2 describes character values of the digits 0 through 9, the uppercase letters A through F, and the lowercase letters a through f, as defined in Table 2-4 of POSIX.2 {9} used when the `xdigit` keyword is not specified.

086(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `blank` keyword, then a call to `localedef` interprets the characters specified in the operands as `<blank>` characters.

087(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition omits the `blank` keyword, then a call to `localedef` treats the characters `<space>` and `<tab>` as belonging to the class of `<blank>` characters.

088(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `blank` keyword and the operands omit some or all of the characters `<space>` and `<tab>`, then a call to `localedef` includes these omitted characters in the `blank` classification.

089(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `blank` keyword and the operands contain the characters `<space>` and `<tab>`, then a call to `localedef` accepts them silently.

090(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `toupper` keyword and each operand consists of a comma-separated pair of characters enclosed in parentheses, then a call to `localedef` interprets the operands as pairs of characters defining the correspondence of the lowercase character (the first of the pair) to its uppercase counterpart (the second of the pair).

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

091(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition omits the `toupper` keyword, then a call to `localedef` includes the lowercase letters a through z and their corresponding uppercase letters A through Z, as defined in Table 2-4 of POSIX.2 {9} in the `toupper` keyword.

092(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `toupper` keyword, and the operands contain the characters lowercase letters a through z and their corresponding uppercase letters A through Z, as defined in Table 2.4 of POSIX.2 {9}, then a call to `localedef` accepts them silently.

093(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `toupper` keyword, and any of the first characters in the character pairs specified by the operands is not specified for the `lower` keyword, then a call to `localedef` treats this as an error.

D29 The PCD.2 describes character values of the lowercase letters a through z, and their corresponding uppercase letters A through Z, as defined in Table 2-4 of POSIX.2 {9}, used when the `toupper` keyword is not specified.

094(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `tolower` keyword and each operand consists of a comma-separated pair of characters enclosed in parentheses, then a call to `localedef` interprets the operands as pairs of characters defining the correspondence of the uppercase character (the first of the pair) to its lowercase counterpart (the second of the pair).

095(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `tolower` keyword, and any of the first characters in the character pairs specified by the operands is not specified for the `upper` keyword, then a call to `localedef` treats this as an error.

096(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition omits the `toupper` and `tolower` keywords, then a call to `localedef` includes the uppercase letters A through Z and their corresponding lowercase letters a through z, as defined in Table 2-4 of POSIX.2 {9}, in the `toupper` and `tolower` keywords.

097(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition contains the `tolower` keyword and the operands contain the characters uppercase letters A through Z and their corresponding lowercase letters a through z, as defined in Table 2-4 of POSIX.2 {9}, then a call to `localedef` accepts them silently.

Documents Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

098(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_CTYPE category source definition omits the `tolower` keyword, then a call to `localedef` uses the reverse mapping of the one specified for the `toupper` keyword as the correspondence between lowercase and uppercase characters.

2.5.2.2 LC_COLLATE

099(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `copy` keyword with an operand that is the name of an existing locale, then a call to `localedef` uses that locale as the source for the definition of this category.

100(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `copy` keyword with an operand that is the name of an existing locale, and also contains other keywords, then a call to `localedef` treats this as an error.

2.5.2.2.1 collating-element Keyword

101(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `collating-element` keyword, then a call to `localedef` interprets the operand as a multicharacter collating element.

NOTE: A multicharacter collating element is not defined.

102(D) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `collating-element` keyword, the format of the operands is `''collating-element %s from \''%s\''\n''`, <*collating-symbol*>, <*string*> where the <*collating-symbol*> operand is a symbolic name, enclosed between angle brackets, and where the <*string*> operand is a string of two or more characters that collate as an entity.
See Reason 3 in Section 5 of POSIX.3 {11}.

103(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `collating-element` keyword, and the collating symbol part of the operand duplicates another symbolic name in the current *charmap* file, then `localedef` treats this as an error.

104(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `collating-element` keyword, and the collating symbol part of the operand duplicates another symbolic name in the current collation definition, then `localedef` treats this as an error.

Documents Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

2.5.2.2.2 collating-symbol Keyword

105(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `collating-symbol` keyword, then a call to `localedef` interprets the operand as a multicharacter collating symbol for use in collation order statements.

106(D) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `collating-symbol` keyword, the format of the operands shall be "`collating-symbol %s\n`", <*collating-symbol*> where the <*collating-symbol*> operand is a symbolic name, enclosed between angle brackets.
See Reason 3 in Section 5 of POSIX.3 {11}.

107(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `collating-symbol` keyword, and the collating symbol operand duplicates another symbolic name in the current *charmap* file, then `localedef` treats this as an error.

108(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `collating-symbol` keyword, and the collating symbol operand duplicates another symbolic name in the current collation definition, then `localedef` treats this as an error.

2.5.2.2.3 order_start Keyword

109(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `order_start` keyword, and no operands are present, one operand containing one forward directive is assumed.

110(D) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `order_start` keyword, and operands are present, the format of the operands are "`order_start %s;%s;...;%s\n`", <*sort-rules*>, <*sort-rules*>, `....`, <*sort-rules*>.
See Reason 3 in Section 5 of POSIX.3 {11}.

111(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `order_start` keyword, and operands are present, then each operand consists of one or more collation directives, separated by commas.

112(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `order_start` keyword, and more than one operand is present, the operands are separated by semicolons.

Documents Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

113(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `order_start` keyword, and more than one operand is present, then `localedef` treats the order in which they appear as the relative weight with which they are applied.

114(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `order_start` keyword, and more than one operand is present, then `localedef` treats the order in which they appear as the order in which they are applied.

115(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `order_start` keyword, and more than {COLL_WEIGHTS_MAX} operands are present, then `localedef` issues a warning message.

116(D) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `order_start` keyword, and one of the operands contains the collation directive `forward`, then `localedef` interprets comparison operations for the given weight level to proceed from the start of string towards the end of string.
See Reason 3 in Section 5 of POSIX.3 {11}.

117(D) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `order_start` keyword, and one of the operands contains the collation directive `backward`, then `localedef` interprets comparison operations for the given weight level to proceed from the end of string towards the beginning of string.
See Reason 3 in Section 5 of POSIX.3 {11}.

118(D) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `order_start` keyword, and one of the operands contains both the `forward` and `backward` collation directives, then `localedef` treats this as an error.
See Reason 3 in Section 5 of POSIX.3 {11}.

119(D) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `order_start` keyword, and one of the operands contains the collation directive `position`, then `localedef` interprets comparison operations for the given weight level to consider the relative position of non-IGNOREd elements in the respective strings, starting with the first non-IGNOREd element after the fewest IGNOREd collating elements from the start of the string.
See Reason 3 in Section 5 of POSIX.3 {11}.

120(D) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains the `order_start` keyword, and one of the operands contains the collation directive `position`, then `localedef` interprets comparison operations for the given weight level to consider the relative position

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

of non-IGNOREd elements in the respective strings, using collating values assigned to the elements to determine comparisons where the elements are in the same relative position.

See Reason 3 in Section 5 of POSIX.3 {11}.

121(D) If {**POSIX2_LOCALEDEF**} is defined:

When the LC_COLLATE category source definition contains the `order_start` keyword, and one of the operands contains the collation directive `position`, then `localedef` interprets comparison operations for the given weight level to consider the relative position of non-IGNOREd elements in the respective strings, using subsequent non-IGNOREd characters when the collating values assigned to the elements to determine comparisons where the elements are in the same relative position are equal.

See Reason 3 in Section 5 of POSIX.3 {11}.

2.5.2.2.4 Collation Order

122(D) If {**POSIX2_LOCALEDEF**} is defined:

When the LC_COLLATE category source definition contains the `order_start` keyword, subsequent entries in the definition contain collating elements in the format: "`%s %s;%s;...;%s\n`", <*collating-identifier*>, <*weight*>, <*weight*>, ..., <*weight*> where a collating-identifier consists of a character, a <*collating-element*>, a <*collating-symbol*>, an ellipsis, or the special symbol UNDEFINED.

See Reason 3 in Section 5 of POSIX.3 {11}.

123(C) If {**POSIX2_LOCALEDEF**} is defined:

When the LC_COLLATE category source definition contains a `collating-element` entry, then `localedef` treats the order in which collating elements are specified as the character order sequence, from lowest to highest.

124(C) If {**POSIX2_LOCALEDEF**} is defined:

When the LC_COLLATE category source definition contains a `collating-element` entry, then `localedef` treats the order in which collating elements are specified as the character collation sequence, with the null character always comparing the lowest.

125(C) If {**POSIX2_LOCALEDEF**} is defined:

When the LC_COLLATE category source definition contains a `collating-element` entry, then `localedef` treats a <*collating-element*> as a multicharacter collating element, specifying the character sequence to be collated as a unit and in the relative order specified by its place.

126(C) If {**POSIX2_LOCALEDEF**} is defined:

When the LC_COLLATE category source definition contains a `collating-element` entry, then `localedef` treats a <*collating-symbol*> as defining the position in the relative order for use in weights.

127(C) If {**POSIX2_LOCALEDEF**} is defined:

When the LC_COLLATE category source definition contains a `collating-element` entry, then `localedef` interprets an ellipsis

Documents Document Sharing Hub

Full document details: ISO IEC 14515-2-2003

symbol as indicating that all characters with a coded character set value higher than the value of the character in the preceding line, and lower than the coded character set value for the character in the following line, in the current coded character set are placed in the character collation order between the previous and the following character in ascending order according to their coded character set values.

128(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains a `collating-element` entry, and the first collating-element entry is an ellipsis symbol, then `localedef` interprets this as if the preceding line specified the null character.

129(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains a `collating-element` entry, and the last collating-element entry is an ellipsis symbol, then `localedef` interprets this as if the following line specified the highest coded character set value in the current coded character set.

130(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains a `collating-element` entry, and the collating-element entry is an ellipsis symbol, and the preceding element does not specify characters in the current coded character set, then `localedef` treats this as an error.

131(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains a `collating-element` entry, and the collating-element entry is an ellipsis symbol, and the following element does not specify characters in the current coded character set, then `localedef` treats this as an error.

132(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains a `collating-element` entry that contains the symbol UNDEFINED, then `localedef` interprets UNDEFINED as meaning all coded character set values not specified explicitly or via the ellipsis symbol.

133(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains a `collating-element` entry that contains the symbol UNDEFINED, then a call to `localedef` inserts the appropriate characters into the character collation order at the point indicated by the symbol, in ascending order according to their coded character set values.

134(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains a `collating-element` entry that contains no UNDEFINED symbol, and the coded character set contains characters not specified elsewhere in the collation order, then a call to `localedef` issues a warning message and places such characters at the end of the character collation order.

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

135(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains a `collating-element` entry that contains one or more optional operands, then a call to `localedef` treats these operands as defining the primary, secondary, or subsequent weights for the collating element.

136(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains a `collating-element` entry that contains one or more optional operands, and this element has been assigned the same primary weight as another element, then a call to `localedef` treats these collating element entries as belonging to the same equivalence class.

137(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains one or more `collating-element` entries that have IGNOREd elements and the same weight level, then a call to `localedef` treats collation as if the IGNOREd elements have been removed and each successive pair of elements were compared according to relative weights for the elements.

138(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains one or more `collating-element` entries that have IGNOREd elements, the position collation directive specified for the corresponding level with the `order_start` keyword, and the same weight level, then a call to `localedef` treats collation as if the IGNOREd elements have not been removed and each successive pair of elements were compared according to the relative weights for the elements.

139(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains one or more `collating-element` entries that have IGNOREd elements, the same weight level, and compare equally, then a call to `localedef` repeats the collation sequence for the next weight level in sequence until an inequality is reached or {COLL_WEIGHTS_MAX} is reached.

140(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains one or more `collating-element` entries that have weights that are not expressed as characters, <*collating-symbol*>s, <*collating-element*>s, an ellipsis, or the special symbol IGNORE, then a call to `localedef` treats this as an error.

141(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains one or more `collating-element` entries that have weights specified as a single character, a <*collating-symbol*>, or a <*collating-element*>, then a call to `localedef` treats these as representing the relative position in the character collating sequence of the character or symbol.

Documents Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

142(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains one or more `collating-element` entries that have weights specified as two or more concatenated characters or symbolic names, then a call to `localedef` treats these as specifying one-to-many mapping.

143(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains one or more `collating-element` entries that have characters specified via an ellipsis, then a call to `localedef` treats these as having unique weights equal to the relative order of characters.

144(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains one or more `collating-element` entries that have characters specified via an explicit or implicit UNDEFINED special symbol, then a call to `localedef` treats these as having the same primary weights.

145(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains one or more `collating-element` entries that have weights specified via an ellipsis, then a call to `localedef` treats each character in the sequence as having a unique weight equal to the relative order of their character in the character collation sequence, with secondary and subsequent weights having unique values.

146(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains one or more `collating-element` entries that have weights specified via an ellipsis, and the collating element is neither an ellipsis nor the special symbol UNDEFINED, then a call to `localedef` treats this as an error.

147(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains one or more `collating-element` entries that have weights specified via the keyword IGNORE, then a call to `localedef` compares these strings as having weights at the level where IGNORE is specified, and the collating element will be ignored.

148(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains one or more `collating-element` entries that have weights specified via the keyword IGNORE, then a call to `localedef` treats all characters that are IGNOREd in their primary weight, as if they were an equivalence class.

149(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains one or more `collating-element` entries with no operands, then a call to `localedef` treats these as if they were the collating element itself.

150(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains one or more `collating-element` entries with collating elements that are

Documents
Document Sharing Hub
Full document details: ISO IEC 14515-2-2003

not ellipses, and an ellipsis is used as an operand in the expression, then a call to `localedef` treats this as an error.

151(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains one or more `collating-element` entries with collating elements that are ellipses, and an ellipsis is used as an operand in the expression, then a call to `localedef` treats this as the value of each character as defined by the ellipsis.

2.5.2.2.5 order_end Keyword

152(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_COLLATE category source definition contains an `order_start` keyword, the list of collating-element entries are terminated with the `order_end` keyword.

2.5.2.3 LC_MONETARY

153(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_MONETARY category source definition contains the `copy` keyword with an operand that is the name of an existing locale, then a call to `localedef` uses that locale as the source for the definition of this category.

154(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_MONETARY category source definition contains the `copy` keyword with an operand that is the name of an existing locale, and also contains other keywords, then a call to `localedef` treats this as an error.

155(C) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_MONETARY category source definition is used, any operand that is not a string is treated as an error by `localedef`.

156(D) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_MONETARY category source definition is used and any operand contains a value set to the empty string (“”), then a call to `localedef` treats the value as unspecified.
See Reason 1 in Section 5 of POSIX.3 {11}.

157(D) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_MONETARY category source definition is used and any integer keyword is set to -1, then a call to `localedef` treats the value as unspecified.
See Reason 1 in Section 5 of POSIX.3 {11}.

158(D) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_MONETARY category source definition is used and any keyword is not specified, then a call to `localedef` treats the value as unspecified.
See Reason 1 in Section 5 of POSIX.3 {11}.

159(D) If {**POSIX2_LOCALEDEF**} is defined:
When the LC_MONETARY category source definition contains the

Documents Document Sharing Hub
Full document details: ISO IEC 14515-2-2003