

IEEE Std 1003.1™-2001/Cor 2-2004

The Open Group Technical Standard
Base Specifications, Issue 6

1003.1™-2001/Cor 2-2004

**Standard for Information Technology—
Portable Operating System Interface (POSIX®)**

Technical Corrigendum 2

Sponsor

**Portable Applications Standards Committee
of the
IEEE Computer Society**

and

The Open Group



THE *Open* GROUP

[This page intentionally left blank]

Abstract

Technical Corrigendum 2 addresses problems discovered since the approval of the 2003 edition of The Open Group Base Specifications, Issue 6, IEEE Std 1003.1, and ISO/IEC 9945. These are mainly due to resolving integration issues raised by the merger of the original Base documents.

Keywords

application program interface (API), argument, asynchronous, basic regular expression (BRE), batch job, batch system, built-in utility, byte, child, command language interpreter, CPU, extended regular expression (ERE), FIFO, file access control mechanism, input/output (I/O), job control, network, portable operating system interface (POSIX[®]), parent, shell, stream, string, synchronous, system, thread, X/Open System Interface (XSI)

All rights reserved. No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without prior written permission from both the IEEE and The Open Group.

Portions of this standard are derived with permission from copyrighted material owned by Hewlett-Packard Company, International Business Machines Corporation, Novell Inc., The Open Software Foundation, and Sun Microsystems, Inc.

Permissions

Authorization to photocopy portions of this standard for internal or personal use is granted provided that the appropriate fee is paid to the Copyright Clearance Center or the equivalent body outside of the U.S. Permission to make multiple copies for educational purposes in the U.S. requires agreement and a license fee to be paid to the Copyright Clearance Center.

Beyond these provisions, permission to reproduce all or any part of this standard must be with the consent of both copyright holders and may be subject to a license fee. Both copyright holders will need to be satisfied that the other has granted permission. Requests to the copyright holders should be sent by email to austin-group-permissions@opengroup.org.

Feedback

This standard has been prepared by the Austin Group. Feedback relating to the material contained in this standard may be submitted using the Austin Group web site at www.opengroup.org/austin/defectform.html.

IEEE

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 the 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. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, explanation, or interpretation of the IEEE. Current interpretations can be accessed at <http://standards.ieee.org/reading/ieee/interp/index.html>.

Errata, if any, for this and all other standards can be accessed at <http://standards.ieee.org/reading/ieee/updates/errata/index.html>. Users are encouraged to check this URL for errata periodically.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with the IEEE.¹ 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, U.S.A.

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.

A patent holder has filed a statement of assurance that it will grant licenses under these rights without compensation or under reasonable rates and non-discriminatory, reasonable terms and conditions to all applicants desiring to obtain such licenses. The IEEE makes no representation as to the reasonableness of rates and/or terms and conditions of the license agreements offered by patent holders. Further information may be obtained from the IEEE Standards Department.

Authorization to photocopy portions of any individual standard for internal or personal use is granted in the U.S. by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to the Copyright Clearance Center.² Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center. To arrange for payment of the licensing fee, please contact:

Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923, U.S.A., Tel.: +1 978 750 8400

Amendments, corrigenda, and interpretations for this standard, or information about the IEEE standards development process, may be found at <http://standards.ieee.org>.

The IEEE publications catalog and ordering information are available at <http://shop.ieee.org/store>.

1. For this standard, please send comments via the Austin Group as requested on page ii.

2. Please refer to the special provisions for this standard on page ii concerning permissions from both copyright holders and arrangements to cover photocopying and reproduction across the world, as well as by commercial organizations wishing to license the material for use in product documentation.

The Open Group

The Open Group is a vendor-neutral and technology-neutral consortium, whose vision of Boundaryless Information Flow will enable access to integrated information within and between enterprises based on open standards and global interoperability. The Open Group works with customers, suppliers, consortia, and other standards bodies. Its role is to capture, understand, and address current and emerging requirements, establish policies, and share best practices; to facilitate interoperability, develop consensus, and evolve and integrate specifications and Open Source technologies; to offer a comprehensive set of services to enhance the operational efficiency of consortia; and to operate the industry's premier certification service, including UNIX certification.

Further information on The Open Group is available at www.opengroup.org.

The Open Group has over 15 years' experience in developing and operating certification programs and has extensive experience developing and facilitating industry adoption of test suites used to validate conformance to an open standard or specification. The Open Group portfolio of test suites includes the *Westwood* family of tests for this standard and the associated certification program for Version 3 of the Single UNIX Specification, as well tests for CDE, CORBA, Motif, Linux, LDAP, POSIX.1, POSIX.2, POSIX Realtime, Sockets, UNIX, XPG4, XNFS, XTI, and X11. The Open Group test tools are essential for proper development and maintenance of standards-based products, ensuring conformance of products to industry-standard APIs, applications portability, and interoperability. In-depth testing identifies defects at the earliest possible point in the development cycle, saving costs in development and quality assurance.

More information is available at www.opengroup.org/testing.

The Open Group publishes a wide range of technical documentation, the main part of which is focused on development of Technical and Product Standards and Guides, but which also includes white papers, technical studies, branding and testing documentation, and business titles. Full details and a catalog are available at www.opengroup.org/pubs.

As with all *live* documents, Technical Standards and Specifications require revision to align with new developments and associated international standards. To distinguish between revised specifications which are fully backwards compatible and those which are not:

- A new *Version* indicates there is no change to the definitive information contained in the previous publication of that title, but additions/extensions are included. As such, it *replaces* the previous publication.
- A new *Issue* indicates there is substantive change to the definitive information contained in the previous publication of that title, and there may also be additions/extensions. As such, both previous and new documents are maintained as current publications.

Readers should note that Corrigenda may apply to any publication. Corrigenda information is published at www.opengroup.org/corrigenda.

The Open Group publications catalog and ordering information are available at www.opengroup.org/pubs.

Participants

IEEE Std 1003.1-2001/Cor 2-2004 was prepared by the Austin Group, sponsored by the Portable Applications Standards Committee of the IEEE Computer Society, The Open Group, and ISO/IEC JTC 1/SC22/WG15.

The Austin Group

At the time of approval, the membership of the Austin Group was as follows:

Andrew Josey, Chair

Donald W. Cragun, Organizational Representative, IEEE PASC

Nicholas Stoughton, Organizational Representative, ISO/IEC JTC 1/SC22/WG15

Mark Brown, Organizational Representative, The Open Group

Cathy Fox, Technical Editor

Austin Group Technical Reviewers

Jay Ashford	Clive D.W. Feather	Rajesh Moorkath
Julian Blake	Yaacov Fenster	Peter Petrov
Mitchell Bonnett	Mark Funkenhauser	Franklin Prindle
Andries Brouwer	Ernesto Garcia	Vikram Punj
Mark Brown	Andrew Gollan	Eusebio Rufian-Zilbermann
Paul Buerger	Michael Gonzalez	Joerg Schilling
Dave Butenhof	Jean-Denis Gorin	Stephen Schwarm
Keith Chow	Matthew Gream	Gil Shultz
Geoff Clare	Scott Gudgel	Keld Simonsen
Donald W. Cragun	Bruno Haible	Nicholas Stoughton
Lee Damico	Charles Hammons	Alexander Terekhov
Maulik Dave	Barry Hedquist	Donn Terry
Juan A. de la Puente	Jon Hitchcock	Mark-Rene Uchida
Guru Dutt Dhingra	Lowell G. Johnson	Thomas Unsicker
Loic Domaigne	Andrew Josey	Scott Valcourt
Ulrich Drepper	Piotr Karocki	Mats Wichmann
Sourav Dutta	David Leciston	Garrett A. Wollman
Larry Dwyer	Ryan Madron	Oren Yuen
Paul Eggert	Roger J. Martin	Mark Ziegast
Joanna Farley	George Miao	

Austin Group Working Group Members

Harold C. Adams
Jay Ashford
Theodore P. Baker
David J. Blackwood
Julian Blake
Mitchell Bonnett
Andries Brouwer
Mark Brown
Paul Buerger
Alan Burns
Dave Butenhof
Keith Chow
Geoff Clare
Donald W. Cragun
Dragan Cvetkovic
Lee Damico
Maulik Dave
Juan A. de la Puente
Loic Demaille
Steven J. Dovich
Ulrich Drepper
Guru Dutt Dhingra
Sourav Dutta
Larry Dwyer
Paul Eggert
Joanna Farley
Clive D.W. Feather
Yaacov Fenster

Mark Funkenhauser
Ernesto Garcia
Lois Goldthwaite
Andrew Gollan
Michael Gonzalez
Karen D. Gordon
Jean-Denis Gorin
Matthew Gream
Scott Gudgel
Joseph M. Gwinn
Steven A. Haaser
Charles Hammons
Ben Harris
Barry Hedquist
Karl Heubaum
Jon Hitchcock
Andreas Jaeger
Lowell G. Johnson
Andrew Josey
Piotr Karocki
Kenneth Lang
Pi-Cheng Law
David Leciston
Wojtek Lerch
Jonathan Lennox
Nick Maclaren
Ryan Madron
Roger J. Martin

George Miao
Rajesh Moorkath
Vilhelm Mueller
Joseph S. Myers
Peter Petrov
Franklin Prindle
Vikram Punj
Kenneth Raeburn
Tim Robbins
Curtis Royster Jr.
Eusebio Rufian-Zilbermann
Joerg Schilling
Stephen Schwarm
Glen Seeds
Gil Shultz
Keld Simonsen
Nicholas Stoughton
Alexander Terekhov
Donn Terry
Mark-Rene Uchida
Thomas Unsicker
Scott Valcourt
Mats Wichmann
Mike Wilson
Garrett A. Wollman
Oren Yuen
Mark Ziegast
Jason Zions

The Open Group

When The Open Group approved the Base Specifications, Issue 6, Technical Corrigendum 2 on 18 December 2003, the membership of The Open Group Base Working Group was as follows:

Andrew Josey, Chair

Mark Brown, Austin Group Liaison

Cathy Fox, Technical Editor

Base Working Group Members

Mark Brown	IBM Corporation
Dave Butenhof	Hewlett-Packard Company
Donald W. Cragun	Sun Microsystems, Inc.
Larry Dwyer	Hewlett-Packard Company
Ulrich Drepper	Red Hat, Inc.
Joanna Farley	Sun Microsystems, Inc.
Andrew Gollan	Sun Microsystems, Inc.
Andrew K. Roach	Sun Microsystems, Inc.
Curtis Royster Jr.	US DoD DISA
Nicholas Stoughton	USENIX Association
Kenjiro Tsuji	Sun Microsystems, Inc.

IEEE

When the IEEE Standards Board approved IEEE Std 1003.1-2001/Cor 2-2004 on 9 February 2004, the membership of the committees was as follows:

Portable Applications Standards Committee (PASC)

Lowell G. Johnson, Chair

Joseph M. Gwinn, Vice-Chair

Jay Ashford, Functional Chair

Andrew Josey, Functional Chair

Curtis Royster Jr., Functional Chair

Nicholas Stoughton, Secretary

Balloting Committee

The following members of the balloting committee voted on IEEE Std 1003.1-2001/Cor 2-2004. Balloters may have voted for approval, disapproval, or abstention:

Jay Ashford

Julian Blake

Mark Brown

Keith Chow

Donald W. Cragun

Juan A. de la Puente

Guru Dutt Dhingra

Ernesto Garcia

Michael Gonzalez

Jean-Denis Gorin

Matthew Gream

Scott Gudgel

Charles Hammons

Barry Hedquist

Andrew Josey

Piotr Karocki

David Leciston

Ryan Madron

Roger J. Martin

George Miao

Rajesh Moorkath

Peter Petrov

Vikram Punj

Eusebio Rufian-Zilbermann

Stephen Schwarm

Gil Shultz

Mark-Rene Uchida

Thomas Unsicker

Scott Valcourt

Oren Yuen

IEEE-SA Standards Board

When the IEEE-SA Standards Board approved IEEE Std 1003.1-2001/Cor 2-2004 on 9 February 2004, the membership was as follows:

Don Wright, Chair

Judith Gorman, Secretary

Chuck Adams

H. Stephen Berger

Mark D. Bowman

Joseph A. Bruder

Bob Davis

Roberto de Boisson

Julian Forster*

Arnold M. Greenspan

Mark S. Halpin

Raymond Hapeman

Richard J. Holleman

Richard H. Hulett

Lowell G. Johnson

Hermann Koch

Joseph L. Koepfinger*

Thomas J. McGean

Steve M. Mills

Daleep C. Mohla

Paul Nikolich

T. W. Olsen

Ronald C. Petersen

Gary S. Robinson

Frank Stone

Malcolm V. Thaden

Doug Topping

Joe D. Watson

Also included are the following non-voting IEEE-SA Standards Board liaisons:

Satish K. Aggarwal, NRC Representative

Richard DeBlasio, DOE Representative

Alan Cookson, NIST Representative

Savoula Amanatidis, IEEE Standards Managing Editor

* Member Emeritus

[This page intentionally left blank]

1. Introduction

1.1 Scope

This technical corrigendum addresses issues raised in defect reports and interpretation requests submitted up to 14th August 2003, and that meet all of the following criteria: a. They are in the scope of the approved standard. b. They contain no new APIs (functions/utilities); however, they may add enumeration symbols, non-function #defines, and reserve additional namespaces. c. They address contradictions between different parts of the standard, or add consistency between the standard and overriding standards, or address security-related problems.

2. Changes to Base Definitions

This section contains the set of changes to the text of the Base Definitions.

0001 Change Number: XBD/TC2/D6/1 [XBD ERN 20]

0002 On Page: xli Line: "ISO/IEC 8859" Section: Referenced Documents (2003 Ed.)

0003 On Page: xxxix Line: "ISO/IEC 8859" Section: Referenced Documents (2001 Ed.)

0004 Add after line starting "Part 10":

0005 "Part 11: Latin/Thai Alphabet"

0006 Add after line starting "Part 15":

0007 "Part 16: Latin Alphabet No. 10"

0008 **Change Number:** XBD/TC2/D6/2 [XSH ERN 4,XBD ERN 3]

0009 On Page: 25 Line: 932-935 Section: Conformance (2003 Ed.)

0010 On Page: 23 Line: 921-924 Section: Conformance (2001 Ed.)

0011 Change from:

0012 "If _POSIX_PRIORITIZED_IO is supported, then asynchronous I/O operations
0013 performed by aio_read(), aio_write(), and lio_listio() shall be submitted
0014 at a priority equal to the scheduling priority of the process minus
0015 aiocbp->aio_reqprio."

0016 To:

0017 "If _POSIX_PRIORITIZED_IO is supported, then asynchronous I/O operations
0018 performed by aio_read(), aio_write(), and lio_listio() shall be submitted
0019 at a priority equal to the scheduling priority equal to a base scheduling
0020 priority minus aiocbp->aio_reqprio. If Thread Priority Scheduling is
0021 not supported then the base scheduling priority is that of the calling
0022 process, otherwise the base scheduling priority is that of the calling
0023 thread."

0024 Rationale:

0025 The previous wording did not take threads into account.

Technical Corrigendum 2

0026 **Change Number:** XBD/TC2/D6/3 [XBD ERN 4]

0027 On Page: 48 Line: 1631-1635 Section: 3.93 Child Process (2003 Ed.)

0028 On Page: 46 Line: 1619-1623 Section: 3.93 Child Process (2001 Ed.)

0029 Change From:

0030 "3.93 Child Process

0031 A new process created (by fork(), posix_spawn(), or posix_spawnp())

0032 by a given process. A child process remains the child of the creating

0033 process as long as both processes continue to exist.

0034 Note: The fork(), posix_spawn(), and posix_spawnp() functions

0035 are defined in detail in the System Interfaces volume of IEEE Std

0036 1003.1-2001."

0037 To:

0038 "3.93 Child Process

0039 A new process created (by fork(), posix_spawn(), posix_spawnp(), or

0040 vfork()) by a given process. A child process remains the child of the

0041 creating process as long as both processes continue to exist.

0042 Note: The fork(), posix_spawn(), posix_spawnp() and vfork() functions

0043 are defined in detail in the System Interfaces volume of IEEE Std

0044 1003.1-2001."

0045 **Change Number:** XBD/TC2/D6/4 [XBD ERN 30]

0046 On Page: 61 Line: 2005-2006 Section: 3.187 Group Database (2003 Ed.)

0047 On Page: 59 Line: 1993-1994 Section: 3.187 Group Database (2001 Ed.)

0048 Change From:

0049 "A system database of implementation-defined format that contains at

0050 least the following information for each group ID:"

0051 To:

0052 "A system database that contains at least the following information for

0053 each group ID:"

0054 **Change Number:** XBD/TC2/D6/5 [XBD ERN 7]

0055 On Page: 76 Line: 2433,2440 Section: 3.295 Process Lifetime (2003 Ed.)

0056 On Page: 74 Line: 2421,2428 Section: 3.295 Process Lifetime(2001 Ed.)

0057 In the Section 3.295 Process Lifetime

0058 Change From:

0059 "After a process is created with a fork() function, it is considered

0060 active."

0061 To:

0062 "After a process is created by fork(), posix_spawn(), posix_spawnp()

0063 or vfork(), it is considered active."

0064 Change From:

0065 "Note: The fork(), wait(), waitid(), and waitpid() functions

0066 are defined in detail in the System Interfaces volume of IEEE Std

0067 1003.1-2001."

0068 To:

0069 "Note: The fork(), posix_spawn(), posix_spawnp(), vfork(), wait(),

0070 waitid(), and waitpid() functions are defined in detail in the System

0071 Interfaces volume of IEEE Std 1003.1-2001."

0072 **Change Number:** XBD/TC2/D6/6 [XBD ERN 5]

0073 On Page: 76 Line: 2448-2454 Section: 3.297 Process Termination (2003 Ed.)

0074 On Page: 74 Line: 2436-2442 Section: 3.297 Process Termination (2001 Ed.)

0075 Change From:

0076 "There are two kinds of process termination:

0077 1. Normal termination occurs by a return from main() or when requested

0078 with the exit() or _exit() functions.

0079 2. Abnormal termination occurs when requested by the abort() function

0080 or when some signals are received.

0081 Note: The _exit(), abort(), and exit() functions are defined in

0082 detail in the System Interfaces volume of IEEE Std 1003.1-2001."

0083 To:

0084 "There are two kinds of process termination:

0085 1. Normal termination occurs by a return from main(), when requested

0086 with the exit(), _exit(), or _Exit() functions; or when the last thread

0087 in the process terminates by returning from its start function, by

0088 calling the pthread_exit() function, or through cancellation.

0089 2. Abnormal termination occurs when requested by the abort() function

0090 or when some signals are received.

0091 Note: The _exit(), _Exit(), abort(), and exit() functions are defined in

0092 detail in the System Interfaces volume of IEEE Std 1003.1-2001."

0093 Rationale:

0094 The definition does not mention the 'passive exit' on termination of

0095 the last thread or the _Exit() function.

0096 **Change Number:** XBD/TC2/D6/7 [XBD ERN 31,32]

0097 On Page: 88 Line: 2778-2779 Section: 3.382 System Console (2003 Ed.)

0098 On Page: 86 Line: 2766-2767 Section: 3.382 System Console (2001 Ed.)

0099 Change From:

0100 "An implementation-defined device that receives messages sent by the

0101 syslog() function, and the fmtmsg() function when the MM_CONSOLE flag

0102 is set."

0103 To:

0104 "A device that receives messages sent by the syslog() function, and the

0105 fmtmsg() function when the MM_CONSOLE flag is set."

0106 **Change Number:** XBD/TC2/D6/8 [XBD ERN 33]

0107 On Page: 88 Line: 2802-2803 Section: 3.385 System Process (2003 Ed.)

0108 On Page: 86 Line: 2790-2791 Section: 3.385 System Process (2001 Ed.)

0109 Change From:

0110 "An implementation-defined object, other than a process executing an

0111 application, that has a process ID."

0112 To:

0113 "An object other than a process executing an application, that is

0114 provided by the system and has a process ID."

Technical Corrigendum 2

0115 **Change Number:** XBD/TC2/D6/9 [TC2d5 ERN 4]

0116 On Page: 88 Line: 2783-2796 Section: 3.383 System Databases (2003 Ed.)

0117 On Page: 86 Line: 2771-2784 Section: 3.383 System Databases (2001 Ed.)

0118 Change From:

0119 "An implementation provides two system databases.

0120 The "group database" contains the following information for each group:

0121 1. Group name

0122 2. Numerical group ID

0123 3. List of all users allowed in the group

0124 The "user database" contains the following information for each user:

0125 1. User name

0126 2. Numerical user ID

0127 3. Numerical group ID

0128 4. Initial working directory

0129 5. Initial user program

0130 If the initial user program field is null, the system default is used. If

0131 the initial working directory field is null, the interpretation of that

0132 field is implementation-defined. These databases may contain other fields

0133 that are unspecified by IEEE Std 1003.1-2001."

0134 To:

0135 "An implementation provides two system databases, the "group database"

0136 (see also Section 3.187) and the "user database" (see also Section

0137 3.424)."

0138 **Change Number:** XBD/TC2/D6/10 [TC2d5 ERN 5]

0139 On Page: 89 Line: 2805-2806 Section: 3.386 System Reboot (2003 Ed.)

0140 On Page: 87 Line: 2793-2794 Section: 3.386 System Reboot (2001 Ed.)

0141 Change From:

0142 "An implementation-defined sequence of events that may result in the loss

0143 of transitory data; that is, data that is not saved in permanent storage."

0144 To:

0145 "An unspecified sequence of events that may result in the loss

0146 of transitory data; that is, data that is not saved in permanent storage."

0147 **Change Number:** XBD/TC2/D6/11 [XBD ERN 36]

0148 On Page: 94 Line: 2939-2940 Section: 3.424 User Database (2003 Ed.)

0149 On Page: 92 Line: 2927-2928 Section: 3.424 User Database (2001 Ed.)

0150 Change From:

0151 "A system database of implementation-defined format that contains at

0152 least the following information for each user ID:"

0153 To:

0154 "A system database that contains at least the following information for

0155 each user ID:"

0156 **Change Number:** XBD/TC2/D6/12 [XBD ERN 28]

0157 On Page: 104 Line: 3222-3223 Section: 4.14 (2003 Ed.)

0158 On Page: 102 Line: 3207-3208 Section: 4.14 (2001 Ed.)

0159 Change From:

0160 "How any changes to the value of seconds since the Epoch
0161 are made to align to a desired relationship with the current
0162 actual time are made is implementation-defined."

0163 To:

0164 "How any changes to the value of seconds since the Epoch
0165 are made to align to a desired relationship with the current
0166 actual time is implementation-defined."

0167 Rationale:

0168 This is an editorial change.

0169 **Change Number:** XBD/TC2/D6/13 [XBD ERN 17]

0170 On Page: 119 Line: 3758 Section: 6.3 (2003 Ed.)

0171 On Page: 117 Line: 3742 Section: 6.3 (2001 Ed.)

0172 Add to the end of the paragraph:

0173 "This standard provides no means of defining a wide character codeset."

0174 **Change Number:** XBD/TC2/D6/14 [XBD ERN 17]

0175 On Page: 121 Line: 3860-3862,3867-3868,3869 Section: 6.4 (2003 Ed.)

0176 On Page: 119 Line: 3844-3846,3851-3852,3853 Section: 6.4 (2001 Ed.)

0177 Change From:

0178 "Bytes shall be treated as unsigned octets, and carry shall be propagated
0179 between the bytes as necessary to represent the range. For example,
0180 the line:"

0181 To:

0182 "Bytes shall be treated as unsigned octets, and carry shall be propagated
0183 between the bytes as necessary to represent the range. However, because
0184 this causes a null byte in the second or subsequent bytes of a character,
0185 such a declaration should not be specified. For example, the line:"

0186 Change From:

0187 <j0103> \d130\d0

0188 <j0104> \d130\d1

0189 To:

0190 <j0103> \d130\d00

0191 <j0104> \d130\d01

0192 Change From:

0193 "The comment is optional."

0194 To:

0195 "The expanded declaration of the symbol <j0103> in the above
0196 example is an invalid specification, because it contains a null
0197 byte in the second byte of a character.

0198 The comment is optional."

0199 Add to the end of the paragraph:

0200 "This standard provides no means of defining a wide character codeset."

Technical Corrigendum 2

0201 **Change Number:** XBD/TC2/D6/15 [XBD ERN 24]

0202 On Page: 121-122 Line: 3873-3885 Section: 6.4 (2003 Ed.)

0203 On Page: 119-120 Line: 3857-3859 Section: 6.4 (2001 Ed.)

0204 Change From:

0205 "WIDTH An unsigned positive integer value defining the column width
0206 (see Section 3.103 (on page 49)) for the printable characters in the
0207 coded character set specified in Table 6-1 (on page 115) and Table 6-2
0208 (on page 120)."

0209 To:

0210 "WIDTH A non-negative integer value defining the column width (see
0211 Section 3.103 (on page 49)) for the printable characters in the coded
0212 character set specified in Table 6-1 (on page 115) and Table 6-2 (on
0213 page 120)."

0214 Change From:

0215 "WIDTH_DEFAULT An unsigned positive integer value defining the default
0216 column width for any printable character not listed by one of the WIDTH
0217 keywords."

0218 To:

0219 "WIDTH_DEFAULT A non-negative integer value defining the default
0220 column width for any printable character not listed by one of the WIDTH
0221 keywords."

0222 Rationale: This change allows the value zero for the width value of WIDTH
0223 and WIDTH_DEFAULT. This is required to cover some existing locales

0224 **Change Number:** XBD/TC2/D6/16 [XBD ERN 25]

0225 On Page: 143 Line: 4836-4839 Section: 7.3.3 (2003 Ed.)

0226 On Page: 141 Line: 4820-4823 Section: 7.3.3 (2001 Ed.)

0227 On page 143 lines 4836-4839 (2003 Ed.), page 141 lines 4820-4823 (2001 Ed.)

0228 Change From:

0229 "p_sep_by_space An integer set to 0 if no space separates the
0230 currency_symbol from the value for a monetary quantity with a non-negative
0231 value, set to 1 if a space separates the symbol from the value, and
0232 set to 2 if a space separates the symbol and the sign string, if adjacent."

0233 To:

0234 "p_sep_by_space Set to a value indicating the separation of the
0235 currency_symbol, the sign string, and the |
0236 value for a non-negative formatted monetary quantity.

0237 The values of p_sep_by_space, n_sep_by_space, int_p_sep_by_space, and
0238 int_n_sep_by_space are interpreted according to the following:

0239 0 No space separates the currency symbol and value.

0240 1 If the currency symbol and sign string are adjacent, a
0241 space separates them from the value; otherwise, a space
0242 separates the currency symbol from the value.

0243 2 If the currency symbol and sign string are adjacent, a
0244 space separates them; otherwise, a space separates the
0245 sign string from the value."

0246 On page 143 lines 4843-4846 (2003 Ed.), page 141 lines 4827-2830 (2001 Ed.)

0247 Change From:

0248 "n_sep_by_space An integer set to 0 if no space separates the
0249 currency_symbol from the value for a monetary quantity with a negative
0250 value, set to 1 if a space separates the symbol from the value, and set
0251 to 2 if a space separates the symbol and the sign string, if adjacent."

0252 To:

0253 "n_sep_by_space Set to a value indicating the separation of
0254 the currency_symbol, the sign string, and the value for a negative

0255 formatted monetary quantity."

0256 On page 144 lines 4864-4871 (2003 Ed.), page 142 lines 4848-4855 (2001 Ed.)

0257 Change From:

0258 "int_p_sep_by_space An integer set to 0 if no space separates the

0259 int_curr_symbol from the value for a monetary quantity with a non-negative

0260 value, set to 1 if a space separates the symbol from the value, and set

0261 to 2 if a space separates the symbol and the sign string, if adjacent."

0262 int_n_sep_by_space An integer set to 0 if no space separates the

0263 int_curr_symbol from the value for a monetary quantity with a negative

0264 value, set to 1 if a space separates the symbol from the value, and set

0265 to 2 if a space separates the symbol and the sign string, if adjacent."

0266 To:

0267 "int_p_sep_by_space Set to a value indicating the separation

0268 of the int_curr_symbol, the sign string, and the value for a

0269 non-negative internationally formatted monetary quantity."

0270 int_n_sep_by_space Set to a value indicating the separation of

0271 the int_curr_symbol, the sign string, and the value for a negative

0272 internationally formatted monetary quantity."

0273 Rationale:

0274 The descriptions of p_sep_by_space, n_sep_by_space, int_p_sep_by_space,

0275 and int_n_sep_by_space need to be updated to match the description of

0276 these keywords in C99 and System Interfaces Volume, localeconv().

0277 **Change Number:** XBD/TC2/D6/17 [XBD ERN 37]

0278 On Page: 185 Line: 6540-6541 Section: 10.2 (2003 Ed.)

0279 On Page: 183 Line: 6508-6509 Section: 10.2 (2001 Ed.)

0280 Change From:

0281 "The implementation shall document which terminal types it supports and

0282 which of these features and utilities are not supported by each terminal."

0283 To:

0284 "The implementation shall document in the system documentation which

0285 terminal types it supports and which of these features and utilities

0286 are not supported by each terminal."

0287 **Change Number:** XBD/TC2/D6/18 [XSH ERN 146]

0288 On Page: 224 Line: 7897-7899 Section: fcntl.h (2003 Ed.)

0289 On Page: 222 Line: 7864-7865 Section: fcntl.h (2001 Ed.)

0290 In the DESCRIPTION section

0291 Change From:

0292 "[ADV] int posix_fadvise(int, off_t, size_t, int);

0293 int posix_fallocate(int, off_t, size_t);"

0294 To:

0295 "[ADV] int posix_fadvise(int, off_t, off_t, int);

0296 int posix_fallocate(int, off_t, off_t);"

0297 Rationale: The previous prototype was not large-file aware, and the standard

0298 developers felt it acceptable to make this change before

0299 implementations of the functions become widespread."

Technical Corrigendum 2

0300 **Change Number:** XBD/TC2/D6/19 [XBD ERN 12]

0301 On Page: 259-260 Line: 9132-9169 Section: limits.h (2003 Ed.)

0302 On Page: 257-258 Line: 9093-9131 Section: limits.h (2001 Ed.)

0303 On line 9132-9134 (2003 Ed.); line 9093-9095 (2001 Ed.)

0304 Change From:

0305 "{INT_MAX}

0306 Maximum value of an int.

0307 Minimum Acceptable Value: -2 147 483 647"

0308 To:

0309 "{INT_MAX}

0310 Maximum value of an int.

0311 [CX]Minimum Acceptable Value: -2 147 483 647[/CX]"

0312 On line 9156-9158 (2003 Ed.); line 9117-9119 (2001 Ed.)

0313 Change From:

0314 "{UINT_MAX}

0315 Maximum value of type unsigned.

0316 Minimum Acceptable Value: 4 294 967 295"

0317 To:

0318 "{UINT_MAX}

0319 Maximum value of type unsigned.

0320 [CX]Minimum Acceptable Value: 4 294 967 295[/CX]"

0321 On line 9165-9167 (2003 Ed.); line 9126-9128 (2001 Ed.)

0322 Change From:

0323 "[XSI] {WORD_BIT}

0324 Number of bits in a word or type int.

0325 Minimum Acceptable Value: 16[/XSI]"

0326 To:

0327 "[XSI] {WORD_BIT}

0328 Number of bits in a type int.

0329 Minimum Acceptable Value: 32[/XSI]"

0330 On Line 9168-9170 (2003 Ed.); line 9129-9131 (2001 Ed.)

0331 Change From:

0332 "{INT_MIN}

0333 Minimum value of type int.

0334 Maximum Acceptable Value: 2 147 483 647"

0335 To:

0336 "{INT_MIN}

0337 Minimum value of type int.

0338 [CX]Maximum Acceptable Value: 2 147 483 647[/CX]"

0339 Rationale: There was conflicting information about the size

0340 of an integer.

0341 **Change Number:** XBD/TC2/D6/20 [XBD ERN 26]

0342 On Page: 260 Line: 9191-9193 Section: limits.h (2003 Ed.)

0343 On Page: 258 Line: 9152-9154 Section: limits.h (2001 Ed.)

0344 In the DESCRIPTION section

0345 Under "Other Invariant Values"

0346 Remove the lines:

0347 "XSI {CHARCLASS_NAME_MAX}

0348 Maximum number of bytes in a character class name.

0349 Minimum Acceptable Value: 14"

0350 Rationale:

0351 CHARCLASS_NAME_MAX was defined under "Runtime Inceasable Values"

0352 and also "Other Invarient Values". This change corrects

0353 an integration issue with the base specifications.

0354 **Change Number:** XBD/TC2/D6/21 [XBD ERN 21]

0355 On Page: 266 Line: 9440-9447 Section: math.h (2003 Ed.)

0356 On Page: 264 Line: 9399-9406 Section: math.h (2001 Ed.)

0357 Change From:

0358 "The following optional macros indicate whether the fma() family of
0359 functions are fast compared with direct code:

0360 FP_FAST_FMA

0361 FP_FAST_FMAF

0362 FP_FAST_FMAL

0363 The FP_FAST_FMA macro shall be defined to indicate that the fma() function
0364 generally executes about as fast as, or faster than, a multiply and an
0365 add of double operands. The other macros have the equivalent meaning
0366 for the float and long double versions."

0367 To:

0368 "The following optional macros indicate whether the fma()
0369 family of functions are fast compared with direct code:

0370 FP_FAST_FMA

0371 FP_FAST_FMAF

0372 FP_FAST_FMAL

0373 If defined, the FP_FAST_FMA macro shall indicate that the fma() function
0374 generally executes about as fast as, or faster than, a multiply and an
0375 add of double operands. If undefined it is unspecified what the speed
0376 of execution is. The other macros have the equivalent meaning for the
0377 float and long double versions."

0378 **Change Number:** XBD/TC2/D6/22 [XBD ERN 15]

0379 On Page: 281 Line: 10000 Section: netinet/in.h (2003 Ed.)

0380 On Page: 279 Line: 9953 Section: netinet/in.h (2001 Ed.)

0381 Change From:

0382 "The <netinet/in.h> header shall define the sockaddr_in structure
0383 that includes at least the following members (all in network
0384 byte order):"

0385 To:

0386 "The <netinet/in.h> header shall define the sockaddr_in structure
0387 that includes at least the following members listed below:"

0388 Add after line 10004 (2003 Ed.); line 9957 (2001 Ed.):

0389 "The sin_port and sin_addr members shall be in network byte order."

0390 On line 10011 (2003 Ed.); line 9964 (2001 Ed.).

0391 Change From:

0392 "The <netinet/in.h> header shall define the sockaddr_in6 structure that
0393 includes at least the following members (all in network byte order):"

0394 To:

0395 "The <netinet/in.h> header shall define the sockaddr_in6 structure that
0396 includes at least the following members:"

0397 Add after line 10017 (2003 Ed.); line 9970 (2001 Ed.):

0398 "The sin6_port and sin6_addr members shall be in network byte order."

Technical Corrigendum 2

0399 **Change Number:** XBD/TC2/D6/23 [XBD ERN 8]

0400 On Page: 297 Line: 10588,10600 Section: <sched.h> (2003 Ed.)

0401 On Page: 295 Line: 10537,10549 Section: <sched.h> (2001 Ed.)

0402 In the DESCRIPTION section

0403 Change From:

0404 "int sched_priority Process execution scheduling priority."

0405 To:

0406 "int sched_priority Process [THR]or thread[/THR] execution scheduling
0407 priority."

0408 Change From:

0409 "Each process is controlled by an associated scheduling policy and
0410 priority."

0411 To:

0412 "Each process [THR]or thread[/THR] is controlled by an associated
0413 scheduling policy and priority."

0414 **Change Number:** XBD/TC2/D6/24 [XBD ERN 18]

0415 On Page: 306 Line: 10907-10918 Section: signal.h (2003 Ed.)

0416 On Page: 304 Line: 10852-10863 Section: signal.h (2001 Ed.)

0417 Change From:

0418 "CX The <signal.h> header shall define the siginfo_t type as a structure
0419 that includes at least the following members:

0420 [CX]	int si_signo	Signal number.
0421 [XSI]	int si_errno	If non-zero, an errno value associated with
0422		this signal, as defined in <errno.h>.
0423 [CX]	int si_code	Signal code.
0424 [XSI]	pid_t si_pid	Sending process ID.
0425	uid_t si_uid	Real user ID of sending process.
0426	void *si_addr	Address of faulting instruction.
0427	int si_status	Exit value or signal.
0428	long si_band	Band event for SIGPOLL.
0429 [RTS]	union sigval si_value	Signal value. "

0430 To:

0431 "[CX] The <signal.h> header shall define the siginfo_t type as a structure
0432 that includes at least the following members:

0433 [CX]	int si_signo	Signal number.
0434	int si_code	Signal code.
0435 [XSI]	int si_errno	If non-zero, an errno value associated with
0436		this signal, as defined in <errno.h>.
0437	pid_t si_pid	Sending process ID.
0438	uid_t si_uid	Real user ID of sending process.
0439	void *si_addr	Address of faulting instruction.
0440	int si_status	Exit value or signal.
0441	long si_band	Band event for SIGPOLL.
0442 [RTS]	union sigval si_value	Signal value. "

0443 Rationale: This is an editorial change to the shading and no

0444 normative change is intended.

0445 **Change Number:** XBD/TC2/D6/25 [XBD ERN 29]

0446 On Page: 358 Line: 12708 Section: <sys/stat.h> (2003 Ed.)
 0447 On Page: 356 Line: 12643 Section: <sys/stat.h> (2001 Ed.)

0448 Change From:
 0449 "Times shall be given in seconds since the Epoch."
 0450
 0451 To:
 0452 "The timespec structure may be defined as described in <time.h>."
 0453 Times shall be given in seconds since the Epoch."

0454 **Change Number:** XBD/TC2/D6/26 [XBD ERN 9]

0455 On Page: 369 Line: 13064 Section: sys/types.h (2003 Ed.)
 0456 On Page: 367 Line: 12988 Section: sys/types.h (2001 Ed.)

0457 Add "pthread_t" to the list of types in lines 13052-13068 (2003 Ed.)
 0458 and lines 12976-12992 (2001 Ed.) with it margin marked as THR and shaded.

0459 Rationale:The intent is to allow pthread_t to be a structure.

0460 **Change Number:** XBD/TC2/D6/27 [XBD ERN 38]

0461 On Page: 373 Line: 13189-13190 Section: sys/utsname.h (2003 Ed.)
 0462 On Page: 371 Line: 13113-13114 Section: sys/utsname.h (2001 Ed.)

0463 Change From:
 0464 "char nodename[] Name of this node within an implementation-defined
 0465 communications network. "

0466 To:
 0467 "char nodename[] Name of this node within the communications
 0468 network to which this node is attached,if any."

0469 **Change Number:** XBD/TC2/D6/28 [XSH ERN 53]

0470 On Page: 398 Line: 14041-14044 Section: ucontext.h (2003 Ed.)
 0471 On Page: 396 Line: 13960-13963 Section: ucontext.h (2001 Ed.)

0472 Margin mark OB (obsolescent) and shade the getcontext(), makecontext(),
 0473 setcontext(), swapcontext() functions.

Technical Corrigendum 2

0474 **Change Number:** XBD/TC2/D6/29 [XBD ERN 16]

0475 On Page: 401-404 Line: 14130-14133 Section: unistd.h (2003 Ed.)

0476 On Page: 399-403 Line: 14049-14052 Section: unistd.h (2001 Ed.)

0477 Change From:

0478 "_POSIX_CHOWN_RESTRICTED

0479 The use of chown() and fchown() is restricted to a process with
0480 appropriate privileges, and to changing the group ID of a file only to
0481 the effective group ID of the process or to one of its supplementary
0482 group IDs. This symbol shall always be set to a value other than -1."

0483 To:

0484 "_POSIX_CHOWN_RESTRICTED

0485 The use of chown() and fchown() is restricted to a process with
0486 appropriate privileges, and to changing the group ID of a file only to
0487 the effective group ID of the process or to one of its supplementary
0488 group IDs. This symbol shall either be undefined or defined with a value
0489 other than -1."

0490 On lines 14167-14169 (2003 Ed.); lines 14083-14085 (2001 Ed.)

0491 Change from:

0492 "_POSIX_NO_TRUNC

0493 Pathname components longer than {NAME_MAX} generate an
0494 error. This symbol shall always be set to a value other than -1."

0495 To:

0496 "_POSIX_NO_TRUNC

0497 Pathname components longer than {NAME_MAX} generate an error. This symbol
0498 shall either be undefined or defined with a value other than -1."

0499 For the Constants for Options and Option Groups on pages 401-406 lines

0500 14121-14350 (2003 Ed.); pages 399-404 lines 14040-14266 (2001 Ed.):

0501 Change occurrences of the text From:

0502 "If this symbol has a value other than -1 or 0, it shall have the
0503 value 200112L."

0504 To:

0505 "If this symbol is defined in <unistd.h>, it shall be defined to be -1,
0506 0, or 200112L. The value of this symbol reported by sysconf() shall
0507 either be -1 or 200112L."

0508 Rationale:

0509 This further clarifies the requirements on when constants in unistd.h
0510 can be defined or undefined.

0511 **Change Number:** XBD/TC2/D6/30 [XSH ERN 106]

0512 On Page: 405 Line: 14309-14320 Section: unistd.h (2003 Ed.)
 0513 On Page: 403 Line: 14225-14236 Section: unistd.h (2003 Ed.)

0514 In the DESCRIPTION section, line 14309 (2003 Ed.); line 14225 (2001 Ed.)
 0515 Change From:
 0516 "_V6_ILP32_OFF32"

0517 To:
 0518 "_POSIX_V6_ILP32_OFF32"

0519 In the DESCRIPTION section, line 14312 (2003 Ed.); line 14228 (2001 Ed.)
 0520 Change From:
 0521 "_V6_ILP32_OFFBIG"

0522 To:
 0523 "_POSIX_V6_ILP32_OFFBIG"

0524 In the DESCRIPTION section, line 14315 (2003 Ed.); line 14231 (2001 Ed.)
 0525 Change From:
 0526 "_V6_LP64_OFF64"

0527 To:
 0528 "_POSIX_V6_LP64_OFF64"

0529 In the DESCRIPTION section, line 14318 (2003 Ed.); line 14234 (2001 Ed.)
 0530 Change From:
 0531 "_V6_LPBIG_OFFBIG"

0532 To:
 0533 "_POSIX_V6_LPBIG_OFFBIG"

0534 Rationale: Consistency with the sysconf() and c99 man pages.

0535 **Change Number:** XBD/TC2/D6/31 [XCU ERN 19]

0536 On Page: 408 Line: 14437-14440 Section: unistd.h (2003 Ed.)
 0537 On Page: 406 Line: 14350-14353 Section: unistd.h (2001 Ed.)

0538 Change From:
 0539 "This value is a <newline>-separated list of names of programming
 0540 environments supported by the implementation in which the widths of
 0541 the blksize_t, cc_t, mode_t, nfds_t, pid_t, ptrdiff_t, size_t, speed_t,
 0542 ssize_t, suseconds_t, tcflag_t, useconds_t, wchar_t, and wint_t types
 0543 are no greater than the width of type long."

0544 To:
 0545 "This value is a <newline>-separated list of names of programming
 0546 environments supported by the implementation in which the widths of
 0547 the blksize_t, cc_t, mode_t, nfds_t, pid_t, ptrdiff_t, size_t, speed_t,
 0548 ssize_t, suseconds_t, tcflag_t, useconds_t, wchar_t, and wint_t types
 0549 are no greater than the width of type long. The format of each name
 0550 shall be suitable for use with the getconf -v option."

0551 **Change Number:** XBD/TC2/D6/32 [XSH ERN 40,61,62]

0552 On Page: 409-411 Line: 14525,14492,14610 Section: unistd.h (2003 Ed.)
 0553 On Page: 407-409 Line: 14438,14405,14520 Section: unistd.h (2001 Ed.)

0554 Delete "_SC_FILE_LOCKING"

0555 Delete "_SC_2_C_VERSION"

0556 Delete "_SC_XOPEN_XCU_VERSION"

Technical Corrigendum 2

0557 **Change Number:** XBD/TC2/D6/33 [XBD ERN 17]

0558 On Page: 410 Line: 14564,14586,14588 Section: unistd.h (2003 Ed.)

0559 On Page: 408 Line: 14475,14496,14498 Section: unistd.h (2001 Ed.)

0560 Add to the list of symbolic constants for sysconf()

0561 _SC_SS_REPL_MAX, _SC_TRACE_EVENT_NAME_MAX, _SC_TRACE_NAME_MAX,

0562 _SC_TRACE_SYS_MAX _SC_TRACE_USER_EVENT_MAX

0563 Rationale: These symbols were omitted.

0564 **Change Number:** XBD/TC2/D6/34 [XBD ERN 10]

0565 On Page: 413 Line: 14692 Section: unistd.h (2003 Ed.)

0566 On Page: 411 Line: 14602 Section: unistd.h (2001 Ed.)

0567 Remove the XSI margin marker and remove the shading

0568 from the following function:

0569 "int symlink(const char *, const char *);"

0570 Rationale: This was inconsistent with the XSH volume.

0571 **Change Number:** XBD/TC2/D6/35 [XBD ERN 11]

0572 On Page: 413 Line: 14471 Section: unistd.h (2003 Ed.)

0573 On Page: 404 Line: 14384 Section: unistd.h (2001 Ed.)

0574 Insert at the start of the symbolic constants list

0575 for pathconf() (before _PC_ALLOC_SIZE_MIN):

0576 "_PC_2_SYMLINKS"

0577 Rationale: This corresponds to the definition of POSIX2_SYMLINKS in

0578 the Shell and Utilities volume.

3. Changes to System Interfaces

This section contains the set of changes to the text of the System Interfaces.

0579 **Change Number:** XSH/TC2/D6/1 [XBD ERN 20]

0580 On Page: xxx Line: "ISO/IEC 8859" Section: Referenced Documents (2003 Ed.)

0581 On Page: xxvi Line: "ISO/IEC 8859" Section: Referenced Documents (2001 Ed.)

0582 Add after line starting "Part 10":

0583 "Part 11: Latin/Thai Alphabet"

0584 Add after line starting "Part 15":

0585 "Part 16: Latin Alphabet No. 10"

0586 **Change Number:** XSH/TC2/D6/2 [XSH ERN 167,TC2d5 ERN 9]

0587 On Page: 15,19 Line: 580-583,713-714 Section: 2.2.2 (2003 Ed.)

0588 On Page: 15,19 Line: 571-574,699-700 Section: 2.2.2 (2001 Ed.)

0589 On lines 580-583 (2003 Ed.); lines 571-574 (2001 Ed.)

0590 Change From:

0591 "Implementations may add symbols to the headers shown in the following

0592 table, provided the identifiers for those symbols begin with the

0593 corresponding reserved prefixes in the following table, and do not use

0594 the reserved prefixes `posix_`, `POSIX_`, or `_POSIX_`."

0595 To:

0596 "Implementations may add symbols to the headers shown in the following

0597 table, provided the identifiers for those symbols either:

0598 1. begin with one of the corresponding reserved prefixes in the table; or

0599 2. have one of the corresponding complete names in the table; or

0600 3. end in the string indicated as a reserved suffix in the table and

0601 do not use the reserved prefixes `posix_`, `POSIX_`, or `_POSIX_`, as long as

0602 the reserved suffix is in that part of the name considered significant

0603 by the implementation.

0604 Symbols that use the reserved prefix `_POSIX_` may be made visible by

0605 implementations in any header defined by IEEE Std 1003.1-2001."

0606 On lines 713-714 (2003 Ed.); lines 699-700 (2001 Ed.):

0607 Change From:

0608 "1. All identifiers that begin with an underscore and either an uppercase

0609 letter or another underscore are always reserved for any use by the

0610 implementation."

0611 To:

0612 "1. With the exception of identifiers beginning with the prefix `_POSIX_`,

0613 all identifiers that begin with an underscore and either an uppercase

0614 letter or another underscore are always reserved for any use by the

0615 implementation."

0616 Rationale:

0617 This change permits implementations to have symbols with the prefix

0618 `_POSIX_` visible in any header. A change in TC1 had disallowed this.

Technical Corrigendum 2

0619 **Change Number:** XSH/TC2/D6/3 [XSH ERN 1]

0620 On Page: 18 Line: 668 Section: 2.2.2 (2003 Ed.)

0621 On Page: 18 Line: 659 Section: 2.2.2 (2001 Ed.)

0622 Add the following line into the table after line 668 (2003 Ed.);

0623 after line 659 (2001 Ed.):

0624 "<math.h> FP_[A-Z]"

0625 Rationale:

0626 This text is added for consistency with the math.h page in the

0627 Base Definitions volume which states "Additional implementation-defined

0628 floating-point classifications, with macro definitions beginning with FP_

0629 and an uppercase letter, may also be specified by the implementation."

0630 **Change Number:** XSH/TC2/D6/4 [XSH ERN 99]

0631 On Page: 33 Line: 1337 Section: 2.4.3 (2003 Ed.)

0632 On Page: 33 Line: 1339 Section: 2.4.3 (2001 Ed.)

0633 Add "socketatmark()" to the table in 2.4.3 (the list of functions that shall

0634 be either reentrant or non-interruptible by signals and shall be

0635 async-signal-safe).

0636 Rationale:

0637 This was an omission when the socketatmark() function was added.

0638 **Change Number:** XSH/TC2/D6/5 [XSH ERN 109]

0639 On Page: 55 Line: 2267 Section: 2.9.5.2 (2003 Ed.)

0640 On Page: 55 Line: 2268 Section: 2.9.5.2 (2001 Ed.)

0641 Add "fdatasync()" into the table of functions after "fcntl()".

0642 Rationale:

0643 This was an omission. Since fdatsync() is often implemented

0644 in terms of fsync() it should also be in this list.

0645 **Change Number:** XSH/TC2/D6/6 [XSH ERN 76]

0646 On Page: 56 Line: 2277 Section: 2.9.5.2 Cancellation Points (2003 Ed.)

0647 On Page: 56 Line: 2278 Section: 2.9.5.2 Cancellation Points (2001 Ed.)

0648 Insert into the list of functions that "may have" a cancellation

0649 point (the second list of functions in this section)

0650 in the appropriate order:

0651 "access
0652 asctime
0653 asctime_r
0654 ctime
0655 ctime_r
0656 fmtmsg
0657 fpathconf
0658 fstat
0659 getaddrinfo
0660 gethostid
0661 getnameinfo
0662 getopt [with footnote if opterr is nonzero]
0663 link
0664 localtime
0665 localtime_r
0666 lstat
0667 mktime
0668 pathconf
0669 posix_openpt
0670 stat
0671 strerror_r
0672 strftime
0673 symlink
0674 sync
0675 tzset
0676 wcsftime
0677 wordexp"

0678 **Change Number:** XSH/TC2/D6/7 [XSH ERN 77]

0679 On Page: 57 Line: 2336-2353 Section: 2.9.5.3 (2003 Ed.)

0680 On Page: 57 Line: 2337-2354 Section: 2.9.5.3 (2001 Ed.)

0681 Change From:

0682 "Each thread maintains a list of cancellation cleanup handlers. The
0683 programmer uses the pthread_cleanup_push() and pthread_cleanup_pop()
0684 functions to place routines on and remove routines from this list.

0685 When a cancellation request is acted upon, the routines in the list
0686 are invoked one by one in LIFO sequence; that is, the last routine
0687 pushed onto the list (Last In) is the first to be invoked (First
0688 Out). The thread invokes the cancellation cleanup handler with
0689 cancellation disabled until the last cancellation cleanup handler
0690 returns. When the cancellation cleanup handler for a scope is
0691 invoked, the storage for that scope remains valid. If the last
0692 cancellation cleanup handler returns, thread execution is terminated
0693 and a status of PTHREAD_CANCELED is made available to any threads
0694 joining with the target. The symbolic constant PTHREAD_CANCELED
0695 expands to a constant expression of type (void *) whose value
0696 matches no pointer to an object in memory nor the value NULL.

0697 The cancellation cleanup handlers are also invoked when the thread
0698 calls pthread_exit().

0699 A side effect of acting upon a cancellation request while in a
0700 condition variable wait is that the mutex is re-acquired before
0701 calling the first cancellation cleanup handler. In addition, the
0702 thread is no longer considered to be waiting for the condition and
0703 the thread shall not have consumed any pending condition signals on
0704 the condition.

Technical Corrigendum 2

0705 A cancellation cleanup handler cannot exit via `longjmp()` or
0706 `siglongjmp()`."

0707 To:

0708 Each thread maintains a list of cancellation cleanup handlers. The
0709 programmer uses the `pthread_cleanup_push()` and `pthread_cleanup_pop()`
0710 functions to place routines on and remove routines from this list.

0711 When a cancellation request is acted upon, or when a thread calls
0712 `pthread_exit()`, the thread first disables cancellation by
0713 setting its cancelability state to `PTHREAD_CANCEL_DISABLE` and its
0714 cancelability type to `PTHREAD_CANCEL_DEFERRED`. The cancelability
0715 state shall remain set to `PTHREAD_CANCEL_DISABLE` until the thread
0716 has terminated. The behavior is undefined if a cancellation cleanup
0717 handler or thread-specific data destructor routine changes the
0718 cancelability state to `PTHREAD_CANCEL_ENABLE`.

0719 The routines in the thread's list of cancellation cleanup handlers
0720 are invoked one by one in LIFO sequence; that is, the last routine
0721 pushed onto the list (Last In) is the first to be invoked (First
0722 Out). When the cancellation cleanup handler for a scope is invoked,
0723 the storage for that scope remains valid. If the last cancellation
0724 cleanup handler returns, thread-specific data destructors (if any)
0725 associated with thread-specific data keys for which the thread has
0726 non-NULL values will be run, in unspecified order, as described for
0727 `pthread_key_create()`.

0728 After all cancellation cleanup handlers and thread-specific data
0729 destructors have returned, thread execution is terminated. If the
0730 thread has terminated because of a call to `pthread_exit()`, the
0731 `value_ptr` argument is made available to any threads joining with the
0732 target. If the thread has terminated by acting on a cancellation
0733 request, a status of `PTHREAD_CANCELED` is made available to any
0734 threads joining with the target. The symbolic constant
0735 `PTHREAD_CANCELED` expands to a constant expression of type `(void *)`
0736 whose value matches no pointer to an object in memory nor the value
0737 `NULL`.

0738 A side effect of acting upon a cancellation request while in a
0739 condition variable wait is that the mutex is re-acquired before
0740 calling the first cancellation cleanup handler. In addition, the
0741 thread is no longer considered to be waiting for the condition and
0742 the thread shall not have consumed any pending condition signals on
0743 the condition.

0744 A cancellation cleanup handler cannot exit via `longjmp()` or
0745 `siglongjmp()`."

0746 **Change Number:** XSH/TC2/D6/8 [XSH ERN 89]

0747 On Page: 58 Line: 2379 Section: Threads (2003 Ed.)

0748 On Page: 58 Line: 2380 Section: Threads (2001 Ed.)

0749 Insert new section:

0750 "2.9.8 Use of Application-Managed Thread Stacks

0751 An "application-managed thread stack" is a region of memory allocated
0752 by the application, for example, memory returned by the `malloc()`
0753 or `mmap()` functions, and designated as a stack through the act of
0754 passing an address related to that memory as the "stackaddr" argument to
0755 `pthread_attr_setstackaddr()` (obsolete) or by passing the address and size
0756 of the stack, respectively, as the `stackaddr` and `stacksize` arguments to
0757 `pthread_attr_setstack()`. Application-managed stacks allow the application
0758 to precisely control the placement and size of a stack.

0759 The application grants to the implementation permanent ownership of and
0760 control over the application-managed stack when the attributes object in
0761 which the `stack` or `stackaddr` attribute has been set is used, either by
0762 presenting that attributes object as the 'attr' argument in a call to
0763 `pthread_create()` that completes successfully, or by storing a pointer to

0764 the attributes object in the sigev_notify_attributes member of a 'struct
 0765 sigevent' and passing that 'struct sigevent' to a function accepting
 0766 such argument that completes successfully. The application may
 0767 thereafter utilize the memory within the stack only within the normal
 0768 context of stack usage within or properly synchronized with a thread
 0769 that has been scheduled by the implementation with stack pointer
 0770 value(s) that are within the range of that stack. In particular, the
 0771 region of memory cannot be freed, nor can it be later specified as the
 0772 stack for another thread.

0773 When specifying an attributes object with an application-managed stack
 0774 through the sigev_notify_attributes member of a 'struct sigevent', the
 0775 results are undefined if the requested signal is generated multiple
 0776 times (as for a repeating timer).

0777 Until an attributes object in which the stack or stackaddr attribute has
 0778 been set is used, the application retains ownership of and control over
 0779 the memory allocated to the stack. It may free or reuse the memory as
 0780 long as it either deletes the attributes object, or before using the
 0781 attributes object replaces the stack by making an additional call to the
 0782 same function, either pthread_attr_setstackaddr() or
 0783 pthread_attr_setstack(), that was used originally to designate the stack.
 0784 There is no mechanism to retract the reference to an application-managed
 0785 stack by an existing attributes object.

0786 Once an attributes object with an application-managed stack has been
 0787 used, that attributes object cannot be used again by a subsequent call
 0788 to pthread_create() or any function accepting a 'struct sigevent' with
 0789 sigev_notify_attributes containing a pointer to the attributes object,
 0790 without designating an unused application-managed stack by making an
 0791 additional call to the function originally used to define the stack,
 0792 pthread_attr_setstack() or pthread_attr_setstackaddr()."

0793 **Change Number:** XSH/TC2/D6/9 [XSH ERN 138]

0794 On Page: 92 Line: 3558-3559 Section: abort (2003 Ed.)
 0795 On Page: 92 Line: 3559-3560 Section: abort (2001 Ed.)

0796 In the APPLICATION USAGE section
 0797 Change From:
 0798 "Catching the signal is intended to provide the application writer with
 0799 a portable means to abort processing, free from possible interference
 0800 from any implementation-defined functions."

0801 To:
 0802 "Catching the signal is intended to provide the application writer with
 0803 a portable means to abort processing, free from possible interference
 0804 from any implementation-supplied functions."

0805 **Change Number:** XSH/TC2/D6/10 [XSH ERN 2]

0806 On Page: 105 Line: 3919 Section: aio_cancel (2003 Ed.)
 0807 On Page: 104 Line: 3898 Section: aio_cancel (2001 Ed.)

0808 In the RETURN VALUE section
 0809 Change From:
 0810 "The aio_cancel() function shall return the value AIO_CANCELED to the
 0811 calling process if the requested operation(s) were canceled."

0812 To:
 0813 "The aio_cancel() function shall return the value AIO_CANCELED
 0814 if the requested operation(s) were canceled."

0815 Rationale:
 0816 The use of the term "to the calling process" was unnecessary and
 0817 incorrect.

Technical Corrigendum 2

0818 **Change Number:** XSH/TC2/D6/11 [XSH ERN 3]

0819 On Page: 108 Line: 4022 Section: aio_fsync (2003 Ed.)

0820 On Page: 107 Line: 4001 Section: aio_fsync (2001 Ed.)

0821 In the RETURN VALUE section

0822 Change From:

0823 "The aio_fsync() function shall return the value 0 to the calling

0824 process if the I/O operation is successfully queued;"

0825 To:

0826 "The aio_fsync() function shall return the value 0 if the I/O operation

0827 is successfully queued;"

0828 Rationale:

0829 The use of the term "to the calling process" was unnecessary and incorrect.

0830 **Change Number:** XSH/TC2/D6/12 [XSH ERN 4,5]

0831 On Page: 110 Line: 4065,4088 Section: aio_read (2003 Ed.)

0832 On Page: 109 Line: 4044,4067 Section: aio_read (2001 Ed.)

0833 In the DESCRIPTION section

0834 Change From:

0835 "If prioritized I/O is supported for this file, then the asynchronous

0836 operation shall be submitted at a priority equal to the scheduling

0837 priority of the process minus aiocbp->aio_reqprio."

0838 To:

0839 "[PIO] If prioritized I/O is supported for this file, then the

0840 asynchronous operation shall be submitted at a priority

0841 equal to a base scheduling priority minus aiocbp->reqprio.

0842 If Thread Priority Scheduling is not supported then the

0843 base scheduling priority is that of the calling process,

0844 [PIO TPS] otherwise the base scheduling priority is that of the calling

0845 thread. [/end shading]"

0846 Rationale:

0847 The previous wording did not take threads into account.

0848 In the RETURN VALUE section

0849 Change From:

0850 "The aio_read() function shall return the value zero to the calling

0851 process if the I/O operation is successfully queued;"

0852 To:

0853 "The aio_read() function shall return the value zero

0854 if the I/O operation is successfully queued;"

0855 Rationale:

0856 The use of the term "to the calling process" was unnecessary and incorrect.

0857 **Change Number:** XSH/TC2/D6/13 [XSH ERN 83]

0858 On Page: 111 Line: 4101-4103 Section: aio_read (2003 Ed.)
 0859 On Page: 110 Line: 4080-4082 Section: aio_read (2001 Ed.)

0860 In the ERRORS section
 0861 Change From:
 0862 "[EINVAL] The file offset value implied by aiocbp->aio_offset would be
 0863 invalid, aiocbp->aio_reqprio is not a valid value, or aiocbp->aio_nbytes
 0864 is an invalid value."

0865 To:
 0866 "[EINVAL] The file offset value implied by aiocbp->aio_offset would be
 0867 invalid, [PIO]aiocbp->aio_reqprio is not a valid value[/PIO],
 0868 or aiocbp->aio_nbytes is an invalid value."

0869 Rationale:
 0870 Detection of an [EINVAL] error for a invalid value of aiocbp->aio_reqprio
 0871 should only be required if _POSIX_PRIORITIZED_IO is supported.

0872 **Change Number:** XSH/TC2/D6/14 [XSH ERN 6,7]

0873 On Page: 116 Line: 4252,4276 Section: aio_write (2003 Ed.)
 0874 On Page: 115 Line: 4231,4255 Section: aio_write (2001 Ed.)

0875 In the DESCRIPTION section
 0876 Change From:
 0877 "If prioritized I/O is supported for this file, then the asynchronous
 0878 operation shall be submitted at a priority equal to the scheduling
 0879 priority of the process minus aiocbp->aio_reqprio."

0880 To:
 0881 "[PIO] If prioritized I/O is supported for this file, then the
 0882 asynchronous operation shall be submitted at a priority
 0883 equal to a base scheduling priority minus aiocbp->reqprio.
 0884 If Thread Priority Scheduling is not supported then the
 0885 base scheduling priority is that of the calling process,
 0886 [PIO TPS] otherwise the base scheduling priority is that of the calling
 0887 thread. [/end shading]"

0888 Rationale:
 0889 The previous wording did not take threads into account.

0890 In the RETURN VALUE section
 0891 Change From:
 0892 "The aio_write () function shall return the value zero to the calling
 0893 process if the I/O operation is successfully queued;"

0894 To:
 0895 "The aio_write () function shall return the value zero if the I/O
 0896 operation is successfully queued;"

0897 Rationale:
 0898 The use of the term "to the calling process" was unnecessary and incorrect.

Technical Corrigendum 2

0899 **Change Number:** XSH/TC2/D6/15 [XSH ERN 83]

0900 On Page: 117 Line: 4290-4292 Section: aio_write (2003 Ed.)

0901 On Page: 116 Line: 4269-4271 Section: aio_write (2001 Ed.)

0902 In the ERRORS section

0903 Change From:

0904 "[EINVAL] The file offset value implied by aiocbp->aio_offset would be
0905 invalid, aiocbp->aio_reqprio is not a valid value, or aiocbp->aio_nbytes
0906 is an invalid value."

0907 To:

0908 "[EINVAL] The file offset value implied by aiocbp->aio_offset would be
0909 invalid, [PIO]aiocbp->aio_reqprio is not a valid value[/PIO],
0910 or aiocbp->aio_nbytes is an invalid value."

0911 Rationale:

0912 Detection of an [EINVAL] error for a invalid value of aiocbp->aio_reqprio
0913 should only be required if _POSIX_PRIORITIZED_IO is supported.

0914 **Change Number:** XSH/TC2/D6/16 [XSH ERN 139]

0915 On Page: 119 Line: 4375-4376 Section: alarm (2003 Ed.)

0916 On Page: 118 Line: 4353-4354 Section: alarm (2001 Ed.)

0917 In the RATIONALE section

0918 Change From:

0919 "In some implementations, including 4.3 BSD, very large values of the
0920 seconds argument are silently rounded down to an implementation-defined
0921 maximum value"

0922 To:

0923 "In some implementations, including 4.3 BSD, very large values of the
0924 seconds argument are silently rounded down to an implementation-specific
0925 maximum value"

0926 **Change Number:** XSH/TC2/D6/17 [XSH ERN 115]

0927 On Page: 122 Line: 4440 Section: asctime (2003 Ed.)

0928 On Page: 121 Line: 4419 Section: asctime (2001 Ed.)

0929 In the RETURN VALUE section

0930 Change From:

0931 "Upon successful completion, asctime() shall return a pointer to the string."

0932 To:

0933 "Upon successful completion, asctime() shall return a pointer to the string.
0934 [CX] If the function is unsuccessful, it shall return NULL. [/CX]"

0935 **Change Number:** XSH/TC2/D6/18 [XSH ERN 123]

0936 On Page: 132 Line: 4738 Section: atan2 (2003 Ed.)

0937 On Page: 131 Line: 4717 Section: atan2 (2001 Ed.)

0938 In the EXAMPLES section

0939 Change From:

0940 "None."

0941 To:

0942 "Converting Cartesian to Polar coordinates system

0943 The function below uses atan2() to convert a 2d vector expressed

0944 in cartesian coordinates (x,y) to the polar coordinates (rho,theta).

0945 There are other ways to compute the angle theta, using

0946 asin(), acos() or atan(). However, atan2() presents here two

0947 advantages:

0948 - the angle's quadrant is automatically determined.

0949 - the singular cases (0,y) are taken into account.

0950 Finally, this example uses hypot() rather than sqrt() since

0951 it is better for special cases, see hypot() for

0952 more information.

0953 #include <math.h>

0954 void

0955 cartesian_to_polar(const double x, const double y,

0956 double *rho, double *theta

0957)

0958 {

0959 *rho = hypot (x,y); /* better than sqrt(x*x+y*y) */

0960 *theta = atan2 (y,x);

0961 }"

0962 **Change Number:** XSH/TC2/D6/19 [XSH ERN 8]

0963 On Page: 137 Line: 4876 Section: atexit (2003 Ed.)

0964 On Page: 136 Line: 4855 Section: atexit (2001 Ed.)

0965 Add to end of the APPLICATION USAGE section:

0966 "Since the behavior is undefined if the exit() function is called

0967 more than once, portable applications calling atexit() must ensure

0968 that the exit() function is not called at normal process termination

0969 when all functions registered by the atexit() function are called.

0970 All functions registered by the atexit() function are called at

0971 normal process termination, which occurs by a call to the exit()

0972 function or a return from main() or on the last thread termination,

0973 when the behavior is as if the implementation called exit() with a

0974 zero argument at thread termination time.

0975 If, at normal process termination, a function registered by the

0976 atexit() function is called and a portable application needs to stop

0977 further exit() processing, it must call the _exit() function or

0978 the _Exit() function or one of the functions which cause abnormal

0979 process termination."

0980 Rationale:

0981 Additional clarifications for application usage are added.

Technical Corrigendum 2

0982 **Change Number:** XSH/TC2/D6/20 [XSH ERN 140]

0983 On Page: 142 Line: 5007-5009 Section: basename (2003 Ed.)

0984 On Page: 141 Line: 4986-4988 Section: basename (2001 Ed.)

0985 In the DESCRIPTION section

0986 Change From:

0987 "If the string consists entirely of the '/' character,
0988 basename() shall return a pointer to the string "/". If the string
0989 is exactly "/", it is implementation-defined whether '/'
0990 or "/" is returned."

0991 To:

0992 "If the string pointed to by path consists entirely of the '/'
0993 character, basename() shall return a pointer to the string "/". If the
0994 string pointed to by path is exactly "/", it is implementation-defined
0995 whether "/" or "/" is returned."

0996 **Change Number:** XSH/TC2/D6/21 [XSH ERN 138]

0997 On Page: 259 Line: 8533-8534 Section: dlopen (2003 Ed.)

0998 On Page: 257 Line: 8488-8489 Section: dlopen (2001 Ed.)

0999 In the DESCRIPTION section

1000 Change From:

1001 "If neither RTLD_GLOBAL nor RTLD_LOCAL are specified, then an
1002 implementation-defined default behavior shall be applied."

1003 To:

1004 "If neither RTLD_GLOBAL nor RTLD_LOCAL are specified, then
1005 the default behavior is unspecified."

1006 **Change Number:** XSH/TC2/D6/22 [XSH ERN 120]

1007 On Page: 289 Line: 9418 Section: erf (2003 Ed.)

1008 On Page: 287 Line: 9360 Section: erf (2001 Ed.)

1009 In the EXAMPLES section

1010 Change From:

1011 "None."

1012 To:

1013 "Computing the probability for a normal variate

1014 This example shows how to use erf() to compute the probability
1015 that a normal variate assumes a value in the range [x1,x2] with x1<=x2.
1016 This example uses the constant M_SQRT1_2 which is an XSI extension.

1017 #include <math.h>

1018 double

1019 Phi(const double x1, const double x2)

1020 {

1021 return (erf(x2*M_SQRT1_2) - erf(x1*M_SQRT1_2)) / 2;

1022 }"

1023 **Change Number:** XSH/TC2/D6/23 [XSH ERN 132]

1024 On Page: 294 Line: 9526 Section: errno (2003 Ed.)

1025 On Page: 292 Line: 9468 Section: errno (2001 Ed.)

1026 In the DESCRIPTION section

1027 Add at the end of the second paragraph:

1028 "The setting of errno after a successful call to a function is

1029 unspecified unless the description of that function specifies

1030 that errno shall not be modified".

1031 **Change Number:** XSH/TC2/D6/24 [XSH ERN 45,46,63]

1032 On Page: 297-299 Line: 9637 Section: exec (2003 Ed.)

1033 On Page: 295-297 Line: 9574 Section: exec (2001 Ed.)

1034 In the DESCRIPTION section

1035 On line 9637 (2003 Ed.); line 9574 (2001 Ed.)

1036 Change From:

1037 "The state of the floating-point environment in the new process image

1038 shall be set to the default."

1039 To:

1040 "The state of the floating-point environment in the new process image

1041 [THR]or in the initial thread of the new process image[/THR] shall be

1042 set to the default."

1043 On line 9650 (2003 Ed.); line 9587 (2001 Ed.)

1044 Change From:

1045 "After a successful call to any of the exec functions, any functions

1046 previously registered by atexit() are no longer registered."

1047 To:

1048 "After a successful call to any of the exec functions, any functions

1049 previously registered by atexit() [THR]or pthread_atfork()[/THR] are no

1050 longer registered."

1051 On lines 9675-9677 (2003 Ed.); lines 9612-9614 (2001 Ed.)

1052 Change From:

1053 "[PS]For the SCHED_FIFO and SCHED_RR scheduling policies, the policy and

1054 priority settings shall not be changed by a call to an exec

1055 function. For other scheduling policies, the policy and priority

1056 settings on exec are implementation-defined.[/PS]"

1057 To:

1058 "When the calling process image does not use the SCHED_FIFO,

1059 SCHED_RR, or SCHED_SPORADIC scheduling policies, the scheduling

1060 policy and parameters of the new process image and the initial

1061 thread in that new process image are implementation defined. [PS]When

1062 the calling process image uses the SCHED_FIFO, SCHED_RR, or

1063 SCHED_SPORADIC scheduling policies, the process policy and

1064 scheduling parameter settings shall not be changed by a call to an

1065 exec function.[/PS] [TPS]The initial thread in the new process image

1066 shall inherit the process scheduling policy and parameters. It shall

1067 have the default system contention scope, but shall inherit its

1068 allocation domain from the calling process image.[/TPS]"

1069 Insert the following after line 9701 (2003 Ed.); line 9638 (2001 Ed.)

1070 (before the list that begins "The new process shall inherit at least..."):

1071 "[THR]The thread id of the initial thread in the new process image is

1072 unspecified.

1073 The size and location of the stack on which the initial thread in the

1074 new process image runs is unspecified.

1075 The initial thread in the new process image shall have its

1076 cancellation type set to PTHREAD_CANCEL_DEFERRED and its cancellation

1077 state set to PTHREAD_CANCEL_ENABLED.

1078 The initial thread in the new process image shall have all

Technical Corrigendum 2

1079 thread-specific data values set to NULL and all thread-specific data
1080 keys shall be removed by the call to exec without running
1081 destructors.

1082 The initial thread in the new process image shall be joinable, as if
1083 created with the detachstate attribute set to PTHREAD_CREATE_JOINABLE.
1084 [/THR]"

1085 Add after line 9722 (2003 Ed.); line 9659 (2001 Ed.)
1086 "The initial thread of the new process shall inherit at least the
1087 following attributes from the calling thread:

1088 * Signal mask (see sigprocmask() and pthread_sigmask())
1089 * Pending signals (see sigpending())"

1090 On lines 9723-9725 (2003 Ed.); lines 9660-9662 (2001 Ed.)
1091 Change From:
1092 "All other process attributes defined in this volume of IEEE Std
1093 1003.1-2001 shall be the same in the new and old process images. The
1094 inheritance of process attributes not defined by this volume of
1095 IEEE Std 1003.1-2001 is implementation-defined."

1096 To:
1097 "All other process attributes defined in this volume of IEEE Std 1003.1-2001
1098 shall be inherited in the new process image from the old process image.
1099 All other thread attributes defined in this volume of IEEE Std 1003.1-2001
1100 shall be inherited in the initial thread in the new process image from
1101 the calling thread in the old process image. The inheritance of process
1102 or thread attributes not defined by this volume of IEEE Std 1003.1-2001
1103 is implementation-defined."

1104 On lines 9726-9728 (2003 Ed.); lines 9663-9665 (2001 Ed.)
1105 Change From:
1106 "A call to any exec function from a process with more than one thread
1107 shall result in all threads being terminated and the new executable
1108 image being loaded and executed. No destructor functions shall be
1109 called."

1110 To:
1111 "[THR]A call to any exec function from a process with more than one
1112 thread shall result in all threads being terminated and the new
1113 executable image being loaded and executed. No destructor functions
1114 or cleanup handlers shall be called.[/THR]"

1115 **Change Number:** XSH/TC2/D6/25 [XSH ERN 9]

1116 On Page: 303 Line: 9890 Section: exec (2003 Ed.)
1117 On Page: 301 Line: 9825 Section: exec (2001 Ed.)

1118 In the RATIONALE section
1119 Change From:
1120 "SIG_IGN, and that the process signal mask be unchanged across an exec."

1121 To:
1122 "SIG_IGN, and that the new process image inherits the signal mask of
1123 the thread that called exec in the old process image"

1124 **Change Number:** XSH/TC2/D6/26 [XSH ERN 119]

1125 On Page: 313 Line: 10264 Section: exp (2003 Ed.)
 1126 On Page: 311 Line: 10192 Section: exp (2001 Ed.)

1127 In the EXAMPLES section
 1128 Change From:
 1129 "None."

1130 To:
 1131 "Computing the density of the standard normal distribution

1132 This function shows an implementation for the density of the standard
 1133 normal distribution using exp(). This example uses the constant M_PI
 1134 which is an XSI extension.

1135 #include <math.h>

1136 double
 1137 normal_density (double x
 1138)
 1139 {
 1140 return exp(-x*x/2) / sqrt (2*M_PI);
 1141 }"

1142 **Change Number:** XSH/TC2/D6/27 [XSH ERN 121]

1143 On Page: 318 Line: 10437 Section: fabs (2003 Ed.)
 1144 On Page: 316 Line: 10365 Section: fabs (2001 Ed.)

1145 In the EXAMPLES section
 1146 Change From:
 1147 "None."

1148 To:
 1149 " Computing the 1-norm of a floating point vector

1150 This example shows the use of fabs() to compute the 1-norm of a
 1151 vector defined as follows:

1152 norm1(v) = |v[0]| + |v[1]| + ... + |v[n-1]|

1153 where |x| denotes the absolute value of x,
 1154 n denotes the vector's dimension
 1155 v[i] denotes the i-th component of v (0<=i<n)

1156 #include <math.h>

1157 double
 1158 norm1(const double v[], const int n)
 1159 {
 1160 int i;
 1161 double nl_v; /* 1-norm of v */

 1162 nl_v = 0;
 1163 for (i=0; i<n; i++) {
 1164 nl_v += fabs (v[i]);
 1165 }

 1166 return nl_v;
 1167 }"

Technical Corrigendum 2

1168 **Change Number:** XSH/TC2/D6/28 [XSH ERN 11]

1169 On Page: 327 Line: 10726 Section: fclose (2003 Ed.)

1170 On Page: 325 Line: 10655 Section: fclose (2001 Ed.)

1171 In the ERRORS section

1172 Change From:

1173 "[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor
1174 underlying stream and the process would be delayed in the write
1175 operation. [/CX]"

1176 To:

1177 "[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor
1178 underlying stream and the thread would be delayed in the write
1179 operation. [/CX]"

1180 Rationale: It is the thread that is delayed not the process.

1181 **Change Number:** XSH/TC2/D6/29 [XSH ERN 134]

1182 On Page: 332 Line: 10935 Section: fcntl (2003 Ed.)

1183 On Page: 330 Line: 10863 Section: fcntl (2001 Ed.)

1184 In the EXAMPLES section

1185 Change From:

1186 "None."

1187 To:

1188 "Locking and unlocking a file

1189 The following example demonstrates how to place a lock on bytes 100 to
1190 109 of a file and then later remove it. F_SETLK is used to perform a
1191 non-blocking lock request so that the process does not have to wait if
1192 an incompatible lock is held by another process; instead the process
1193 can take some other action.

1194 #include <stdlib.h>

1195 #include <unistd.h>

1196 #include <fcntl.h>

1197 #include <errno.h>

1198 int

1199 main(int argc, char *argv[])

1200 {

1201 int fd;

1202 struct flock fl;

1203 fd = open("testfile", O_RDWR);

1204 if (fd == -1)

1205 /* Handle error */;

1206 /* Make a non-blocking request to place a write lock
1207 on bytes 100-109 of testfile */

1208 fl.l_type = F_WRLCK;

1209 fl.l_whence = SEEK_SET;

1210 fl.l_start = 100;

1211 fl.l_len = 10;

1212 if (fcntl(fd, F_SETLK, &fl) == -1) {
1213 if (errno == EACCES || errno == EAGAIN) {
1214 printf("Already locked by another process\n");

1215 /* We can't get the lock at the moment */

1216 } else {

1217 /* Handle unexpected error */;

1218 }

1219 } else { /* Lock was granted... */


```

1220      /* Perform I/O on bytes 100 to 109 of file */
1221      /* Unlock the locked bytes */
1222      fl.l_type = F_UNLCK;
1223      fl.l_whence = SEEK_SET;
1224      fl.l_start = 100;
1225      fl.l_len = 10;
1226      if (fcntl(fd, F_SETLK, &fl) == -1)
1227          /* Handle error */;
1228      }
1229      exit(EXIT_SUCCESS);
1230  } /* main */

1231 Setting the close-on-exec flag

1232 The following example demonstrates how to set the close on exec flag
1233 for the file descriptor fd.

1234 #include <unistd.h>
1235 #include <fcntl.h>
1236 ...
1237     int flags;

1238     flags = fcntl(fd, F_GETFD);
1239     if (flags == -1)
1240         /* Handle error */;
1241     flags |= FD_CLOEXEC;
1242     if (fcntl(fd, F_SETFD, flags) == -1)
1243         /* Handle error */;

1244 Change Number: XSH/TC2/D6/30 [XSH ERN 141]

1245 On Page: 343   Line: 11245-11247   Section: fdopen (2003 Ed.)
1246 On Page: 341   Line: 11173-11175   Section: fdopen (2001 Ed.)

1247 In the RATIONALE section
1248 Change From:
1249 "The file descriptor may have been obtained from open(), creat(), pipe(),
1250 dup(), or fcntl(); inherited through fork() or exec; or perhaps obtained
1251 by implementation-defined means, such as the 4.3 BSD socket() call."

1252 To:
1253 The file descriptor may have been obtained from open(), creat(), pipe(),
1254 dup(), fcntl() or socket(); inherited through fork(), posix_spawn()
1255 or exec; or perhaps obtained by other means.

1256 Change Number: XSH/TC2/D6/31 [XSH ERN 12]

1257 On Page: 360   Line: 11686   Section: fflush (2003 Ed.)
1258 On Page: 358   Line: 11614   Section: fflush (2001 Ed.)

1259 In the ERRORS section
1260 Change From:
1261 "[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor
1262 underlying stream and the process would be delayed in the write
1263 operation. [/CX]"

1264 To:
1265 "[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor
1266 underlying stream and the thread would be delayed in the write
1267 operation. [/CX]"

```

Technical Corrigendum 2

1268 **Change Number:** XSH/TC2/D6/32 [XSH ERN 13]

1269 On Page: 364 Line: 11802 Section: fgetc (2003 Ed.)

1270 On Page: 362 Line: 11730 Section: fgetc (2001 Ed.)

1271 In the ERRORS section

1272 Change From:

1273 "[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor
1274 underlying stream and the process would be delayed in the fgetc()
1275 operation. [/CX]"

1276 To:

1277 "[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor
1278 underlying stream and the thread would be delayed in the fgetc()
1279 operation. [/CX]"

1280 **Change Number:** XSH/TC2/D6/33 [XSH ERN 14]

1281 On Page: 370 Line: 11969 Section: fgetwc (2003 Ed.)

1282 On Page: 368 Line: 11897 Section: fgetwc (2001 Ed.)

1283 In the ERRORS section

1284 Change From:

1285 "[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor
1286 underlying stream and the process would be delayed in the fgetwc()
1287 operation. [/CX]"

1288 To:

1289 "[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor
1290 underlying stream and the thread would be delayed in the fgetwc()
1291 operation. [/CX]"

1292 **Change Number:** XSH/TC2/D6/34 [XBD ERN 11]

1293 On Page: 399 Line: 12942 Section: fpathconf (2003 Ed.)

1294 On Page: 397 Line: 12860 Section: fpathconf (2001 Ed.)

1295 In the DESCRIPTION section

1296 Insert after the "{PIPE_BUF}" entry:

1297 "{POSIX2_SYMLINKS} _PC_2_SYMLINKS 4"

1298 **Change Number:** XSH/TC2/D6/35 [XSH ERN 15]

1299 On Page: 399-402 Line: 12943-12947,12975,13054 Section: fpathconf (2003 Ed.)

1300 On Page: 397-400 Line: 12861-12865,12893,12972 Section: fpathconf (2001 Ed.)

1301 In the DESCRIPTION section

1302 Insert the number "10" into the third column for the rows beginning

1303 {POSIX_ALLOC_SIZE_MIN}, {POSIX_REC_INCR_XFER_SIZE},

1304 {POSIX_REC_MAX_XFER_SIZE}, {POSIX_REC_MIN_XFER_SIZE}, and

1305 {POSIX_REC_XFER_ALIGN}.

1306 Add a new bullet item after Item 9 after line 12975 (2003 Ed.);

1307 line 12893 (2001 Ed.)

1308 "10. If path or fildes does not refer to a regular file, it is

1309 unspecified whether an implementation supports an

1310 association of the variable name with the specified file. If

1311 an implementation supports such an association for other than

1312 a regular file, the value returned is unspecified."

1313 Add to the RATIONALE section as a new paragraph, after line 13054 (2003

1314 Ed.); line 12972 (2001 Ed.):

1315 "It was the intention of 1003.1d that the variables

1316 {POSIX_ALLOC_SIZE_MIN}, {POSIX_REC_INCR_XFER_SIZE},

1317 {POSIX_REC_MAX_XFER_SIZE}, {POSIX_REC_MIN_XFER_SIZE}, and

1318 {POSIX_REC_XFER_ALIGN} only applied to regular files, but note 10 also

1319 permits implementation of the advisory semantics on other file types

1320 unique to an implementation (e.g. a character special device.) "

1321 **Change Number:** XSH/TC2/D6/36 [XBD ERN 27]

1322 On Page: 400 Line: 12991,13020 Section: fpathconf (2003 Ed.)

1323 On Page: 398 Line: 12909,12938 Section: fpathconf (2001 Ed.)

1324 Add to the end of the RETURN VALUE section as a new paragraph:

1325 "If the variable corresponding to name is dependent on an unsupported
1326 option the results are unspecified."

1327 Add to the end of the APPLICATION USAGE section:

1328 "Application writers should check whether an option, such as

1329 _POSIX_ADVISORY_INFO is supported, prior to obtaining and using values
1330 for related variables such as POSIX_ALLOC_SIZE_MIN."

1331 **Change Number:** XSH/TC2/D6/37 [XSH ERN 16]

1332 On Page: 416 Line: 13661 Section: fputc (2003 Ed.)

1333 On Page: 414 Line: 13576 Section: fputc (2001 Ed.)

1334 In the ERRORS section

1335 Change From:

1336 "[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor
1337 underlying stream and the process would be delayed in the write
1338 operation. [/CX]"

1339 To:

1340 "[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor
1341 underlying stream and the thread would be delayed in the write
1342 operation. [/CX]"

1343 **Change Number:** XSH/TC2/D6/38 [XSH ERN 17]

1344 On Page: 420 Line: 13778 Section: fputwc (2003 Ed.)

1345 On Page: 418 Line: 13693 Section: fputwc (2001 Ed.)

1346 In the ERRORS section

1347 Change From:

1348 "[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor
1349 underlying stream and the process would be delayed in the write
1350 operation. [/CX]"

1351 To:

1352 "[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor
1353 underlying stream and the thread would be delayed in the write
1354 operation. [/CX]"

1355 **Change Number:** XSH/TC2/D6/39 [XSH ERN 126]

1356 On Page: 428 Line: 14076 Section: freeaddrinfo (2003 Ed.)

1357 On Page: 426 Line: 13979 Section: freeaddrinfo (2001 Ed.)

1358 In the ERRORS section

1359 Change From:

1360 "The getaddrinfo() function shall fail and return the corresponding value if:"

1361 To:

1362 "The getaddrinfo() function shall fail and return the corresponding
1363 error value if:"

Technical Corrigendum 2

1364 **Change Number:** XSH/TC2/D6/40 [XSH ERN 142]

1365 On Page: 430 Line: 14141-14144 Section: freopen (2003 Ed.)

1366 On Page: 428 Line: 14031-14034 Section: freopen (2001 Ed.)

1367 In the DESCRIPTION section

1368 Change From:

1369 "If filename is a null pointer, the freopen() function attempt
1370 to change the mode of the stream to that by mode, as if the
1371 name of the file currently with the stream had been used. It is
1372 implementation-defined which changes of mode are permitted (if any)
1373 and under what circumstances".

1374 To:

1375 "If filename is a null pointer, the freopen() function shall attempt to
1376 change the mode of the stream to that specified by mode, as if the name
1377 of the file currently associated with the stream had been used. In this
1378 case, the file descriptor associated with the stream need not be closed
1379 if the call to freopen() succeeds. It is implementation-defined which
1380 changes of mode are permitted (if any) and under what circumstances".

1381 **Change Number:** XSH/TC2/D6/41 [XSH ERN 142]

1382 On Page: 430,431 Line: 14158,14179 Section: freopen (2003 Ed.)

1383 On Page: 428,429 Line: 14048,14069 Section: freopen (2001 Ed.)

1384 In the ERRORS section

1385 Add after line 14158 (2003 Ed.); line 14048 (2001 Ed.):

1386 "[CX][EBADF] The file descriptor underlying the stream is not a valid file
1387 descriptor when filename is a null pointer. [/CX]"

1388 Add after line 14179 (2003 Ed.); line 14069 (2001 Ed.):

1389 "[CX][EBADF] The mode with which the file descriptor underlying the stream
1390 was opened does not support the requested mode when filename is
1391 a null pointer. [/CX]"

1392 **Change Number:** XSH/TC2/D6/42 [XSH ERN 18]

1393 On Page: 443 Line: 14598 Section: fseek (2003 Ed.)

1394 On Page: 441 Line: 14488 Section: fseek (2001 Ed.)

1395 In the ERRORS section

1396 Change From:

1397 "[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor and
1398 the process would be delayed in the write operation. [/CX]"

1399 To:

1400 "[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor and
1401 the thread would be delayed in the write operation. [/CX]"

1402 **Change Number:** XSH/TC2/D6/43 [XSH ERN 19]

1403 On Page: 445 Line: 14679 Section: fsetpos (2003 Ed.)

1404 On Page: 443 Line: 14569 Section: fsetpos (2001 Ed.)

1405 In the ERRORS section

1406 Change From:

1407 "[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor and
1408 the process would be delayed in the write operation. [/CX]"

1409 To:

1410 "[CX][EAGAIN] The O_NONBLOCK flag is set for the file descriptor and
1411 the thread would be delayed in the write operation. [/CX]"

1412 **Change Number:** XSH/TC2/D6/44 [XSH ERN 143]

1413 On Page: 452 Line: 14894-14896 Section: fsync (2003 Ed.)

1414 On Page: 450 Line: 14784-14786 Section: fsync (2001 Ed.)

1415 In the DESCRIPTION section

1416 Change From:

1417 "The fsync() function shall request that all data for the open file
1418 descriptor named by fildes is to be transferred to the storage device
1419 associated with the file described by fildes in an implementation-defined
1420 manner."

1421 To:

1422 "The fsync() function shall request that all data for the open file
1423 descriptor named by fildes is to be transferred to the storage device
1424 associated with the file described by fildes. The nature of the transfer
1425 is implementation-defined."

1426 Rationale: This is an editorial wording change with no change
1427 intended in meaning.

1428 **Change Number:** XSH/TC2/D6/45 [XSH ERN 53]

1429 On Page: 491 Line: 16190,16229 Section: getcontext (2003 Ed.)

1430 On Page: 489 Line: 16077,16116 Section: getcontext (2001 Ed.)

1431 In the SYNOPSIS section

1432 Change the margin marker from "XSI" to "OB XSI", to denote that
1433 the getcontext() and setcontext() functions are obsolescent.

1434 In the APPLICATION USAGE section for getcontext add at the end:

1435 "The obsolescent functions getcontext(), makecontext() and
1436 swapcontext() can be replaced using POSIX threads functions."

1437 **Change Number:** XSH/TC2/D6/46 [XSH ERN 20]

1438 On Page: 554,556 Line: 18219,18266 Section: getrlimit (2003 Ed.)

1439 On Page: 551,553 Line: 18094,18139 Section: getrlimit (2001 Ed.)

1440 In the DESCRIPTION section

1441 Change From:

1442 "RLIMIT_STACK This is the maximum size of a process stack, in bytes."

1443 To:

1444 "RLIMIT_STACK This is the maximum size of the initial thread's stack,
1445 in bytes."

1446 In the RATIONALE section

1447 Change From:

1448 "None."

1449 To:

1450 "It should be noted that RLIMIT_STACK applies "at least" to the stack of
1451 the initial thread in the process, and not to the sum of all the stacks
1452 in the process, as that would be very limiting unless the value is so
1453 big as to provide no value at all with a single thread."

1454 **Change Number:** XSH/TC2/D6/47 [XSH ERN 21,22,23]

1455 On Page: 563 Line: 18492-18497 Section: getsockopt (2003 Ed.)

1456 On Page: 560 Line: 18359-18364 Section: getsockopt (2001 Ed.)

1457 Change From:

1458 "If SO_LINGER is set, the system blocks the process during close() until
1459 it can transmit the data or until the end of the interval indicated by the
1460 l_linger member, whichever comes first. If SO_LINGER is not specified,
1461 and close() is issued, the system handles the call in a way that allows
1462 the process to continue as quickly as possible."

Technical Corrigendum 2

1463 To:
1464 "If SO_LINGER is set, the system shall block the calling thread during
1465 close() until it can transmit the data or until the end of the interval
1466 indicated by the l_linger member, whichever comes first. If SO_LINGER
1467 is not specified, and close() is issued, the system handles the call in
1468 a way that allows the calling thread to continue as quickly as possible."

1469 **Change Number:** XSH/TC2/D6/48 [XSH ERN 48]

1470 On Page: 580 Line: 19104-19108 Section: gmtime (2003 Ed.)
1471 On Page: 577 Line: 18970-18974 Section: gmtime (2001 Ed.)

1472 In the RETURN VALUE section
1473 Change From:
1474 "Upon successful completion, gmtime_r() shall return the address of
1475 the structure pointed to by the argument result. If an error is
1476 detected, gmtime_r() shall return a null pointer."

1477 To:
1478 "Upon successful completion, gmtime_r() shall return the
1479 address of the structure pointed to by the argument result.
1480 If an error is detected, gmtime_r() shall return a null
1481 pointer and set errno to indicate the error".

1482 In the ERRORS section of the 2003 Ed.
1483 Change From:
1484 "The gmtime() function shall fail if:

1485 [Eoverflow] The result cannot be represented."

1486 To:
1487 "The gmtime() and gmtime_r() function shall fail if:

1488 [Eoverflow] The result cannot be represented."

1489 with "and gmtime_r()" shaded and margin marked TSF.

1490 In the ERRORS section of the 2001 Ed.
1491 Change from:
1492 "No errors are defined"

1493 To:
1494 "The gmtime() and gmtime_r() function shall fail if:

1495 [Eoverflow] The result cannot be represented."

1496 with "and gmtime_r()" shaded and margin marked TSF.

1497 **Change Number:** XSH/TC2/D6/49 [TC2d5 ERN 3]

1498 On Page: 589 Line: 19374 Section: hypot (2003 Ed.)
1499 On Page: 586 Line: 19238 Section: hypot (2001 Ed.)

1500 In the EXAMPLES section
1501 Change From:
1502 "None."

1503 To:
1504 "See atan2() EXAMPLES."

1505 **Change Number:** XSH/TC2/D6/50 [XSH ERN 24]

1506 On Page: 648 Line: 21400 Section: isunordered (2003 Ed.)

1507 On Page: 645 Line: 21256 Section: isunordered (2001 Ed.)

1508 Change From:

1509 "If x or y is NaN, 0 shall be returned."

1510 To:

1511 "If x or y is NaN, 1 shall be returned."

1512 Rationale:

1513 The previous statement is incorrect. If x is NaN, then isunordered()

1514 must return 1. Likewise if y is NaN, then isunordered() must return 1.

1515 **Change Number:** XSH/TC2/D6/51 [XSH ERN 144]

1516 On Page: 669 Line: 22175-22176 Section: kill (2003 Ed.)

1517 On Page: 666 Line: 22031-22032 Section: kill (2001 Ed.)

1518 In the RATIONALE section

1519 Change From:

1520 "The implementation-defined processes to which a signal cannot be sent

1521 may include the scheduler or init."

1522 To:

1523 "The unspecified processes to which a signal cannot be sent

1524 may include the scheduler or init."

1525 Rationale: The RATIONALE section conflicted with the normative

1526 text in the DESCRIPTION section.

1527 **Change Number:** XSH/TC2/D6/52 [XSH ERN 135]

1528 On Page: 671 Line: 22533 Section: killpg (2003 Ed.)

1529 On Page: 668 Line: 22109 Section: killpg (2001 Ed.)

1530 In the EXAMPLES section

1531 Change From:

1532 "None."

1533 To:

1534 "Sending a signal to all other members of a process group

1535 The following example shows how the calling process could send a

1536 signal to all other members of its process group. To prevent

1537 itself from receiving the signal it first makes itself immune to

1538 the signal by ignoring it.

1539 #include <signal.h>

1540 #include <unistd.h>

1541 ...

1542 if (signal(SIGUSR1, SIG_IGN) == SIG_ERR)

1543 /* Handle error */;

1544 if (killpg(getpgrp(), SIGUSR1) == -1)

1545 /* Handle error */;

Technical Corrigendum 2

1546 **Change Number:** XSH/TC2/D6/53 [XSH ERN 78]

1547 On Page: 686 Line: 22699 Section: lio_listio (2003 Ed.)

1548 On Page: 683 Line: 22555 Section: lio_listio (2001 Ed.)

1549 In the DESCRIPTION section

1550 Add after line 22699 (2003 Ed.); line 22555 (2001 Ed.):

1551 "If the buffer pointed to by list or the aiocb structures pointed to
1552 by the elements of the array list become illegal addresses before all
1553 asynchronous I/O completed and, if necessary, the notification is sent,
1554 then the behavior is undefined. If the buffers pointed to by the
1555 aio_buf member of the aiocb structure pointed to by the elements of
1556 the array list become illegal addresses prior to the asynchronous I/O
1557 associated with that aiocb structure being completed, the behavior is
1558 undefined."

1559 Rationale: This text is added for symmetry with the aio_read()

1560 and aio_write() functions.

1561 **Change Number:** XSH/TC2/D6/54 [XSH ERN 79]

1562 On Page: 686 Line: 22719 Section: lio_listio (2003 Ed.)

1563 On Page: 683 Line: 22575 Section: lio_listio (2001 Ed.)

1564 In the DESCRIPTION Section (after the last paragraph)

1565 Add after line 22719 (2003 Ed.); line 22575 (2001 Ed.):

1566 "If sig->sigev_notify is SIGEV_THREAD and
1567 sig->sigev_notify_attributes is a non-NULL pointer and the block
1568 pointed to by this pointer becomes an illegal address prior to all
1569 asynchronous I/O being completed, then the behavior is undefined."

1570 Rationale:

1571 This text is added to make it explicit that the user is required to keep

1572 the structure pointed to by sig->sigev_notify_attributes valid until the

1573 last asynchronous operation finished and the notification has been sent.

1574 **Change Number:** XSH/TC2/D6/55 [XSH ERN 49]

1575 On Page: 702 Line: 23186-23190 Section: localtime (2003 Ed.)

1576 On Page: 699 Line: 23038-23041 Section: localtime (2001 Ed.)

1577 In the RETURN VALUE

1578 Change From:

1579 "Upon successful completion, localtime_r() shall return a
1580 pointer to the structure pointed to by the argument result."

1581 To:

1582 "Upon successful completion, localtime_r() shall return a
1583 pointer to the structure pointed to by the argument result. If
1584 an error is detected, localtime_r() shall return
1585 a null pointer and set errno to indicate the error."

1586 In the ERRORS section of the 2003 Ed.

1587 Change From:

1588 "The localtime() function shall fail if:

1589 [EOVERFLOW] The result cannot be represented."

1590 To:

1591 "The localtime() and localtime_r() functions shall fail if:

1592 [EOVERFLOW] The result cannot be represented."

1593 with "and localtime_r()" shaded and margin marked TSF.

1594 In the ERRORS section of the 2001 Ed.

1595 Change From:

1596 "No errors are defined."

1597 To:

1598 "The localtime() and localtime_r() functions shall fail if:

1599 [EOVERFLOW] The result cannot be represented."

1600 with "and localtime_r()" shaded and margin marked TSF.

1601 **Change Number: XSH/TC2/D6/56** [XSH ERN 84,85]

1602 On Page: 702,704 Line: 23181,23245 Section: localtime (2003 Ed.)

1603 On Page: 699,701 Line: 23035,23096 Section: localtime (2001 Ed.)

1604 In the DESCRIPTION section

1605 Add a new paragraph following the last paragraph

1606 with TSF and XSI shading:

1607 "If the reentrant version does not set tzname, it shall not set daylight

1608 and shall not set timezone."

1609 In the SEE ALSO section

1610 Add "tzset()," before "utime(),".

1611 Rationale:

1612 On systems supporting XSI, the daylight, timezone, and tzname variables

1613 should all be set to provide information for the same timezone.

1614 This updates the description of localtime_r() to mention

1615 daylight and timezone as well as tzname.

1616 **Change Number: XSH/TC2/D6/57** [XSH ERN 53]

1617 On Page: 732,733 Line: 24081-24083,24143-24146 Section: makecontext (2003 Ed.)

1618 On Page: 729,730 Line: 23930-23932,23993-23995 Section: makecontext (2001 Ed.)

1619 In the SYNOPSIS section

1620 Change the "XSI" margin marker to "OB XSI"

1621 Change the function prototype from:

1622 "void makecontext(ucontext_t *ucp, void (*func)(void), int argc, ...);"

1623 To:

1624 "void makecontext(ucontext_t *ucp, void (*func)(), int argc, ...);"

1625 In the APPLICATION USAGE section

1626 Change From:

1627 "None."

1628 To:

1629 "The obsolescent functions getcontext(), makecontext() and

1630 swapcontext() can be replaced using POSIX threads functions."

1631 In the RATIONALE section

1632 Change From:

1633 "None."

1634 To:

1635 "With the incorporation of ISO/IEC 9899:1999 (C99) into this

1636 specification it was found that the ISO C Standard (subclause 6.11.6)

1637 specifies that the use of function declarators with empty parentheses

1638 is an obsolescent feature. Therefore, using the function prototype:

1639 void makecontext(ucontext_t *ucp, void (*func)(), int argc, ...);

1640 is making use of an obsolescent feature of ISO C. Therefore, a strictly

1641 conforming POSIX application can't use this form. Therefore, use of

1642 getcontext(), makecontext() and swapcontext() is marked obsolescent.

1643 There is no way in ISO C to specify a non-obsolescent function prototype

Technical Corrigendum 2

1644 indicating that a function will be called with an arbitrary number
1645 (including zero) of arguments of arbitrary types (including integers,
1646 pointers to data, pointers to functions, and composite types).

1647 Replacing makecontext() with a number of ISO C compatible functions
1648 handing various numbers and types of arguments would have forced all
1649 existing uses of makecontext() to be rewritten for little or no gain.
1650 There are very few applications today that use the *context() routines.
1651 Those that do use them are almost always using them to implement
1652 co-routines. By maintaining the XSH5 specification for makecontext(),
1653 existing applications will continue to work, although they won't be able
1654 to be classified as strictly conforming applications.

1655 There is no way in ISO C (without using obsolescent behavior) to specify
1656 functionality that was standard, strictly conforming behavior in XSH5
1657 using the 1990 C Standard. Threads can be used to implement the
1658 functionality provided by makecontext(), getcontext() and swapcontext()
1659 but they are more complex to use. It was felt inventing new ISO C compatible
1660 interfaces that describe what can be done with the XSH5 functions and
1661 then converting applications to use them would cause more difficulty
1662 than just converting applications that use them to use threads instead."

1663 **Change Number:** XSH/TC2/D6/58 [XSH ERN 50]

1664 On Page: 768 Line: 25150-25152,25154 Section: mktime (2003 Ed.)

1665 On Page: 765 Line: 24997-24999,25001 Section: mktime (2001 Ed.)

1666 In the RETURN VALUE section

1667 Change From:

1668 "The mktime() function shall return the specified time since the Epoch
1669 encoded as a value of type time_t. If the time since the Epoch cannot
1670 be represented, the function shall return the value (time_t)-1."

1671 To:

1672 "The mktime() function shall return the specified time since the Epoch
1673 encoded as a value of type time_t. If the time since the Epoch cannot
1674 be represented, the function shall return the value (time_t)-1 [CX]
1675 and may set errno to indicate the error.[/CX]."

1676 In the ERRORS section

1677 Change From:

1678 "None."

1679 To:

1680 (Shaded and marked with the CX margin code):

1681 "The mktime() function may fail if:

1682 [Eoverflow] The result cannot be represented."

1683 **Change Number:** XSH/TC2/D6/59 [XSH ERN 86]

1684 On Page: 769 Line: 25185 Section: mktime (2003 Ed.)

1685 On Page: 766 Line: 25032 Section: mktime (2001 Ed.)

1686 In the SEE ALSO section

1687 Add "tzset()," before "utime()," "

1688 Rationale: This is editorial.

1689 **Change Number:** XSH/TC2/D6/60 [XSH ERN 72]

1690 On Page: 774,777 Line: 25357,25469 Section: mmap (2003 Ed.)

1691 On Page: 771,774 Line: 25204,25316 Section: mmap (2001 Ed.)

1692 In the DESCRIPTION section

1693 Add a new paragraph after line 25357 (2003 Ed.); line 25204 (2001 Ed.) :

1694 "If len is zero mmap() shall fail and no mapping shall be

1695 established."

1696 In the ERRORS section (shall fail)

1697 Add after line 25469 (2003 Ed.); line 25316 (2001 Ed.):

1698 "[EINVAL] The value of len is zero."

1699 **Change Number:** XSH/TC2/D6/61 [XSH ERN 47]

1700 On Page: 786 Line: 25817 Section: mq_getattr (2003 Ed.)

1701 On Page: 783 Line: 25660 Section: mq_getattr (2001 Ed.)

1702 In the ERRORS section

1703 Change From:

1704 "The mq_getattr() function shall fail if:"

1705 To:

1706 "The mq_getattr() function may fail if:"

1707 **Change Number:** XSH/TC2/D6/62 [XSH ERN 87]

1708 On Page: 791 Line: 25933 Section: mq_open (2003 Ed.)

1709 On Page: 788 Line: 25776 Section: mq_open (2001 Ed.)

1710 In the DESCRIPTION section (O_CREAT mode)

1711 Change From:

1712 "The file permission bits shall be set to the value of mode. When

1713 bits in mode other than file permission bits are set, the effect is

1714 implementation-defined."

1715 To:

1716 "The permission bits of the message queue shall be set to the value of the

1717 mode argument except those set in the file mode creation mask of the

1718 process. When bits in mode other than the file permission bits are

1719 specified, the effect is unspecified."

1720 Rationale: Consistency with shm_open(), sem_open().

1721 **Change Number:** XSH/TC2/D6/63 [XSH ERN 25]

1722 On Page: 823 Line: 26879-26880 Section: nanosleep (2003 Ed.)

1723 On Page: 820 Line: 26720-26721 Section: nanosleep (2001 Ed.)

1724 In the RATIONALE section

1725 Change From:

1726 "It is common to suspend execution of a process for an interval in order

1727 to poll the status of a non-interrupting function."

1728 To:

1729 "It is common to suspend execution of a thread for an interval in order

1730 to poll the status of a non-interrupting function."

Technical Corrigendum 2

1731 **Change Number:** XSH/TC2/D6/64 [XSH ERN 73]

1732 On Page: 829-831 Line: 27026,27069,27102,27114 Section: nftw (2003 Ed.)

1733 On Page: 826-828 Line: 26865,26908,26941,26953 Section: nftw (2001 Ed.)

1734 In the SYNOPSIS section, line 27026 (2003 Ed.); line 26865 (2001 Ed.)

1735 Change From:

```
1736 "int nftw(const char *path, int (*fn)(const char *, const struct stat *,
1737         int, struct FTW *), int depth, int flags);"
```

1738 To:

```
1739 "int nftw(const char *path, int (*fn)(const char *, const struct
1740         stat *, int, struct FTW *), int fd_limit, int flags); "
```

1741 In the DESCRIPTION section, line 27069 (2003 Ed.); line 26908 (2001 Ed.)

1742 Change From:

```
1743 "The argument depth sets the maximum number of file descriptors that
1744 shall be used by nftw() while traversing the file tree."
```

1745 To:

```
1746 "The argument fd_limit sets the maximum number of file descriptors that
1747 shall be used by nftw() while traversing the file tree."
```

1748 In the EXAMPLES section, line 27102 (2003 Ed.); line 26941 (2001 Ed.)

1749 Change From:

```
1750 "The following example walks the /tmp directory and its subdirectories,
1751 calling the nftw() function for every directory entry, to a maximum
1752 of 5 levels deep."
```

1753 To:

```
1754 "The following example walks the /tmp directory and its subdirectories,
1755 calling the nftw() function for every directory entry, using a maximum
1756 of 5 file descriptors."
```

1757 On line 27114 (2003 Ed.); line 26953 (2001 Ed.)

1758 Change From:

```
1759 "int depth = 5;
1760 int flags = FTW_CHDIR | FTW_DEPTH | FTW_MOUNT;
1761 int ret;
1762 ret = nftw(startpath, nftwfunc, depth, flags);"
```

1763 To:

```
1764 "int fd_limit = 5;
1765 int flags = FTW_CHDIR | FTW_DEPTH | FTW_MOUNT;
1766 int ret;
1767 ret = nftw(startpath, nftwfunc, fd_limit, flags);"
```

1768 **Change Number:** XSH/TC2/D6/65 [XSH ERN 127]

1769 On Page: 856 Line: 27866 Section: pipe (2003 Ed.)

1770 On Page: 853 Line: 27705 Section: pipe (2001 Ed.)

1771 In the EXAMPLES section

1772 Change From:

1773 "None."

1774 To:

1775 "Using a pipe to pass data between a parent process and a child process

```
1776 The following example demonstrates the use of a pipe to transfer data
1777 between a parent process and a child process. Error handling is excluded,
1778 but otherwise this code demonstrates good practice when using pipes:
1779 after the fork() the two processes close the unused ends of the pipe
1780 before they commence transferring data.
```

```
1781 #include <stdlib.h>
```

```
1782 #include <unistd.h>
```

```
1783 ...
```

```
1784 int fildes[2];
```

```

1785 const int BSIZE = 100;
1786 char buf[BSIZE];
1787 ssize_t nbytes;
1788 int status;

1789 status = pipe(fildes);
1790 if (status == -1 ) {
1791     /* an error occurred */
1792     ....
1793 }

1794 switch (fork()) {
1795 case -1: /* Handle error */
1796     break;

1797 case 0: /* Child - reads from pipe */
1798     close(fildes[1]); /* Write end is unused */
1799     nbytes = read(fildes[0], buf, BSIZE); /* Get data from pipe */
1800     /* At this point, a further read would see end of file... */
1801     close(fildes[0]); /* Finished with pipe */
1802     exit(EXIT_SUCCESS);

1803 default: /* Parent - writes to pipe */
1804     close(fildes[0]); /* Read end is unused */
1805     write(fildes[1], "Hello world\n", 12); /* Write data on pipe */
1806     close(fildes[1]); /* Child will see EOF */
1807     exit(EXIT_SUCCESS);
1808 }"

```

1809 **Change Number:** XSH/TC2/D6/66 [XSH ERN 145]

1810 On Page: 860 Line: 27995-27999 Section: poll (2003 Ed.)

1811 On Page: 857 Line: 27834-27838 Section: poll (2001 Ed.)

1812 In the EXAMPLES section

1813 Change From:

```

1814 "/* Open STREAMS device. */
1815 fds[0].fd = open("/dev/dev0", ...);
1816 fds[1].fd = open("/dev/dev1", ...);
1817     fds[0].events = POLLOUT | POLLWRBAND;
1818     fds[1].events = POLLOUT | POLLWRBAND;"

```

1819 To:

```

1820 "/* Open STREAMS device. */
1821 fds[0].fd = open("/dev/dev0", ...);
1822 fds[1].fd = open("/dev/dev1", ...);
1823 fds[0].events = POLLOUT | POLLWRBAND;
1824 fds[1].events = POLLOUT | POLLWRBAND;"

```

Technical Corrigendum 2

1825 **Change Number:** XSH/TC2/D6/67 [XSH ERN 128]

1826 On Page: 863 Line: 28077 Section: popen (2003 Ed.)

1827 On Page: 860 Line: 27916 Section: popen (2001 Ed.)

1828 In the EXAMPLES section

1829 Change From:

1830 "None."

1831 To:

1832 "Using popen() to obtain a list of files from the ls utility

1833 The following example demonstrates the use of popen()

1834 and pclose() to execute the command "ls *" in order to

1835 obtain a list of files in the current directory:

1836 #include <stdio.h>

1837 ...

1838 FILE *fp;

1839 int status;

1840 char path[PATH_MAX];

1841 fp = popen("ls *", "r");

1842 if (fp == NULL)

1843 /* Handle error */;

1844 while (fgets(path, PATH_MAX, fp) != NULL)

1845 printf("%s", path);

1846 status = pclose(fp);

1847 if (status == -1) {

1848 /* Error reported by pclose() */

1849 ...

1850 } else {

1851 /* Use macros described under wait() to inspect 'status' in order

1852 to determine success/failure of command executed by popen() */

1853 ...

1854 }"

1855 **Change Number:** XSH/TC2/D6/68 [XSH ERN 146]

1856 On Page: 864 Line: 28122 Section: posix_fadvise (2003 Ed.)

1857 On Page: 861 Line: 27961 Section: posix_fadvise (2001 Ed.)

1858 In the SYNOPSIS section

1859 Change From:

1860 "int posix_fadvise(int fd, off_t offset, size_t len, int advice);"

1861 To:

1862 "int posix_fadvise(int fd, off_t offset, off_t len, int advice);"

1863 Rationale: The previous prototype was not large-file aware, and the standard

1864 developers felt it acceptable to make this change before

1865 implementations of the functions become widespread."

1866 **Change Number:** XSH/TC2/D6/69 [XSH ERN 146]

1867 On Page: 866 Line: 28177 Section: posix_fallocate (2003 Ed.)

1868 On Page: 863 Line: 28016 Section: posix_fallocate (2001 Ed.)

1869 In the SYNOPSIS section

1870 Change From:

1871 "int posix_fallocate(int fd, off_t offset, size_t len);"

1872 To:

1873 "int posix_fallocate(int fd, off_t offset, off_t len);"

1874 Rationale: The previous prototype was not large-file aware, and the standard

1875 developers felt it acceptable to make this change before

1876 implementations of the functions become widespread."

1877 **Change Number:** XSH/TC2/D6/70 [XSH ERN 110]

1878 On Page: 975 Line: 31069-31071 Section: pselect (2003 Ed.)

1879 On Page: 972 Line: 30903-30905 Section: pselect (2001 Ed.)

1880 In the DESCRIPTION section

1881 Change From:

1882 "If sigmask is not a null pointer, then the pselect() function shall

1883 replace the signal mask of the process by the set of signals pointed

1884 to by sigmask before examining the descriptors, and shall restore the

1885 signal mask of the process before returning."

1886 To:

1887 "If sigmask is not a null pointer, then the pselect() function shall

1888 replace the signal mask of the caller by the set of signals pointed to by

1889 sigmask before examining the descriptors, and shall restore the signal

1890 mask of the calling thread before returning."

1891 **Change Number:** XSH/TC2/D6/71 [XSH ERN 147,148]

1892 On Page: 981 Line: 31293 Section: pthread_attr_destroy (2003 Ed.)

1893 On Page: 978 Line: 31127 Section: pthread_attr_destroy (2001 Ed.)

1894 In the ERRORS section

1895 Add

1896 "The pthread_attr_destroy() may fail if:

1897 [EINVAL] The value specified by attr does not refer to an initialized

1898 thread attribute object.

1899 The pthread_attr_init() may fail if:

1900 [EBUSY] The implementation has detected an attempt to reinitialize the thread

1901 attribute referenced by attr, a previously initialized, but not yet

1902 destroyed, thread attribute."

Technical Corrigendum 2

1903 **Change Number:** XSH/TC2/D6/72 [XSH ERN 125]

1904 On Page: 984 Line: 31407 Section: pthread_attr_getdetachstate (2003 Ed.)

1905 On Page: 981 Line: 31241 Section: pthread_attr_getdetachstate (2001 Ed.)

1906 In the EXAMPLES section

1907 Change From:

1908 "None."

1909 To:

1910 "Retrieving the detachstate attribute."

1911 This example shows how to obtain the detachstate attribute of a thread

1912 attribute object.

```
1913 #include <pthread.h>
```

```
1914 pthread_attr_t thread_attr;
```

```
1915 int detachstate;
```

```
1916 int rc;
```

```
1917 /* code initializing thread_attr */
```

```
1918 ...
```

```
1919 rc = pthread_attr_getdetachstate (&thread_attr, &detachstate);
```

```
1920 if (rc!=0) {
```

```
1921     /* handle error */
```

```
1922     ...
```

```
1923 }
```

```
1924 else {
```

```
1925     /* legal values for detachstate are:
```

```
1926     * PTHREAD_CREATE_DETACHED or PTHREAD_CREATE_JOINABLE
```

```
1927     */
```

```
1928     ...
```

```
1929 }"
```

1930 **Change Number:** XSH/TC2/D6/73 [XSH ERN 149]

1931 On Page: 984 Line: 31404 Section: pthread_attr_getdetachstate (2003 Ed.)

1932 On Page: 981 Line: 31238 Section: pthread_attr_getdetachstate (2001 Ed.)

1933 In the ERRORS section

1934 Add

1935 "The pthread_attr_getdetachstate() may fail if:

1936 [EINVAL] The value specified by attr does not refer to an initialized

1937 thread attribute object.

1938 The pthread_attr_setdetachstate() may fail if:

1939 [EINVAL] The value specified by attr does not refer to an initialized

1940 thread attribute object."

1941 **Change Number:** XSH/TC2/D6/74 [XSH ERN 150]

1942 On Page: 986 Line: 31461,31462 Section: pthread_attr_getguardsize (2003 Ed.)

1943 On Page: 983 Line: 31295,31296 Section: pthread_attr_getguardsize (2001 Ed.)

1944 In the ERRORS section

1945 On line 31461 (2003 Ed.); line 31295 (2001 Ed.) delete:

1946 "[EINVAL] The attribute attr is invalid."

1947 In the ERRORS section

1948 Add:

1949 "The pthread_attr_getguardsize() and pthread_attr_setguardsize may fail

1950 if:

1951 [EINVAL] The value specified by attr does not refer to an initialized

1952 thread attribute object."

1953 **Change Number:** XSH/TC2/D6/75 [XSH ERN 151,152]

1954 On Page: 986 Line: 31500,31516,31517 Section: pthread_attr_getinheritsched (2003 Ed.)

1955 On Page: 983 Line: 31334,31351,31352 Section: pthread_attr_getinheritsched (2001 Ed.)

1956 In the DESCRIPTION section

1957 After line 31500 (2003 Ed.); line 31334 (2001 Ed.) insert a new

1958 paragraph:

1959 "The supported values of inheritsched shall be:"

1960 In the ERRORS section

1961 Insert before line 31516 (2003 Ed.); line 31350 (2001 Ed.)

1962 "The pthread_attr_getinheritsched() may fail if:

1963 [EINVAL] The value specified by attr does not refer to an initialized

1964 thread attribute object."

1965 In the ERRORS section

1966 Add after line 31517 (2003 Ed.); 31351 (2001 Ed.):

1967 "[EINVAL] The value specified by attr does not refer to an initialized

1968 thread attribute object."

Technical Corrigendum 2

1969 **Change Number:** XSH/TC2/D6/76 [XSH ERN 124]

1970 On Page: 987 Line: 31465 Section: pthread_attr_getguardsize (2003 Ed.)

1971 On Page: 984 Line: 31299 Section: pthread_attr_getguardsize (2001 Ed.)

1972 In the EXAMPLES section

1973 Change From:

1974 "None."

1975 To:

1976 "Retrieving the guardsize attribute."

1977 This example shows how to obtain the guardsize attribute of a thread

1978 attribute object.

```
1979 #include <pthread.h>
```

```
1980 pthread_attr_t thread_attr;
```

```
1981 size_t guardsize;
```

```
1982 int rc;
```

```
1983 /* code initializing thread_attr */
```

```
1984 ...
```

```
1985 rc = pthread_attr_getguardsize (&thread_attr, &guardsize);
```

```
1986 if (rc != 0) {
```

```
1987     /* handle error */
```

```
1988     ...
```

```
1989 }
```

```
1990 else {
```

```
1991     if (guardsize > 0) {
```

```
1992         /* a guard area of at least guardsize bytes is provided */
```

```
1993         ...
```

```
1994     }
```

```
1995     else {
```

```
1996         /* no guard area provided */
```

```
1997         ...
```

```
1998     }
```

```
1999 }
```

2000 **Change Number:** XSH/TC2/D6/77 [XSH ERN 55]

2001 On Page: 988 Line: 31524 Section: pthread_attr_getinheritsched (2003 Ed.)

2002 On Page: 985 Line: 31358 Section: pthread_attr_getinheritsched (2001 Ed.)

2003 In the APPLICATION USAGE section

2004 Add:

2005 "See Section 2.9.4 for further details on thread scheduling

2006 attributes and their default settings."

2007 **Change Number:** XSH/TC2/D6/78 [XSH ERN 154]

2008 On Page: 990 Line: 31567,31569 Section: pthread_attr_getschedparam (2003 Ed.)

2009 On Page: 987 Line: 31401,31403 Section: pthread_attr_getschedparam (2001 Ed.)

2010 In the ERRORS section

2011 Add after line 31567 (2003 Ed.); line 31401 (2001 Ed.):

2012 "The pthread_attr_getschedparam() function may fail if:

2013 [EINVAL] The value specified by attr does not refer to an initialized

2014 thread attribute object."

2015 On line 31569 (2003 Ed.);31403 (2001 Ed.)

2016 Change From:

2017 "[EINVAL] The value of param is not valid."

2018 To:

2019 "[EINVAL] The value of param is not valid, or the value specified by attr does

2020 not refer to an initialized thread attribute object."

2021 **Change Number:** XSH/TC2/D6/79 [XSH ERN 56]

2022 On Page: 992 Line: 31621 Section: pthread_attr_getschedpolicy (2003 Ed.)

2023 On Page: 989 Line: 31455 Section: pthread_attr_getschedpolicy (2001 Ed.)

2024 In the APPLICATION USAGE section

2025 Add:

2026 "See Section 2.9.4 for further details on thread scheduling

2027 attributes and their default settings."

2028 **Change Number:** XSH/TC2/D6/80 [XSH ERN 155]

2029 On Page: 992 Line: 31612,31614 Section: pthread_attr_getschedpolicy (2003 Ed.)

2030 On Page: 989 Line: 31446,31448 Section: pthread_attr_getschedpolicy (2001 Ed.)

2031 In the ERRORS section

2032 Add after line 31612 (2003 Ed.); line 31446 (2001 Ed.):

2033 "The pthread_attr_getschedpolicy() function may fail if:

2034 [EINVAL] The value specified by attr does not refer to an initialized

2035 thread attribute object."

2036 On line 31614 (2003 Ed.);31448 (2001 Ed.)

2037 Change From:

2038 "[EINVAL] The value of policy is not valid."

2039 To:

2040 "[EINVAL] The value of policy is not valid, or the value specified by

2041 attr does not refer to an initialized thread attribute object."

2042 **Change Number:** XSH/TC2/D6/81 [XSH ERN 57]

2043 On Page: 994 Line: 31669 Section: pthread_attr_getscope (2003 Ed.)

2044 On Page: 991 Line: 31503 Section: pthread_attr_getscope (2001 Ed.)

2045 In the APPLICATION USAGE section

2046 Add:

2047 "See Section 2.9.4 for further details on thread scheduling

2048 attributes and their default settings."

Technical Corrigendum 2

2049 **Change Number:** XSH/TC2/D6/82 [XSH ERN 157]

2050 On Page: 994 Line: 31660,31662 Section: pthread_attr_getscope (2003 Ed.)

2051 On Page: 991 Line: 31494,31496 Section: pthread_attr_getscope (2001 Ed.)

2052 In the ERRORS section

2053 Add after line 31660 (2003 Ed.); line 31494 (2001 Ed.):

2054 "The pthread_attr_getscope() function may fail if:

2055 [EINVAL] The value specified by attr does not refer to an initialized

2056 thread attribute object."

2057 On line 31662 (2003 Ed.);31496 (2001 Ed.)

2058 Change From:

2059 "[EINVAL] The value of contentionscope is not valid."

2060 To:

2061 "[EINVAL] The value of contentionscope is not valid, or the value

2062 specified by attr does not refer to an initialized thread attribute

2063 object."

2064 **Change Number:** XSH/TC2/D6/83 [XSH ERN 89]

2065 On Page: 996 Line: 31731 Section: pthread_attr_getstack (2003 Ed.)

2066 On Page: 993 Line: 31565 Section: pthread_attr_getstack (2001 Ed.)

2067 In the APPLICATION USAGE

2068 Add at the end:

2069 "After a successful call to pthread_attr_setstack(), the storage

2070 area specified by the stackaddr parameter is under the control of the

2071 implementation as described in Section 2.9.8."

2072 **Change Number:** XSH/TC2/D6/84 [XSH ERN 158]

2073 On Page: 996 Line: 31714,31716 Section: pthread_attr_getstack (2003 Ed.)

2074 On Page: 993 Line: 31548,31550 Section: pthread_attr_getstack (2001 Ed.)

2075 In the ERRORS section

2076 Add after line 31774 (2003 Ed.); line 31716 (2001 Ed.):

2077 "The pthread_attr_getstack() function may fail if:

2078 [EINVAL] The value specified by attr does not refer to an initialized

2079 thread attribute object."

2080 On line 31716 (2003 Ed.);31550 (2001 Ed.)

2081 Change From:

2082 "[EINVAL] The value of stackaddr does not have proper alignment to be

2083 used as a stack, or if (stackaddr + stacksize) lacks proper alignment."

2084 To:

2085 "[EINVAL] The value of stackaddr does not have proper alignment to be

2086 used as a stack, or (stackaddr + stacksize) lacks proper alignment,

2087 or the value specified by attr does not refer to an initialized thread

2088 attribute object."

2089 **Change Number:** XSH/TC2/D6/85 [XSH ERN 89]

2090 On Page: 998 Line: 31777 Section: pthread_attr_getstackaddr (2003 Ed.)

2091 On Page: 995 Line: 31611 Section: pthread_attr_getstackaddr (2001 Ed.)

2092 In the APPLICATION USAGE

2093 Add at the end:

2094 "After a successful call to pthread_attr_setstackaddr(), the storage
2095 area specified by the stackaddr parameter is under the control of the
2096 implementation as described in Section 2.9.8."

2097 **Change Number:** XSH/TC2/D6/86 [XSH ERN 159]

2098 On Page: 998 Line: 31761 Section: pthread_attr_getstackaddr (2003 Ed.)

2099 On Page: 995 Line: 31595 Section: pthread_attr_getstackaddr (2001 Ed.)

2100 In the ERRORS section

2101 Change From:

2102 "No errors are defined."

2103 To:

2104 "The pthread_attr_getstackaddr() may fail if:

2105 [EINVAL] The value specified by attr does not refer to an initialized
2106 thread attribute object.

2107 The pthread_attr_setstackaddr() may fail if:

2108 [EINVAL] The value specified by attr does not refer to an initialized
2109 thread attribute object."

2110 **Change Number:** XSH/TC2/D6/87 [XSH ERN 160]

2111 On Page: 1000 Line: 31815 Section: pthread_attr_getstacksize (2003 Ed.)

2112 On Page: 997 Line: 31649 Section: pthread_attr_getstacksize (2001 Ed.)

2113 In the ERRORS section

2114 Add:

2115 "The pthread_attr_getstacksize() may fail if:

2116 [EINVAL] The value specified by attr does not refer to an initialized
2117 thread attribute object.

2118 The pthread_attr_setstacksize() may fail if:

2119 [EINVAL] The value specified by attr does not refer to an initialized
2120 thread attribute object."

2121 **Change Number:** XSH/TC2/D6/88 [XSH ERN 64]

2122 On Page: 1024 Line: 32233 Section: pthread_cleanup_pop (2003 Ed.)

2123 On Page: 1020 Line: 32064 Section: pthread_cleanup_pop (2001 Ed.)

2124 In the DESCRIPTION section

2125 Add a new final paragraph:

2126 "The effect of the use of 'return', 'break', 'continue', and 'goto' to
2127 prematurely leave a code block described by a pair of pthread_cleanup_push()
2128 and pthread_cleanup_pop() functions calls is undefined."

Technical Corrigendum 2

2129 **Change Number:** XSH/TC2/D6/89 [XSH ERN 90]

2130 On Page: 1036 Line: 32652-32655 Section: pthread_cond_timedwait (2003 Ed.)

2131 On Page: 1033 Line: 32483-32486 Section: pthread_cond_timedwait (2001 Ed.)

2132 In the DESCRIPTION section

2133 Change From:

2134 "The effect of using more than one mutex for concurrent
2135 pthread_cond_timedwait() or pthread_cond_wait() operations on
2136 the same condition variable is undefined; that is, a condition
2137 variable becomes bound to a unique mutex when a thread waits on the
2138 condition variable, and this (dynamic) binding shall end when the
2139 wait returns."

2140 To:

2141 "When a thread waits on a condition variable, having specified
2142 a particular mutex to either the pthread_cond_timedwait() or the
2143 pthread_cond_wait() operation, a dynamic binding is formed between that
2144 mutex and condition variable that remains in effect as long as at least
2145 one thread is blocked on the condition variable. During this time, the
2146 effect of an attempt by any thread to wait on that condition variable
2147 using a different mutex is undefined. Once all waiting threads have been
2148 unblocked (as by the pthread_cond_broadcast() operation), the next wait
2149 operation on that condition variable shall form a new dynamic binding
2150 with the mutex specified by that wait operation. Even though the dynamic
2151 binding between condition variable and mutex may be removed or replaced
2152 between the time a thread is unblocked from a wait on the condition
2153 variable and the time that it returns to the caller or begins cancellation
2154 cleanup, the unblocked thread shall always re-acquire the mutex specified
2155 in the condition wait operation call from which it is returning."

2156 Rationale:

2157 This change is for consistency with pthread_cond_destroy() which says
2158 it is safe to destroy an initialized condition variable upon which no
2159 threads are currently blocked.

2160 **Change Number:** XSH/TC2/D6/90 [XSH ERN 58]

2161 On Page: 1036 Line: 32656-32659 Section: pthread_cond_timedwait (2003 Ed.)

2162 On Page: 1032 Line: 32487-32490 Section: pthread_cond_timedwait (2001 Ed.)

2163 In the DESCRIPTION section

2164 Change From:

2165 "When the cancelability enable state of a thread is set to
2166 PTHREAD_CANCEL_DEFERRED, a side effect of acting upon a
2167 cancellation request while in a condition wait is that the mutex is
2168 (in effect) re-acquired before calling the first cancellation
2169 cleanup handler."

2170 To:

2171 "When the cancelability type of a thread is set to PTHREAD_CANCEL_DEFERRED,
2172 a side effect of acting upon a cancellation request while in a condition
2173 wait is that the mutex is (in effect) re-acquired before calling the
2174 first cancellation cleanup handler."

2175 **Change Number:** XSH/TC2/D6/91 [XSH ERN 26]

2176 On Page: 1037 Line: 32690-32693 Section: pthread_cond_timedwait (2003 Ed.)
 2177 On Page: 1033 Line: 32521-32524 Section: pthread_cond_timedwait (2001 Ed.)

2178 In the ERRORS section
 2179 Change From:
 2180 "The pthread_cond_timedwait() and pthread_cond_wait() functions may fail if:

2181 [EINVAL] The value specified by cond, mutex, or abstime is invalid.

2182 [EINVAL] Different mutexes were supplied for concurrent
 2183 pthread_cond_timedwait() or pthread_cond_wait() operations on the same
 2184 condition variable."

2185 To:
 2186 "The pthread_cond_timedwait() function shall fail if:

2187 [ETIMEDOUT] The time specified by abstime to pthread_cond_timedwait()
 2188 has passed.

2189 [EINVAL] The value specified by abstime is invalid.

2190 The pthread_cond_timedwait() and pthread_cond_wait() functions may
 2191 fail if:

2192 [EINVAL] The value specified by cond or mutex is invalid."

2193 **Change Number:** XSH/TC2/D6/92 [XSH ERN 118]

2194 On Page: 1037 Line: 32705 Section: pthread_cond_timedwait (2003 Ed.)
 2195 On Page: 1033 Line: 32536 Section: pthread_cond_timedwait (2001 Ed.)

2196 In the RATIONALE section
 2197 Add a new second paragraph:
 2198 "The application needs to recheck the predicate on any return because it
 2199 cannot be sure there is another thread waiting on the thread to handle
 2200 the signal, and if there is not then the signal is lost. The burden
 2201 is on the application to check the predicate."

2202 **Change Number:** XSH/TC2/D6/93 [XSH ERN 136]

2203 On Page: 1050 Line: 33036,33038 Section: pthread_create (2003 Ed.)
 2204 On Page: 1046 Line: 32866,32868 Section: pthread_create (2001 Ed.)

2205 In the ERROR RETURN section
 2206 Delete the following from the "shall fail" section:
 2207 "[EINVAL] The value specified by attr is invalid."

2208 Add after line 33038 (2003 Ed.); 32868 (2001 Ed.):
 2209 "The pthread_create() function may fail if:

2210 [EINVAL] The attributes specified by attr are invalid."

Technical Corrigendum 2

2211 **Change Number:** XSH/TC2/D6/94 [XSH ERN 91]

2212 On Page: 1051 Line: 33043 Section: pthread_create (2003 Ed.)

2213 On Page: 1047 Line: 32873 Section: pthread_create (2001 Ed.)

2214 In the APPLICATION USAGE section

2215 Change From:

2216 "None."

2217 To:

2218 "There is no requirement on the implementation that the ID of the created
2219 thread be available before the newly created thread starts executing.

2220 The calling thread can obtain the ID of the created thread through the

2221 return value of the pthread_create() function, and the newly created

2222 thread can obtain its ID by a call to pthread_self()."

2223 **Change Number:** XSH/TC2/D6/95 [XSH ERN 27]

2224 On Page: 1053 Line: 33123-33127 Section: pthread_detach (2003 Ed.)

2225 On Page: 1049 Line: 32951-32954 Section: pthread_detach (2001 Ed.)

2226 In the ERRORS section

2227 Change From:

2228 "The pthread_detach() function shall fail if:

2229 [EINVAL] The implementation has detected that the value specified by
2230 thread does not refer to a joinable thread.

2231 [ESRCH] No thread could be found corresponding to that specified by the
2232 given thread ID."

2233 To:

2234 "The pthread_detach() function may fail if:

2235 [EINVAL] The implementation has detected that the value specified by
2236 thread does not refer to a joinable thread.

2237 [ESRCH] No thread could be found corresponding to that specified by the
2238 given thread ID."

2239 **Change Number:** XSH/TC2/D6/96 [XSH ERN 100]

2240 On Page: 1064 Line: 33451 Section: pthread_getspecific (2003 Ed.)

2241 On Page: 1060 Line: 33279 Section: pthread_getspecific (2001 Ed.)

2242 In the ERRORS Section

2243 Change From:

2244 "[ENOMEM] Insufficient memory exists to associate the value with the key."

2245 To:

2246 "[ENOMEM] Insufficient memory exists to associate the non-NULL value
2247 with the key."

2248 **Change Number:** XSH/TC2/D6/97 [XSH ERN 28]

2249 On Page: 1066 Line: 33494-33500 Section: pthread_join (2003 Ed.)

2250 On Page: 1062 Line: 33322-33328 Section: pthread_join (2001 Ed.)

2251 Change From:

2252 "The pthread_join() function shall fail if:

2253 [EINVAL] The implementation has detected that the value specified by

2254 thread does not refer to a joinable thread.

2255 [ESRCH] No thread could be found corresponding to that specified by

2256 the given thread ID.

2257 The pthread_join() function may fail if:

2258 [EDEADLK] A deadlock was detected or the value of thread specifies the

2259 calling thread."

2260 To:

2261 "The pthread_join() function shall fail if:

2262 [ESRCH] No thread could be found corresponding to that specified by

2263 the given thread ID.

2264 The pthread_join() function may fail if:

2265 [EDEADLK] A deadlock was detected or the value of thread specifies the

2266 calling thread.

2267 [EINVAL] The value specified by thread does not refer to a joinable

2268 thread."

2269 **Change Number:** XSH/TC2/D6/98 [XSH ERN 59,65]

2270 On Page: 1085 Line: 34135,34150 Section: pthread_mutex_lock (2003 Ed.)

2271 On Page: 1081 Line: 33962,33977 Section: pthread_mutex_lock (2001 Ed.)

2272 In the ERRORS section

2273 Change From:

2274 "[EDEADLK] The current thread already owns the mutex."

2275 To:

2276 "[EDEADLK] A deadlock condition was detected or the

2277 current thread already owns the mutex."

2278 In the RATIONALE section

2279 Change From:

2280 "For example, deadlocking on a double-lock is explicitly allowed behavior

2281 in order to avoid requiring more overhead in the basic mechanism than is

2282 absolutely necessary."

2283 To:

2284 "For example, on systems not supporting the XSI extended mutex types,

2285 deadlocking on a double-lock is explicitly allowed behavior in order to

2286 avoid requiring more overhead in the basic mechanism than is absolutely

2287 necessary."

2288 Add to the end of the RATIONALE section:

2289 "For further rationale on the XSI extended mutex types please

2290 see the Rationale Volume."

Technical Corrigendum 2

2291 **Change Number:** XSH/TC2/D6/99 [XSH ERN 60]

2292 On Page: 1088 Line: 34211-34214 Section: pthread_mutex_timedlock (2003 Ed.)

2293 On Page: 1084 Line: 34038-34041 Section: pthread_mutex_timedlock (2001 Ed.)

2294 In the DESCRIPTION section (last paragraph)

2295 Margin mark the following text TPI and shade:

2296 "As a consequence of the priority inheritance rules (for mutexes
2297 initialized with the PRIO_INHERIT protocol), if a timed mutex wait
2298 is terminated because its timeout expires, the priority of the
2299 owner of the mutex shall be adjusted as necessary to reflect the fact that
2300 this thread is no longer among the threads waiting for the mutex."

2301 **Change Number:** XSH/TC2/D6/100 [XSH ERN 65]

2302 On Page: 1089 Line: 34231 Section: pthread_mutex_timedlock (2003 Ed.)

2303 On Page: 1085 Line: 34058 Section: pthread_mutex_timedlock (2001 Ed.)

2304 In the ERRORS section

2305 Change From:

2306 "[EDEADLK] The current thread already owns the mutex."

2307 To:

2308 "[EDEADLK] A deadlock condition was detected or the current thread
2309 already owns the mutex."

2310 **Change Number:** XSH/TC2/D6/101 [XSH ERN 65]

2311 On Page: 1114 Line: 34988 Section: pthread_rwlock_rdlock (2003 Ed.)

2312 On Page: 1110 Line: 34811 Section: pthread_rwlock_rdlock (2001 Ed.)

2313 In the ERRORS section

2314 Change From:

2315 "[EDEADLK] The current thread already owns the read-write lock for
2316 writing."

2317 To:

2318 "[EDEADLK] A deadlock condition was detected or the current thread
2319 already owns the read-write lock for writing."

2320 **Change Number:** XSH/TC2/D6/102 [XSH ERN 65]

2321 On Page: 1115 Line: 35054 Section: pthread_rwlock_timedrdlock (2003 Ed.)

2322 On Page: 1111 Line: 34877 Section: pthread_rwlock_timedrdlock (2003 Ed.)

2323 In the ERRORS section

2324 Change From:

2325 "[EDEADLK] The calling thread already holds a write lock on rwlock."

2326 To:

2327 "[EDEADLK] A deadlock condition was detected or the calling thread
2328 already holds a write lock on rwlock."

2329 **Change Number:** XSH/TC2/D6/103 [XSH ERN 65]

2330 On Page: 1117 Line: 35112 Section: pthread_rwlock_timedwrlock (2003 Ed.)

2331 On Page: 1113 Line: 34935 Section: pthread_rwlock_timedwrlock (2001 Ed.)

2332 In the ERRORS section

2333 Change From:

2334 "[EDEADLK] The calling thread already holds the rwlock."

2335 To:

2336 "[EDEADLK] A deadlock condition was detected or the calling thread
2337 already holds the rwlock."

2338 **Change Number:** XSH/TC2/D6/104 [XSH ERN 65]

2339 On Page: 1120 Line: 35178 Section: pthread_rwlock_trywrlock (2003 Ed.)

2340 On Page: 1116 Line: 35001 Section: pthread_rwlock_trywrlock (2001 Ed.)

2341 In the ERRORS section

2342 Change From:

2343 "[EDEADLK] The current thread already owns the read-write lock for
2344 writing or reading."

2345 To:

2346 "[EDEADLK] A deadlock condition was detected or the current thread
2347 already owns the read-write lock for writing or reading."

2348 **Change Number:** XSH/TC2/D6/105 [XSH ERN 30]

2349 On Page: 1139-1140 Line: 35589,35615 Section: pthread_sigmask (2003 Ed.)

2350 On Page: 1135-1136 Line: 35412,35438 Section: pthread_sigmask (2001 Ed.)

2351 In the DESCRIPTION section, line 35589 (2003 Ed.); line 35412 (2001 Ed.)

2352 Change From:

2353 "If set is a null pointer, the value of the argument how is not
2354 significant and the process signal mask shall be unchanged; thus the
2355 call can be used to enquire about currently blocked signals."

2356 To:

2357 "If set is a null pointer, the value of the argument how is not
2358 significant and the thread signal mask shall be unchanged; thus the call
2359 can be used to enquire about currently blocked signals."

2360 In the RATIONALE section, line 35615 (2003 Ed.); line 35438 (2001 Ed.)

2361 Change From:

2362 "When a process signal mask is changed in a signal-catching function that
2363 is installed by sigaction(), the restoration of the signal mask
2364 on return from the signal-catching function overrides that change
2365 (see sigaction())."

2366 To:

2367 "When a thread's signal mask is changed in a signal-catching function
2368 that is installed by sigaction(), the restoration of the signal mask
2369 on return from the signal-catching function overrides that change
2370 (see sigaction())."

Technical Corrigendum 2

2371 **Change Number:** XSH/TC2/D6/106 [XSH ERN 114]

2372 On Page: 1140 Line: 35611 Section: pthread_sigmask (2003 Ed.)

2373 On Page: 1136 Line: 35434 Section: pthread_sigmask (2001 Ed.)

2374 In the EXAMPLES section

2375 Change From:

2376 "None."

2377 To:

2378 "Signalling in multi-threaded process."

2379 This example shows the use of pthread_sigmask() in order to deal with
2380 signals in a multi-threaded process. It provides a fairly general
2381 framework that could be easily adapted/extended by the reader.

```
2382 #include <stdio.h>
2383 #include <stdlib.h>
2384 #include <pthread.h>
2385 #include <signal.h>
2386 #include <string.h>
2387 #include <errno.h>
2388 ...

2389     static sigset_t    signal_mask; /* signals to block          */

2390     int main (int argc, char *argv[])
2391     {
2392         pthread_t    sig_thr_id;      /* signal handler thread ID */
2393         int          rc;              /* return code              */

2394         sigemptyset (&signal_mask);
2395         sigaddset (&signal_mask, SIGINT);
2396         sigaddset (&signal_mask, SIGTERM);
2397         rc = pthread_sigmask (SIG_BLOCK, &signal_mask, NULL);
2398         if (rc != 0) {
2399             /* handle error */
2400             ...
2401         }
2402         /* any newly created threads inherit the signal mask */

2403         rc = pthread_create (&sig_thr_id, NULL, signal_thread, NULL);
2404         if (rc != 0) {
2405             /* handle error */
2406             ...
2407         }

2408         /* APPLICATION CODE */
2409         ...
2410     }

2411     void *signal_thread (void *arg)
2412     {
2413         int          sig_caught;      /* signal caught            */
2414         int          rc;              /* returned code            */

2415         rc = sigwait (&signal_mask, &sig_caught);
2416         if (rc != 0) {
2417             /* handle error */
2418         }
2419         switch (sig_caught)
2420         {
2421             case SIGINT:      /* process SIGINT */
2422                 ...
2423                 break;
2424             case SIGTERM:    /* process SIGTERM */
2425                 ...
2426                 break;
2427             default:         /* should normally not happen */
2428                 fprintf (stderr, "\nUnexpected signal %d\n", sig_caught);
```

```

2429         break;
2430     }
2431 }

```

2432 **Change Number:** XSH/TC2/D6/107 [XSH ERN 65]

2433 On Page: 1143 Line: 35723 Section: pthread_spin_lock (2003 Ed.)
 2434 On Page: 1139 Line: 35546 Section: pthread_spin_lock (2001 Ed.)

2435 In the ERRORS section

2436 Change From:

2437 "[EDEADLK] The calling thread already holds the lock."

2438 To:

2439 "[EDEADLK] A deadlock condition was detected or the calling thread
 2440 already holds the lock."

2441 **Change Number:** XSH/TC2/D6/108 [XSH ERN 43]

2442 On Page: 1174 Line: 36533 Section: read (2003 Ed.)
 2443 On Page: 1169 Line: 36335 Section: read (2001 Ed.)

2444 In the ERRORS section

2445 Change From:

2446 "[EAGAIN] The O_NONBLOCK flag is set for the file descriptor and the
 2447 process would be delayed."

2448 To:

2449 "[EAGAIN] The O_NONBLOCK flag is set for the file descriptor and the
 2450 thread would be delayed."

2451 **Change Number:** XSH/TC2/D6/109 [XSH ERN 165]

2452 On Page: 1177 Line: 36652-36654 Section: read (2003 Ed.)
 2453 On Page: 1172 Line: 36454-36456 Section: read (2001 Ed.)

2454 In the RATIONALE section

2455 Change From:

2456 "IEEE Std 1003.1-2001 does not specify when an implementation that
 2457 buffers read()s actually moves the data into the user-supplied buffer,
 2458 so an implementation may chose to do this at the latest possible moment."

2459 To:

2460 "IEEE Std 1003.1-2001 does not specify when an implementation that
 2461 buffers read()s actually moves the data into the user-supplied buffer,
 2462 so an implementation may chose to do this at the latest possible moment."

2463 **Change Number:** XSH/TC2/D6/110 [XSH ERN 111]

2464 On Page: 1190 Line: 37082,37091 Section: realpath() (2003 Ed.)
 2465 On Page: 1185 Line: 36862,36872 Section: realpath() (2001 Ed.)

2466 Change From:

2467 "[ELOOP] A loop exists in symbolic links encountered during resolution
 2468 of the path argument."

2469 To:

2470 "[ELOOP] A loop exists in symbolic links encountered during resolution
 2471 of the file_name argument."

2472 Change From:

2473 "[ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
 2474 resolution of the path argument."

2475 To:

2476 "[ELOOP] More than {SYMLOOP_MAX} symbolic links were encountered during
 2477 resolution of the file_name argument."

Technical Corrigendum 2

2478 Rationale:
2479 These ERROR descriptions erroneously referred to a non-existent
2480 path argument.

2481 **Change Number:** XSH/TC2/D6/111 [XSH Errata 2003-1]

2482 On Page: 1247 Line: 38887 Section: select (2003 Ed.)
2483 On Page: 1242 Line: 38658 Section: select (2001 Ed.)

2484 In the SYNOPSIS section
2485 Change from:
2486 "#include <sys/time.h>"

2487 To:
2488 "#include <sys/select.h>"

2489 Rationale: This is an editorial correction to the pointer page
2490 to match the actual manual page.

2491 **Change Number:** XSH/TC2/D6/112 [Errata Item 1]

2492 On Page: 1247 Line: 38887 Section: select (2003 Ed.)
2493 On Page: 1242 Line: 38656 Section: select (2001 Ed.)

2494 In the SYNOPSIS section

2495 Change From:
2496 "#include <sys/time.h>"

2497 To:
2498 "#include <sys/select.h>"

2499 Rationale: This corrects the pointer page to be consistent with the
2500 pselect() manual page.

2501 **Change Number:** XSH/TC2/D6/113 [XSH ERN 66]

2502 On Page: 1248 Line: 38915 Section: sem_close (2003 Ed.)
2503 On Page: 1243 Line: 38686 Section: sem_close (2001 Ed.)

2504 In the ERRORS section
2505 Change From:
2506 "The sem_close() function shall fail if:"

2507 To:
2508 "The sem_close() function may fail if:"

2509 **Change Number:** XSH/TC2/D6/114 [XSH ERN 67]

2510 On Page: 1249 Line: 38952-38955 Section: sem_destroy (2003 Ed.)
2511 On Page: 1244 Line: 38723-38726 Section: sem_destroy (2001 Ed.)

2512 In the ERRORS section
2513 Change From:
2514 "The sem_destroy() function shall fail if:

2515 [EINVAL] The sem argument is not a valid semaphore.

2516 The sem_destroy() function may fail if:

2517 [EBUSY] There are currently processes blocked on the semaphore."

2518 To:
2519 "The sem_destroy() function may fail if:

2520 [EINVAL] The sem argument is not a valid semaphore.

2521 [EBUSY] There are currently processes blocked on the semaphore."

2522 **Change Number:** XSH/TC2/D6/115 [XSH ERN 68]

2523 On Page: 1250 Line: 38993 Section: sem_getvalue (2003 Ed.)

2524 On Page: 1245 Line: 38764 Section: sem_getvalue (2001 Ed.)

2525 In the ERRORS section

2526 Change From:

2527 "The sem_getvalue() function shall fail if:"

2528 To:

2529 "The sem_getvalue() function may fail if:"

2530 **Change Number:** XSH/TC2/D6/116 [XSH ERN 102]

2531 On Page: 1252 Line: 39025-39036 Section: sem_init (2003 Ed.)

2532 On Page: 1246 Line: 38795-38806 Section: sem_init (2001 Ed.)

2533 In the DESCRIPTION section

2534 Change From:

2535 "The sem_init() function shall initialize the unnamed semaphore
2536 referred to by sem. The value of the initialized semaphore shall be
2537 value. Following a successful call to sem_init(), the semaphore may be
2538 used in subsequent calls to sem_wait(), sem_trywait(), sem_post(), and
2539 sem_destroy(). This semaphore shall remain usable until the semaphore
2540 is destroyed.

2541 If the pshared argument has a non-zero value, then the semaphore is shared
2542 between processes; in this case, any process that can access the semaphore
2543 sem can use sem for performing sem_wait(), sem_trywait(), sem_post(),
2544 and sem_destroy() operations.

2545 Only sem itself may be used for performing synchronization. The result of
2546 referring to copies of sem in calls to sem_wait(), sem_trywait(), sem_post(),
2547 and sem_destroy() is undefined.

2548 If the pshared argument is zero, then the semaphore is shared between
2549 threads of the process; any thread in this process can use sem for
2550 performing sem_wait(), sem_trywait(), sem_post(), and sem_destroy()
2551 operations. "

2552 To:

2553 "The sem_init() function shall initialize the unnamed semaphore
2554 referred to by sem. The value of the initialized semaphore shall be
2555 value. Following a successful call to sem_init(), the semaphore may
2556 be used in subsequent calls to sem_wait(), [TMO]sem_timedwait(), [TMO]
2557 sem_trywait(), sem_post(), and sem_destroy(). This semaphore shall
2558 remain usable until the semaphore is destroyed.

2559 If the pshared argument has a non-zero value, then the semaphore
2560 is shared between processes; in this case, any process that can
2561 access the semaphore sem can use sem for performing sem_wait(),
2562 [TMO]sem_timedwait(), [TMO]sem_trywait(), sem_post(), and
2563 sem_destroy() operations.

2564 Only sem itself may be used for performing synchronization. The
2565 result of referring to copies of sem in calls to sem_wait(),
2566 [TMO]sem_timedwait(), [TMO]sem_trywait(), sem_post(), and sem_destroy()
2567 is undefined.

2568 If the pshared argument is zero, then the semaphore is shared between
2569 threads of the process; any thread in this process can use sem for
2570 performing sem_wait(), [TMO]sem_timedwait(), [TMO]sem_trywait(),
2571 sem_post(), and sem_destroy() operations."

2572 Rationale: The sem_timedwait() function is added to the DESCRIPTION
2573 for alignment with 1003.1d-1999.

Technical Corrigendum 2

2574 **Change Number:** XSH/TC2/D6/117 [XSH ERN 103]

2575 On Page: 1254 Line: 39079-39080 Section: sem_open (2003 Ed.)

2576 On Page: 1248 Line: 38849-38850 Section: sem_open (2001 Ed.)

2577 In the DESCRIPTION section

2578 Change From:

2579 "This semaphore may be used in subsequent calls to sem_wait(),

2580 sem_trywait(), sem_post(), and sem_close(). "

2581 To:

2582 "This semaphore may be used in subsequent calls to sem_wait(),

2583 [TMO]sem_timedwait(),[/TMO] sem_trywait(), sem_post(), and sem_close(). "

2584 Rationale: The sem_timedwait() function is added to the DESCRIPTION

2585 for alignment with 1003.1d-1999.

2586 **Change Number:** XSH/TC2/D6/118 [XSH ERN 97]

2587 On Page: 1254-1255 Line: 39115-39117 Section: sem_open (2003 Ed.)

2588 On Page: 1248-1249 Line: 38885-38887 Section: sem_open (2001 Ed.)

2589 In the DESCRIPTION section

2590 Change From:

2591 "If a process makes multiple successful calls to sem_open() with the same

2592 value for name, the same semaphore address shall be returned for each such

2593 successful call, provided that there have been no calls to sem_unlink()

2594 for this semaphore."

2595 To:

2596 "If a process makes multiple successful calls to sem_open() with the same

2597 value for name, the same semaphore address shall be returned for each such

2598 successful call, provided that there have been no calls to sem_unlink()

2599 for this semaphore and at least one previous successful sem_open()

2600 call for this semaphore has not been matched with sem_close() call."

2601 **Change Number:** XSH/TC2/D6/119 [XSH ERN 69]

2602 On Page: 1257 Line: 39192 Section: sem_post (2003 Ed.)

2603 On Page: 1251 Line: 38962 Section: sem_post (2001 Ed.)

2604 In the ERRORS section

2605 Change From:

2606 "The sem_post() function shall fail if:"

2607 To:

2608 "The sem_post() function may fail if:"

2609 **Change Number:** XSH/TC2/D6/120 [XSH ERN 70]

2610 On Page: 1259 Line: 39247 Section: sem_timedwait (2001 Ed.)

2611 On Page: 1253 Line: 39017 Section: sem_timedwait (2001 Ed.)

2612 In the ERRORS section

2613 Move the following text on line 39247 from the "shall fail"

2614 to the "may fail" section:

2615 "[EINVAL] The sem argument does not refer to a valid semaphore."

2616 **Change Number:** XSH/TC2/D6/121 [XSH ERN 71]

2617 On Page: 1261 Line: 39298 Section: sem_trywait (2003 Ed.)

2618 On Page: 1255 Line: 39068 Section: sem_trywait (2001 Ed.)

2619 In the ERRORS section

2620 Move the following text on line 39298 from the "shall fail" to the

2621 "may fail" section:

2622 "[EINVAL] The sem argument does not refer to a valid semaphore."

2623 **Change Number:** XSH/TC2/D6/122 [XSH ERN 133]

2624 On Page: 1269 Line: 39510 Section: semget (2003 Ed.)

2625 On Page: 1263 Line: 39280 Section: semget (2001 Ed.)

2626 In the DESCRIPTION section

2627 Change From:

2628 "The data structure associated with each semaphore in the set shall not
2629 be initialized."

2630 To:

2631 "The data structure associated with each semaphore in the set need not
2632 be initialized."

2633 **Change Number:** XSH/TC2/D6/123 [XSH ERN 44]

2634 On Page: 1290 Line: 40250-40251 Section: seteuid (2003 Ed.)

2635 On Page: 1284 Line: 40018-40019 Section: seteuid (2001 Ed.)

2636 In the RATIONALE section

2637 Change From:

2638 "[EPERM] The process does not have appropriate privileges and uid does
2639 not match the real group ID or the saved set-group-ID."

2640 To:

2641 "[EPERM] The process does not have appropriate privileges and uid does
2642 not match the real user ID or the saved set-user-ID."

2643 Rationale. This is a correction of an editorial error.

2644 **Change Number:** XSH/TC2/D6/124 [XSH ERN 92]

2645 On Page: 1299 Line: 40449,40456 Section: setlocale (2003 Ed.)

2646 On Page: 1293 Line: 40217,40224 Section: setlocale (2001 Ed.)

2647 In the DESCRIPTION section

2648 Change From:

2649 " "" Specifies an implementation-defined native environment. [CX]
2650 This corresponds to the value of the associated environment
2651 variables, LC_* and LANG ; see the Base Definitions volume of IEEE Std
2652 1003.1-2001, Chapter 7, Locale and the Base Definitions volume of IEEE
2653 Std 1003.1-2001, Chapter 8, Environment Variables. [/CX] "

2654 To:

2655 " "" Specifies an implementation-defined native environment. [CX]
2656 The determination of the name of the new locale for the specified
2657 category depends on the value of the associated environment
2658 variables, LC_* and LANG ; see the Base Definitions volume of IEEE Std
2659 1003.1-2001, Chapter 7, Locale and the Base Definitions volume of IEEE
2660 Std 1003.1-2001, Chapter 8, Environment Variables. [/CX] "

2661 In the DESCRIPTION section

2662 Insert before the last paragraph:

2663 "[CX] Setting all of the categories of the locale of the process is similar
2664 to successively setting each individual category of the locale of the
2665 process, except that all error checking is done before any actions
2666 are performed. To set all the categories of the locale of the process,
2667 setlocale() is invoked as:

2668 setlocale(LC_ALL, "");

2669 In this case, setlocale() shall first verify that the values of all
2670 the environment variables it needs according to the precedence rules,
2671 described in the Base Definitions Volume of IEEE Std 1003.1-8, Chapter 8,
2672 Environment Variables, indicate supported locales. If the value of any of

Technical Corrigendum 2

2673 these environment-variable searches yields a locale that is not supported
2674 (and nonnull), setlocale() shall return a null pointer and the
2675 locale of the process shall not be changed. If all environment variables name
2676 supported locales, setlocale() shall proceed as if it had been called
2677 for each category, using the appropriate value from the associated
2678 environment variable or from the implementation-defined default if there
2679 is no such value. [/CX]"

2680 **Change Number:** XSH/TC2/D6/125 [XSH ERN 31,32]

2681 On Page: 1319 Line: 40905-40910 Section: setsockopt (2003 Ed.)
2682 On Page: 1313 Line: 40671-40674 Section: setsockopt (2001 Ed.)

2683 Change From:

2684 "If SO_LINGER is set, the system shall block the process during close()
2685 until it can transmit the data or until the time expires. If
2686 SO_LINGER is not specified, and close() is issued, the system
2687 handles the call in a way that allows the process to continue as
2688 quickly as possible."

2689 To:

2690 "If SO_LINGER is set, the system shall block the calling thread during close()
2691 until it can transmit the data or until the time expires. If
2692 SO_LINGER is not specified, and close() is issued, the system
2693 handles the call in a way that allows the calling thread to continue as
2694 quickly as possible."

2695 **Change Number:** XSH/TC2/D6/126 [XSH ERN 129]

2696 On Page: 1330 Line: 41250 Section: shm_open (2003 Ed.)
2697 On Page: 1324 Line: 41014 Section: shm_open (2001 Ed.)

2698 In the EXAMPLES section

2699 Change From:

2700 "None."

2701 To:

2702 "Creating and mapping a shared memory object

2703 The following code segment demonstrates the use of shm_open()
2704 to create a shared memory object which is then sized using
2705 ftruncate() before being mapped into the process address
2706 space using mmap():

```
2707 #include <unistd.h>
2708 #include <sys/mman.h>
2709 ...
```

```
2710 #define MAX_LEN 10000
2711 struct region {          /* Defines "structure" of shared memory */
2712     int len;
2713     char buf[MAX_LEN];
2714 };
2715 struct region *rptr;
2716 int fd;
```

2717 /* Create shared memory object and set its size */

```
2718 fd = shm_open("/myregion", O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);
2719 if (fd == -1)
2720     /* Handle error */;
```

```
2721 if (ftruncate(fd, sizeof(struct region)) == -1)
2722     /* Handle error */;
```

2723 /* Map shared memory object */

```
2724 rptr = mmap(NULL, sizeof(struct region),
2725             PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
```

2726 if (rptr == MAP_FAILED)
2727 /* Handle error */;

2728 /* Now we can refer to mapped region using fields of rptr,
2729 for example, rptr->len */
2730 ..."

2731 **Change Number:** XSH/TC2/D6/127 [XSH ERN 93]

2732 On Page: 1344 Line: 41688-41696 Section: sigaction (2003 Ed.)

2733 On Page: 1338 Line: 41449-41458 Section: sigaction (2001 Ed.)

2734 In the DESCRIPTION

2735 Change From:

2736 "If the SA_SIGINFO flag (see below) is cleared in the sa_flags field of
2737 the sigaction structure, the sa_handler field identifies the
2738 action to be associated with the specified signal. [XSI|RTS] If the SA_SIGINFO
2739 flag is set in the sa_flags field, and the implementation supports the
2740 Realtime Signals Extension option or the XSI Extension option,
2741 the sa_sigaction field specifies a signal-catching function. [/XSI|RTS]
2742 If the SA_SIGINFO bit is cleared and the sa_handler field specifies a
2743 signal-catching function, or if the SA_SIGINFO bit is set, the sa_mask
2744 field identifies a set of signals that shall be added to the signal
2745 mask of the thread before the signal-catching function is invoked. If
2746 the sa_handler field specifies a signal-catching function, the sa_mask
2747 field identifies a set of signals that shall be added to the process
2748 signal mask before the signal-catching function is invoked."

2749 To:

2750 "If the SA_SIGINFO flag (see below) is cleared in the sa_flags field of
2751 the sigaction structure, the sa_handler field identifies the
2752 action to be associated with the specified signal. [XSI|RTS] If the SA_SIGINFO
2753 flag is set in the sa_flags field, and the implementation supports the
2754 Realtime Signals Extension option or the XSI Extension option,
2755 the sa_sigaction field specifies a signal-catching function. [/XSI|RTS]

2756 Rationale:

2757 The removed text repeats requirements stated later in the
2758 DESCRIPTION section on page 1346 line 41765 (2003 Ed.);
2759 page 1340 line 41527 (2001 Ed.).

2760 **Change Number:** XSH/TC2/D6/128 [XSH ERN 33,34,35,36,37]

2761 On Page: 1345-1348 Line: 41736,41761,41843 Section: sigaction (2003 Ed.)

2762 On Page: 1339-1342 Line: 41498,41523,41605 Section: sigaction (2001 Ed.)

2763 In the DESCRIPTION of SA_SIGINFO, line 41736 (2003 Ed.); line 41498 (2001 Ed.)

2764 Change From:

2765 "The second argument shall point to an object of type siginfo_t explaining
2766 the reason why the signal was generated; the third argument can be cast
2767 to a pointer to an object of type ucontext_t to refer to the receiving
2768 process context that was interrupted when the signal was delivered."

2769 To:

2770 "The second argument shall point to an object of type siginfo_t explaining
2771 the reason why the signal was generated; the third argument can be cast
2772 to a pointer to an object of type ucontext_t to refer to the receiving
2773 thread's context that was interrupted when the signal was delivered."

2774 In the DESCRIPTION of SA_NODEFER, line 41761 (2003 Ed.); line 41523 (2001 Ed.)

2775 Change From:

2776 "If set and sig is caught, sig shall not be added to the process
2777 signal mask on entry to the signal handler unless it is included in
2778 sa_mask. Otherwise, sig shall always be added to the process signal mask
2779 on entry to the signal handler."

2780 To:

2781 "If set and sig is caught, sig shall not be added to the thread's

Technical Corrigendum 2

2782 signal mask on entry to the signal handler unless it is included in
2783 sa_mask. Otherwise, sig shall always be added to the thread's signal mask
2784 on entry to the signal handler."

2785 In the APPLICATION USAGE , line 41843 (2003 Ed.); line 41605 (2001 Ed.)
2786 Change From:
2787 "When the signal handler returns, the receiving process resumes execution
2788 at the point it was interrupted unless the signal handler makes other
2789 arrangements. If longjmp() or _longjmp() is used to leave the signal
2790 handler, then the signal mask must be explicitly restored by the process."

2791 To:
2792 "When the signal handler returns, the receiving thread resumes execution
2793 at the point it was interrupted unless the signal handler makes other
2794 arrangements. If longjmp() or _longjmp() is used to leave the signal
2795 handler, then the signal mask must be explicitly restored."

2796 **Change Number:** XSH/TC2/D6/129 [XSH ERN 130]

2797 On Page: 1347 Line: 41807 Section: sigaction (2003 Ed.)
2798 On Page: 1341 Line: 41569 Section: sigaction (2001 Ed.)

2799 In the EXAMPLES section
2800 Change From:
2801 "None."

2802 To:
2803 "Establishing a signal handler

2804 The following example demonstrates the use of sigaction to
2805 establish a handler for the SIGINT signal.

```
2806 #include <signal.h>

2807 static void handler(int signum)
2808 {
2809     /* Take appropriate actions for signal delivery */
2810 }

2811 int main()
2812 {
2813     struct sigaction sa;

2814     sa.sa_handler = handler;
2815     sigemptyset(&sa.sa_mask);
2816     sa.sa_flags = SA_RESTART;      /* Restart functions if
2817                                   interrupted by handler */
2818     if (sigaction(SIGINT, &sa, NULL) == -1)
2819         /* Handle error */;

2820     /* Further code */
2821 }
```

2822 **Change Number:** XSH/TC2/D6/130 [XSH ERN 38]

2823 On Page: 1377 Line: 42765-42766 Section: sigtimedwait (2003 Ed.)
2824 On Page: 1371 Line: 42520-42521 Section: sigtimedwait (2001 Ed.)

2825 Change From:
2826 "An implementation only checks for this error if no signal is pending in
2827 set and it is necessary to wait."

2828 To:
2829 "An implementation should only check for this error if no signal is pending in
2830 set and it is necessary to wait."

2831 Rationale: This restores the wording from the original base document.

2832 **Change Number:** XSH/TC2/D6/131 [XSH ERN 94]

2833 On Page: 1380 Line: 42881-42883 Section: sigwait (2003 Edition)
 2834 On Page: 1374 Line: 42636-42638 Section: sigwait (2001 Edition)

2835 In the DESCRIPTION section

2836 Change From:

2837 "If more than one thread is using sigwait() to wait for the same signal,
 2838 no more than one of these threads shall return from sigwait()
 2839 with the signal number. Which thread returns from sigwait() if
 2840 more than a single thread is waiting is unspecified."

2841 To:

2842 "If more than one thread is using sigwait() to wait for the same signal,
 2843 no more than one of these threads shall return from sigwait() with the
 2844 signal number. If more than a single thread is blocked in sigwait() for a
 2845 signal when that signal is generated for the process, it is unspecified
 2846 which of the waiting threads returns from sigwait(). If the signal
 2847 is generated for a specific thread, as by pthread_kill(), only that thread
 2848 shall return."

2849 **Change Number:** XSH/TC2/D6/132 [XSH ERN 38]

2850 On Page: 1388 Line: 43101-43103 Section: sleep (2003 Ed.)
 2851 On Page: 1382 Line: 42856-42858 Section: sleep (2001 Ed.)

2852 In the RATIONALE section

2853 Change From:

2854 "One is to use the alarm() function to schedule a SIGALRM signal and
 2855 then suspend the process waiting for that signal."

2856 To:

2857 "One is to use the alarm() function to schedule a SIGALRM signal and
 2858 then suspend the calling thread waiting for that signal."

Technical Corrigendum 2

2859 **Change Number:** XSH/TC2/D6/133 [XSH ERN 131]

2860 On Page: 1443 Line: 44720 Section: `strptime` (2003 Ed.)

2861 On Page: 1436 Line: 44467 Section: `strptime` (2001 Ed.)

2862 In the EXAMPLES section

2863 Change From:

2864 "None."

2865 To:

2866 "Convert a data-plus-time string to broken-down time and then into seconds

2867 The following example demonstrates the use `strptime()` to convert a

2868 string into broken-down time. The broken-down time is then converted

2869 into seconds since the Epoch using `mktime()`.

2870 `#include <time.h>`

2871 `...`

2872 `struct tm tm;`

2873 `time_t t;`

2874 `if (strptime("6 Dec 2001 12:33:45", "%d %b %Y %H:%M:%S", &tm) == NULL)`

2875 `/* Handle error */;`

2876 `printf("year: %d; month: %d; day: %d;\n",`

2877 `tm.tm_year, tm.tm_mon, tm.tm_mday);`

2878 `printf("hour: %d; minute: %d; second: %d\n",`

2879 `tm.tm_hour, tm.tm_min, tm.tm_sec);`

2880 `printf("week day: %d; year day: %d\n", tm.tm_wday, tm.tm_yday);`

2881 `tm.tm_isdst = -1; /* Not set by strptime(); tells mktime()`

2882 `to determine if daylight saving time`

2883 `is in effect */`

2884 `t = mktime(&tm);`

2885 `if (t == -1)`

2886 `/* Handle error */;`

2887 `printf("seconds since the Epoch: %ld\n", (long) t);"`

2888 **Change Number:** XSH/TC2/D6/134 [XSH ERN 105]

2889 On Page: 1474 Line: 45617 Section: `sysconf` (2003 Ed.)

2890 (not present in the 2001 Edition)

2891 In the DESCRIPTION section

2892 Delete the following entry from the table

2893 `_POSIX_SYMLINK_MAX` `_SC_SYMLINK_MAX`

2894 Rationale:

2895 There is an entry for `SYMLINK_MAX` in this table, there thus does not

2896 need to be an entry for `_POSIX_SYMLINK_MAX` (the minimum maximum) here.

2897 This corrects an error in Technical Corrigendum 1.

2898 **Change Number:** XSH/TC2/D6/135 [XSH ERN 40,61,62]

2899 On Page: 1474-1476 Line: 45593,45603,45645,45691,45711-45718 Section: `sysconf` (2003 Ed.)

2900 On Page: 1467-1469 Line: 45341,45350,45390,45434,45452-45459 Section: `sysconf` (2001 Ed.)

2901 In the DESCRIPTION, line 45593 (2003 Ed.); line 45341 (2001 Ed.)

2902 Delete the following entry from the table

2903 `_POSIX_FILE_LOCKING` `_SC_FILE_LOCKING`

2904 In the DESCRIPTION, line 45603 (2003 Ed.); line 45350 (2001 Ed.)

2905 Delete the following entry from the table

2906 `_POSIX_MULTI_PROCESS _SC_MULTI_PROCESS`

2907 In the DESCRIPTION, line 45645 (2003 Ed.); line 45390 (2001 Ed.)

2908 Delete the following entry from the table

2909 `_POSIX2_C_VERSION _SC_2_C_VERSION`

2910 In the DESCRIPTION, line 45691 (2003 Ed.); line 45434 (2001 Ed.)

2911 Delete the following entry from the table

2912 `_XOPEN_XCU_VERSION _SC_XOPEN_XCU_VERSION`

2913 In the APPLICATION USAGE, lines 45711-45718 (2003 Ed.); lines 45452-45459

2914 (2001 Ed.)

2915 Delete the paragraph

2916 "If the value of `sysconf(_SC_2_VERSION)` is not equal to the value of

2917 the `_POSIX2_VERSION` symbolic constant, the utilities available via

2918 `system()` or `popen()` might not behave as described in the Shell and

2919 Utilities volume of IEEE Std 1003.1-2001. This would mean that the

2920 application is not running in an environment that conforms to the Shell

2921 and Utilities volume of IEEE Std 1003.1-2001. Some applications might be

2922 able to deal with this, others might not. However, the functions defined

2923 in this volume of IEEE Std 1003.1-2001 continue to operate as specified,

2924 even if `sysconf(_SC_2_VERSION)` reports that the utilities no longer

2925 perform as specified."

2926 Rationale: `_POSIX_FILE_LOCKING` and `_POSIX_MULTI_PROCESS` were erroneous

2927 entries. `_POSIX2_C_VERSION` was removed by P1003.1a/P1003.2b and should

2928 have been removed here too. `_XOPEN_XCU_VERSION` should have been removed

2929 as per The Open Group Base Working Group resolution bwg2001-006.

2930 **Change Number: XSH/TC2/D6/136** [XBD ERN 27]

2931 On Page: 1474 Line: 45616,45632,45635,45701,45718 Section: `sysconf` (2003 Ed.)

2932 On Page: 1467 Line: 45362,45377,45380,45442,45459 Section: `sysconf` (2001 Ed.)

2933 In the DESCRIPTION section:

2934 Add into the table after the `_POSIX_SPORADIC_SERVER` entry:

2935 `_POSIX_SS_REPL_MAX _SC_SS_REPL_MAX`

2936 Add into the table after the `_POSIX_TRACE_EVENT_FILTER` entry:

2937 `_POSIX_TRACE_EVENT_NAME_MAX _SC_TRACE_EVENT_NAME_MAX`

2938 Add into the table after the `_POSIX_TRACE_MEMORY_OBJECTS` entry:

2939 `_POSIX_TRACE_NAME_MAX _SC_TRACE_NAME_MAX`

2940 `_POSIX_TRACE_SYS_MAX _SC_TRACE_SYS_MAX`

2941 `_POSIX_TRACE_USER_EVENT_MAX _SC_TRACE_USER_EVENT_MAX`

2942 Add to the end of the RETURN VALUE section as a new paragraph:

2943 "If the variable corresponding to name is dependent on an unsupported

2944 option the results are unspecified."

2945 Add to the end of the APPLICATION USAGE section:

2946 "Application writers should check whether an option, such as `_POSIX_TRACE`

2947 is supported, prior to obtaining and using values for related

2948 variables such as `_POSIX_TRACE_NAME_MAX`."

Technical Corrigendum 2

2949 **Change Number:** XSH/TC2/D6/137 [XSH ERN 107]

2950 On Page: 1475 Line: 45659 Section: sysconf (2003 Ed.)

2951 On Page: 1468 Line: 45403 Section: sysconf (2001 Ed.)

2952 In the DESCRIPTION section

2953 Delete the following entry from the table:

2954 "_REGEX_VERSION" "_SC_REGEX_VERSION"

2955 Rationale:

2956 The text in P1003.1a indicates that this was to be used for versioning

2957 information related to the _SC_REGEX_VERSION (sic)

2958 (.1a D17): "172 POSIX.2RE: The version of ISO/IEC 9945-2 that defines

2959 _POSIX2_VERSION to be the value indicated by the return value of

2960 sysconf(_SC_REGEX_VERSION)."

2961 Given that the approach for "_SC_2_C_VERSION" and "_SC_XOPEN_XCU_VERSION"

2962 has been to remove them since they are unnecessary this is removed

2963 for consistency.

2964 **Change Number:** XSH/TC2/D6/138 [XSH ERN 95]

2965 On Page: 1521,1522 Line: 47034,47050 Section: timer_create (2003 Ed.)

2966 On Page: 1514,1515 Line: 46762,46780 Section: timer_create (2001 Ed.)

2967 In the DESCRIPTION section

2968 Add as a new last paragraph, shaded and margin marked "TSA":

2969 "If evp->sigev_sigev_notify is SIGEV_THREAD and

2970 sev->sigev_notify_attributes is not NULL, if the attribute pointed to

2971 by sev->sigev_notify_attributes has a thread stack address specified

2972 by a call to pthread_attr_setstack() or pthread_attr_setstackaddr()

2973 the results are unspecified if the signal is generated more than once."

2974 In the APPLICATION USAGE section:

2975 Change From:

2976 "None."

2977 To:

2978 "If a timer is created which has evp->sigev_sigev_notify set to

2979 SIGEV_THREAD and the attribute pointed to by evp->sigev_notify_attributes

2980 has a thread stack address specified by a call to either

2981 pthread_attr_setstack() or pthread_attr_setstackaddr(), the memory

2982 dedicated as a thread stack can not be recovered. The reason for

2983 this is that the threads created in response to a timer expiration are

2984 created detached, or in an unspecified way if the thread attribute's

2985 detachstate is PTHREAD_CREATE_JOINABLE. In neither case is it valid to

2986 call pthread_join(), which makes it impossible to determine the lifetime

2987 of the created thread which thus means the stack memory can not

2988 be reused."

2989 **Change Number:** XSH/TC2/D6/139 [XSH ERN 74]

2990 On Page: 1524 Line: 47131 Section: timer_delete (2003 Ed.)

2991 On Page: 1517 Line: 46859 Section: timer_delete (2001 Ed.)

2992 In the ERRORS section

2993 Change From:

2994 "The timer_delete() function shall fail if:"

2995 To:

2996 "The timer_delete() function may fail if:"

2997 **Change Number:** XSH/TC2/D6/140 [XSH ERN 75]

2998 On Page: 1526 Line: 47211-47213 Section: timer_getoverrun (2003 Ed.)

2999 On Page: 1519 Line: 46939-46941 Section: timer_getoverrun (2001 Ed.)

3000 Change From:
 3001 "The timer_getoverrun(), timer_gettime(), and timer_settime() functions
 3002 shall fail if:

3003 [EINVAL] The timerid argument does not correspond to an ID returned by
 3004 timer_create() but not yet deleted by timer_delete()."

3005 To:
 3006 "The timer_getoverrun(), timer_gettime(), and timer_settime() functions
 3007 may fail if:

3008 [EINVAL] The timerid argument does not correspond to an ID returned by
 3009 timer_create() but not yet deleted by timer_delete()."

3010 Notes to Editors:
 3011 Move the revised text section after the current following [EINVAL] error
 3012 so that the "shall fail" case occurs before the "may fail" cases.

3013 **Change Number: XSH/TC2/D6/141** [XSH ERN 95]

3014 On Page: 1526 Line: 47217,47221 Section: timer_getoverrun (2003 Ed.)
 3015 On Page: 1519 Line: 46945,46949 Section: timer_getoverrun (2001 Ed.)

3016 In the ERRORS section
 3017 Add the following additional text

3018 "The timer_settime() function may fail if:

3019 [EINVAL] The it_interval member of value is not zero and the timer was
 3020 created with notification by creation of a new thread (sigev_sigev_notify
 3021 was SIGEV_THREAD) and a fixed stack address has been set in the thread
 3022 attribute pointed to by sigev_notify_attributes."

3023 In the APPLICATION USAGE section
 3024 Change From:
 3025 "None."

3026 To:
 3027 "Using fixed stack addresses is problematic when timer expiration is
 3028 signalled by the creation of a new thread. Since it can not be assumed
 3029 that the thread created for one expiration is finished before the next
 3030 expiration of the timer, it could happen that two threads use the same
 3031 memory as stack at the same time. This is invalid and produces
 3032 undefined results."

3033 **Change Number: XSH/TC2/D6/142** [XSH ERN 137]

3034 On Page: 1533 Line: 47433-47434 Section: tmpnam (2003 Ed.)
 3035 On Page: 1526 Line: 47161-47162 Section: tmpnam (2001 Ed.)

3036 In the DESCRIPTION section
 3037 Change From:
 3038 "The implementation shall behave as if no function defined in this volume
 3039 of IEEE Std 1003.1-2001 calls tmpnam()."

3040 To:
 3041 "The implementation shall behave as if no function defined in this volume
 3042 of IEEE Std 1003.1-2001 except tmpnam() calls tmpnam()."

3043 **Change Number: XSH/TC2/D6/143** [XSH ERN 52,98]

3044 On Page: 1555 Line: 48027-48028,48051 Section: umask (2003 Ed.)
 3045 On Page: 1548 Line: 47755-47756,47779 Section: umask (2001 Ed.)

3046 In the DESCRIPTION section
 3047 Change From:
 3048 "The process' file mode creation mask is used during open(), creat(),
 3049 mkdir(), and mkfifo() to turn off permission bits in the mode

Technical Corrigendum 2

3050 argument supplied."

3051 To:

3052 "The process' file mode creation mask is used during open(), creat(),
3053 mkdir(), mkfifo(), mknod(), mq_open() and sem_open() to turn off
3054 permission bits in the mode argument supplied."

3055 (Ed note:mq_open is shaded and margin marked MSG, sem_open is shaded
3056 and margin marked SEM, mknod is shaded and margin marked XSI)

3057 In the SEE ALSO section

3058 Add "mknod(), mq_open()" and "sem_open()" in alphabetical order.

3059 **Change Number:** XSH/TC2/D6/144 [XSH ERN 41]

3060 On Page: 1568 Line: 48450-48452,48465 Section: usleep (2003 Ed.)

3061 On Page: 1561 Line: 48178-48180,48193 Section: usleep (2001 Ed.)

3062 Change From:

3063 "It is also unspecified whether the SIGALRM signal is blocked, unless
3064 the process signal mask is restored as part of the environment."

3065 To:

3066 "It is also unspecified whether the SIGALRM signal is blocked, unless
3067 the thread's signal mask is restored as part of the environment."

3068 Add as a new final paragraph to the DESCRIPTION:

3069 "The usleep() function need not be reentrant. A function that is not
3070 required to be reentrant is not required to be thread-safe."

3071 **Change Number:** XSH/TC2/D6/145 [XSH ERN 116]

3072 On Page: 1592 Line: 49061 Section: wait (2003 Ed.)

3073 On Page: 1585 Line: 48789 Section: wait (2001 Ed.)

3074 In the EXAMPLES section

3075 Change From:

3076 "None."

3077 To:

3078 "Waiting for a child process and then checking its status

3079 The following example demonstrates the use of waitpid(), fork(), and the
3080 macros used to interpret the status value returned by waitpid() (and
3081 wait()). The code segment creates a child process which does some
3082 unspecified work. Meanwhile the parent loops performing calls to
3083 waitpid() to monitor the status of the child. The loop terminates when
3084 child termination is detected.

```
3085 #include <stdio.h>
3086 #include <stdlib.h>
3087 #include <unistd.h>
3088 #include <sys/wait.h>
3089 ...
```

```
3090 pid_t child_pid, wpid;
3091 int status;
```

```
3092 child_pid = fork();
3093 if (child_pid == -1) {          /* fork() failed */
3094     perror("fork");
3095     exit(EXIT_FAILURE);
3096 }
```

```
3097 if (child_pid == 0) {          /* This is the child */
3098     /* Child does some work and then terminates */
3099     ...
```

```

3100 } else {                                /* This is the parent */
3101     do {
3102         wpid = waitpid(child_pid, &status, WUNTRACED
3103 #ifdef WCONTINUED        /* Not all implementations support this */
3104                         | WCONTINUED
3105 #endif
3106                     );
3107         if (wpid == -1) {
3108             perror("waitpid");
3109             exit(EXIT_FAILURE);
3110         }
3111
3112         if (WIFEXITED(status)) {
3113             printf("child exited, status=%d\n", WEXITSTATUS(status));
3114
3115         } else if (WIFSIGNALED(status)) {
3116             printf("child killed (signal %d)\n", WTERMSIG(status));
3117
3118         } else if (WIFSTOPPED(status)) {
3119             printf("child stopped (signal %d)\n", WSTOPSIG(status));
3120
3121 #ifdef WIFCONTINUED        /* Not all implementations support this */
3122         } else if (WIFCONTINUED(status)) {
3123             printf("child continued\n");
3124         }
3125 #endif
3126     } else { /* Non-standard case -- may never happen */
3127         printf("Unexpected status (0x%x)\n", status);
3128     }
3129 } while (!WIFEXITED(status) && !WIFSIGNALED(status));
3130 }

```

3126 **Change Number:** XSH/TC2/D6/146 [XSH ERN 166]

3127 On Page: 1659 Line: 51172-51173 Section: write (2003 Ed.)

3128 On Page: 1652 Line: 50898-50899 Section: write (2001 Ed.)

3129 In the ERRORS section

3130 Change From:

3131 "In the latter case, if the socket is of type SOCK_STREAM, the SIGPIPE
3132 signal is generated to the calling process."

3133 To:

3134 "In the latter case, if the socket is of type SOCK_STREAM, a SIGPIPE
3135 signal shall also be sent to the thread."

3136 **Change Number:** XSH/TC2/D6/147 [XSH ERN 42]

3137 On Page: 1660 Line: 51223 Section: write (2003 Ed.)

3138 On Page: 1653 Line: 50949 Section: write (2001 Ed.)

3139 In the RATIONALE section

3140 Change From:

3141 "Otherwise, the process may block; that is, pause until enough space is
3142 available for writing."

3143 To:

3144 "Otherwise, the calling thread may block; that is, pause until enough space is
3145 available for writing."

3146 **Change Number:** XSH/TC2/D6/148 [XSH ERN 108]

3147 On Page: 1667 Line: 51442,51453 Section: y0 (2003 Ed.)

3148 On Page: 1660 Line: 51168,51179 Section: y0 (2001 Ed.)

3149 In the RETURN VALUE section

3150 Change From:

3151 "If x is 0.0, -HUGE_VAL shall be returned and a range error
3152 may occur".

Technical Corrigendum 2

3153 To:
3154 "If x is 0.0, -HUGE_VAL shall be returned and a pole error may occur."

3155 In the ERRORS section
3156 Insert before line 51453 (2003 Ed.); line 51179 (2001 Ed.):
3157 "Pole error The value of x is zero.

3158 If the integer expression (math_errhandling & MATH_ERRNO) is
3159 non-zero, then errno shall be set to [ERANGE]. If the integer
3160 expression (math_errhandling & MATH_ERREXCEPT) is non-zero,
3161 then the divide-by-zero floating-point exception shall be raised."

3162 In the ERRORS section, line 51453 (2003 Ed.); line 51179 (2001 Ed.)
3163 Change From:
3164 "Range Error The value of x is 0.0, or the correct result would cause
3165 overflow."

3166 To:
3167 "Range Error The correct result would cause overflow."

3168 Rationale:
3169 The changes are made for consistency with the general rules stated in
3170 "Treatment of Error Conditions for Mathematical Functions" in XBD.

4. Changes to Shell and Utilities

This section contains the set of changes to the text of the Shell and Utilities.

3171 **Change Number:** XCU/TC2/D6/1 [XBD ERN 20]

3172 On Page: xxxii Line: "ISO/IEC 8859" Section: Referenced Documents (2003 Ed.)

3173 On Page: xxviii Line: "ISO/IEC 8859" Section: Referenced Documents (2001 Ed.)

3174 Add after line starting "Part 10":

3175 "Part 11: Latin/Thai Alphabet"

3176 Add after line starting "Part 15":

3177 "Part 16: Latin Alphabet No. 10"

3178 **Change Number:** XCU/TC2/D6/2 [XCU ERN 29]

3179 On Page: 5 Line: 139-150,170 Section: 1.7.1.4 (2003 Ed.)

3180 On Page: 5 Line: 139-150,170 Section: 1.7.1.4 (2001 Ed.)

3181 In Table 1-1, change the dash (-) entries to letter "U" on lines

3182 140, 141 and 150 (rows for types B, C and T respectively).

3183 Change all the dash (-) entries in the T column to letter "U".

3184 Add after line 170

3185 "U The effect is unspecified unless specified by the utility description."

3186 **Change Number:** XCU/TC2/D6/3 [XRAT ERN 2]

3187 On Page: 41 Line: 1704 Section: 2.6.4 (2003 Ed.)

3188 On Page: 41 Line: 1700 Section: 2.6.4 (2001 Ed.)

3189 Add a new paragraph before the existing paragraph that

3190 commences "As an extension, the shell may recognize arithmetic expressions

3191 beyond those listed...":

3192 "All changes to variables in an arithmetic expression shall be in effect

3193 after the arithmetic expansion, as in the parameter expansion

3194 \${x=value}.

3195 If the shell variable x contains a value that forms a valid

3196 integer constant then the arithmetic expansions \$((x)) and \$((x))

3197 shall return the same value."

3198 **Change Number:** XCU/TC2/D6/4 [XCU ERN 17]

3199 On Page: 48 Line: 1986-1990 Section: 2.9.1.1 (2003 Ed.)

3200 On Page: 48 Line: 1982-1986 Section: 2.9.1.1 (2001 Ed.)

3201 In Section 2.9.1.1 Command Search and Execution

3202 Change From:

3203 "If the `execve()` function fails due to an error equivalent to the

3204 `[ENOEXEC]` error defined in the System Interfaces volume of IEEE Std

3205 1003.1-2001, the shell shall execute a command equivalent to having

3206 a shell invoked with the command name as its first operand, with any

3207 remaining arguments passed to the new shell."

3208 To:

3209 "If the `execve()` function fails due to an error equivalent to the

3210 `[ENOEXEC]` error defined in the System Interfaces volume of IEEE Std

3211 1003.1-2001, the shell shall execute a command equivalent to having a

3212 shell invoked with the pathname resulting from the search as its first

3213 operand, with any remaining arguments passed to the new shell, except

Technical Corrigendum 2

3214 that the value of \$0 in the new shell may be set to the command name."

3215 **Change Number:** XCU/TC2/D6/5 [XCU ERN 30]

3216 On Page: 212 Line: 8323-8324 Section: c99 (2003 Ed.)

3217 On Page: 212 Line: 8243-8244 Section: c99 (2001 Ed.)

3218 In the OPTIONS section

3219 Change From:

3220 "If a directory specified by the -L option contains files named libc.a,
3221 libm.a, libl.a, or liby.a, the results are unspecified."

3222 To:

3223 "If a directory specified by the -L option contains files with names
3224 starting with any of the strings "libc.", "libl.", "libpthread.",
3225 "libm.", "librt.", "libtrace.", "libxnet.", or "liby.", the results
3226 are unspecified."

3227 **Change Number:** XCU/TC2/D6/6 [XCU ERN 19]

3228 On Page: 215,218 Line: 8466-8467,8583-8588 Section: c99 (2003 Ed.)

3229 On Page: 215,218 Line: 8385-8386,8502-8507 Section: c99 (2001 Ed.)

3230 In the EXTENDED DESCRIPTION section, Programming Environments

3231 Change From:

3232 "shall be output by a getconf command using the
3233 _POSIX_V6_WIDTH_RESTRICTED_ENVS argument."

3234 To:

3235 "shall be output by a getconf command using the
3236 POSIX_V6_WIDTH_RESTRICTED_ENVS argument, as a <newline>-separated list
3237 of names suitable for use with the getconf -v option."

3238 (Note the removed leading underscore on POSIX_V6_WIDTH_RESTRICTED_ENVS.)

3239 In the EXAMPLES section

3240 Change From:

3241 "# ... if there are no additional constraints, the first one will do:
3242 CENV=\$(getconf _POSIX_V6_WIDTH_RESTRICTED_ENVS | head -n 1)

3243 # ... or, if an environment that supports large files is preferred,
3244 # look for names that contain "OFF64" or "OFFBIG". (This chooses
3245 # the last one in the list if none match.)
3246 for CENV in \$(getconf _POSIX_V6_WIDTH_RESTRICTED_ENVS)"

3247 To:

3248 "# ... if there are no additional constraints, the first one will do:
3249 CENV=\$(getconf POSIX_V6_WIDTH_RESTRICTED_ENVS | head -n 1)

3250 # ... or, if an environment that supports large files is preferred,
3251 # look for names that contain "OFF64" or "OFFBIG". (This chooses
3252 # the last one in the list if none match.)
3253 for CENV in \$(getconf POSIX_V6_WIDTH_RESTRICTED_ENVS) "

3254 Rationale:

3255 This change corrects the name of the variable not to include the
3256 underscore.

3257 **Change Number:** XCU/TC2/D6/7 [XCU ERN 33]

3258 On Page: 343 Line: 13238-13240,13451 Section: ed (2003 Ed.)

3259 On Page: 341 Line: 13113-13115,13326 Section: ed (2001 Ed.)

3260 In the EXTENDED DESCRIPTION section (List command)

3261 Delete the sentence:

3262 "If the size of a byte on the system is greater than nine bits, the

3263 format used for non-printable characters is implementation-defined."

3264 In the RATIONALE section:

3265 Change From:

3266 "Systems with bytes too large to fit into three octal digits must devise

3267 other means of displaying non-printable characters. Consideration was

3268 given to requiring that the number of octal digits be large enough

3269 to hold a byte, but this seemed to be too confusing for applications

3270 on the vast majority of systems where three digits are adequate. It

3271 would be theoretically possible for the application to use the getconf

3272 utility to find out the CHAR_BIT value and deal with such an algorithm;

3273 however, there is really no portable way that an application can use

3274 the octal values of the bytes across various coded character sets,

3275 so the additional specification was not worthwhile."

3276 To:

3277 "Prior versions of this standard allowed for implementations with

3278 bytes other than eight bits, but this has been modified in this

3279 version."

3280 **Change Number:** XCU/TC2/D6/8 [XCU ERN 31]

3281 On Page: 370 Line: 14333-14334 Section: ex (2003 Ed.)

3282 On Page: 368 Line: 14205-14206 Section: ex (2001 Ed.)

3283 In the EXTENDED DESCRIPTION section

3284 Change From:

3285 "Implementations may restrict the set of characters accepted in lhs

3286 or rh, except that printable characters and <blank>s shall not be

3287 restricted. Additional restrictions shall be implementation-defined."

3288 To:

3289 "Implementations may restrict the set of characters accepted in lhs

3290 or rhs, except that printable characters and <blank>s shall not be

3291 restricted. Additional restrictions shall be implementation-defined."

3292 Note to editor: The change is from "rh" to "rhs".

3293 **Change Number:** XCU/TC2/D6/9 [XCU ERN 33]

3294 On Page: 380 Line: 14705-14709,16154 Section: ex (2003 Ed.)

3295 On Page: 378 Line: 14577-14581,16025 Section: ex (2001 Ed.)

3296 In the EXTENDED DESCRIPTION section (Print command)

3297 Delete the sentence:

3298 "If the size of a byte on the system is greater than 9 bits, the format used

3299 for non-printable characters is implementation-defined."

3300 Add to the RATIONALE section:

3301 "Prior versions of this standard allowed for implementations with

3302 bytes other than eight bits, but this has been modified in this

3303 version."

Technical Corrigendum 2

3304 **Change Number:** XCU/TC2/D6/10 [XCU ERN 26,28]

3305 On Page: 447-448 Line: 17290-17292,17297-17300,17317-17322 Section: file (2003 Ed.)

3306 On Page: 445-446 Line: 17135-17137,17142-17147,17162-17167 Section: file (2001 Ed.)

3307 On lines 17290-17292 (2003 Ed.); lines 17135-17137 (2001 Ed.)

3308 Change From:

3309 "The type shall consist of the type specification characters c, d, f, s,

3310 and u, specifying character, signed decimal, floating point, string,

3311 and unsigned decimal, respectively."

3312 To:

3313 "The type shall consist of the type specification characters d, s, and u,

3314 specifying signed decimal, string, and unsigned

3315 decimal, respectively."

3316 On lines 17297-17300 (2003 Ed.); lines 17142-17147 (2001 Ed.)

3317 Change From:

3318 "The type specification characters d, f, and u can be

3319 followed by an optional unsigned decimal integer that

3320 specifies the number of bytes represented by the type.

3321 The type specification character f can be followed by

3322 an optional F, D, or L, indicating that the value is

3323 of type float, double, or long double, respectively."

3324 To:

3325 "The type specification characters d and u can be

3326 followed by an optional unsigned decimal integer that

3327 specifies the number of bytes represented by the type."

3328 On lines 17317-17322 (2003 Ed.); lines 17162-17167 (2001 Ed.)

3329 Delete the paragraph:

3330 "For the type specifier f, the default number of bytes shall correspond

3331 to the number of bytes in the basic double precision floating-point data

3332 type of the underlying implementation. The implementation shall support

3333 values of the optional number of bytes to be converted corresponding

3334 to the number of bytes in the C-language types float, double, and long

3335 double. These numbers can also be specified by an application as the

3336 characters F, D, and L, respectively."

3337 **Change Number:** XCU/TC2/D6/11 [XCU ERN 27]

3338 On Page: 447-448 Line: 17280 Section: file (2003 Ed.)

3339 On Page: 445-446 Line: 17125 Section: file (2001 Ed.)

3340 In the EXTENDED DESCRIPTION section

3341 On line 17280 (2003 Ed.); line 17125 (2001 Ed.)

3342 Change From:

3343 "Each line shall be composed of the following four <blank>-separated fields:"

3344 To:

3345 "Each line shall be composed of the following four <tab>-separated* fields:

3346 Footnote:* Implementations may allow any combination of one

3347 or more white space characters other than <newline>

3348 to act as field separators."

3349 On lines 17336-17341 (2003 Ed.); lines 17181-17186 (2001 Ed.)

3350 Change From:

3351 "\character

3352 The backslash-escape sequences as specified in the Base Definitions

3353 volume of IEEE Std 1003.1-2001, Table 5-1, Escape Sequences and

3354 Associated Actions ('\\', '\a', '\b', '\f', '\n', '\r', '\t',

3355 '\v'). The results of using any other character, other than an octal

3356 digit, following the backslash are unspecified."

3357 To:

3358 "\character

3359 The backslash-escape sequences as specified in the Base Definitions
 3360 volume of IEEE Std 1003.1-2001, Table 5-1, Escape Sequences and Associated
 3361 Actions ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v'). In
 3362 addition, the escape sequence '\ ' (the <backslash> character followed
 3363 by the <space> character) shall be recognized to represent a <space>
 3364 character. The results of using any other character, other than an
 3365 octal digit, following the backslash are unspecified."

3366 **Change Number:** XCU/TC2/D6/12 [XCU ERN 33]

3367 On Page: 448 Line: 17345-17347,17462 Section: file (2003 Ed.)
 3368 On Page: 446 Line: 17190-17192,17278 Section: file (2001 Ed.)

3369 In the EXTENDED DESCRIPTION section
 3370 Delete the sentence:
 3371 "If the size of a byte on the system is greater than 9 bits, the valid
 3372 escape sequence used to represent a byte implementation-defined."

3373 In the RATIONALE section
 3374 Add the following text at the end:
 3375 "Prior versions of this standard allowed for implementations with
 3376 bytes other than eight bits, but this has been modified in this
 3377 version."

3378 **Change Number:** XCU/TC2/D6/13 [XCU ERN 4]

3379 On Page: 458 Line: 17761-17762,17765-17766 Section: find (2003 Ed.)
 3380 On Page: 455 Line: 17572-17573,17576-17577 Section: find (2001 Ed.)

3381 In the RATIONALE section
 3382 Change From:
 3383 "The descriptions of -atime, -ctime, and -mtime were changed from the
 3384 SVID description of n 'days' to '24-hour period'."

3385 To:
 3386 "The descriptions of -atime, -ctime, and -mtime were changed from the
 3387 SVID description of n 'days' to n being the result of the integer
 3388 division of the time difference in seconds by 86400".

3389 Change From:
 3390 "For example, -atime 3 is true if the file was accessed any time in the
 3391 period from 72 hours to 48 hours ago."

3392 To:
 3393 "For example, -atime 2 is true if the file was accessed any time in the
 3394 period from 72 hours to 48 hours ago."

3395 Rationale:
 3396 The rationale for -atime/-mtime/-ctime needs updating to match
 3397 changes made in the normative text since POSIX.2-1992.

3398 **Change Number:** XCU/TC2/D6/14 [XCU ERN 33]

3399 On Page: 540 Line: 20802-20806,21020-21023 Section: lex (2003 Ed.)
 3400 On Page: 537 Line: 20593-20598,20811-20814 Section: lex (2001 Ed.)

3401 In the EXTENDED DESCRIPTION section (Table 4-10)
 3402 Delete the sentence:
 3403 "If the size of a byte on the system is greater than nine bits, the escape
 3404 sequence used to represent a byte is implementation-defined."

3405 In the RATIONALE section
 3406 Change from:
 3407 "The description of octal and hexadecimal-digit escape sequences agrees
 3408 with the ISO C standard use of escape sequences. See the RATIONALE for ed
 3409 for a discussion of bytes larger than 9 bits being represented by octal
 3410 values. Hexadecimal values can represent larger bytes and multi-byte
 3411 characters directly, using as many digits as required."

Technical Corrigendum 2

3412 To:
3413 "The description of octal and hexadecimal-digit escape sequences agrees
3414 with the ISO C standard use of escape sequences.

3415 Prior versions of this standard allowed for implementations with
3416 bytes other than eight bits, but this has been modified in this
3417 version."

3418 **Change Number:** XCU/TC2/D6/15 [XCU ERN 18]

3419 On Page: 559 Line: 21535 Section: localedef (2003 Ed.)
3420 On Page: 556 Line: 21324 Section: localedef (2001 Ed.)

3421 In the OPERANDS section
3422 Change From:
3423 "This capability may be restricted to users with appropriate privileges."

3424 To:
3425 "The ability to create public locales in this way may be restricted to
3426 users with appropriate privileges."

3427 **Change Number:** XCU/TC2/D6/16 [XCU ERN 13]

3428 On Page: 560-561 Line: 21593-21594,21611 Section: localedef (2003 Ed.)
3429 On Page: 557-558 Line: 21382-21383,21400 Section: localedef (2001 Ed.)

3430 In the EXTENDED DESCRIPTION section
3431 Change From:
3432 "If a non-printable character in the charmap has a width specified
3433 that is not -1, localedef shall generate a warning."

3434 To:
3435 "If a non-printable character in the charmap has a width specified
3436 that is not -1 the result will be undefined."

3437 In the CONSEQUENCES OF ERRORS section
3438 Delete the fourth bullet item:

3439 "* If a non-printable character has a width specified other than -1."

3440 Rationale:
3441 The previous text was conflicting with the descriptions of WIDTH and
3442 WIDTH_DEFAULT in XBD.

3443 **Change Number:** XCU/TC2/D6/17 [XCU ERN 34]

3444 On Page: 593 Line: 22946-22947 Section: mailx (2003 Ed.)
3445 On Page: 590 Line: 22721-22722 Section: mailx (2001 Ed.)

3446 In the EXTENDED DESCRIPTION section (Internal Variables in Mailx)
3447 Change From:
3448 "The defaults specified here may be changed by the implementation-defined
3449 system start-up file unless the user specifies the -n option."

3450 To:
3451 "The defaults specified here may be changed by the unspecified
3452 system start-up file unless the user specifies the -n option."

3453 Rationale: The change is made for consistency with elsewhere in the
3454 EXTENDED DESCRIPTION.

3455 **Change Number:** XCU/TC2/D6/18 [XCU ERN 35]

3456 On Page: 666 Line: 25769-25772 Section: nice (2003 Ed.)

3457 On Page: 663 Line: 25539-25542 Section: nice (2001 Ed.)

3458 In the RATIONALE section

3459 Delete the paragraph:

3460 "Due to the text about the limits of the nice value being
3461 implementation-defined, nice is not actually required to change the nice
3462 value of the executed command; the limits could be zero differences from
3463 the system default, although the implementor is required to document this
3464 fact in the conformance document."

3465 **Change Number:** XCU/TC2/D6/19 [XCU ERN 33]

3466 On Page: 682 Line: 26379-26380,26539 Section: od (2003 Ed.)

3467 On Page: 679 Line: 26149-26150,26309 Section: od (2001 Ed.)

3468 In the EXTENDED DESCRIPTION section

3469 Delete the sentence:

3470 "If the size of a byte on the system is greater than nine bits, the format
3471 used for non-printable characters is implementation-defined."

3472 Add to the RATIONALE section:

3473 "Prior versions of this standard allowed for implementations with
3474 bytes other than eight bits, but this has been modified in this
3475 version."

3476 **Change Number:** XCU/TC2/D6/20 [XCU ERN 21]

3477 On Page: 701 Line: 27113 Section: pax (2003 Ed.)

3478 On Page: 698 Line: 26876 Section: pax (2001 Ed.)

3479 In the SYNOPSIS section

3480 Change From:

3481 "pax -r -w[-diklntuvX][-H|-L][-p string]...[-s replstr]..."

3482 To:

3483 "pax -r -w[-diklntuvX][-H|-L][-o options]...[-p string]...[-s replstr]..."

3484 Change From:

3485 "pax [-cdnv][-H|-L][-f archive][-s replstr]...[pattern...]..."

3486 To:

3487 "pax [-cdnv][-H|-L][-o options][-f archive][-s replstr]...[pattern...]..."

3488 Rationale:

3489 The SYNOPSIS was inconsistent with the normative text.

3490 **Change Number:** XCU/TC2/D6/21 [XCU ERN 24]

3491 On Page: 702 Line: 27167-27168 Section: pax (2003 Ed.)

3492 On Page: 699 Line: 26930-26931 Section: pax (2001 Ed.)

3493 Change From:

3494 "If the selected archive format supports the specification of linked
3495 files, it shall be an error if these files cannot be linked when the
3496 archive is extracted."

3497 To:

3498 "If the selected archive format supports the specification of linked
3499 files, it shall be an error if these files cannot be linked when the
3500 archive is extracted, except that if the files to be linked are symbolic
3501 links and the system is not capable of making hard links to symbolic
3502 links then separate copies of the symbolic link shall be created instead."

Technical Corrigendum 2

3503 **Change Number:** XCU/TC2/D6/22 [XCU ERN 22]

3504 On Page: 704 Line: 27266 Section: pax (2003 Ed.)

3505 On Page: 701 Line: 27029 Section: pax (2001 Ed.)

3506 In the OPTIONS section,

3507 Add new text to the end of the subsection for "delete=pattern"

3508 "When multiple -odelete=pattern options are specified, the patterns shall

3509 be additive; all keywords matching the specified string patterns shall

3510 be omitted from extended header records that pax produces."

3511 **Change Number:** XCU/TC2/D6/23 [XCU ERN 9]

3512 On Page: 706 Line: 27336-27339 Section: pax (2003 Ed.)

3513 On Page: 703 Line: 27097-27100 Section: pax (2001 Ed.)

3514 In the OPTIONS section

3515 Change From:

3516 "write In read or copy mode, pax shall write the file, translating or

3517 truncating the name, regardless of whether this may overwrite an

3518 existing file with a valid name. In list mode, pax shall behave

3519 identically to the bypass action."

3520 To:

3521 "write In read or copy mode, pax shall write the file, translating the

3522 name, regardless of whether this may overwrite an existing file

3523 with a valid name. In list mode, pax shall behave identically

3524 to the bypass action."

3525 Rationale:

3526 The text currently written in the standard allows the pax utility to

3527 truncate pathnames of files being extracted from an archive or copied

3528 from one place to another in arbitrary ways, even if this action

3529 overwrites existing files. We agree that translation may cause this to

3530 occur, but the pax utility should not truncate file names on its

3531 own. It should be capable of restoring files even if the pathname in

3532 the archive is longer than {PATH_MAX}. (This may require the pax

3533 utility to perform one or more change directory operations to get to a

3534 point where the file being extracted can be created using a pathname

3535 shorter than {PATH_MAX}.) If the underlying file system performs

3536 filename truncation (which is not allowed on file systems conforming to

3537 this standard), the pax utility need not attempt to prevent or detect

3538 the action. But, the pax utility itself should never truncate a

3539 pathname component.

3540 **Change Number:** XCU/TC2/D6/24 [XCU ERN 7]

3541 On Page: 708 Line: 27469 Section: pax (2003 Ed.)

3542 On Page: 705 Line: 27230 Section: pax (2001 Ed.)

3543 In the OPTIONS section

3544 On line 27469 (2003 Ed.); line 27230 (2001 Ed.)

3545 add as a new paragraph before the paragraph beginning

3546 "The options that operate on the names of files..."

3547 "Specifying more than one of the mutually-exclusive options -H and -L

3548 shall not be considered an error and the last option specified

3549 shall determine the behavior of the utility."

3550 **Change Number:** XCU/TC2/D6/25 [XCU ERN 8]

3551 On Page: 715 Line: 27723-27727 Section: pax (2003 Ed.)

3552 On Page: 712 Line: 27484-27488 Section: pax (2001 Ed.)

3553 In the EXTENDED DESCRIPTION section (Pax Extended Header)

3554 Delete the paragraph at line 27723 (2003 Ed.); line 27484 (2001 Ed.):

3555 "ctime The file creation time for the following file(s), equivalent

3556 to the value of the st_ctime member of the stat structure for
 3557 a file, as described by the stat() function. The creation time
 3558 shall be restored if the process has the appropriate privilege
 3559 required to do so. The format of the <value> shall be as
 3560 described in pax Extended Header File Times (on page 717)."

3561 Rationale:

3562 There is a contradiction in the definition of the ctime keyword for
 3563 pax's extended header, in that the st_ctime member of the stat structure
 3564 does not refer to a file creation time. No field in the standard stat
 3565 structure from <sys/stat.h> includes a file creation time.

3566 **Change Number:** XCU/TC2/D6/26 [XCU ERN 23]

3567 On Page: 720 Line: 27939-27940,27961 Section: pax (2003 Ed.)
 3568 On Page: 717 Line: 27700-27701,27722 Section: pax (2001 Ed.)

3569 In the EXTENDED DESCRIPTION section (ustar Interchange Format)
 3570 On lines 27939-27940 (2003 Ed.); lines 27700-27701 (2001 Ed.)

3571 Change From:

3572 "Such files are identified by each file having the same device and
 3573 file serial number."

3574 To:

3575 "Such files are identified by having the same device and file
 3576 serial numbers, and pathnames that refer to different directory
 3577 entries. All such files shall be archived as linked files."

3578 Before line 27961 (2003 Ed.); line 27722 (2001 Ed.)

3579 Insert the following paragraph:

3580 "It is unspecified whether files with pathnames that refer to
 3581 the same directory entry are archived as linked files or as
 3582 separate files. If they are archived as linked files this means
 3583 that attempting to extract both pathnames from the resulting
 3584 archive will always cause an error (unless the -u option is used)
 3585 because the link cannot be created."

3586 It is unspecified whether files with the same device and file
 3587 serial numbers being appended to an archive are treated as
 3588 linked files to members that were in the archive before the append."

3589 **Change Number:** XCU/TC2/D6/27 [XCU ERN 10]

3590 On Page: 722,724 Line: 28039-28040,28099 Section: pax (2003 Ed.)
 3591 On Page: 719,721 Line: 27800-27801,27860 Section: pax (2001 Ed.)

3592 In the EXTENDED DESCRIPTION section (cpio header)

3593 Change From:

3594 "c_nlink Contains the number of links referencing the file at
 3595 the time the archive was created."

3596 To:

3597 "c_nlink Contains a number greater than or equal to the
 3598 number of links in the archive referencing the file. If the -a option is
 3599 used to append to a cpio archive, then the pax utility need not account
 3600 for the files in the existing part of the archive when calculating the
 3601 c_nlink values for the appended part of the archive, and need not alter
 3602 the c_nlink values in the existing part of the archive if additional
 3603 files with the same c_dev and c_ino values are appended to the archive."

3604 In the APPLICATION USAGE section

3605 Insert before line 28099 (2003 Ed.); line 27860 (2001 Ed.)

3606 "Caution is advised when using the -a option to append to a
 3607 cpio format archive. If any of the files being appended happen
 3608 to be given the same c_dev and c_ino values as a file in the
 3609 existing part of the archive, then they may be treated as links
 3610 to that file on extraction. Thus it is risky to use -a with cpio

Technical Corrigendum 2

3611 format except when it is done on the same system that the original
3612 archive was created on, and with the same pax utility, and in
3613 the knowledge that there has been little or no file system
3614 activity since the original archive was created that could lead
3615 to any of the files being appended being given the same c_dev and
3616 c_ino values as an unrelated file in the existing part of the
3617 archive. Also, when (intentionally) appending additional links
3618 to a file in the existing part of the archive, the c_nlink values
3619 in the modified archive can be smaller than the number of links
3620 to the file in the archive, which may mean that the links are not
3621 preserved on extraction."

3622 **Change Number:** XCU/TC2/D6/28 [XCU ERN 33]

3623 On Page: 845 Line: 32699-32701,32892 Section: sed (2003 Ed.)

3624 On Page: 842 Line: 32450-32452,32643 Section: sed (2001 Ed.)

3625 In the EXTENDED DESCRIPTION section

3626 Delete the sentence:

3627 "If the size of a byte on the system is greater than nine bits, the format
3628 used for non-printable characters is implementation-defined."

3629 Add to the RATIONALE section:

3630 "Prior versions of this standard allowed for implementations with
3631 bytes other than eight bits, but this has been modified in this
3632 version."

3633 **Change Number:** XCU/TC2/D6/29 [XCU ERN 11]

3634 On Page: 846 Line: 32770-32771 Section: sed (2003 Ed.)

3635 On Page: 843 Line: 32521-32522 Section: sed (2001 Ed.)

3636 Change From:

3637 "If the delimiter is not n, within string1 and string2, the delimiter
3638 itself can be used as a literal character if it preceeded by
3639 a backslash."

3640 To:

3641 "If the delimiter is not 'n', within string1 and string2, the delimiter
3642 itself can be used as a literal character if it preceeded by
3643 a backslash."

3644 Note to editor the change is from italic n to a quoted character n.

3645 **Change Number:** XCU/TC2/D6/30 [XCU ERN 14]

3646 On Page: 907-905 Line: 35124-35169 Section: test (2003 Ed.)

3647 On Page: 904-905 Line: 34872-34917 Section: test (2001 Ed.)

3648 In the OPERANDS section

3649 Change From:

3650 "-b file

3651 True if file exists and is a block special file.

3652 -c file

3653 True if file exists and is a character special file.

3654 -d file

3655 True if file exists and is a directory.

3656 -e file

3657 True if file exists.

3658 -f file

3659 True if file exists and is a regular file.

3660 -g file

3661 True if file exists and its set-group-ID flag is set.

```

3662 -h file
3663     True if file exists and is a symbolic link.

3664 -L file
3665     True if file exists and is a symbolic link.

3666 -n string
3667     True if the length of string is non-zero.

3668 -p file
3669     True if file is a FIFO.

3670 -r file
3671     True if file exists and is readable. True shall indicate that
3672     permission to read from file will be granted, as defined in Section
3673     1.7.1.4 (on page 4).

3674 -S file
3675     True if file exists and is a socket.

3676 -s file
3677     True if file exists and has a size greater than zero.

3678 -t file_descriptor
3679     True if the file whose file descriptor number is file_descriptor is
3680     open and is associated with a terminal.

3681 -u file
3682     True if file exists and its set-user-ID flag is set.

3683 -w file
3684     True if file exists and is writable. True shall indicate that
3685     permission to write from file will be granted, as defined in
3686     Section 1.7.1.4 (on page 4).

3687 -x file
3688     True if file exists and is executable. True shall indicate that
3689     permission to execute file will be granted, as defined in Section
3690     1.7.4 (on page 4). If file is a directory, true shall indicate that
3691     permission to search file will be granted.

3692 -z string
3693     True if the length of string string is zero.

3694 string
3695     True if the string string is not the null string.

3696 s1 = s2
3697     True if the strings s1 and s2 are identical.

3698 s1 != s2
3699     True if the strings s1 and s2 are not identical.

3700 n1 -eq n2
3701     True if the integers n1 and n2 are algebraically equal.

3702 n1 -ne n2
3703     True if the integers n1 and n2 are not algebraically equal.

3704 n1 -gt n2
3705     True if the integer n1 is algebraically greater than the integer n2.

3706 n1 -ge n2
3707     True if the integer n1 is algebraically greater than or equal to
3708     the integer n2.

3709 n1 -lt n2
3710     True if the integer n1 is algebraically less than the integer n2.

3711 n1 -le n2

```

Technical Corrigendum 2

3712 True if the integer n1 is algebraically less than or equal to the
3713 integer n2.

3714 expression1 -a expression2
3715 [XSI] True if both expression1 and expression2 are true. The -a
3716 binary primary is left associative. It has a higher precedence than
3717 -o. [/XSI]

3718 expression1 -o expression2
3719 [XSI] True if either expression1 or expression2 is
3720 true. The -o binary primary is left associative. [/XSI]

3721 With the exception of the -h file and -L file primaries, if a file
3722 argument is a symbolic link, test shall evaluate the expression by
3723 resolving the symbolic link and using the file referenced by the link.

3724 These primaries can be combined with the following operators:

3725 ! expression
3726 True if expression is false.

3727 (expression)
3728 [XSI] True if expression is true. The parentheses can be used to
3729 alter the normal precedence and associativity. [/XSI]"

3730 To:

3731 "-b pathname
3732 True if pathname resolves to a file that exists and is a block special
3733 file. False if pathname cannot be resolved, or if pathname resolves
3734 to a file that exists but is not a block special file.

3735 -c pathname
3736 True if pathname resolves to a file that exists and is a character
3737 special file. False if pathname cannot be resolved, or if pathname
3738 resolves to a file that exists but is not a character special file.

3739 -d pathname
3740 True if pathname resolves to a file that exists and is a directory.
3741 False if pathname cannot be resolved, or if pathname resolves to a
3742 file that exists but is not a directory.

3743 -e pathname
3744 True if pathname resolves to a file that exists. False if pathname
3745 cannot be resolved.

3746 -f pathname
3747 True if pathname resolves to a file that exists and is a regular file.
3748 False if pathname cannot be resolved, or if pathname resolves to a
3749 file that exists but is not a regular file.

3750 -g pathname
3751 True if pathname resolves to a file that exists and has its set-group-ID
3752 flag set. False if pathname cannot be resolved, or if pathname resolves
3753 to a file that exists but does not have its set-group-ID flag set.

3754 -h pathname
3755 True if pathname resolves to a file that exists and is a symbolic link.
3756 False if pathname cannot be resolved, or if pathname resolves to a file
3757 that exists but is not a symbolic link. If the final component of
3758 pathname is a symlink that symlink is not followed.

3759 -L pathname
3760 True if pathname resolves to a file that exists and is a symbolic link.
3761 False if pathname cannot be resolved, or if pathname resolves to a
3762 file that exists but is not a symbolic link. If the final component
3763 of pathname is a symlink that symlink is not followed.

3764 -n string
3765 True if the length of string is non-zero, otherwise false.

3766 -p pathname
3767 True if pathname resolves to a file that exists and is a FIFO.


```

3768 False if pathname cannot be resolved, or if pathname resolves to a
3769 file that exists but is not a FIFO.

3770 -r pathname
3771 True if pathname resolves to a file that exists and for which permission
3772 to read from the file will be granted, as defined in Section 1.7.1.4
3773 (on page 4). False if pathname cannot be resolved, or if pathname
3774 resolves to a file for which permission to read from the file will
3775 not be granted.

3776 -S pathname
3777 True if pathname resolves to a file that exists and is a socket.
3778 False if pathname cannot be resolved, or if pathname resolves to a
3779 file that exists but is not a socket.

3780 -s pathname
3781 True if pathname resolves to a file that exists and has a size greater
3782 than zero. False if pathname cannot be resolved, or if pathname
3783 resolves to a file that exists but does not have a size greater
3784 than zero.

3785 -t file_descriptor
3786 True if file descriptor number file_descriptor is open and is
3787 associated with a terminal. False if file_descriptor is not a valid
3788 file descriptor number, or if file descriptor number file_descriptor
3789 is not open, or if it is open but is not associated with a terminal.

3790 -u pathname
3791 True if pathname resolves to a file that exists and has its set-user-ID
3792 flag set. False if pathname cannot be resolved, or if pathname resolves
3793 to a file that exists but does not have its set-user-ID flag set.

3794 -w pathname
3795 True if pathname resolves to a file that exists and for which permission
3796 to write to the file will be granted, as defined in Section 1.7.1.4 (on
3797 page 4). False if pathname cannot be resolved, or if pathname resolves
3798 to a file for which permission to write to the file will not be granted.

3799 -x pathname
3800 True if pathname resolves to a file that exists and for which
3801 permission to execute the file (or search it, if it is a directory)
3802 will be granted, as defined in Section 1.7.1.4 (on page 4). False if
3803 pathname cannot be resolved, or if pathname resolves to a file for
3804 which permission to execute (or search) the file will not be granted.

3805 -z string
3806 True if the length of string string is zero, otherwise false.

3807 string
3808 True if the string string is not the null string, otherwise false.

3809 s1 = s2
3810 True if the strings s1 and s2 are identical, otherwise false.

3811 s1 != s2
3812 True if the strings s1 and s2 are not identical, otherwise false.

3813 n1 -eq n2
3814 True if the integers n1 and n2 are algebraically equal, otherwise
3815 false.

3816 n1 -ne n2
3817 True if the integers n1 and n2 are not algebraically equal,
3818 otherwise false.

3819 n1 -gt n2
3820 True if the integer n1 is algebraically greater than the integer n2,
3821 otherwise false.

3822 n1 -ge n2
3823 True if the integer n1 is algebraically greater than or equal to

```

Technical Corrigendum 2

3824 the integer n2, otherwise false.

3825 n1 -lt n2
3826 True if the integer n1 is algebraically less than the integer n2,
3827 otherwise false.

3828 n1 -le n2
3829 True if the integer n1 is algebraically less than or equal to the
3830 integer n2, otherwise false.

3831 expression1 -a expression2
3832 [XSI] True if both expression1 and expression2 are true, otherwise
3833 false. The -a binary primary is left associative. It has a higher
3834 precedence than -o. [/XSI]

3835 expression1 -o expression2
3836 [XSI] True if either expression1 or expression2 is true, otherwise
3837 false. The -o binary primary is left associative. [/XSI]

3838 With the exception of the -h file and -L file primaries, if a file
3839 argument is a symbolic link, test shall evaluate the expression by
3840 following the symbolic link and using the file referenced by the link.

3841 These primaries can be combined with the following operators:

3842 ! expression
3843 True if expression is false. False if expression is true.
3844 (expression)
3845 [XSI] True if expression is true. False if expression is false. The
3846 parentheses can be used to alter the normal precedence and
3847 associativity. [/XSI]

3848 Rationale:
3849 The previous descriptions of the file existence primaries for the test utility
3850 have been clarified.

3851 **Change Number: XCU/TC2/D6/31** [XCU ERN 33]

3852 On Page: 927 Line: 35871-35872,36041 Section: tr (2003 Ed.)
3853 On Page: 924 Line: 35616-35617,35786 Section: tr (2001 Ed.)

3854 In the EXTENDED DESCRIPTION section
3855 Delete the sentence:
3856 "If the size of a byte on the system is greater than nine bits,
3857 the valid escape sequence to represent a byte is
3858 implementation-defined."

3859 Add to the RATIONALE section:
3860 "Prior versions of this standard allowed for implementations with
3861 bytes other than eight bits, but this has been modified in this
3862 version."

3863 **Change Number: XCU/TC2/D6/32** [XCU ERN 12]

3864 On Page: 930 Line: 35992 Section: tr (2003 Ed.)
3865 On Page: 927 Line: 35737 Section: tr (2001 Ed.)

3866 In the EXAMPLES section
3867 Change From:
3868 "tr "[=e]" e <file1 >file2"

3869 To:
3870 "tr "[=e]" "[e*]" <file1 >file2"

3871 Rationale:
3872 The previous example depended on unspecified behavior.

3873 **Change Number:** XCU/TC2/D6/33 [XCU ERN 20]

3874 On Page: 931 Line: 36041 Section: tr (2003 Ed.)

3875 On Page: 928 Line: 35786 Section: tr (2001 Ed.)

3876 In the RATIONALE section,

3877 Change From:

3878 "IEEE Std 1003.1-2001 specifies that octal sequences always refer to

3879 single byte binary values."

3880 To:

3881 "IEEE Std 1003.1-2001 specifies that octal sequences always refer to

3882 single byte binary values when used to specify an endpoint of a range

3883 of collating elements."

3884 Rationale:

3885 This clarification is for consistency with the normative text.

3886 **Change Number:** XCU/TC2/D6/34 [XCU ERN 35]

3887 On Page: 945 Line: 36539-36540 Section: umask (2003 Ed.)

3888 On Page: 663 Line: 25539-25542 Section: umask (2001 Ed.)

3889 In the RATIONALE section

3890 Change From:

3891 "The default output style is implementation-defined to permit implementors

3892 to provide migration to the new symbolic style at the time most

3893 appropriate to their users."

3894 To:

3895 "The default output style is unspecified to permit implementors

3896 to provide migration to the new symbolic style at the time most

3897 appropriate to their users."

3898 Rationale: The change is made for consistency with the normative text.

3899 **Change Number:** XCU/TC2/D6/35 [XCU ERN 16]

3900 On Page: 969 Line: 37351 Section: udecode (2003 Ed.)

3901 On Page: 966 Line: 37089 Section: udecode (2003 Ed.)

3902 In the DESCRIPTION section

3903 Add at the end of the first paragraph:

3904 "If either of the op characters '+' and '-' (see chmod) are specified

3905 in symbolic mode, the initial mode on which those operations are based

3906 is unspecified."

3907 Note to editors: "op" is in italics

5. Changes to Rationale

This section contains the set of changes to the text of the Rationale.

3908 **Change Number:** XRAT/TC2/D6/1 [XBD ERN 20]

3909 On Page: xxxiii Line: "ISO/IEC 8859" Section: Referenced Documents (2003 Ed.)

3910 On Page: xxix Line: "ISO/IEC 8859" Section: Referenced Documents (2001 Ed.)

3911 Add after line starting "Part 10":

3912 "Part 11: Latin/Thai Alphabet"

3913 Add after line starting "Part 15":

3914 "Part 16: Latin Alphabet No. 10"

3915 **Change Number:** XRAT/TC2/D6/2 [XBD ERN 35]

3916 On Page: 7 Line: 191 Section: A.1.4 (2003 Ed.)

3917 On Page: 7 Line: 190 Section: A.1.4 (2001 Ed.)

3918 Add to the end of the paragraph:

3919 "In some places the text refers to facilities supplied by the
3920 implementation that are outside of the standard as implementation-supplied
3921 or implementation-provided. This is not intended to imply a requirement
3922 for documentation. If it were the term "implementation-defined" would
3923 have been used."

3924 **Change Number:** XRAT/TC2/D6/3 [XBD ERN 17]

3925 On Page: 42 Line: 1652 Section: A.6.3 (2003 Ed.)

3926 On Page: 42 Line: 1648 Section: A.6.3 (2001 Ed.)

3927 Change From:

3928 "There is no additional rationale provided for this section."

3929 To:

3930 "The standard does not specify how wide characters are encoded or
3931 provide a method for defining wide characters in a charmap.
3932 It specifies ways of translating between wide characters and multibyte
3933 characters. The standard does not prevent an extension from
3934 providing a method to define wide characters."

3935 **Change Number:** XRAT/TC2/D6/4 [XBD ERN 25]

3936 On Page: 49 Line: 1962-1966 Section: A.7.3.3 (2003 Ed.)

3937 On Page: 49 Line: 1958-1962 Section: A.7.3.3 (2001 Ed.)

3938 Change From:

3939 "The locale definition is an extension of the ISO C standard localeconv()
3940 specification. In particular, rules on how currency_symbol is treated
3941 are extended to also cover int_curr_symbol, and p_set_by_space and
3942 n_sep_by_space have been augmented with the value 2, which places a
3943 <space> between the sign and the symbol (if they are adjacent; otherwise,
3944 it should be treated as a 0). The following table shows the result of
3945 various combinations:"

3946 To:

3947 "The locale definition is an extension of the ISO C standard localeconv()
3948 specification. In particular, rules on how currency_symbol is treated
3949 are extended to also cover int_curr_symbol, and p_set_by_space and
3950 n_sep_by_space have been augmented with the value 2, which places a
3951 <space> between the sign and the symbol. This has been updated to match
3952 the C99 requirements and is an incompatible change from UNIX 98 and
3953 POSIX.2-1992 and POSIX.1-1996 requirements. The following table shows

3954 the result of various combinations:"

3955 **Change Number:** **XRAT/TC2/D6/5** [XSH ERN 92]

3956 On Page: 55 Line: 2254-2257,2258-2259 Section: 8.2 (2003 Ed.)
 3957 On Page: 56 Line: 2244-2247,2248-2249 Section: 8.2 (2001 Ed.)

3958 In Section 8.2 Internationalization Variables
 3959 Change From:
 3960 "The text about locale implies that any utilities written in standard
 3961 C and conforming to IEEE Std 1003.1-2001 must issue the following call:

3962 `setlocale(LC_ALL, "")`

3963 If this were omitted, the ISO C standard specifies that the C locale
 3964 would be used."

3965 To:
 3966 "Utilities conforming to the Shell and Utilities volume of IEEE Std
 3967 1003.1-2001 and written in standard C can access the locale variables
 3968 by issuing the following call:

3969 `setlocale(LC_ALL, "")`

3970 If this were omitted, the ISO C standard specifies that the C locale
 3971 would be used."

3972 Change From:
 3973 "If any of the environment variables are invalid, it makes sense to
 3974 default to an implementation-defined, consistent locale environment."

3975 To:
 3976 "The DESCRIPTION of `setlocale()` requires that when setting all
 3977 categories of a locale, that if the value of any of
 3978 the environment-variable searches yields a locale that is not supported
 3979 (and nonnull), the `setlocale()` function returns a null pointer and the
 3980 locale of the process is unchanged.

3981 For the standard utilities, if any of the environment variables
 3982 are invalid, it makes sense to default to an implementation-defined,
 3983 consistent locale environment."

3984 **Change Number:** **XRAT/TC2/D6/6** [XRAT ERN 1]

3985 On Page: 72 Line: 2960-2961 Section: A.12.2 (2003 Ed.)
 3986 On Page: 73 Line: 2948-2949 Section: A.12.2 (2001 Ed.)

3987 Change From:
 3988 "Guidelines 1 and 2 are offered as guidance for locales using Latin
 3989 alphabets. No recommendations are made by IEEE Std 1003.1-2001 concerning
 3990 utility naming in other locales."

3991 To:
 3992 "Guidelines 1 and 2 encourage utility writers to use only characters from
 3993 the portable character set because use of locale specific characters
 3994 may make the utility inaccessible from other locales. Use of uppercase
 3995 letters is discouraged due to problems associated with porting utilities
 3996 to systems that don't distinguish between uppercase and lowercase
 3997 characters in file names. Use of non-alphanumeric characters is
 3998 discouraged due to the number of utilities that treat non-alphanumeric
 3999 characters in "special" ways depending on context (such as the shell
 4000 using whitespace characters to delimit arguments, various quote characters
 4001 for quoting, <dollar-sign> to introduce variable expansion,...)"

Technical Corrigendum 2

4002 **Change Number:** **XRAT/TC2/D6/7** [XSH ERN 89]

4003 On Page: 177 Line: 7479 Section: B.2 Threads (2003 Ed.)

4004 On Page: 178 Line: 7442 Section: B.2 Threads (2001 Ed.)

4005 Insert new section before B.2.10:

4006 "B.2.9.8 Use of Application-Managed Thread Stacks

4007 IEEE Std 1003.1-2001/Cor 2-200x, item XSH/TC2/Dx/x is applied, adding
4008 this new section. It was added to make it clear that the current
4009 standard does not allow an application to determine when a stack can be
4010 reclaimed. This may be addressed in a future revision."

4011 Notes to Editors: The reference should be updated to reflect the
4012 final draft.

4013 **Change Number:** **XRAT/TC2/D6/8** [XBD ERN 11]

4014 On Page: 224 Line: 9264-9265 Section: C.1.9 (2003 Ed.)

4015 On Page: 224 Line: 9226-9227 Section: C.1.9 (2001 Ed.)

4016 Delete the paragraph:

4017 " {POSIX2_SYMLINKS} was developed even though there is no comparable
4018 configuration value for the system interfaces."

4019 Rationale: The pathconf() symbolic constant _PC_2_SYMLINKS
4020 has been introduced as a result of this defect report.

4021 **Change Number:** **XRAT/TC2/D6/9** [XRAT ERN 2]

4022 On Page: 240,241 Line: 9903-9911,9921,9925,9958 Section: C.2.6.4 (2003 Ed.)

4023 On Page: 239,240 Line: 9852-9860,9870,9874,9907 Section: C.2.6.4 (2001 Ed.)

4024 Change From:

4025 "The "({})" form of KornShell arithmetic in early proposals was
4026 omitted. The standard developers concluded that there was a strong
4027 desire for some kind of arithmetic evaluator to replace expr, and that
4028 relating it to '\$' makes it work well with the standard shell language,
4029 and it provides access to arithmetic evaluation in places where accessing
4030 a utility would be inconvenient.

4031 The syntax and semantics for arithmetic were changed for the ISO/IEC
4032 9945-2:1993 standard. The language is essentially a pure arithmetic
4033 evaluator of constants and operators (excluding assignment) and represents
4034 a simple subset of the previous arithmetic language (which was derived
4035 from the KornShell "({})" construct). "

4036 To:

4037 "The standard developers agreed that there was a strong desire for
4038 some kind of arithmetic evaluator to provide functionality similar to
4039 expr, that relating it to '\$' makes it work well with the standard
4040 shell language and provides access to arithmetic evaluation in places
4041 where accessing a utility would be inconvenient.

4042 The syntax and semantics for arithmetic were revised for the ISO/IEC
4043 9945-2:1993 standard. The language represents a simple subset of the
4044 previous arithmetic language (which was derived from the KornShell "({})"
4045 construct). "

4046 On line 9921 (2003 Ed.); line 9870 (2001 Ed.):

4047 Add before the paragraph commencing "In early proposals"

4048 "The standard requires assignment operators to be supported (as listed
4049 in Section 2.7.1), and since arithmetic expansions are not specified to
4050 be evaluated in a subshell environment, changes to variables there have
4051 to be in effect after the arithmetic expansion, just as in the parameter
4052 expansion \${x=value}.

4053 Note, however, that \$((x=5)) need not be equivalent to \$((\$x=5)).

4054 If the value of the environment variable x is the string "y=", the
 4055 expansion of `$((x=5))` would set x to 5 and output 5, but `$(($x=5))`
 4056 output 0 if the value of the environment variable y is not 5 and would
 4057 output 1 if the environment variable y is 5. Similarly, if the value of
 4058 the environment variable is 4, the expansion of `$((x=5))` would still
 4059 set x to 5 and output 5, but `$(($x=5))` (which would be equivalent to
 4060 `$((4=5))`) would yield a syntax error. "

4061 In the paragraph on line 9925 (2003 Ed.); line 9874, (2001 Ed.)
 4062 Change From:
 4063 "The portion of the ISO C standard arithmetic operations selected
 4064 corresponds to the operations historically supported in the KornShell."

4065 To:
 4066 "The portion of the ISO C standard arithmetic operations selected
 4067 corresponds to the operations historically supported in the KornShell.
 4068 In addition to the exceptions listed in section 2.6.4, the use of the
 4069 following are explicitly outside the scope of the rules defined in
 4070 section 1.7.2.1:

4071 * The prefix operator & and the [], -> and . operators.

4072 * Casts."

4073 On line 9958 (2003 Ed.); line 9907 (2001 Ed.)
 4074 Add before the paragraph commencing: "Although the ISO/IEC 9899:1999
 4075 Standard now..." :

4076 "The standard is intentionally silent about how a variable's numeric
 4077 value in an expression is determined from its normal "sequence of bytes"
 4078 value. It could be done as a text substitution, as a conversion like
 4079 that performed by `strtol()`, or even recursive evaluation. Therefore,
 4080 the only cases for which the standard is clear are those for which
 4081 both conversions produce the same result. The cases where they give
 4082 the same result are those where the sequence of bytes form a valid
 4083 integer constant. Therefore, if a variable does not contain a valid
 4084 integer constant, the behavior is unspecified.

4085 For the commands:
 4086 `x=010; echo $((x += 1))`
 4087 the output must be 9.

4088 For the commands:
 4089 `x=' 1'; echo $((x += 1))`
 4090 the results are unspecified.

4091 For the commands:
 4092 `x=1+1; echo $((x += 1))`
 4093 the results are unspecified."